

Makine Öğrenimi Algoritmalarını Kullanarak Jeokonumsal Verilerin Tahmini

Gamze Erdoğan Erten

DOKTORA TEZİ

Maden Mühendisliği Anabilim Dalı

Temmuz 2021

Estimation of Geospatial Data by Using Machine Learning Algorithms

Gamze Erdoğan Erten

DOCTORAL DISSERTATION

Department of Mining Engineering

July 2021

Estimation of Geospatial Data by Using Machine Learning Algorithms

A thesis submitted to the Eskişehir Osmangazi University
Graduate School of Natural and Applied Sciences in partial
fulfillment of the requirements for the degree of Doctor of Philosophy
in Discipline of Mining of the Department of Mining Engineering
by
Gamze Erdoğan Erten

Supervisor: Prof. Dr. Mahmut Yavuz

July 2021

ETHICAL STATEMENT

I hereby declare that this thesis study titled “Estimation of Geospatial Data by Using Machine Learning Algorithms” has been prepared in accordance with the thesis writing rules of Eskisehir Osmangazi University, Graduate School of Natural and Applied Sciences under the academic consultancy of my supervisor Prof. Dr. Mahmut Yavuz. I hereby declare that the work presented in this thesis is original. I also declare, that I have respected scientific ethical principles and rules in all stages of my thesis study, all information and data presented in this thesis have been obtained within the scope of scientific and academic ethical principles and rules, all materials used in this thesis which are not original to this work have been fully cited and referenced, and all knowledge, documents, and results have been presented in accordance with scientific ethical principles and rules. 15/07/2021

Gamze Erdoğan Erten

ÖZET

Jeostatistik, jeokonumsal değişkenleri doğru bir şekilde tahmin etmek için günümüzde kullanılan en yaygın yöntemdir. Ancak, etkin bir şekilde uygulanabilmesi için verilere ilişkin durağanlık ve doğrusallık gibi bazı varsayımlara ihtiyaç duyar. Son yıllarda, makine öğrenimi (ML) algoritmaları özellikle karmaşık koşullarda tahmin için verimli çözümler sunabildiklerinden giderek popüler olmaya başlamıştır. Bununla birlikte, bu algoritmalar: (1) verilerin birbirinden bağımsız olduğunu varsaymakta ve (2) verileri konumlarında yeniden üretememektedirler. Dahası, çoğu ML algoritması tahmin haritalarında koordinat yönleri boyunca önemli yapay görüntüler vermektedir. Bu tezde ML'yi temel alan iki yeni jeokonumsal tahmin yönteminin sunulması amaçlanmaktadır. Topluluk süper öğrenci (ESL) modeli olarak adlandırılan ilk yöntem, ML tahmin haritalarındaki yapaylıkları yönetmek için sunulmuştur. Bu model ML ile modelleme yapabilmek için süper öğrenci (SL) modelinden yararlanmakta ve bir koordinat rotasyon stratejisi ile orijinal veri setinden çok sayıda farklı eğitim seti oluşturmaktadır. İncelenen vaka çalışmaları; ESL modelinin durağanlık ve doğrusallık varsayımlarına ihtiyaç duymadan, tahmin doğruluğu açısından geleneksel jeostatistiksel yöntemlerle karşılaştırılabilir sonuçlar verdiğini ve ESL modelinin anizotropiyi dikkate alıp, tahmin haritalarındaki yapaylıkları yönettiğini göstermektedir. Sunulan ikinci yöntem; kriging ve ML yöntemlerini, her iki yöntemin jeokonumsal verilerin tahminindeki dezavantajlarını azaltmak, daha doğru ve tutarlı tahminler elde etmek için biraraya getirmektedir. ML ve krigingden elde edilen tahmin sonuçları, kriging varyansına dayalı bir ağırlıklandırma fonksiyonu ile birleştirilmekte ve ağırlıklar sıralı ikinci dereceden programlama kullanılarak optimize edilmektedir. Birleştirilmiş yöntem çok sayıda benzetilmiş ve gerçek veri seti üzerinde test edilmiş ve sonuçlar bu yöntemin, Gauss veri seti dışındaki tüm veri setlerinde hem kriging hem de ML' den elde edilenlere kıyasla tahmin sonuçlarını iyileştirdiğini göstermiştir.

Anahtar Kelimeler: Jeokonumsal tahmin, kriging, makine öğrenimi

SUMMARY

Geostatistics is the most prevalent method used today to estimate geospatial variables accurately. However, some assumptions about the data such as stationarity and linearity are needed for geostatistical methods to be effectively applied. In recent years, machine learning (ML) models have started to become popular, as these models promise to provide efficient solutions for estimation, especially in complex cases. However, these models have two major limitations: (1) the data is considered to be independent, and (2) the data is not reproduced at their locations. Moreover, most ML models produce visible artifacts in the resulting estimates along the coordinate directions, which is not realistic in terms of modeling the geological deposits. This thesis aims to present two new geospatial estimation methods based on ML. The first method called the ensemble super learner (ESL) model is presented to manage estimation artifacts in ML geospatial estimation. This model makes use of the super learner (SL) model for ML modeling and creates numerous different training sets from the original dataset by a coordinate rotation strategy. The case studies demonstrate that the ESL model provides results comparable to the traditional geostatistical methods in terms of estimation accuracy without the need for stationarity and linearity assumptions. The ESL model also incorporates anisotropy information and manages the artifacts in ML spatial estimation. The second method combines kriging and ML to mitigate the disadvantages of each method in geospatial estimation as well as obtain more accurate and consistent estimates. In the proposed method, estimation results from both ML and kriging are combined by a weighting function based on the kriging variance, and weights are optimized using sequential quadratic programming. The combined model is demonstrated in numerous synthetic and real case studies and the results indicate that this method improves the estimation results in comparison to the ones obtained both from kriging and the ML in all cases, except in the truly Gaussian dataset.

Keywords: Geospatial estimation, kriging, machine learning

LIST OF CONTENTS

	<u>Page</u>
ÖZET.....	vi
SUMMARY	vii
LIST OF CONTENTS	viii
LIST OF FIGURES	xi
LIST OF TABLES	xvii
LIST OF ABBREVIATIONS AND SYMBOLS.....	xix
1. INTRODUCTION.....	1
2. LITERATURE REVIEW.....	6
2.1. Introduction.....	6
2.2. Random Forest.....	6
2.3. Support Vector Machine.....	8
2.4. Artificial Neural Network.....	10
3. THEORETICAL BACKGROUND.....	16
3.1. Introduction.....	16
3.2. Machine Learning.....	16
3.2.1. Supervised learning.....	17
3.2.2. Generalization, overfitting, and underfitting.....	18
3.2.3. Bias-variance dilemma.....	19
3.2.4. Model assessment.....	20
3.2.5. Parameters and hyper-parameters	22
3.3. Hyper-parameter Tuning.....	22
3.3.1. Grid search algorithm.....	22
3.3.2. Random search algorithm.....	24
3.3.3. Hyper-parameters of the selected ML models	25
3.4. Main Steps of the Applied Machine Learning Process.....	27
3.5. Machine Learning Models	29
3.5.1. <i>k</i> -nearest neighbor	29
3.5.2. Decision tree.....	31
3.5.3. Bagging	32

LIST OF CONTENTS (continued)

	<u>Page</u>
3.5.4. Random forest	33
3.5.5. Extra trees.....	35
3.5.6. AdaBoost.....	35
3.5.7. Gradient boosting machine.....	36
3.5.8. Support vector machine.....	38
3.5.9. Artificial neural network	41
3.5.10. Ensemble models.....	45
3.6. Measures of Evaluation	46
3.7. Geostatistics	49
3.7.1. Random variable and random function	49
3.7.2. Stationarity	50
3.7.3. The variogram function.....	51
<u>3.7.3.1. Experimental variogram.....</u>	<u>52</u>
<u>3.7.3.2. Interpretation of variograms.....</u>	<u>54</u>
<u>3.7.3.3. Variogram models.....</u>	<u>56</u>
3.7.4. Geostatistical estimation	58
<u>3.7.4.1. Simple kriging.....</u>	<u>58</u>
<u>3.7.4.2. Ordinary kriging.....</u>	<u>61</u>
3.7.5. Geostatistical simulation	62
3.7.6. Declustering	64
4. MATERIALS AND METHODS	67
4.1. Introduction.....	67
4.2. The Super Learner Model	67
4.3. Application of the SL Model for the Geospatial Estimation	71
4.4. Application of Kriging for the Geospatial Estimation.....	72
4.5. Partitioning Drill Hole Data into k -Folds.....	73
4.6. Managing Estimation Artifacts in Machine Learning Geospatial Estimation	75
4.6.1. Coordinate system rotation.....	76
4.6.2. Ensemble the estimations of individual SL models	78

LIST OF CONTENTS (continued)

	<u>Page</u>
4.7. Combination of Machine Learning and Kriging for Geospatial Estimation	81
4.7.1. Sequential quadratic programming	86
4.7.1.1. Quadratic programming	86
4.7.1.2. Sequential quadratic programming	89
5. RESULTS AND DISCUSSION	92
5.1. Introduction.....	92
5.2. Case Study – 1: Managing Artifacts in Machine Learning Estimation Through Ensemble Rotation.....	92
5.2.1. Walker Lake case study.....	92
5.2.2. Swiss Jura case study	98
5.3. Case Study – 2: Merging Machine Learning and Kriging Estimates	108
5.3.1. Non-stationary synthetic data example	108
5.3.2. Non-Gaussian synthetic data example	116
5.3.3. Lognormal synthetic data example.....	122
5.3.4. Gaussian synthetic data example.....	128
5.4. Case Study - 3: Implementation of the Methodologies in Mining Cases	134
5.4.1. Oilsands data example.....	135
5.4.2. Gold data example.....	144
6. CONCLUSION AND RECOMMENDATIONS	153
6.1. Statement of Contributions	155
6.2. Recommendation for Future Work	155
REFERENCES	157
APPENDIX	169
APPENDIX – A	170
APPENDIX – B	174
APPENDIX – C	176

LIST OF FIGURES

<u>Figure</u>	<u>Page</u>
2.1 (a) Triangulation neighbor samples space configuration. (b) fixed grid quadrant/octant search neighbor samples space configuration (c) flexible quadrant/octant search neighbor samples space configuration (after Kapageridis, 2005).....	12
2.2. The workflow of the ML and geostatistical estimation (after Samson and Deutsch, 2021)	14
3.1. The relationship between model complexity and training and test errors (after Kanevski et al., 2009).....	19
3.2. Bias-variance trade-off as a function of model complexity	20
3.3. A schematic display of 5-fold cross-validation.	21
3.4. An example of a grid search algorithm flowchart	24
3.5. Visual representation of grid and random search algorithms on two hyper-parameters	25
3.6. The steps of the applied ML process (after Rijmenam, 2019)	27
3.7. An example of the regions defined by a tree-based model.....	31
3.8. The flowchart of RF for regression problems	34
3.9. (a) Possible hyperplanes and (b) optimal hyperplane for completely separable two classes.....	39
3.10. One-dimensional linear SVR (Drucker et al., 1997)	41
3.11. A simple artificial neuron model	42
3.12. A diagram of a neural network with two hidden layers (Kanevski, Pozdnoukhov, Pozdnoukhov and Timonin, 2009).....	43
3.13. The comparison of the bagging and boosting.....	45
3.14. The general flowchart of the stacking learner.	46
3.15. An example of a negative R-squared value	48
3.16. Covariance and variogram.....	52
3.17. The parameters required to calculate an experimental variogram (Deutsch and Journel, 1998).	53
3.18. Features of a variogram	54
3.19. (a) Bouguer dataset (b) omnidirectional variogram (c) variogram on N60E	55
3.20. A variogram shape for cyclicity (after Rossi and Deutsch, 2013).....	55

LIST OF FIGURES (continued)

<u>Figure</u>	<u>Page</u>
3.21. The variogram shapes (a) for geometric and (b) zonal anisotropy (after Rossi and Deutsch, 2013)	56
3.22. Comparison of the variogram models. A standardized sill of 1 is used in the illustration.	57
3.23. An example of the nested model	58
3.24. The comparison of kriging and geostatistical simulation (after Rossi and Deutsch, 2013).	63
3.25. Illustration of the cell declustering method (Deutsch, 2020)	65
3.26. The cell declustering mean versus different cell sizes (after Rossi and Deutsch, 2013)	66
4.1. The simplified workflow of the SL model	69
4.2. An example for a random 5-fold cross-validation.....	73
4.3. Two folds with (a) random and (b) controlling the closeness	75
4.4. Two-fold spatial partitioning.....	75
4.5. An example aims to capture the true anisotropy	77
4.6. Rotation of azimuth in 2D (after Deutsch, 2015)	78
4.7. The rotated training datasets in 5° increments.....	79
4.8. Flowchart of the ESL model.....	80
4.9. Data reproduction for kriging and SL model.....	82
4.10. The workflow of the combination of ML and kriging for spatial estimation strategy.....	84
4.11. Parallel gradients in the Lagrange method (after Chinneck, 2006).....	87
5.1. (a) the sample and (b) exhaustive datasets for variable V	93
5.2. Cell size declustering of variable V.....	93
5.3. Histogram plots of the (a) sample and (b) exhaustive datasets for variable V	94
5.4. Modeled variograms for variable V.....	94
5.5. Estimation performance of OK, SL, and ESL models for variable V in Walker Lake dataset.....	96
5.6. Omnidirectional variogram of raw data and the residuals of ESL model estimations .	97
5.7. Spatial estimation maps of OK, SL, and ESL models for variable V in Walker Lake dataset.....	97

LIST OF FIGURES (continued)

<u>Figure</u>	<u>Page</u>
5.8. The location maps of the continuous variables in the Jura training dataset	98
5.9. The location maps of the continuous variables in the Jura validation dataset.....	99
5.10. The histograms of the continuous variables in the Jura training dataset.....	100
5.11. The histograms of the continuous variables in the Jura validation dataset	100
5.12. The omnidirectional variograms of the continuous variables in the Jura training dataset	101
5.13. OK model spatial estimation maps of the continuous variables in the Jura dataset.	103
5.14. SL model spatial estimation maps of the continuous variables in the Jura dataset..	104
5.15. ESL model spatial estimation maps of the continuous variables in the Jura dataset	105
5.16. The omnidirectional variogram for both raw data and the residuals from the ESL model in the Jura dataset.....	107
5.17. (a) Reference realization plan map, (b) sampling plan map at the elevation of 190 m, (c) reference realization, and (d) sampling E-W section map looking from North (non-stationary synthetic data).....	109
5.18. Histogram of (a) reference realization and (b) sampling data (non-stationary synthetic data).....	110
5.19. Variogram map of the target value for non-stationary synthetic data	110
5.20. (a) The distribution and (b) plan maps of the weights at an elevation of 190 m for SK and the SL models for the nonstationary synthetic data	112
5.21. Illustration of objective contours and the feasible solution of parameters of weight for the nonstationary synthetic data.	113
5.22. Spatial estimation maps of all models in (a) plan view at an elevation of 190 m and (b) E-W section looking from North for the nonstationary synthetic data.	114
5.23. Estimation performance of all models for the nonstationary synthetic data.	115
5.24. Omnidirectional variogram of raw data and SK+SL model residuals for the nonstationary synthetic data.....	116
5.25. Data reproduction for SK, SL, and SK+SL models	116
5.26. (a) Reference realization plan map, (b) sampling plan map at an elevation of 190 m, (c) reference realization, and (d) sampling E-W section map looking from North (non-Gaussian synthetic data).....	117
5.27. Histogram of (a) reference realization (b) sampling data (non-Gaussian synthetic data).	118

LIST OF FIGURES (continued)

<u>Figure</u>	<u>Page</u>
5.28. Spatial estimation maps of all models in (a) plan view at an elevation of 190 m and (b) E-W section looking from North for the non-Gaussian synthetic data.....	119
5.29. Estimation performance of all models for the non-Gaussian synthetic data	120
5.30. (a) The distribution and (b) plan maps of the weights at the elevation of 190 m for SK and the SL models for the non-Gaussian synthetic data	121
5.31. Illustration of objective contours and the feasible solution of parameters of weight for the non-Gaussian synthetic data.....	121
5.32. Omnidirectional variogram of raw data and SK+SL model residuals for the non-Gaussian synthetic data	122
5.33. (a) Reference realization plan map, (b) sampling plan map at the elevation of 190 m, (c) reference realization, and (d) sampling E-W section map looking from North (lognormal synthetic data).....	123
5.34. Histogram of (a) reference realization (b) sampling data (lognormal synthetic data).	123
5.35. Estimation performance of all models for the lognormal synthetic data.....	125
5.36. Spatial estimation maps of all models in (a) plan view at an elevation of 190 m and (b) E-W section looking from North for the lognormal synthetic data.	126
5.37. (a) The distribution and (b) plan maps of the weights at an elevation of 190 m for SK and the SL models for lognormal synthetic data.....	127
5.38. Illustration of objective contours and the feasible solution of parameters of weight for the lognormal synthetic data.	127
5.39. Omnidirectional variogram of raw data and SK+SL model residuals for the lognormal synthetic data.....	128
5.40. (a) Reference realization plan map, (b) sampling plan map at an elevation of 190 m, (c) reference realization, and (d) sampling E-W section map looking from North (Gaussian synthetic data).	129
5.41. Histogram of (a) reference realization (b) sampling data (Gaussian synthetic data).	129
5.42. Estimation performance of all models for the Gaussian synthetic data.	131
5.43. Spatial estimation maps of all models in (a) plan view at an elevation of 190 m and (b) E-W section looking from North for the Gaussian synthetic data.	132
5.44. (a) The distribution and (b) plan maps of the weights at the elevation of 190 m for SK and the SL models for the Gaussian synthetic data	133

LIST OF FIGURES (continued)

<u>Figure</u>	<u>Page</u>
5.45. Illustration of objective contours and the feasible solution of parameters of weight for the Gaussian synthetic data.	133
5.46. Omnidirectional variogram of raw data and SK+SL model residuals for the Gaussian synthetic data.....	134
5.47. (a) Location plan map of Bitumen at an elevation of 235 m, (b) location map in 3-D view	135
5.48. Modelled Bitumen variograms.	135
5.49. (a) Cell size declustering, (b) histogram with declustering of bitumen.....	136
5.50. Original, training, and validation dataset plan map for the first fold at the elevation of 235 m.....	137
5.51. The histograms of the original, training, and validation datasets for the first fold ..	137
5.52. Estimation performance of OK, SL, and OK+SL models for each fold of oilsands data	139
5.53. Spatial estimation maps of all models in (a) plan view at an elevation of 235 m and (b) E-W section looking from North for the oilsands data	141
5.54. The variogram of raw data and cross-validation residuals	142
5.55. The dataset division for the performance comparison of models in different numbers of DHs for the oilsands data.....	143
5.56. The comparison of OK and OK+SL models in different numbers of DHs for oilsands data	143
5.57. (a) Location plan map of gold at an elevation of 210 m, (b) location map in 3-D view	144
5.58. (a) Cell size declustering, (b) histogram with declustering of the gold variable.....	145
5.59. Variograms of the gold variable in three directions	145
5.60. Original, training, and validation datasets plan maps for the first fold at the elevation of 210 m	146
5.61. The histograms of the original, training, and validation datasets for the first fold ..	147
5.62. Spatial estimation maps of all models in (a) plan view at an elevation of 210 m and (b) E-W section looking from North for the gold data	148
5.63. The variogram of raw data and cross-validation residuals	150
5.64 The dataset division for the performance comparison of models in the different number of DHs for the gold data.....	151

LIST OF FIGURES (continued)

<u>Figure</u>	<u>Page</u>
5.65. The comparison of SK and SK+ESL models in different numbers of DHs for the gold data	152



LIST OF TABLES

<u>Table</u>	<u>Page</u>
3.1. Illustration of underfitting, good-fitting, and overfitting (after James et al., 2013)	19
3.2. Grid Search algorithm with cross-validation to tune the hyper-parameters (after Kuhn and Johnson, 2019).	23
3.3. Hyper-parameter types and definitions for each ML model (Notations of hyper-parameters are from the respective scikit-learn packages in Python (after Pedregosa et al., 2011)).	26
3.4. The k NN for regression problems	29
3.5. A simple tree structure example	31
3.6. The gradient boosting (after Hastie et al., 2009)	37
5.1. The best configuration of hyper-parameters for variable V in Walker Lake dataset. ..	95
5.2. The comparison of general metrics and the basic statistics of OK, SL, and ESL models for V variable in the Walker Lake dataset	96
5.3. The best configuration of hyper-parameters for the continuous variables in the Jura dataset.	102
5.4. The comparison of general metrics of OK, SL, and ESL models for continuous variables in the Jura dataset.	106
5.5. The basic statistics of OK, SL, and ESL models for continuous variables in the Jura dataset.	106
5.6. The best configuration of hyper-parameters for the nonstationary synthetic data	111
5.7. The comparison of general metrics and the basic statistics of all models for the nonstationary synthetic data	115
5.8. The best configuration of hyper-parameters for the non-Gaussian synthetic data	118
5.9. The comparison of general metrics and the basic statistics of all estimation models for non-Gaussian synthetic data	120
5.10. The best configuration of hyper-parameters for the lognormal synthetic data	124
5.11. The comparison of general metrics and the basic statistics of all models for the lognormal synthetic data	125
5.12. The best configuration of hyper-parameters for the Gaussian synthetic data	130
5.13. The comparison of general metrics and the basic statistics of all models for the Gaussian synthetic data	131
5.14. The best configuration of hyper-parameters for the oilsands data	138

LIST OF TABLES (continued)

<u>Table</u>	<u>Page</u>
5.15. The optimized b_0 and b_1 for each fold	138
5.16. The comparison of general metrics of all estimation models for oilsands data	140
5.17. The comparison of the basic statistics of all estimation models and of oilsands data in each fold	140
5.18. The comparison of OK and OK+SL models in different numbers of DHs for oilsands data	144
5.19. The best configuration of hyper-parameters for the gold data	146
5.20. The optimized b_0 and b_1 for each fold	147
5.21. The comparison of general metrics of all estimation models for the gold data.....	149
5.22. The comparison of the basic statistics of all estimation models and of gold data in each fold.	149
5.23. The comparison of SK and SK+ESL models in different numbers of DHs for the gold data	151

LIST OF ABBREVIATIONS AND SYMBOLS

<u>Abbreviations</u>	<u>Descriptions</u>
2D	2 (Two) Dimensional
3D	3 (Three) Dimensional
ADA	AdaBoost
ANN	Artificial Neural Network
BAG	Bagging
C	Cost
CDF	Cumulative Distribution Function
DH	Drillhole
DT	Decision Tree
ELM	Extreme Learning Machine
ESL	Ensemble Super Learner
EXT	Extra Trees
FNN	Feed Forward Neural Network
g-ROKY	Grouped Random Forest + Kriging
GA	Genetic Algorithm
GBM	Gradient Boosting Machine
GPR	Gaussian Process
GSLIB	Geostatistical Software Library
IK	Indicator Kriging
IDS	Inverse Distance Squared
KED	Kriging with External Drift
kNN	k - Nearest Neighbor
KKT	Karush–Kuhn–Tucker
LMBP	Levenberg-Marquardt backpropagation
LR	Linear Regression
MAE	Mean Absolute Error

LIST OF ABBREVIATIONS AND SYMBOLS (continued)

<u>Abbreviations</u>	<u>Descriptions</u>
ME	Mean Error
MIK	Median Indicator Kriging
ML	Machine Learning
MLP	Multi-Layered Perceptron
MSE	Mean Squared Error
NMSE	Normalized Mean Squared Error
OK	Ordinary Kriging
OOB	Out-of-Bag
QP	Quadratic Programming
QRF	Quantile Random Forest
RBF	Radial Basis Function Network
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Networks
ROKY	Random Forest + Kriging
R-squared	The Coefficient of Determination
RVM	Relevance Vector Machine
st.dev.	Standard Deviation
SK	Simple Kriging
SL	Super Learner
SGS	Sequential Gaussian Simulation
SQP	Sequential Quadratic Programming
SSE	Sum of Squares Error
SVM	Support Vector Machine
SVR	Support Vector Regression

1. INTRODUCTION

The accurate estimation of geological variables is one of the most important stages in the evaluation of a given deposit (Abzalov, 2016). Generally, a small fraction of the deposit can be sampled due mainly to the cost associated with sampling. Therefore, the attributes of interest must be estimated at unsampled locations in a deposit (Rossi and Deutsch, 2013). Over the past 40 years, the traditional geostatistical estimation techniques have become the established methodology for geospatial data analysis, as it incorporates the spatial variability into the estimation process through the variogram which measures the spatial correlation between the data and the unsampled locations (Deutsch and Journel, 1998; Pyrcz and Deutsch, 2014). The main steps involved in the spatial estimation using geostatistics: (1) calculate the experimental variogram, (2) fit a positive definite analytical model to the variogram, and (3) generate estimates using kriging.

Kriging algorithms have been used as the best unbiased linear estimation technique for years (Goovaerts, 1997). However, they assume that the relationships between covariates and dependent are linear, and the required structural information is based on two-point statistics (i.e. covariances or variogram) (Yamamoto, 2000). Therefore, these methods may not be sufficient to capture the broad nature of spatial nonlinearity and complex features. They must also rely on random function model assumptions such as second-order stationarity (mean and variance are constant across the study region, and the covariance depends only on the lag distance $\mathbf{h}$) which may be questionable and difficult to verify (Wang et al., 2019). Moreover, geostatistical methods require expert knowledge for variogram modeling and parameter selection.

In some cases, data can be multivariate, non-Gaussian, non-stationary and the phenomenon being sampled can be generated by non-linear processes. Considering these cases, the spatial estimation of continuous variables could be a challenging task. ML algorithms/models allow one to identify the patterns in the complex datasets and to make estimations based on them. The key feature of ML models is that they learn from data, and they do not require rigid statistical assumptions (such as stationarity and linearity). In

addition, ML models are more flexible towards incorporating, combining, and extending covariates of different types (Kanevski et al., 2009).

The implementation of ML models for spatial estimation is not an easy task, as these methods consider that the data are independent and identically distributed. In fact, ML models are not designed to handle spatial data; that is, the particular relationship between the spatial proximity among observational units and the numeric similarity among their values are ignored. Moreover, ML models do not reproduce data at known locations (contrary to the exactitude property in kriging algorithms). Besides all these, ML models could only learn anisotropic features provided that these features are aligning with the coordinate system. Furthermore, most of the ML models produce visible artifacts in the resulting estimates along the coordinate directions.

The first aim of this thesis is twofold: (1) to manage visual artifacts in ML spatial estimation maps and (2) to incorporate anisotropy information into the estimation process. For these aims, a method called the ESL model which uses the SL model for ML modeling has been proposed. The SL model is an ensemble method in which the optimal weights for merging the estimates from a group of ML models are adjusted to obtain a final SL estimation that performs at least as well as any of its component models. The SL model has been conducted in a way to get maximum information from the spatial data by using the coordinates of the known data as input. The hyper-parameters of the candidate ML models that form the SL model have been tuned by the grid search algorithm with 5-fold cross-validation.

The ESL model uses a set of trained SL models on a set of rotated datasets obtained from a coordinate rotation strategy and ensembles the produced estimates. Comprehensive analyses through three case studies have been performed in order to investigate the behavior of the ESL model and compare it with kriging and the SL model. Coefficient of determination (R-squared), root mean squared error (RMSE), and mean absolute error (MAE) have been used as performance evaluation criteria for comparison and validation. Moreover, the reproduction of basic statistics (i.e. mean and standard deviation) has been evaluated and visual inspection of the estimates for artifacts has been made. The presence of the spatial structures in the residuals has also been assessed to prove that the ESL model

extracts spatially structured information from data. The results show that the estimation performance of the ESL model is at least as good as kriging without the need for stationarity and linearity assumptions. It also improves the estimation performance of the SL model and manages the visual artifacts in estimation maps. Therefore, the ESL model can be considered as a general and automated method for geospatial estimation.

The second aim of this thesis is to combine ML and kriging to generate more accurate, consistent, and unbiased geospatial estimates. In this method, the weights of ML, which will also be the SL model, and kriging are determined according to the kriging variance and optimized with sequential quadratic programming. Considering the geostatistical estimation for the geospatial datasets, there are several approaches each with different implementation advantages and disadvantages. However, in this thesis, simple kriging (SK) and ordinary kriging (OK) are the main geostatistical estimation techniques that have been selected to introduce and demonstrate the methodology. Four simulated synthetic datasets and two real datasets have been examined to validate the estimation performance of the combined model. The synthetic datasets contain underlying reference data against which the estimates from the methods can be directly validated and compared. For the real datasets, on the other hand, the exhaustive data is not known. Therefore, a new k -fold cross-validation approach that considers the spatial configuration of the data has been implemented. This approach spatially divides the data into k subsets. Then, each time, one of the k subsets is kept aside as a validation dataset where the results are compared, and the remaining $k - 1$ fold is used as a training dataset. In this thesis, as it is frequently used in the literature, the value of k was chosen as 5 (Hastie et al., 2009; Kuhn and Johnson, 2013). The combined model takes advantage of the unbiasedness and exact data reproduction properties of kriging and the ability of ML to model complex phenomena. Therefore, it performs better than the kriging and ML model in all examined cases, except in the Gaussian case, in terms of R-squared, RMSE, and MAE criteria.

Proposed Thesis Statement:

A coordinate rotation ensemble strategy (ESL model) manages the ML estimation artifacts and incorporates anisotropy information into the ML estimation process. Thus, it provides more accurate geospatial estimates with ML. Moreover, the combination of ML and kriging will improve the geospatial estimates in many situations that may be difficult for only using the kriging or only using ML.

This thesis is divided into six chapters, each of which reflects the components required to achieve the aims and objectives of the study undertaken. The outline of the thesis is as follows:

Chapter 2 begins with a literature review on the implementation of the most widely applied ML models, such as random forest (RF), support vector machine (SVM), and artificial neural network (ANN), for geospatial data estimation.

Chapter 3 gives a brief review on the basic concepts of the essentials of ML and geostatistics. In particular, the ML models that will be used to perform the geospatial estimation and classical geostatistical estimation methods that account for two-point statistics are reviewed and discussed.

Chapter 4 is the core chapter describing the proposed methodologies in detail. This chapter firstly discusses the details of the SL model, that will be used for the ML modeling. The general process of applying both the SL model and kriging for geospatial estimation is given. A data partitioning procedure compatible with spatial data estimation is presented. After that, the methodology for managing visual artifacts in ML estimation maps and incorporating anisotropy information into the ML estimation process is proposed. Finally, the methodology for combining ML and kriging based on the kriging variance and sequential quadratic programming to generate more accurate geospatial estimates is developed and discussed.

The first section of **Chapter 5** illustrates and tests the proposed method for managing artifacts in ML estimation through ensemble rotation, called the ESL model throughout the

thesis, using two public domain datasets including Walker Lake and Swiss Jura datasets. The practical details of the application of the developed method are discussed. The estimation results are compared with those produced by the kriging and SL model.

In the second section of **Chapter 5**, the main attention is paid to the application of the combined model on four different synthetic datasets including nonstationary, non-Gaussian, lognormal, and Gaussian conditions. The synthetic spatial datasets are generated through simulations with known spatial characteristics because it is difficult to measure the success of one method over another without comparing the results against reference values. The comparison of the estimation performance of the kriging, the SL model, and the combined model with the underlying reference data is discussed for all cases.

The last section of **Chapter 5** illustrates the proposed methods with actual data from oilsands and gold deposits as well as shows the importance of considering the ML into the geospatial estimation. The demonstrations are discussed using case studies with (1) the combined models that include the kriging and SL model and (2) that include the kriging and the ESL model. Issues in their practical implementation are highlighted. The results are evaluated and compared through several performance evaluation criteria with 5-fold cross-validations. The value of the improvement in performance measures with the proposed methodologies is also discussed in terms of practical economic engineering sets.

Chapter 6 wraps up the thesis with conclusions and future work. The advantages and limitations of using the proposed methodologies in geospatial estimation are evaluated. The future developments related to the proposed methodologies are contemplated.

2. LITERATURE REVIEW

2.1. Introduction

In recent years, ML has appeared to gain popularity among researchers as a reliable tool for geospatial estimation due to its proven ability to model nonlinear and complex relationships in datasets. In this chapter, for geospatial estimation, the ML models that have recently yielded promising results and are prominent among others such as RF, SVM, and ANN are reviewed. This chapter also emphasizes the importance of the proposed methodologies. The details of most of the ML models presented in this chapter will be given in Section 3.2.

2.2. Random Forest

In recent years, there is an increasing interest in using RF for geospatial regression and classification problems. For example, Li et al. (2011) applied 23 methods, including RF, SVM, OK, inverse distance squared (IDS), and their different combinations, to the estimation of mud content samples. For the combination of the methods, firstly the authors applied ML models to the data, and then implemented OK or IDS to the residuals of these models. The estimated values of each model and the corresponding estimated residual values were added together to produce the final estimations of each combined method. The authors assessed the accuracy of methods using cross-validation and then visually examined the spatial estimation maps of the most accurate methods. The results indicated that the RF and its combinations were the most accurate methods in terms of root MAE and relative RMSE.

Matin and Chelgani (2016) investigated the performance of the RF to estimate the gross calorific value of coal samples from a variety of US states. The authors made use of the ability of the RF to model large dimensional datasets and to rank features (input variables) through its inbuilt variable importance measures. RF estimated the complicated relationships accurately in coal features in terms of R-squared and mean error (ME) and yielded a reasonable variable importance rank. The authors suggested that for the models

with the uncomplicated relationships between inputs and outputs, well-understood techniques such as linear regression (LR) could be used rather than more complicated methods such as RF.

Hengl et al., (2018) proposed an RF methodology, called RFsp, for spatial estimation that uses buffer distances from sampling locations as explanatory variables alongside the available inputs such as coordinates. In the proposed approach, the geographical proximity effects were incorporated into the estimation process through buffer distances to consider the spatial autocorrelation. The performance of the RFsp was compared with kriging methods, such as OK and universal kriging, in terms of a variety of the model evaluation criteria such as RMSE, R-squared, and ME. The authors showed that the RF yielded equally accurate and unbiased estimations as different versions of kriging. However, the success of the RF depends heavily on the quantity and quality of the data. It may not yield reasonable results for large datasets where the derivation of buffer distances is a challenging task.

Kıvrak (2018) proposed two different hybrid methods that combines RF and OK and called them ROKY (random forest + kriging) and g-ROKY (grouped random forest + kriging), respectively. These methods mainly aimed to consider the grade variables in domain modeling. In the ROKY method, firstly RF was trained using continuous variables, such as grade, as inputs and categorical variables, such as rock type, as outputs. Then, the OK produced estimates on the unsampled locations, and these estimates were feed into the RF to estimate the domains. In the g-ROKY method, the sample and estimates obtained from kriging were grouped in a standardized form. Then, RF produced estimates on these grouped datasets. The g-ROKY method was developed to avoid the smoothing effect of kriging which may cause misclassifications. The proposed methods were evaluated with the Jura dataset and a dataset from a coal deposit. The g-ROKY method showed better performance than ROKY and indicator kriging (IK) in terms of the classification matrix.

Jafrasteh et al. (2018) evaluated the performances of RF, multi-layered perceptron (MLP), and Gaussian process (GPR) in the problem of the copper grade estimations of a porphyry deposit. A GPR is a class of nonparametric regression techniques extensively used for solving non-linear regression and classification problems. Their results showed that the specially designed GPR had the best overall performance among the ML models as well as

the IK and OK in terms of mean squared error (MSE) and normalized mean squared error (NMSE). In this study, the importance of (1) the spatial partitioning of the dataset, (2) considering spatial anisotropy distribution explicitly in the predictive model, and (3) incorporating information on the rock type as input in ML models were emphasized to improve the estimation performances.

Fouedjio and Klump (2019) compared the kriging with external drift (KED) and quantile random forest (QRF) with respect to their accuracy in making point estimations and their success in modeling estimation uncertainty of spatial data. For this comparison, they used different simulated datasets and a real dataset from a geochemical soil survey. Their results briefly showed that the QRF outperformed KED in terms of estimation accuracy when there was a nonlinear relationship between the variable of interest and the secondary variable. However, as the QRF is a non-parametric model and does not consider spatial autocorrelation, it did not produce reliable estimation uncertainties for data that has a strong spatial correlation, which leads to sub-optimal estimations.

2.3. Support Vector Machine

Another ML model gaining significant attention in geospatial estimation is SVM. Kanevski et al. (2002) are one of the earliest researchers that applied SVM for spatially distributed data. The authors reviewed the use of SVM for modeling spatially distributed data and combined it with geostatistical approaches such as exploratory data analysis and exploratory variography. They validated the SVM in the binary classification of spatial categorical data and the spatial regression of continuous porosity data with tuned hyperparameters (the cost parameter C , kernel parameter γ , and insensitive value ϵ). The obtained results were compared with IK and OK in case of classification and regression problems, respectively, and SVM proved that it was a promising method with low validation errors.

Al-Anazi and Gates (2010a) examined the potential of an SVM formulation, together with feature selection based on fuzzy theory, to classify lithology from good logs in a heterogeneous sandstone reservoir. They optimized the hyperparameters, i.e. C , γ , and another kernel parameter K , of SVM using both grid and pattern search along with cross-

validation. The authors revealed that the proposed methodology showed better performance compared to discriminant analysis and probabilistic ANN in terms of accuracy metrics. In another study, Al-Anazi and Gates (2010b) explored the support vector regression (SVR) to correlate porosity to well log data from a heterogeneous sandstone reservoir. The results showed that SVR was typically more accurate and robust than ANN methods in terms of RMSE, MAE, and average absolute error. However, it needed larger memory and much more computational time than ANN methods.

Dutta et al. (2010) applied SVM and ANN (Levenberg-Marquardt backpropagation (LMBP) learning model) to estimate the grade values of a placer gold deposit where samples were sparsely distributed. The input and output variables were determined as coordinate information (i.e., Easting, Nothing, and water depth) and the gold grade value, respectively. The hyperparameters of the SVM (i.e., C and gamma) were tuned with the grid search algorithm. The performance of ML models compared with OK in terms of ME, MAE, RMSE, and R-squared. SVM yielded better results than other methods. However, the authors declared that because of the presence of extreme data values in the examined datasets, the performance improvement was only marginal. Samui and Sitharam (2010) examined the applicability of SVM and relevance vector machine (RVM) to build models for spatial variability of reduced levels of rock. The hyperparameters of the models were selected by a trial-and-error approach. The results showed that both models had the ability to build accurate models and the performance of RVM is slightly better than SVM in terms of R-squared.

Chatterjee and Bandopadhyay (2011) applied a version of SVM, called least square support vector regression, to the estimation of a platinum placer deposit that had high variability and sparsely sampled data. The input space of the SVM consisted of a combination of different neighborhood samples, their distances, and angles from the estimated point. The hyper-parameters, which were C and gamma, were tuned by the genetic algorithm (GA). The proposed SVM was compared with traditional SVM, where just spatial coordinates were used as input, and OK in terms of ME, MAE, and R-squared. Results demonstrated that the input space configuration strategy improved the performance of the model and thus the proposed input space-configured SVM model outperformed the other

two models. Moreover, it was expected further performance improvement as the number of neighbor samples increased.

Zhang et al. (2013) proposed a weighted least square SVR to estimate the ore grade of a seafloor hydrothermal sulfide deposit, where a large proportion of samples were lack and distributed spatially discontinuously. The hyper-parameters, i.e. C and gamma, were tuned with coupled simulated annealing and grid search algorithms. The data division was conducted with the particle swarm optimization (PSO) algorithm, rather than the random division, to obtain training and validation datasets that had similar statistical properties without large discrepancies. The proposed methodology demonstrated robust predictive and generalization ability. Moreover, it yielded superior results in comparison to inverse distance weight, OK, and backpropagation neural networks in terms of selected performance criteria such as ME, MAE, RMSE, and R-squared.

Li et al. (2013) applied an SVR model that was optimized by self-adaptive learning-based particle swarm optimization for ore grade estimation of several porphyry deposits. The experimental results showed that the proposed model optimized the hyper-parameters of the SVR, which were C , gamma, and ε , efficiently and reliably. It also showed better estimation performance than ANN in terms of RMSE and did not trap in the local optimum. Goswami et al. (2016) used the SVR to assess the grade of an iron ore deposit. The input parameters were taken as three-dimensional spatial coordinates (i.e., X, Y, and Z) and ten different lithological features, and the output variable was taken as ore grade. The performance of the SVR was compared with the GPR and ordinary kriging, using ME, MAE, and RMSE as the performance measures. The results revealed that the performance of GPR was the best, followed by SVR.

2.4. Artificial Neural Network

For many years, ANN has been the most frequently used ML model for geospatial estimation problems. For example, Wu and Zhou (1993) applied an MLP with a sigmoid activation function for modeling the heterogeneous sample data obtained from a copper deposit. The authors used the drill hole coordinates as the input layer and Cu grade values as the output layer. Although the proposed method indicated that ANN could be employed

as a generic reserve estimation tool, it overestimated and underestimated low-frequency values (e.g., high-grade values) because of its smoothing effect. Kapageridis (1999) examined the potential of two types of ANN methods, which were MLP and the radial basis function network (RBF), for the ore grade estimation. The results indicated that the RBF was a better choice for geospatial estimation because it constructs local approximations as opposed to global approximations of the MLP. The authors also built an ML-based software that analyzes and estimates grade values for given training datasets automatically. Koike et al. (2002) implemented a feedforward neural network (FFN) to estimate the metal contents of Kuroko deposits. The relationship between input variables, which are coordinates and lithology, and corresponding target values was detected by repetitive network learning. This study demonstrated that the ANN had the potential to be an alternative to geostatistical methods.

Matias et al. (2004) evaluated and compared the performance of MLP, ensemble neural networks, RBF, and OK using simulated datasets with different scenarios. The results revealed that the OK outperformed the ANN methods for the examined datasets in terms of RMSE and MAE. Samanta et al. (2004) implemented ANN and different types of kriging for grade estimation of a part of the Nome offshore placer gold deposit. The authors also proposed a combined approach of data segmentation and a Kohonen network to produce appropriate data division rather than random division that produces statistically dissimilar sub-datasets. ANN and kriging methods performed equally in terms of ME, MAE, MSE, and R-squared.

Kapageridis (2005) investigated the effects of dimensionality, referred to the number and type of ANN input parameter, effects on the performance of RBF applied to grade estimation. The author configured the available data to form different multidimensional input spaces and tested them on different datasets. X–Y coordinate space, triangulation neighbor samples space (i.e. three grades + three distances = 6 dimensions), and quadrant/octant search neighbor samples space, with a fixed and flexible grid, can be given as an example of the 2D input space configuration (Figure 2.1). The performance of the RBF network systems in all input space configurations showed that there was no globally feasible configuration for all datasets and sampling scheme types.

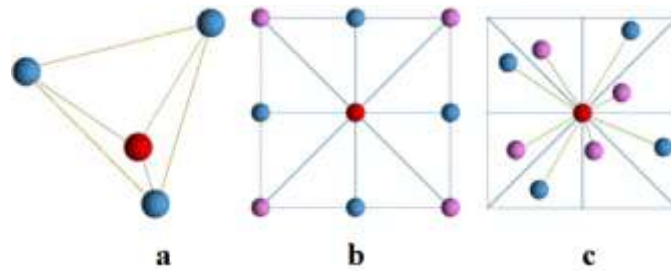


Figure 2.1 (a) Triangulation neighbor samples space configuration. (b) fixed grid quadrant/octant search neighbor samples space configuration (c) flexible quadrant/octant search neighbor samples space configuration (after Kapageridis, 2005)

Samanta et al. (2005) investigated the performance of ANN in estimating five variables of a bauxite deposit. They used GA to divide the dataset to obtain statistically similar training and validation datasets. The ANN and OK performed equally in terms of ME, MAE, and MSE. Mahmoudabadi et al. (2009) presented a hybrid method where MLP was trained using the combination of LMBP and GA. In this method, the initial weights of the MLP were optimized especially for the size of the training data set was small. The results showed significant improvements compared to both conventional MLP and kriging methods. Li et al. (2010) investigated the wavelet neural network approach, which is a nonlinear and intelligent method, to provide fast and reliable ore grade estimation with minimum assumptions. Many case studies revealed that this approach improved the estimation accuracy and the learning speed of the conventional approaches by capturing the local nonlinearity in the datasets.

Chatterjee et al. (2010) developed an ensemble-based ANN for ore grade estimation of a lead-zinc deposit. They used the k-means clustering method to select the optimum subset of candidate networks and improve diversity. GA was used to improve accuracy by calculating the weights of ensemble members. The 3D spatial coordinates and lithology were used as input features. To estimate the lithology at data points where lithological information is not available, SVM was used. The result of this genetic ensemble network model compared with average ensemble model, weighted average ensemble model, and OK. Results showed that the proposed model performed better than the others in terms of such as ME, MAE, MSE, and R-squared, etc.

Badel et al. (2011) compared the median indicator kriging (MIK) and an ANN in the estimation of grades of an iron ore deposit. The results showed that the estimates of MIK had better local precision than the ANN and yielded more reasonable estimation maps. Tahmasebi and Hezarkhani (2011) proposed a modular FFN that had a special structure and links for modeling the grade of an iron deposit. In this method, the interconnections and the network size were reduced according to their final performance. Rather than finding the optimum topology and weights with optimization methods such as GA, the modular FFN selected a network that yielded the minimum error from a range of network architectures. Thus, it offered a less complex and more rapid method, as well as yielded improved performance in smaller training samples. The proposed method showed better performance than OK and MLP while escaping from the local minima. However, the results could be biased as most parts of the processing were determined by the user's interference.

Leuenberger and Kanevski (2015) investigated the potential of the extreme learning machine (ELM), which follows the structure of an MLP with just one single hidden layer, especially for the complex spatial estimation problems. The number of neurons which is the unique hyperparameter of the ELM model and controls its complexity was optimized by cross-validation among a range of different numbers. The analysis of the ELM model was performed on both the simulated dataset and the Lake Geneva dataset that contains noisy and irrelevant data and features. The results revealed that the model had capable of modeling the structured information in data and did not show a spatial structure in their residuals.

Jalloh et al. (2016) presented a method that integrates ANN and geostatistical techniques for mineral reserve evaluation. In the presented method, called ANNMG, firstly validated ANN was used to generalize the known and unknown sampled locations, and then they were fed to the geostatistics to develop the gridded 3D geological block model. The ANNMG yielded accurate grade estimates in terms of MSE and R-squared as well as reduced the drilling requirement but it underpredicted the high-grade values that had relatively few training cases. Jahangiri et al. (2018) employed MLP with the Gustafson-Kessel algorithm to estimate the concentration of different geochemical elements considering coordinates as input features. The available data were clustered with the Gustafson-Kessel algorithm to obtain similar samples in the clusters and then estimators were designed for these clusters. The optimum number of clusters was identified based on

the validation indices which were such as partition coefficient and classification entropy. The clustering methodology improved the performance and estimation accuracy of MLP in terms of R-squared and MSE.

Kaplan and Topal (2020) proposed a hybrid model that consisted of MLP and k-nearest neighbor (kNN) to estimate the grade distribution within a gold deposit. In their approach, firstly rock types and alteration levels were estimated at unsampled locations with the kNN and then spatial coordinates with these geological features created input space of the MLP. The ML models in this study underestimated any high-grade values. The results also revealed that the inclusion of the categorical features improved the performance of the MLP in terms of MAE, MSE, and R-squared. However, this study did not present any analysis for the spatial distributions of the estimates. Samson and Deutsch (2021) proposed a hybrid estimation technique that appears to outperform kriging in complex spatial problems such as nonstationary and nonlinear by maximizing the value of data. In this method, ordinary intrinsic collocated cokriging used the estimates of elliptical radial basis function neural networks as secondary data. The general workflow is given in Figure 2.2. The combination of these two methods improved the mineral resource estimations in terms of R-squared and data reproductions. The authors also validated the modeling artifacts were not present in their estimation and error maps.

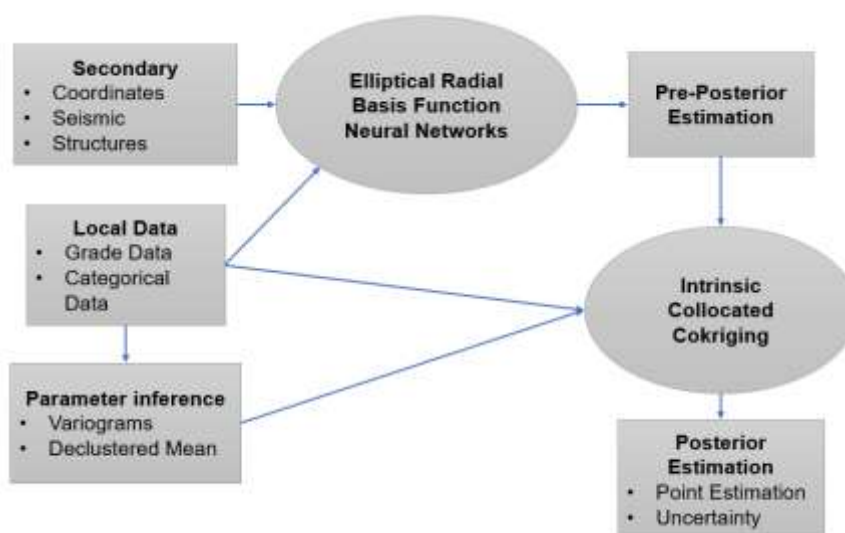


Figure 2.2. The workflow of the ML and geostatistical estimation (after Samson and Deutsch, 2021)

Literature review shows that the ML models give promising results for geospatial estimation. These models have been generally preferred as they require less data preprocessing, the ability to handle complex non-linear relationships, no assumption requirement about the underlying spatial distribution of the data, such as stationarity. Some of the examined studies used the combination of several geospatial parameters, such as lithology, rock types, and geochemical features, as just coordinates may not be enough to obtain reliable geospatial estimates with ML modeling. As long as there are enough unique covariates available that explain the majority of geophysical and geological processes and enough remote sensing data, the uniqueness of each location can be explained in detail so that there is probably no need for including spatial location in the estimation mapping at all (Hengl et al., 2018). However, these types of covariates are not always easily accessible. This thesis firstly aims to demonstrate that how estimates can be produced with ML models in the absence of those types of covariates, using a coordinate rotation strategy. Moreover, most of the proposed studies have not visually validated the produced spatial estimation maps. Visual validation is just as important as proving the accuracy of the methods with regard to practical implementation. In this thesis, thanks to visual examinations, the artifacts produced by ML models in their estimation maps are showed and the mentioned rotation methodology is proposed for this problem.

The studies that compare the estimation performance of ML and geostatistics showed that there is no absolute best method between the two, based on the selected performance criteria. Moreover, although the strong potential and important advantages of ML models in geospatial predictions, they have been often used as a complementary tool to validate results from geostatistical methods rather than as an individual estimation technique. The main reason for this situation is geostatistical estimation methods have been tried, tested, and modified for a long time to produce reliable estimates and they were designed especially for geospatial estimation. Nevertheless, geostatistical methods require assumptions, knowledge, and skills to be effectively applied while their results are not always easy to justify. Thus, this thesis aims to combine the advantages of these methods to build a hybrid model with improved accuracy and obtain reliable estimates. Finally, the literature shows that there is no one-fit-all ML model for all geospatial estimation problems. Thus, in this thesis, an ensemble model called the SL model that combines many different well-defined ML models is used.

3. THEORETICAL BACKGROUND

3.1. Introduction

This chapter briefly outlines the theoretical framework used in this thesis with a short overview of the basic concepts in ML. The concept of hyper-parameter tuning is introduced and the main steps of the applied ML process are summarized. This section gives detailed descriptions of the nine most popular ML models that will be examined for the estimation of geospatial data. The criteria for assessing the performance of the methods in comparison to traditional techniques are presented. The chapter also discusses the general principles of geostatistical estimation and simulation. The conventional estimation methods that will be used to generate the unbiased estimation of geospatial data are reviewed in detail.

3.2. Machine Learning

ML, also known as statistical learning, is about understanding data and extracting knowledge from it. It is a popular research area where statistics, artificial intelligence, and computer science intersect (Müller and Guido, 2016). ML uses statistical theory in constructing mathematical models because the main task is to make inferences from a sample. It uses computer science in training because it is needed efficient algorithms for solving optimization problems. Finally, it is intelligent because it has the ability to learn in a changing environment (Alpaydin, 2014).

Most statistical learning problems can be categorized mainly as supervised or unsupervised learning. Supervised learning generally constructs a statistical model to estimate the *output* based on one or more *inputs* (James et al., 2013). The problems in the space of geographical coordinates (inputs) based on the set of available measurements (outputs) can be given as an example for supervised learning problems. In supervised learning, the output variables vary depending on the nature of the problem being addressed. These variables can be described as either quantitative or categorical. While quantitative variables take numerical values, categorical variables take values in one of k different

classes, categories, or labels. Examples of categorical variables include lithology and rock type, whereas quantitative variables involve grade and thickness. The problems with a quantitative output are generally referred to as regression problems and those including a categorical output are referred to as classification problems (James et al., 2013).

On the other hand, unsupervised learning is interested in problems with no outputs available. They aim at finding structures and dependencies in the input space in order to learn more about the data: clustering, dimensionality reduction, ranking, probability density modeling, outlier detection, etc. The unsupervised problem is generally dealing with estimating the input space's probability density (Kanevski, 2013). This thesis makes use of supervised regression algorithms to estimate geospatial continuous data.

3.2.1. Supervised learning

In supervised learning, the training data comes in pairs of inputs $(\mathbf{x}, y)$, where $\mathbf{x} \in \mathcal{R}^d$ is input variables and y is its output. The training data is shown as:

$$D = \{(x_1, y_1), \dots, (x_n, y_n)\} \subseteq \mathcal{R}^d \times \mathcal{O} \quad (3.1)$$

where, $\mathcal{R}^d$ is the d-dimensional input space, x_i is the input space of the i^{th} data, y_i is the output space of the i^{th} data and $\mathcal{O}$ is the output space. The data (x_i, y_i) are taken from some fixed but unknown distribution $P(X, Y)$, referred to as independent and identically distributed. A function f is trying to be learned such that for a new input data x , $f(x) = y$ is available with high probability, in other words, $f(x) \approx y$.

There are two fundamental steps included in learning a function $f()$. The first step is the selection of the type of ML model that will be appropriate for a particular problem. This step is called *model selection* that defines the function class F , which is the set of functions that can probably be learned. The second one is the *model assessment* step and defines a loss function to be minimized in order to find the best function within this class, $f \in F$. For regression problems, generally, the squared loss function is used:

$$L(f) = \frac{1}{n} \sum_{i=1}^n (f(\mathbf{x}_i) - y_i)^2 \quad (3.2)$$

where n is the total number of training samples, $f(\mathbf{x}_i)$ is the estimated value and y_i is the response value. The loss function measures how well the estimates of the selected function match the samples. It affects the whole learning process which include also generalization and robustness concepts.

3.2.2. Generalization, overfitting, and underfitting

In supervised learning, it is desired to build a model on the training dataset and then be able to make accurate estimates on the test dataset. The test dataset is new data that has the same features like the training set and is unseen during training (Müller and Guido, 2016). How well a model trained in the training data estimates the correct output for new samples is called *generalization* (Alpaydin, 2014). An important aim of learning is to build a model whose generalization ability is as accurate as possible.

Building a model that is too complex could estimate the response variable of the training dataset accurately, but will be unsuccessful to estimate the response variable for new data. Thus, it cannot generalize well to new data. In essence, a too complex model learns the detail and noise in the training data to the extent that negatively impacts the ability of models to generalize. This is called *overfitting* (Hackeling, 2017). When the model is too simple, on the other hand, the features of and variability in the data could not be able to capture, which is called *underfitting*. In the underfitting effect, both training and test errors are similarly large. Figure 3.1 shows the relationship between model complexity and training and test errors.

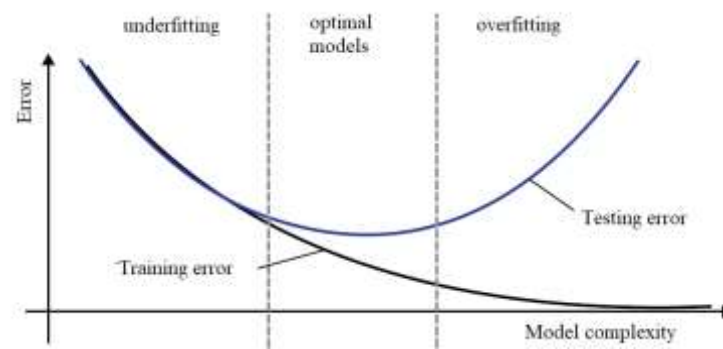
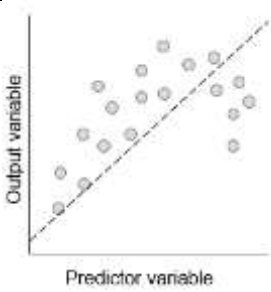
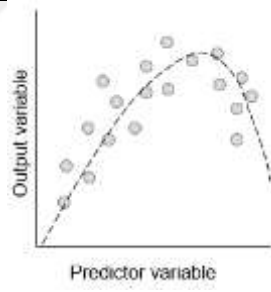
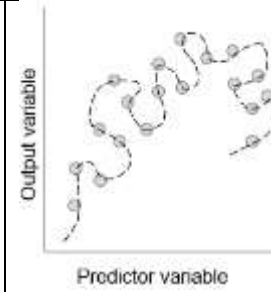


Figure 3.1. The relationship between model complexity and training and test errors (after Kanevski et al., 2009)

It is clear from Figure 3.1 that as the complexity of the model increases, a monotone decrease is observed in the training error and a U-shape in the test error. The optimal model complexity lies in a region between underfitting and overfitting. Table 3.1 illustrates an example of the underfitting, overfitting, and good fitting; and summarizes their symptoms and solutions for the regression problems.

Table 3.1. Illustration of underfitting, good-fitting, and overfitting (after James et al., 2013)

	Underfitting	Good fitting	Overfitting
Model			
Symptoms	- High training error - Training error close to the test error - High bias	Training error slightly lower than test error	- Low training error - Training error much lower than test error - High variance
Solutions	- Complexify model - Add more features - Train longer		- Regularize - Get more data

3.2.3. Bias-variance dilemma

The *bias* term expresses the error that is caused by approximating a problem, which can be extremely complex, by a much simpler model. Linear algorithms such as LR have

generally a high bias; they learn fast and understand easier but are generally less flexible. On the other hand, *variance* refers to the model sensitivity to small fluctuations in training data. A model with higher variance is generally more complex. The complexity of a model is generally linked with its parameter configuration. A model that has a high variance is overfitting the training dataset, whereas a model that has high bias is underfitting the training data.

The balance between bias and variance is expressed as the bias-variance trade-off. One of the most important goals of a supervised ML model is getting an ideal model that has low variance and low bias. However, it can be very hard because the effort to decrease one generally increases the other, which explains the term of trade-off. Figure 3.2 shows an example of bias-variance trade-off as a function of model complexity, where generalization error, which is the error in unseen data, is the sum of bias, variance, and irreducible error. It is also clear from Figure 3.2 that as the complexity of the model increases; the variance increases and the bias decreases.

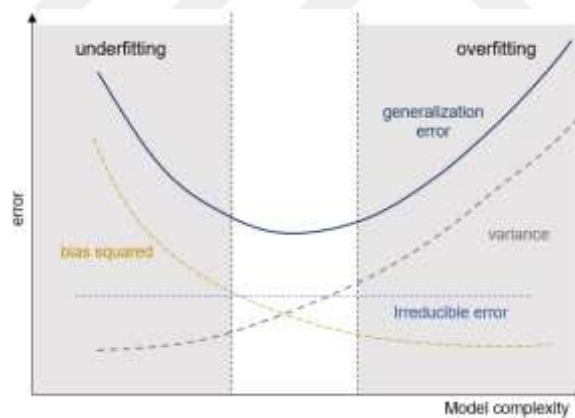


Figure 3.2. Bias-variance trade-off as a function of model complexity

3.2.4. Model assessment

In order to validate a model and measure its generalization ability, it is needed data that is outside the training set. Thus, the dataset is divided randomly into two sets, the first part of which is the *training set* and the second part is the *validation set* (Hackeling, 2017). The candidate ML model is fit on the training dataset and the error of the fitted model is validated on the validation dataset. However, the validation using a single dataset cannot be

adequate to make sensible evaluations (Molinaro et al., 2005). Moreover, the simple train/test split assumes that data is large enough, but data is generally scarce and it is not desired to waste data on validation. Therefore, cross-validation methods, where different subversions of the training dataset are generated, are used to quantify how well the model operates. James et al. (2013) have declared that these methods can be used to estimate the test error associated with a model for its performance evaluation.

k -fold cross-validation is one of the most popular resampling methods to evaluate ML models (Kuhn and Johnson, 2013). It generally results in a less biased and more accurate estimate of the model's performance than a simple train/test split. In the k -fold cross-validation, the whole training set is randomly partitioned into k sets of roughly equal size. The training of the models is performed by $k-1$ folds while the validation is conducted on the remaining **one-fold**. This process is repeated k times with different subsets being treated as the validation set (Russell and Norvig, 2002). In this procedure, each observation is used for validation exactly once. The models' overall accuracy on the training set is acquired by averaging the accuracy carried out in each fold. Typical choices of k are generally 5 or 10. Nevertheless, during the determination of the k value, the bias and computation time should be considered (Picard and Berk, 1990). Figure 3.3 illustrates the process of 5-fold cross-validation.

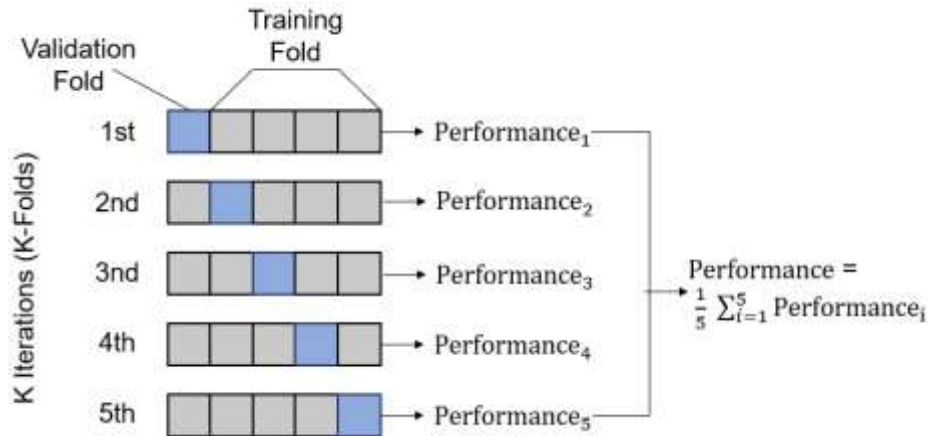


Figure 3.3. A schematic display of 5-fold cross-validation.

k -fold cross-validation can be used for both determining how well a model is expected to perform on a new data (model assessment) and for identifying a model with the lowest test error among several models or model configurations (model selection).

3.2.5. Parameters and hyper-parameters

ML models have at least one *hyper-parameter* that allows tailoring the behavior of the model to a given dataset. The hyper-parameters are different from *parameters* that are internal to the model and determined during training, such as the coefficients in LR or logistic regression. Unlike parameters, hyper-parameters must be set manually for helping the learning. The parameter k that determines the number of neighbors used in the k-nearest neighborhood and the number of neurons in a neural network can be given as examples for the hyper-parameter. The hyper-parameters are related to the complexity of the particular model.

3.3. Hyper-parameter Tuning

The estimation performance of numerous ML models depends heavily on the selected hyper-parameters (Weerts et al., 2020). The selection of hyper-parameters is often more important than the selection of the ML model (Lavesson and Davidsson, 2006). The hyper-parameters must be chosen such that they correspond to the optimal range of model complexities shown in Figure 3.1 that minimizing the test error while having an acceptable training error. It is required to search for the best hyper-parameters that satisfying these conditions on a given dataset (Probst et al., 2019). This is called *hyper-parameter tuning* or *hyper-parameter optimization*. A hyper-parameter tuning strategy comprises defining a search space that is the set of hyper-parameters and their ranges. Then, an optimization algorithm is used to search for the best hyper-parameters within the defined search space. The most basic and widely used approaches for optimizing the hyper-parameters are the grid search and random search algorithms.

3.3.1. Grid search algorithm

The grid search algorithm, which is an exhaustive search, trains and evaluates an ML model for every single combination of the parameters of interest on grids (Hackeling, 2017). In order to find the optimum hyper-parameters of several ML models at the same time by the grid search algorithm, the instruction in Table 3.2 is followed.

Table 3.2. Grid Search algorithm with cross-validation to tune the hyper-parameters (after Kuhn and Johnson, 2019).

```

Define sets of each models' hyper-parameter values to evaluate.
  for each model do
    for each hyper-parameter set do
      for the resampling method do
        Hold out specific samples
        Fit the model on the remainder
        Estimate the hold-out samples
      end
      Calculate the average performance across hold-out estimations
    end
  end
  Choose the optimal parameter set for each model

```

In the instruction given in Table 3.2, the resampling method is often chosen as k -fold cross-validation. In k -fold cross-validation, the candidate ML models are trained on the $k-1$ folds with possible combinations of the parameters, and the performance is measured on the remaining fold. The process is iterated k times with diverse subsets being used as the validating set. Finally, the overall performance of the candidate ML models on the training set is calculated for each possible parameter combination by taking the average performance of the k -folds (Kohavi, 1995). In this instruction, MSE is generally used as the performance evaluation, and the sets of hyper-parameters that yield a lower MSE value are considered to be more accurate. The optimum hyper-parameters which give the minimum MSE are selected (Kuhn and Johnson, 2019). An example of the hyper-parameter tuning flowchart for gradient boosting machine whose details will be given in Section 3.5.7 is illustrated in Figure 3.4. The hyper-parameters of each ML model used in this thesis have been tuned through this flowchart.

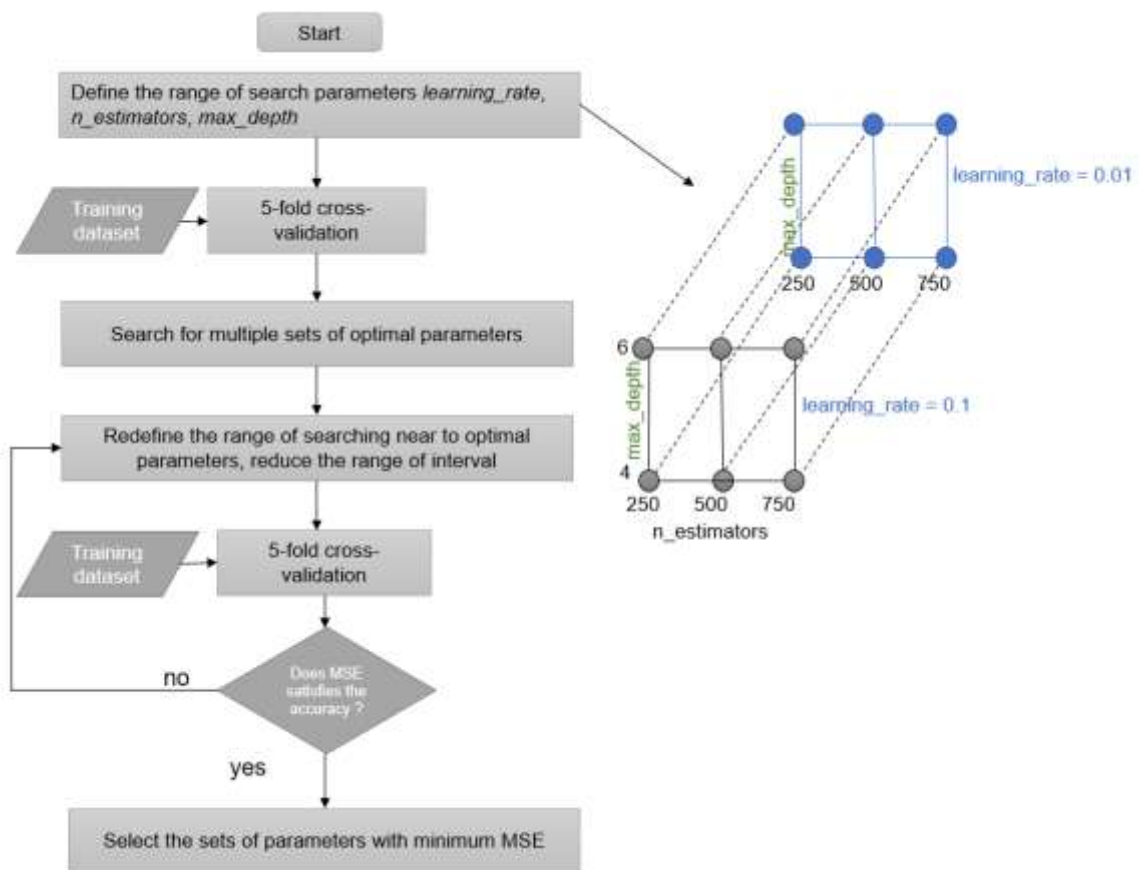


Figure 3.4. An example of a grid search algorithm flowchart

3.3.2. Random search algorithm

In random search, a range of hyper-parameter values is set up and *random* combinations are selected to train and evaluate an ML model (Bergstra and Bengio, 2012). In this approach, the number of combinations that are attempted can be specified explicitly. The grid search and random search algorithms are similar to each other but they differ in the way the hyper-parameters are tested. Figure 3.5 shows the search patterns of these approaches on 2 features for hyper-parameter tuning.

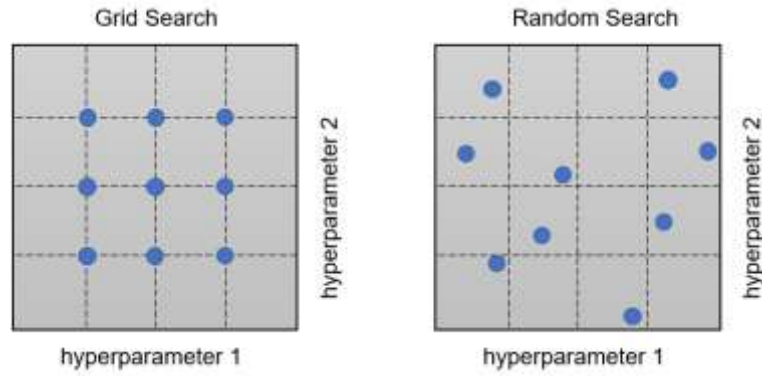


Figure 3.5. Visual representation of grid and random search algorithms on two hyper-parameters

As seen from Figure 3.5, the grid search algorithm sequentially tests the hyper-parameters, while the random search algorithm does not test sequentially all combinations of the hyper-parameters. The random search algorithm can test a wider range of hyper-parameter values within the same computation time as the grid search algorithm, or test the same ones in much less time. However, it does not guarantee to find the best combination due to all combinations are not tested (Liashchynskyi and Liashchynskyi, 2019).

3.3.3. Hyper-parameters of the selected ML models

In this thesis, the grid search algorithm with k -fold validation is applied to find the best hyper-parameters by testing model parameters of the selected ML models. A list of the hyper-parameters of the selected ML models in scikit-learn packages of Python and their descriptions are given in Table 3.3. The hyper-parameters have been selected by considering the ones which have a considerable impact on the performance of the ML model. All other hyper-parameters have been set to their defaults in scikit-learn (Pedregosa et al., 2011).

Table 3.3. Hyper-parameter types and definitions for each ML model (Notations of hyper-parameters are from the respective scikit-learn packages in Python (after Pedregosa et al., 2011))

ML model	Hyper-parameter	Definition
k-Nearest Neighbor	n_neighbors:	Number of neighboring samples to use for imputation (parameter k)
Decision Tree	max_depth:	The maximum depth of the tree.
Bagging	n_estimators:	The number of base estimators (trees) in the ensemble.
Random Forest	n_estimators:	The number of trees in the forest.
Extra Trees	n_estimators:	The number of trees in the forest.
AdaBoost	n_estimators:	The maximum number of estimators at which boosting is terminated.
Gradient Boosting Machine	learning_rate:	shrinks the contribution of each tree by learning rate
	n_estimators:	the number of boosting stages to perform.
	max_depth:	The maximum depth of the tree.
Support Vector Machine	C:	the regularization parameter – the cost parameter
	epsilon:	For epsilon-SVR model. It defines a margin of tolerance where no penalty is given to errors.
Artificial Neural Network	hidden_layer_sizes:	The ith element represents the number of neurons in the ith hidden layer
	neurons:	the number of neurons in a layer
	batch_size:	total number of training examples present in a single batch
	epoch:	one pass through all of the rows in the training dataset.

3.4. Main Steps of the Applied Machine Learning Process

In ML, generally, a 7-step process is followed for supervised learning problems (Yufeng, 2017). These steps are illustrated in Figure 3.6 and summarized in this section briefly.



Figure 3.6. The steps of the applied ML process (after Rijmenam, 2019)

The *data collection* is the first and most important step of the ML process. In this step, input space is constructed in which inputs and outputs variables are selected for an estimation model. This step includes understanding the information available that will be employed to build up a ML model. The data is summarized with statistical methods and visualized with various plots and charts.

The first step is followed by the *data preparation* that transforms given data into a form that is more suitable for modeling. The data preparation is interested in discovering how to best expose the underlying structure of the problem to the ML models. It is highly specific to the data, to the goals of the project, and to the algorithms that will be used to model the data. This step is composed of *data cleaning*, *feature selection*, *data transformation*, *feature engineering*, and *dimensionality reduction*. Data cleaning is interested in the removal or fixing of missing data. The feature selection identifies the input variables that are most relevant to the task. In data transformation, the scale or distribution of variables is changed. Many ML models especially neural networks and support vector machines require the scaled variables to fit the data effectively. When input variables are not scaled for these types of models, the learning process can be slow. Moreover, on the

regression problem, unscaled response variables can conduce to large error gradient values causing the learning process to be unsuccessful. For rescaling the data, normalization or standardization is considered. Generally, if the data distribution is normal, then it is standardized, otherwise, the data is normalized. Normalization rescales data into $[0,1]$ interval by using the following equation:

$$T_{new} = \frac{T - \min(T)}{\max(T) - \min(T)} \quad (3.3)$$

where T is the value of the target variable, $\min(T)$ and $\max(T)$ are the minimum and maximum values of the target variable. On the other hand, standardizing rescales the distribution of data so that the mean is 0 and the standard deviation is 1. The following equation shows the data standardization process:

$$T_{new} = \frac{T - \text{mean}(T)}{\text{standart deviation}(T)} \quad (3.4)$$

Feature engineering derives new variables from available data. Finally, dimensionality reduction creates compact projections of the data. The data preparation step has an important effect on the performance of a ML model (Kuhn and Johnson, 2019).

Step three is *choosing a model* that requires developing a trustworthy *test harness* to improve and spot-check models. The test harness means that the data that will be trained and tested an ML model with, and the performance measure that will be employed to evaluate its performance. The performance measure (i.e. cost function), which is typically MSE for regression problems, is a way to evaluate model estimation skills and should be meaningful to the problem. After the performance measure is selected, a model evaluation procedure needs to be selected. In this procedure, data is split into training and test datasets so as to evaluate the trained model using the training dataset on the test dataset. The most common data split method is the k -fold cross-validation whose details are given in section 3.2.4. When having defined a test harness and a model evaluation procedure, a variety of ML models are spotted checked. Spot-checking of the models means that getting a quick assessment of a group of different ML models on the problem so that knowing what model to focus on and what to discard (Conway and White, 2012).

In step four, the model that performs well in the previous step is *trained* to find useful patterns in given data and it is *evaluated* in step five with the defined performance measure and model evaluation procedure. Step six makes *hyper-parameter tuning* which is the most widely used strategy to improve the results. It is like a search problem, which tries to find the best models using the defined test harness, through hyper-parameter space. The details of the hyper-parameter tuning are given in section 3.3. The final step of the applied ML process is to make estimates on new real-world data.

3.5. Machine Learning Models

The mostly used ML models in the literature, which will be examined in this thesis, are described in detail below. All of these models aim to model the hidden relationship between input and output variables by trying to minimize the estimation errors and complexity of the models. They are adaptive models used to solve both classification and regression problems.

3.5.1. *k*-nearest neighbor

The *k*NN is a nonparametric model that is described as instance-based learning. It stores all available data, and classifies new instance, for classification problems, or estimate the numeric target, for regression problem, based on a measure of similarity like distance functions (Cover and Hart, 1967). Generally, the *k*NN finds the closest data points, nearest neighbors, in the training dataset to estimate a new data point. For regression problems, it is implemented simply by calculating the mean or median of the numeric target of the *k* nearest neighbors (Harrington, 2012). The *k*NN for regression problems is summarized in Table 3.4 (Mitchell, 1996).

Table 3.4. The *k*NN for regression problems

For a training dataset $(\mathbf{x}, \mathbf{f}(\mathbf{x}))$:

Given an instance $\mathbf{x}_p$ to be estimated:

Let the $\mathbf{x}_1 \dots \mathbf{x}_k$ represent instances from the training dataset that are nearest to $\mathbf{x}_p$

Return

$$\hat{f}(\mathbf{x}_p) = \frac{\sum_{i=1}^k f(\mathbf{x}_i)}{k}$$

The basic k NN depends on the definition of the distance function between samples. The most commonly used distance functions in k NN are Euclidean and Minkowski that are defined by:

$$\text{Euclidian Distance Function: } \sqrt{\sum_{i=1}^k (x_i - y_i)^2} \quad (3.5)$$

$$\text{Minkowski Distance Function: } (\sum_{i=1}^k (|x_i - y_i|)^q)^{1/q} \quad (3.6)$$

where x_i and y_i are two individual samples. Euclidean distance calculates the straight-line distance between two samples, while Minkowski distance is a generalization of Euclidean distance, which adds an order parameter $q > 0$. It is common to use the Euclidean distance when the input variables are similar in type and to use the Minkowski distance when they are not. Other popular distance measures include Hamming and Manhattan which are used for specific types of predictors. The optimal number of k is another important issue in the k NN and it can be determined by hyper-parameter tuning. Many different values for k are tried in this approach and checked what works best for the examined problem. Typically, small values of k can cause overfitting and large values of k can lead to underfitting (Larose and Larose, 2014).

k NN is a simple learning model based on the assumption that “*things that look alike must be alike*” (Shalev-Shwartz and Ben-David, 2014). It is much faster than other ML models because it does not require a training period. It stores the entire training data and learns from it only at the time of making estimates. Moreover, its implementation is very easy. However, as the k NN is based on the distance between samples, all predictors need to be rescaled before performing k NN. It is also very sensitive to noisy data, missing values, and outliers. In these cases, it is needed to manually remove outliers and impute missing values. Furthermore, the estimation can be slow when the number of samples or the number of features is very large. It shows bad performance on datasets with many features (>100) and with sparse data.

3.5.2. Decision tree

The decision tree (DT) is widely used both for regression and classification problems (Quinlan, 1986). It consists of several nested *if-then* statements for the predictors which divides up the data. Table 3.5 shows an example of a simple tree in which two-dimensional predictor space is cut into three regions (Figure 3.7). In this example, the outcome is estimated by a single value within each region.

Table 3.5. A simple tree structure example

<i>if predictor $X \geq 109$ then</i>
<i>if predictor $Y \geq 4$ then outcome = 5.5</i>
<i>else outcome = 4.2</i>
<i>else outcome = 3.7</i>

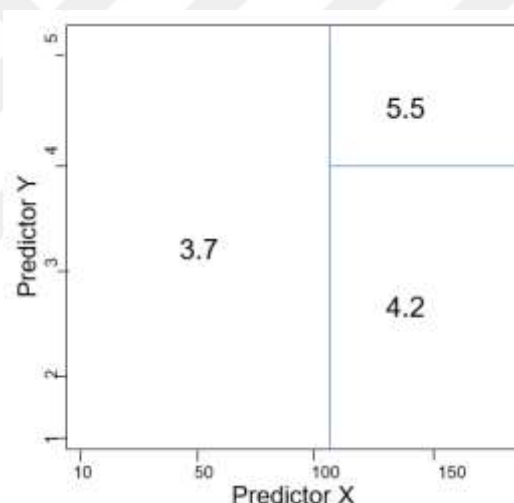


Figure 3.7. An example of the regions defined by a tree-based model

In Table 3.5 and Figure 3.7, there are two *splits* ($X \geq 109$ and $Y \geq 4$) and three *terminal nodes* or *leaves* (outcome) in the terminology of tree models. For regression trees, the predictor space is divided into non-overlapping sub-regions. Within each separate sub-region, the same estimations are made for every observation. The estimation here is the mean or median of the response values of the observations in specified sub-regions. The regions in DT are constructed by recursive partitioning. This method firstly takes a single region (whole dataset), R , and then looks for every separate value of every predictor to discover the predictor and split value that divides up the data into two regions ($R1$ and $R2$) so as to minimize the overall sums of squares error (SSE). SEE is described by:

$$SSE = \sum_{i \in R_1} (y_i - \bar{y}_1)^2 + \sum_{i \in R_2} (y_i - \bar{y}_2)^2 \quad (3.7)$$

where $\bar{y}_1$ and $\bar{y}_2$ are the mean of the response of the training data in regions R_1 and R_2 , respectively (Kuhn and Johnson, 2013). The process is then repeated within each region and searches for the best predictor and the best split value for splitting the data further. The partitioning is iteratively continuing until the stopping criterion is met. The most commonly used stopping criterion is a threshold on the number of training data assigned to each region. When the number is less than the threshold, the split will not be accepted and the node will be taken as a final leaf node. The DT is easy to understand and also to interpret its results. In order to increase the performance of the tree, it can be pruned back to a smaller tree with fewer splits because building a full tree may cause overfitting.

The DT is easy to understand and also to interpret its results. It can also handle many types of predictors without the need for pre-processing. Moreover, it can handle missing data effectively. However, it has some weaknesses such as model instability. DTs may show suboptimal estimation performance as the predictor space can only be divided up into rectangular regions.

3.5.3. Bagging

Bagging, short for bootstrap aggregating, is a simple and powerful ensemble model that combines the estimates from numerous ML models together in order to obtain more accurate estimations than any individual ML model (Breiman, 1996). It is a common procedure that is used to reduce the variance for high-variance ML models, such as DTs (James et al., 2013). Bagging makes use of the bootstrap sample which is a sample randomly chosen from the training set *with replacement* (Efron and Tibshirani, 1986). In the bootstrap procedure, a training dataset of n samples is sampled n times (with replacement) to give another dataset of n samples. The size of the bootstrap sample is the same as the training dataset. Thus, some samples are shown multiple times in the bootstrap sample while others are not picked at all. The probability of a particular sample not being picked all, referred to as *out-of-bag* (OOB) samples, is described by:

$$\left(1 - \frac{1}{n}\right)^n \approx e^{-1} = 0.368 \quad (3.8)$$

where n is the size of the dataset. Bagging of the DTs creates B different trees where each DT is formed from a *bth* bootstrapped sample of the training dataset. Then, the estimations, $\hat{f}^{*b}(x)$, from the trees across all DTs are averaged, $\hat{f}_{bag}(x)$, for regression problems (James et al., 2013):

$$\hat{f}_{bag}(x) = \frac{1}{B} \sum_{b=1}^B \hat{f}^{*b}(x) \quad (3.9)$$

In bagging, for efficiency, the DTs are not pruned and are grown in depth. Thus, these trees have both high variance and low bias. Bagging reduces the variance of DTs through its aggregation process. The most important hyper-parameter for bagging DTs is the number of trees to create. The bagging gives its own internal estimate of estimation performance by using the OOB samples described above. The OOB samples can be used to estimate the test error of a specific model as they were not used during building the model. Thus, the average of the OOB test error of every model in the ensemble yields the test error of the entire ensemble. This value is especially appropriate when performing bagging on big datasets for which cross-validation would be computationally tedious. The advantage of bagging is providing improved accuracy overestimation using a single tree. However, it is much more difficult to interpret a bagged model with respect to a model that is not bagged.

3.5.4. Random forest

A RF is a great improvement over bagging of the DTs. The RF also builds a number of DTs on bootstrapped training samples like bagging. However, the trees in RF are randomized by choosing a random sample of m predictors from the original p predictors as the split candidate (Breiman, 2001). Thus, only a subset of predictors, m , is used at every split of every tree. Generally, a good default value for m is almost equal to the square root of the p , $m \approx \sqrt{p}$ for classification, and $m \approx 1/p$ for the regression problems (James et al., 2013). This value is a hyper-parameter of the RF and can be tuned using cross-validation.

The fundamental difference between bagging and RF is the choice of predictor subset size. While RF selects only a subset of predictors at random out of the total and uses the best split predictors from the subset to split each node in a tree, bagging considers all predictors for splitting a node. The process of RF provides de-correlated trees resulting in more stable and hence more reliable estimations. Bagging, on the other hand, uses highly correlated DTs in its procedure. This results in no considerable reduction in variance over a single tree (Biau and Scornet, 2016; Breiman, 2001). RF has also its internal estimate of estimation performance, without the need to make cross-validation. Each model in the ensemble generates a measure of estimation performance for the OOB samples. The average of the OOB performance metrics is then used to measure the estimation performance of the entire ensemble. This performance correlates well with cross-validation performance assessment (Kuhn and Johnson, 2013). The general workflow of the RF for regression problems can be seen in Figure 3.8.

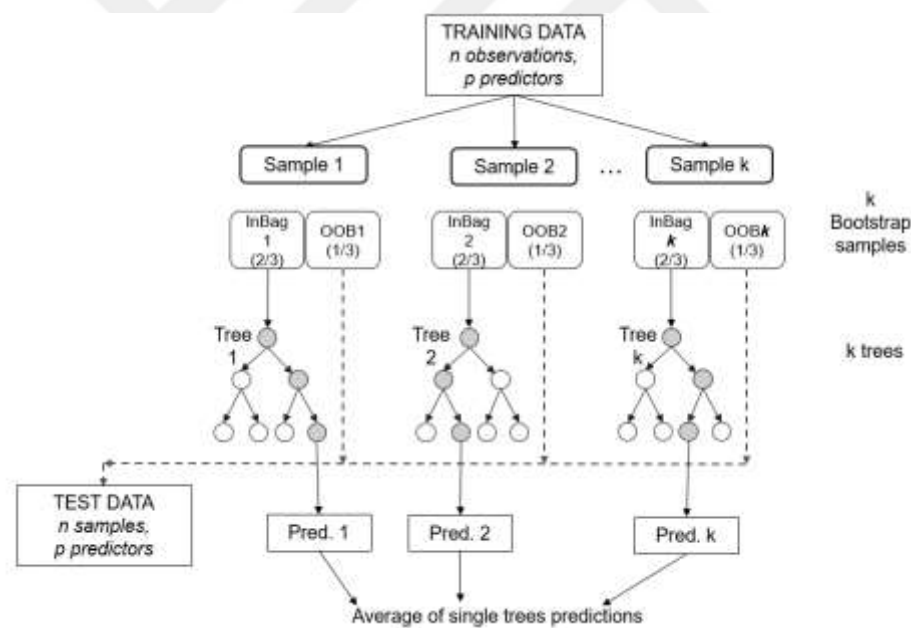


Figure 3.8. The flowchart of RF for regression problems

RF is known as a very powerful model. It does not need to scale the data and often works well without heavy hyper-parameter tuning. Breiman (2001) also demonstrated that RF does not suffer from overfitting. Thus, when a large number of trees are constructed for the forest, RF will not be affected negatively. However, it may not show good performance on sparse data. It also requires more memory and performs slowly on very large datasets.

3.5.5. Extra trees

Extremely Randomized Trees (EXT), or Extra Trees for short, is a tree-based ensemble method for supervised problems (Geurts et al., 2006). The EXT is very similar to the RF. Both RF and EXT are perturb-and-combine methods designed for trees (Breiman, 1998). They are composed of an unpruned large number of DTs, where the final decision is obtained considering the estimation of each tree. Specifically, by averaging in regression problems, and by majority vote in classification problems. As in the RF, the EXT randomly chooses the features at each split point of a DT. However, the EXT fits each DT on the whole training dataset, unlike RF that builds each DT from a sample drawn with replacement (a bootstrap sample) from the training set. Moreover, the EXT selects a cut-point (split point) at random, while RF uses a greedy algorithm to select an optimal split point. Thus, in the EXT, randomness goes one step further in the way splits are computed.

The EXT has mainly three hyper-parameters: (1) the number of DTs in the ensemble, (2) the number of input features to randomly select and consider for each cut-point, and (3) the minimum sample size required in a node to create a new cut-point. The random selection of cut-points leads to the making of multiple de-correlated DTs in the ensemble, although this increases the variance of the model. The increase in variance can be countered by increasing the number of trees used in the ensemble. Geurts, et al. (2006) proved that the EXT is generally competitive with the RF in terms of accuracy and sometimes superior to it.

3.5.6. AdaBoost

Adaptive Boosting, or AdaBoost (ADA) for short, is the most popular *boosting* model (Freund and Schapire, 1997). Boosting is originally designed for classification problems and was later extended to solve the regression problems. The basic idea of them is to combine the outputs of a number of weak learners to develop a strong committee. The boosting works in a similar way with bagging, but in boosting, the trees are grown sequentially. In other words, each tree is grown using information from previously grown trees to reduce the errors from the previous model (James et al., 2013). Moreover, unlike

bagging, boosting does not include bootstrap sampling. Each tree in boosting fits on a modified version of the dataset.

The DTs with one level is the most commonly used model with ADA. These trees are weak learners and are called *decision stumps* that include just one node and two leaves. ADA produces a sequence of weak predictors, in which the model looks for the best predictor at each iteration based on the current sample weights. All samples receive the same weight at the start, which is $1/\text{total}$ number of samples (Schapire, 2013). Then, the sample weights are separately modified at each iteration and the weight of incorrectly estimated samples in the current iteration is increased in the subsequent step. The purpose of this approach is to ensure that samples that are difficult to estimate will get more weight until the ADA recognizes a model that accurately estimates these samples. This process is repeated until a present number of trees are trained or no further improvement can be made on the dataset.

ADA can create a highly accurate predictor by combining many relatively weak and inaccurate predictors. It also needs less tweaking of hyper-parameters, unlike other powerful models such as support vector machines. The most important hyper-parameter of the ADA is the number of trees, which is used during the training phase. The boosting approach can overfit data when the number of trees is too large, unlike bagging or RF (James et al., 2013). Thus, this parameter is generally tuned. Although the ADA has a lot of advantages, it is sensitive to outliers and noisy data.

3.5.7. Gradient boosting machine

The ADA was attached to the statistical concepts of loss functions and additive modeling by Friedman et al. (2000). The authors proposed the boosting as a stage-wise additive model which minimizes exponential loss. This framework is called Gradient Boosting Machines (GBM) that can be used for both regression and classification problems (Friedman et al., 2001). GBM boosts the weak learners (e.g. DTs) iteratively by moving the focus towards observations that were difficult to estimate in previous iterations (Dangeti, 2017).

GBM includes a differentiable loss function such as squared error for regression, a weak learner to make estimations, and an additive model to minimize the loss function. When the DTs are used as the weak learner, the tree depth and the number of iterations are firstly selected. Then, the mean of the response (for regression) is calculated and this response is used as the initial estimated value for each sample. For the current iteration, the residual (the difference between the current estimated value and the observed value) is computed for each sample. After that, a DT is fitted on the residuals and each sample is estimated by using this fitted DT. Finally, the estimated value of each sample is updated by adding the estimated value of the previous iteration to the current estimated value. This procedure is repeated as a fixed number of trees are added or until loss reaches an acceptable level. Table 3.6 shows a detailed description of the GBM.

Table 3.6. The gradient boosting (after Hastie et al., 2009)

-
1. Initialize $f_0(x) = \mathbf{argmin}_{\gamma} \sum_{i=1}^N L(y_i, \gamma)$
 2. For $m = 1$ to M :
 - A. Compute $r_{im} = - \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)} \right]_{f=f_{m-1}(x)}$ for $i = 1, \dots, N$
 - B. Fit a regression tree to the residual values, r_{im} , and create terminal regions, R_{jm} , for $j = 1 \dots J_m$
 - C. For $j = 1 \dots J_m$ compute

$$\gamma_{jm} = \mathbf{argmin}_{\gamma} \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma)$$
 - D. Update $f_m(x) = f_{m-1}(x) + \lambda \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$
 3. Output $\hat{f}(x) = f_M(x)$
-

In Table 3.6, the training dataset $\{(x_i, y_i)\}_{i=1}^n$, where x_i is the input value, y_i is the output value and n is the number of samples, is examined. In the first step of the GBM, a constant model $f_0(x)$ is initialized, where L is the differentiable loss function, γ is the estimated value and $\mathbf{argmin}$ over γ indicates the search for an estimated value that minimizes the presented sum. Then, steps between 2.A and 2.D are repeated M times, where m is the individual tree and M is total number of trees. Step 2.A presents the negative derivative of loss function which equals to r_{im} , where r is short for residual and i is the sample number. The derivative is the *gradient* that GBM is named after. Step 2.B states

building a regression tree on the residuals r_{im} . In this step, the residuals are estimated instead of the weights and terminal regions R_{jm} are created, where j is the index for each leaf in the tree and J_m is the sizes of each of the constituent trees. In step 2. C, the output values γ_{jm} are determined for each leaf in the new tree. The output value is the value for γ that minimizes the presented summation, where $f_{m-1}(x_i)$ is the previous estimation. In step 2.D, a new estimation for each sample is done. The summation in this part says that the output values γ_{jm} should be added up for all the leaves R_{jm} in which sample x can be found, where λ is the learning rate. In the final step, the output is equal to $f_M(x)$.

In the GBM, there are three main parameters to be considered. The first one is the number of trees and it is recommended to continue adding trees until no further improvement is observed. The second one is the depth of the tree and generally shorter trees are preferred. The last one is λ which is a value between 1 and 0. A small learning rate (<0.01) reduces the effect of each tree on the final estimation, and this improves accuracy in the long run (Ridgeway, 2007). However, there is a trade-off between the λ and the computation time. The GBM is widely used in industry and is known as one of the most powerful techniques to build estimation models. It offers competitive estimation performance with RF. However, the GBM is mostly more sensitive to parameter settings and needs greater computation time. It could also be susceptible to overfitting (Friedman, 2001).

3.5.8. Support vector machine

The theory behind the SVM is developed by Vapnik (2013) in the mid-1960 for the classification of both linear and nonlinear data. This theory has been then generalized for regression as an extension of classification. Since the classification theory contains a more natural definition, firstly classification and then regression concepts are explained in this section.

SVM is seeking for finding a *hyperplane* in a p -dimensional space (where p is the number of features) that distinctly classifies the data points. Indeed, there is an infinite number of hyperplanes that could be selected for separating the classes of data points. Figure 3.9 (a) shows an example of the possible hyperplanes with completely separable two classes. The objective of the SVM is to discover a plane that has the largest *margin*, which is the

farthest distance between the training data points of each class. A test observation is then classified as regards which side of the maximal margin hyperplane it lies (Noble, 2006). Figure 3.9 (b) shows an example of the optimal (maximal margin) hyperplane with a solid line on the same dataset shown in Figure 3.9 (a). The optimal hyperplane can also be defined here as the middle-line of the widest piece that can be placed between two classes.

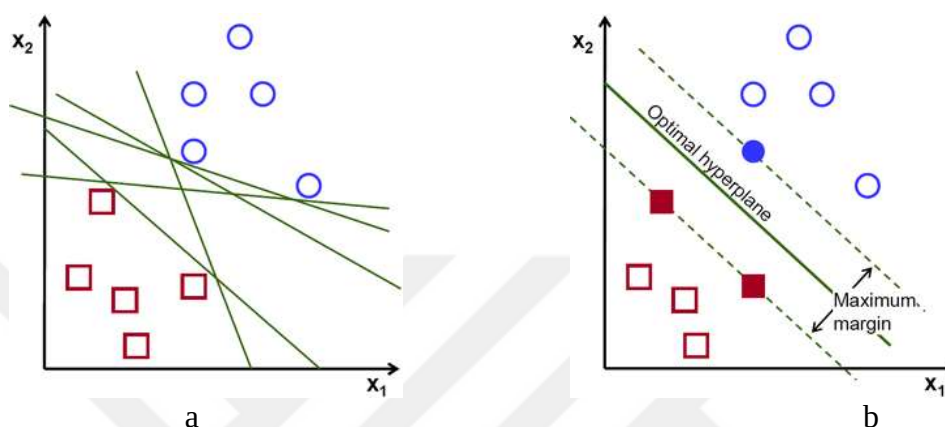


Figure 3.9. (a) Possible hyperplanes and (b) optimal hyperplane for completely separable two classes

It can be seen from Figure 3.9 (b) that three data points highlighted with solid symbols are equidistant from the optimal hyperplane and placed along the dashed lines specifying the margin's width. These three points are described as *support vectors* and optimal hyperplane directly depends only on them (Kuhn and Johnson, 2013). When the classes are not perfectly separable by a hyperplane, the constraint of maximizing the margin must be relaxed. This is called the *soft margin classifier* that allows some observations to be on the wrong side of the margin or even the hyperplane. Therefore, it uses a tuning parameter called *Cost* (C) that defines the amount of violation of the margin allowed (Cortes and Vapnik, 1995). When $C=0$, it defines that there is no budget for violation. When budget C increases, the tolerance of violations to the margin increases, and a wider margin is obtained. The parameter C controls the complexity of the SVM and as its value increases, the SVM becomes more complex (lower variance and higher bias).

In order to accommodate a non-linear boundary between the classes, the SVM is implemented using a *kernel function* that quantifies the similarity between the data points. The kernel function is used to map lower-dimensional data points into higher-dimensional data points. This function allows for more complex models that are not determined simply

by hyperplanes in the input feature space. The general formalization of the SVM that uses the kernel function is described by:

$$f(u) = \beta_0 + \sum_{i=1}^n \alpha_i K(x_i, u) \quad (3.10)$$

where; $K(\cdot, \cdot)$ is the kernel function that computes the inner product between each training sample x_i and new sample u . Moreover, β_0 is intercept and α_i unknown parameters which are estimated from the training data by the learning model. The linear kernel uses the simple inner product $K(x_i, u) = \mathbf{x}_i' \mathbf{u}$ as the kernel function. On the other hand, nonlinear kernels such as polynomial kernel, radial basis kernel (also known as the gaussian kernel), and hyperbolic tangent can be applied to transform the input space into higher dimensions:

$$\text{Polynomial kernel: } f(x) = (\text{scale}(x'u) + 1)^{\text{degree}} \quad (3.11)$$

$$\text{Radial basis: } f(x) = \exp(\phi \|x - u\|^2) \quad (3.12)$$

$$\text{Hyperbolic tangent: } f(x) = \tanh(\text{scale}(x'u) + 1) \quad (3.13)$$

where ϕ is the gamma that controls the width of the kernel. This transformation is called the kernel trick and provides extremely flexible decision boundaries for the SVM modeling produce. The other important tuning parameters except for C are the choice of the kernel and the specific kernel parameters such as gamma in radial basis kernel. The parameter gamma adjusts the curvature of the decision boundary. When its value is low, the curve of the decision boundary is high, which constructs islands of decision boundaries around data points.

When the response variable is quantitative, *Support Vector Regression (SVR)* is applied (Vapnik, 1998). SVR uses the same principles as the SVM with only some minor differences (Witten and Frank, 2002). It makes use of the ε -insensitive error function that is shown in the right panel of Figure 3.10 (Smola and Schölkopf, 2004). A flexible tube, called ε -tube, is formed around the estimated function and points outside this tube are penalized (Drucker et al., 1997). More specifically, SVR firstly defines a convex ε -insensitive loss

function to be minimized and then finds the flattest tube that covers most of the training points (Awad and Khanna, 2015). The support vectors are defined as all points outside the tube and on its border for the SVR. Figure 3.10 shows a one-dimensional example of a linear SVR. The kernel function is used for nonlinear modeling in SVR like in SVM.

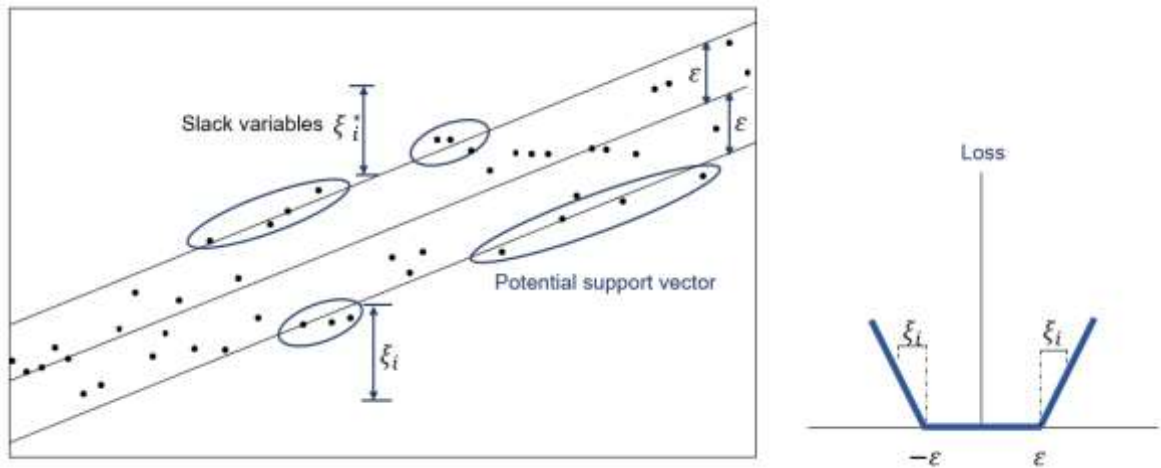


Figure 3.10. One-dimensional linear SVR (Drucker et al., 1997)

SVM generates a good generalization capacity and could work well on a variety of datasets. It allows for complex decision boundaries and performs well on low-dimensional and high-dimensional data. However, it needs careful data preprocessing and hyper-parameter tuning (Müller and Guido, 2016).

3.5.9. Artificial neural network

ANN takes its inspiration from the structure of neural networks in the human brain (Bishop, 1995). The brain consists of a very large number of processing units (neurons) that are connected to each other in a complicated communication network. ANN imitates this idea to learn complex tasks. In ANN, parameters given to the system as input are processed in nodes (neurons). The inputs are multiplied by the randomly selected weight coefficients in the first stage, and their sum is computed with a summation function Σ . Then, a constant value (bias) is added to the neuron and an activation function (usually nonlinear) is applied. The constant value here is a term used to increase or decrease the input of the activation function and again chosen randomly in the first stage. The value obtained from the activation

function is considered as the output node (Cheng and Titterington, 1994). Figure 3.11 presents a simplified artificial neuron model.

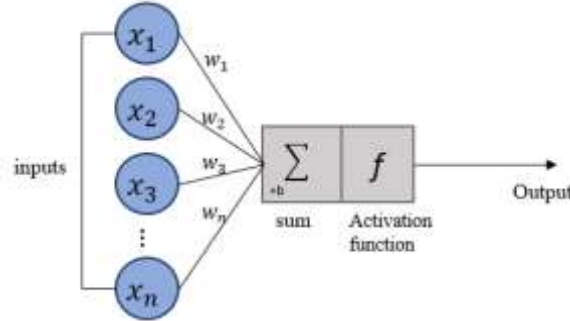


Figure 3.11. A simple artificial neuron model

A neuron as a computational unit is also described by:

$$Z = f\left(\sum_{i=1}^K w_i x^i + b\right) \quad (3.14)$$

where x^i is input features, w_i is weight, b is the bias, $f(\cdot)$ is the activation function and Z is the output. A neuron processes the information through an activation function and passes this information throughout the network. An activation function manages the threshold at which the neuron is activated and the strength of the output signal. Generally, differentiable nonlinear activation functions such as sigmoid and hyperbolic tangent are chosen:

$$\text{Sigmoid: } f(x) = \frac{1}{1+e^{-x}} \quad (3.15)$$

$$\text{Hyperbolic tangent: } f(x) = \frac{1-e^{-x}}{1+e^{-x}} \quad (3.16)$$

An activation function is generally selected based on the nature of the data and the distribution of the target variables (Bishop, 2006). The typical network topology (FNN) consists of an input layer, one or more hidden layers, and an output layer (Haykin, 2009). Figure 3.12 shows an example diagram of ANN consisting of one input layer with 3 inputs, 2 hidden layers with 6 neurons in each, and one output layer with 2 outputs.

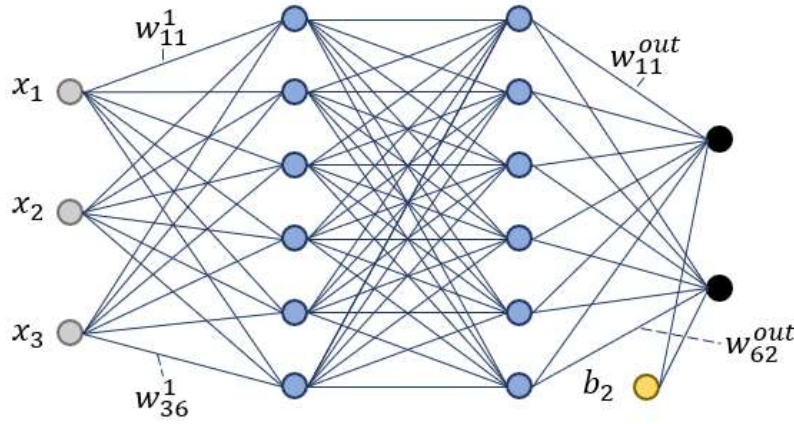


Figure 3.12. A diagram of a neural network with two hidden layers (Kanevski, Pozdnoukhov, Pozdnukhov and Timonin, 2009)

A general simple formula to compute estimations based on learned weights and inputs for Figure 3.12 is described by:

$$f^{out} \left\{ \sum_{h_2=1}^{H_2} w_{h_2,m}^{(out)} f^{h_2} \left[\sum_{h_1=1}^{H_1} w_{h_2,h_1}^{(2)} f^{h_1} \left(\sum_{k=1}^K w_{k,h_1}^{(1)} x_k + b_{h_1} \right) + b_{h_2} \right] + b_{h_2} \right\} \quad (3.17)$$

where $w_{h_a,h_b}^{(H)}$ is the weight of the link from the neuron h_a of the previous layer to the neuron h_b in the layer H , $w_{h_a,m}^{(out)}$ is the weight of the link from the neuron h_a in the last hidden layer and the output m , $f^{h_c}(\cdot)$ is the activation functions for the hidden layers, $f^{out}(\cdot)$ is the activation functions for output layer, m is the index for output, H_1, H_2 are the number of hidden neurons in the first and second layers, K is the number of inputs, and finally b_k, b_{h_1} and b_{h_2} are the biases of the layers (Kanevski et al., 2009).

There are two main architectures for building an ANN which include (FNN) and recurrent neural networks (RNN). In FNN, which is a commonly used network, input information is transmitted to the output in only one direction. No feedback is given to previously active nodes. The most common type of FNN is the multi-layer perceptron. In RNN, on the other hand, the output of neurons is connected back to the inputs via a direct connection or sometimes through a path including other neurons (Rumelhart et al., 1986). These networks have dynamic memories and the output at a time reflects both the current and previous inputs.

A neural network is simply trained by tuning the weights to match some desired criteria of fitting to the data. Each result produced in the model is compared with the true/actual value corresponding to the input used to produce this result. The difference is calculated using an error function. After that, the weight coefficients and bias are updated to reduce the defined error. The *backpropagation algorithms* are commonly used learning algorithms in the learning process (Rumelhart et al., 1986). The backpropagation refers to the correction of each weighting coefficient and bias point by transmitting the error calculated in the output layer back to the hidden layer. The backpropagation is widely used in conjunction with an optimization algorithm such as *gradient descent* to minimize the error function (Zhou, 2012). Gradient descent is an iterative optimization algorithm and computes the gradients of the error function based on the network weights. Then, it sends these gradients to the gradient-based algorithms to update the weights.

The error to be minimized in regression problems is generally chosen as MSE as its partial derivative can be calculated according to individual weights. A simple error notation with a single target t can be described by:

$$E_{MSE}(w) = \frac{1}{2} E[t - Z^{out}(x, w)]^2 \quad (3.18)$$

where x is input, w is weight and Z^{out} is an output estimated for the desired t . One cycle of updating the network for the full training dataset is called an *epoch*. It is a hyper-parameter of ANN and training a neural network can take commonly tens, hundreds, or thousands of epochs. The right number of epochs has an important effect on the performance of the ANN. The one epoch is generally divided into several smaller *batches* as it can be too big to feed to the computer at once. The *batch size* is defined as the total number of training examples existing in a single batch. The other hyper-parameter of the ANN is the learning rate. This parameter controls the step size in the gradient descent algorithm. The main advantage of the ANN is that it can build incredibly complex models and find dependencies between different inputs. However, they often have a long training time. They also require careful data preprocessing and selection of hyper-parameters.

3.5.10. Ensemble models

The ensemble models are commonly applied techniques to improve the stability and accuracy, as well as to decrease the variance of the individual ML models (Breiman, 1996). The ensemble models could be listed as bagging, boosting, averaging, and stacking (also called stacking generalization (Wolpert, 1992)). While stacking and averaging are trained using different types of learning models, bagging and boosting are generally trained using the same types of learning models, such as DTs. Bagging creates an ensemble with sampling from training data with bootstrap, random sampling with replacement, and averaging (regression problems) or voting over class labels (classification problems). In the bagging, any observation has the same probability to seem in a new dataset. For the bagging, the training is done parallel in other words each model is built independently. A detailed explanation of bagging is given in section 3.5.3. On the other hand, boosting builds ensemble by merging weak learners with the expectation that the following models will compensate for errors made by previous models. In boosting, the observations are weighted and thus some of them take part more often in the new datasets. Unlike bagging, boosting builds the new learner in a sequential way by considering the previous models' success. The weights of observations are redistributed after each training step. The weights of the misestimated data are increased to emphasize the most difficult cases. In that way, subsequent learners focus on them during their training. The examples of boosting are given in sections 3.5.6 and 3.5.7. Figure 3.13 shows a comparison of single which is DT, bagging, and boosting.

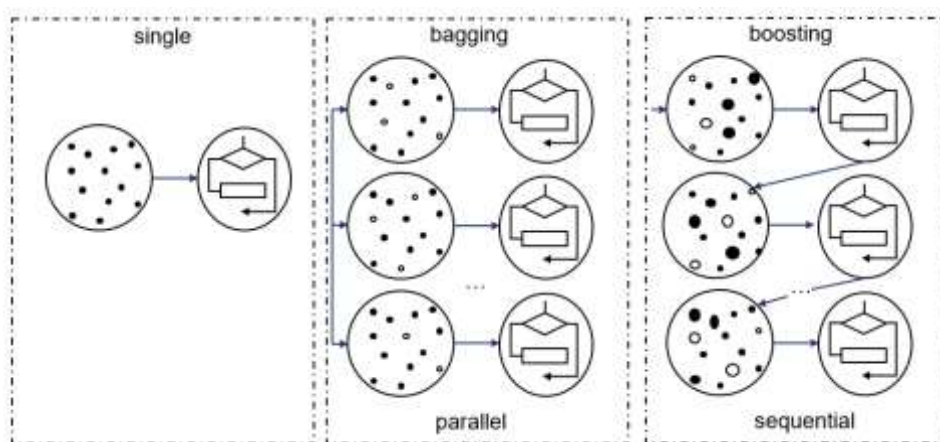


Figure 3.13. The comparison of the bagging and boosting

It is clear from Figure 3.13 that while bagging builds individual models separately, in boosting, each new model is influenced by the performance of those built previously. Stacking, on the other hand, takes the estimation output of the candidate learners which uses the training data and implements another learning model, called meta-learner, on them to estimate the final response values (Sagi and Rokach, 2018). It aims to maximize the generalization accuracy and often performs better than all individual learners. Figure 3.14 shows a general methodology of the stacking.

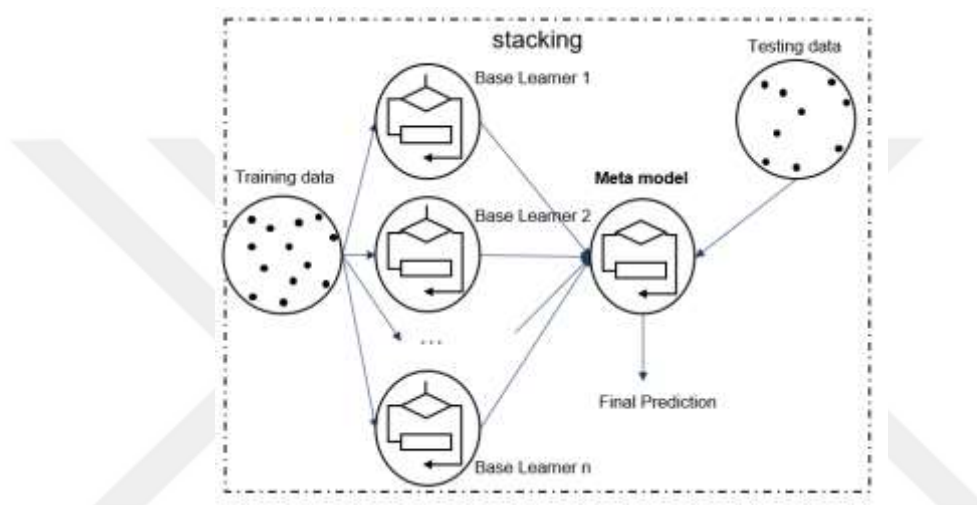


Figure 3.14. The general flowchart of the stacking learner.

Bagging and boosting approaches focus on decreasing variance and bias of the individual ML models, respectively. The stacking approach aims both at finding the optimal way to merge individual ML models and finding the best trade-off between bias and variance. Finally, the averaging approach takes the estimation output of individual learners and calculates the average estimations to obtain final response values. This method often reduces overfit and creates a smoother regression learner. However, the major limitation of the averaging ensemble is that each model contributes equally to the ensemble estimation regardless of how well each learner performs.

3.6. Measures of Evaluation

To validate and measure the performance of an ML model on a given dataset, some performance metrics are used. The most common performance metrics for the regression

problems are listed below (Bose, 2020; Doan and Kalita, 2015; Fernandes, 2020; Loterman et al., 2012):

Root Mean Squared Error (RMSE): It is the square root of the variance of the estimation errors (residuals). RMSE defines the absolute fit of the model to the data. In other words, it measures how close the true data are to the estimated values of a model. RMSE is computed using the following equation:

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (z_i^* - z_i)^2} \quad (3.19)$$

where z_i^* and z_i are estimated and true i th values, respectively, and n denotes the number of data. The RMSE is more easily interpreted than the mean square error (MSE) since it has the same units as the original output value. While MSE is generally used to train the regression models, RMSE is used for performance evaluation. The lower the RMSE value, the better is the performance of the model. Its perfect value is equal to zero.

Mean Absolute Error (MAE): It is calculated as the average of the absolute error values, which is given by the following equation:

$$MAE = \frac{\sum_{i=1}^n |z_i^* - z_i|}{n} \quad (3.20)$$

Like RMSE, MAE considers the units of the error are the same as the units of the true output value. The changes in MAE, unlike the RMSE, are linear and hence intuitive. RMSE penalizes larger errors more severely than MAE. MAE should also be equal to zero.

Coefficient of determination (R-squared): It gives the percentage of data variance explained by the model and, thus, represents the estimation capacity of the model. Unlike RMSE which is an absolute measure of fit, R-squared is a relative measure of fit. *R-squared* is calculated using the following equation:

$$R\text{-squared} = 1 - \frac{RSS}{TSS} = 1 - \frac{\sum_{i=1}^n (z_i - z_i^*)^2}{\sum_{i=1}^n (z_i - \bar{z})^2} \quad (3.21)$$

Where RSS is the sum of squares of residuals, TSS is the total sum of squares and $\bar{z}$ denotes the mean values of the true data. The best possible R-squared value is 1 (or 100%), where 1 indicates that estimations perfectly fit the data. It would get 0 when a constant model that always estimates the expected value of response by ignoring the input features. It is also possible to get a negative R-squared value when a clearly inappropriate model is used or when Equation 3.21 is used although the regressor does not contain an intercept (Chatterjee and Hadi, 2015; Kvålseth, 1985). In other words, R-squared compares the fit of the model with that of a horizontal straight line and when the chosen model fits worse than a horizontal line or does not follow the trend of data, R-squared can be negative. Figure 3.15 illustrates an example of a model that fits the data poorly, where the value of R-squared is negative.

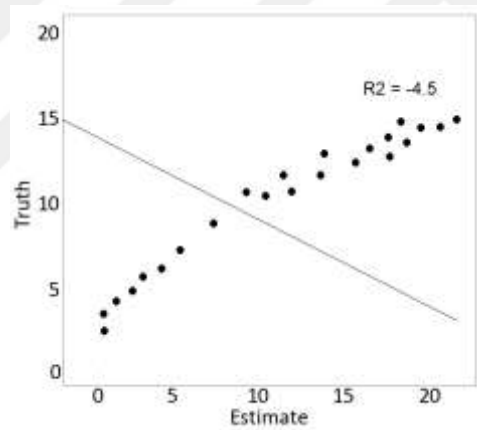


Figure 3.15. An example of a negative R-squared value

In Figure 3.15, the model does not make any sense given the trend of the true data and thus the ratio of RSS to TSS is greater than 1 which leads to a negative R-squared.

Spatial autocorrelation in residuals: Besides the statistical performance metrics, the quality of the proposed models can be quantified by checking the presence of the spatial structures in the residuals. For this purpose, variogram models are assessed on both true data and residuals. It is expected that the variograms do not show a structure (pure nugget effect) on residuals to say that a model is good enough (Leuenberger and Kanevski, 2015). Hengl et al. (2018) have proposed that spatial autocorrelation in residuals is a measure of local

spatial bias in estimation. The remaining spatial correlation in the residuals implies that the proposed model is suboptimal.

3.7. Geostatistics

Geostatistics is a branch of applied statistics that focuses on (i) the geologic structure of the data, (ii) spatial relationship between the data, and (iii) data measured with different volumetric support and precision (Deutsch, 2003). Geostatistics originated from the mining industry, starting with the work by Krige (1951) and was further developed by Matheron (1963) to help improve ore reserve calculation. Then, it has been extended to a variety of fields interested in spatial modeling including geochemistry, geography, soil sciences, forestry, and landscape ecology. Geostatistics offers a collection of deterministic and statistical tools aimed at understanding and modeling spatial variability (Deutsch and Journel, 1998). This section gives a review of the basic concepts in geostatistics.

3.7.1. Random variable and random function

The geostatistical methods are based on the theory of regionalized variables which are characterized by generally two or three orthogonal coordinates (i.e. easting, northing, and elevation) and are specific to a certain region such as a mineral deposit. A regionalized value is considered as the realization of a random variable (Journel and Huijbregts, 1978). A random variable is a variable that may take various outcome values based on a prescribed probability distribution. The random variable is denoted by $Z(\mathbf{u})$ and its outcome values are denoted $z(\mathbf{u})$, where $\mathbf{u}$ is a location within the study domain D . A random variable is defined by its related cumulative distribution function (CDF) (Equation 3.22) (Deutsch and Journel, 1998; Goovaerts, 1997; Wackernagel, 2013).

$$F_Z(\mathbf{u}; z) = \text{Prob}\{Z(\mathbf{u}) \leq z\} \forall \mathbf{u} \in D \quad (3.22)$$

where $F_Z(\mathbf{u}; z)$ represent the CDF of a continuous random variable $Z(\mathbf{u})$ at location $\mathbf{u}$.

A set of RVs $\{Z(\mathbf{u}_i), \mathbf{u}_i \in D, i = 1, \dots, n\}$ at any number of locations is called a random function. The probabilistic framework of random function considers both local and random characteristics of the regionalized variable that accounts for the local irregularities and a general structural characteristic, which reflects large-scale tendencies. The random function is characterized by the multivariate CDF also known as a spatial law:

$$F_{Z(\mathbf{u}_1), \dots, Z(\mathbf{u}_N)}(z(\mathbf{u}_1), \dots, z(\mathbf{u}_N)) = \text{Prob}\{Z(\mathbf{u}_1) \leq z(\mathbf{u}_1), \dots, Z(\mathbf{u}_N) \leq z(\mathbf{u}_N)\} \\ \forall \mathbf{u}_i \in D \quad i = 1, \dots, N \quad (3.23)$$

where N is the number of locations in the domain D and locations $(\mathbf{u}_1), \dots, (\mathbf{u}_N)$ are jointly considered.

3.7.2. Stationarity

Stationarity means that the characteristics of random function $Z(\mathbf{u})$ are invariant under the process of moving from one place to another (Wackernagel, 2013). It is related to a random function rather than the true data (Isaaks and Srivastava, 1989). For geostatistical inference of the random function, stationarity needs only the first two moments (the mean and the covariance) to be constant; this is called second-order stationary. A random function could be second-order stationary, when:

1. The expected value (or mean) of $Z(\mathbf{u})$ does not depend on the point $\mathbf{u}$, i.e, it is constant throughout the domain D :

$$E\{Z(\mathbf{u})\} = m, \quad \forall \mathbf{u} \in D \quad (3.24)$$

2. The covariance function for each pair of RVs $\{Z(\mathbf{u}), Z(\mathbf{u}+\mathbf{h})\}$ depends only on the separation distance ($\mathbf{h}$) between data in the study domain D and not on their true locations:

$$\begin{aligned}
C[Z(\mathbf{u}), Z(\mathbf{u} + \mathbf{h})] &= E[\{Z(\mathbf{u}) - m\}\{Z(\mathbf{u} + \mathbf{h}) - m\}] \\
&= E[\{Z(\mathbf{u})\}\{Z(\mathbf{u} + \mathbf{h})\} - m^2] \\
&= C(\mathbf{h}) \\
&\quad \forall \mathbf{u}, \mathbf{h}, \mathbf{u} + \mathbf{h} \in D
\end{aligned} \tag{3.25}$$

The covariance stationary calls for the stationary of variance. The covariance between RVs at the same locations is the variance of the random function:

$$Var\{Z(\mathbf{u})\} = E[Z(\mathbf{u}) - m]^2 = C(0) \quad \forall \mathbf{u} \in D \tag{3.26}$$

Matheron (1963) suggested the *intrinsic stationarity* which weakens the second-order stationarity. The intrinsic stationarity considers the increments of the function $Z(\mathbf{u} + \mathbf{h}) - Z(\mathbf{u})$ (i.e. variations of grade) rather than the function itself (i.e. grade) (Goovaerts, 1997; Isaaks and Srivastava, 1989; Journel et al., 2013). In this hypothesis, the stationarity of the first two moments of the difference between pairs of regionalized values is assumed. This leads to the notation of the variogram given by:

$$\begin{aligned}
Var\{Z(\mathbf{u}) - Z(\mathbf{u} + \mathbf{h})\} &= 2\gamma(\mathbf{h}) \\
\gamma(\mathbf{h}) &= \frac{1}{2}E\{[Z(\mathbf{u} + \mathbf{h}) - Z(\mathbf{u})]^2\} \\
&\quad \forall \mathbf{u}, \mathbf{h}, \mathbf{u} + \mathbf{h} \in D
\end{aligned} \tag{3.27}$$

Second-order stationary always satisfies the intrinsic hypothesis but the converse is not true.

3.7.3. The variogram function

In geostatistical analysis, a variogram $\gamma(\mathbf{h})$ (this is also called semi-variogram) is a basic tool for quantitative measures of spatial variability (Rendu and Readdy, 1982). It is half the expected squared difference between two random variables $Z(\mathbf{u})$ and $Z(\mathbf{u} + \mathbf{h})$ separated by a distance vector $\mathbf{h}$ (Chiles and Delfiner, 2009; Journel and Huijbregts, 1978). A variogram $\gamma(\mathbf{h})$ does not depend on location just like the mean and the variance, and increases as samples become more dissimilar. If a random function is second-order

stationary and also intrinsic, the variogram $\gamma(\mathbf{h})$ and its covariance $C(\mathbf{h})$ are equivalent functions (David, 1977):

$$\begin{aligned}
 \gamma(\mathbf{h}) &= \frac{1}{2} E\{[Z(\mathbf{u} + \mathbf{h}) - Z(\mathbf{u})]^2\} \\
 &= \frac{1}{2} E[Z(\mathbf{u})]^2 + \frac{1}{2} E[Z(\mathbf{u} + \mathbf{h})]^2 - E[Z(\mathbf{u})Z(\mathbf{u} + \mathbf{h})] \\
 &= E[Z(\mathbf{u})]^2 - E[Z(\mathbf{u})Z(\mathbf{u} + \mathbf{h})] \\
 &= \text{Var}[Z(\mathbf{u})] - \{E[Z(\mathbf{u})Z(\mathbf{u} + \mathbf{h})] - E[Z(\mathbf{u})]E[Z(\mathbf{u} + \mathbf{h})]\} \\
 &= C(0) - C(\mathbf{h}) \\
 &\quad \forall \mathbf{u}, \mathbf{h}, \mathbf{u} + \mathbf{h} \in D
 \end{aligned} \tag{3.28}$$

The basic relation between the variogram $\gamma(\mathbf{h})$ and the corresponding covariance $C(\mathbf{h})$ is also illustrated in Figure 3.16.

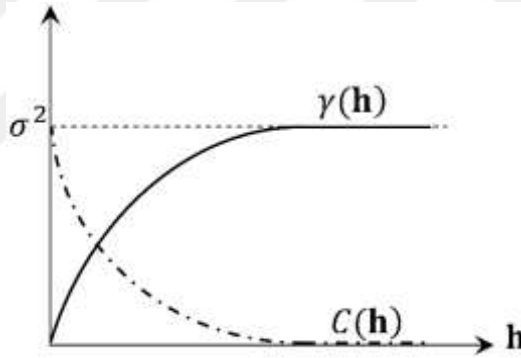


Figure 3.16. Covariance and variogram

3.7.3.1. Experimental variogram

The classical experimental variogram $\hat{\gamma}(\mathbf{h})$ is calculated as the average of squared differences between paired observed values separated by a distance lag of $\mathbf{h}$ (Matheron, 1971):

$$\hat{\gamma}(\mathbf{h}) = \frac{1}{2N(\mathbf{h})} \sum_{i=1}^{N(\mathbf{h})} (z(\mathbf{u}_i) - z(\mathbf{u}_i + \mathbf{h}))^2 \quad \forall \mathbf{u}_i, \mathbf{h}, \mathbf{u}_i + \mathbf{h} \in D \tag{3.29}$$

where $N(\mathbf{h})$ is the number of pairs used in calculating the variogram value at lag distance $\mathbf{h}$. $N(\mathbf{h})$ within each lag is important to compute a robust variogram. In practice, lag tolerances and angles are determined to find the reasonable $N(\mathbf{h})$ (Rossi and Deutsch,

2013). Moreover, reasonable choices in the parameters of the number of lags, lag separation distance, and lag tolerances generally help to obtain a reliable estimate of the variogram. The parameters required for a variogram computation are illustrated in Figure 3.17.

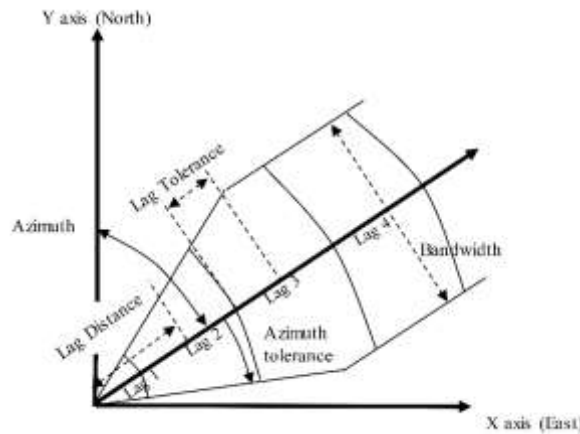


Figure 3.17. The parameters required to calculate an experimental variogram (Deutsch and Journel, 1998).

In practice, lag separation distance should coincide with data spacing. Lag tolerance should be chosen to be half of the lag separation distance. In cases of erratic variograms, lag tolerance can be selected to be more than half of the lag separation to overlap calculations. The variogram is only valid for a distance of one-half of the study domain, thus, the number of lags should be chosen accordingly. Finally, the directions should be selected based on the orientation of the reservoir and/or drill holes (Deutsch, 2003).

The main features of the variogram are the range, sill, and nugget effect. The *range* expresses the distance (if any) at which zero correlation is reached. The *sill* represents the variogram value that corresponds to zero linear correlation. The sill value consists of variation because of measurement error (nugget effect) and the structured variance (Atkinson and Tate, 2000). Finally, the *nugget effect* is the discontinuity of the variogram at the origin because of both measurement errors and short-scale variabilities (Rossi and Deutsch, 2013). The nugget effect should be calculated from the best-informed direction (Deutsch, 2003). An experimental variogram with these three main parameters could be seen in Figure 3.18.

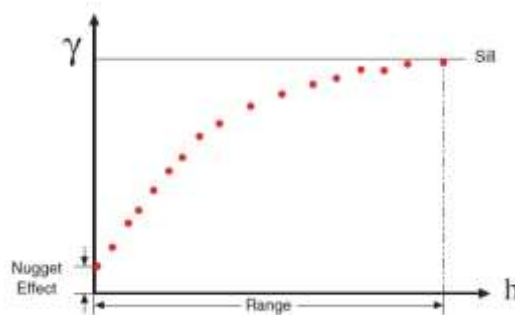
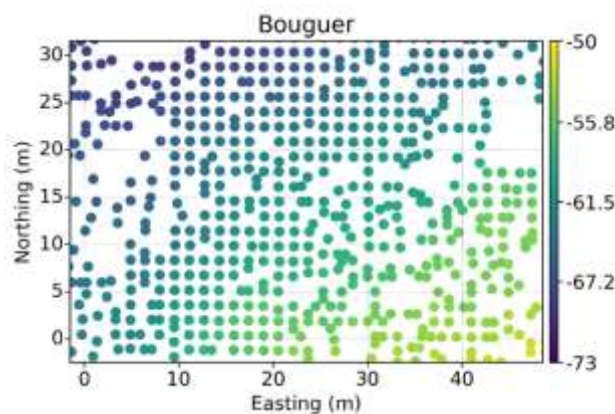


Figure 3.18. Features of a variogram

3.7.3.2. Interpretation of variograms

The most common cases in variogram interpretation consist of *trends*, *cyclicity*, *zonal* and *geometric anisotropy*. *Trends* make the variogram increase above the theoretical sill variance and represent a negative correlation at a large distance. These trends often inquiry the definition of stationarity chosen. The presence of trends is frequently encountered in mineral deposits. For example, systematic decrease in quality of reservoir from proximal to distal portions of the deposition. It may be appropriate to explicitly account for the trend in the later simulation and modeling process (Gringarten and Deutsch, 2001; Rossi and Deutsch, 2013). Bouguer gravity dataset accessed from Olea (2006) can be given as an example for the presence of the trend. In this dataset (Figure 3.19 (a)), there is a systematic increase in the gravity from Northwest to Southeast, which describes the trend. The experimental variogram increasing beyond the sill illustrated in Figure 3.19 (b) demonstrates this trend more clear. On the other hand, Figure 3.19 (c) shows the experimental variogram in the trend-free direction, which is a direction that on average has a constant mean, on approximately N60E.



a

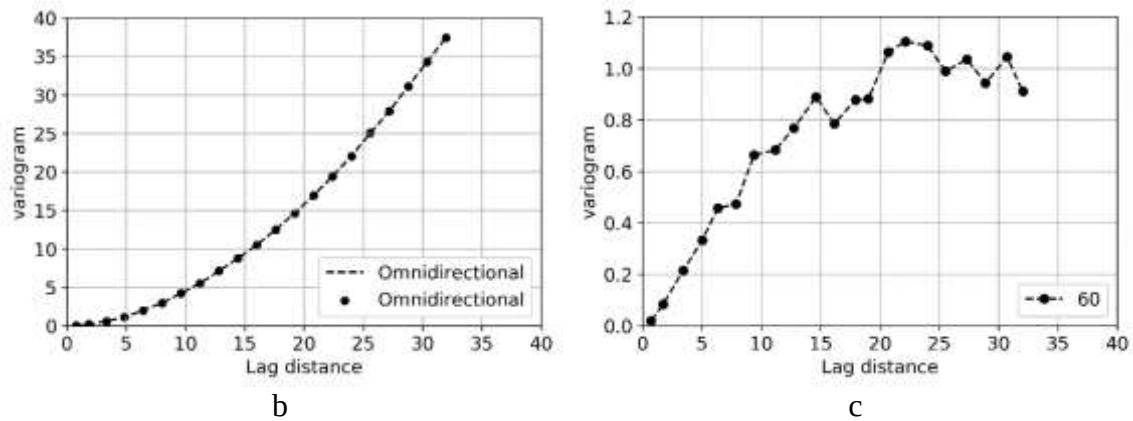


Figure 3.19. (a) Bouguer dataset (b) omnidirectional variogram (c) variogram on N60E

The mineral depositional phenomena frequently occur repetitively over geologic time causing repetitive or *cyclic* variations in mineral grades. This gives a cyclic behavior to the variogram. It means that the variogram will show positive and negative correlations at the length scale of the geologic cycles. The cyclicity could be due to limited data. It is also referred to as a *hole effect* and is usually modeled with sinusoidal functions. For these cases, it should be focused on the nugget effect and a reasonable estimate of the range. Figure 3.20 illustrates an example of the variogram for cyclicity.

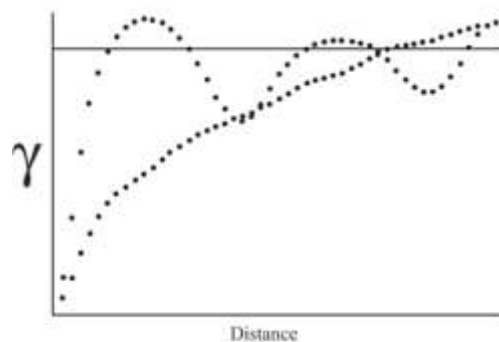


Figure 3.20. A variogram shape for cyclicity (after Rossi and Deutsch, 2013)

The pattern of the continuity can change depending on the direction, and thus the variogram is said to be anisotropic. These types of variograms are frequently encountered in geostatistical applications. The degree and directions of anisotropy can be determined by calculating the experimental variograms in several directions. If the range of the variograms is a function of the direction, then this variable is said to be *geometric anisotropic*. If the ranges remain the same but the sill values change then, the variable is called *zonal*

anisotropic. In zonal anisotropy cases, the vertical variogram reaches a lower sill in presence of areal trends and a higher sill in presence of stratigraphic layering. Figure 3.21 illustrates the examples of variograms for geometric and zonal anisotropy, respectively.

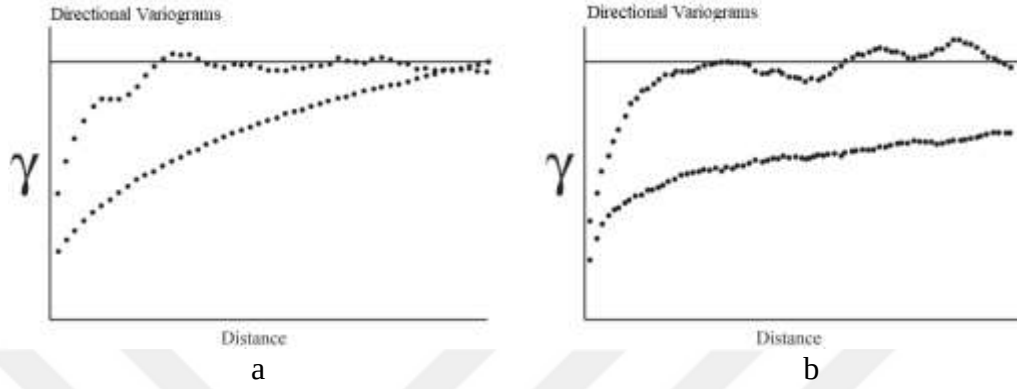


Figure 3.21. The variogram shapes (a) for geometric and (b) zonal anisotropy (after Rossi and Deutsch, 2013)

3.7.3.3. Variogram models

Experimental variograms are calculated only at certain distances but subsequent calculations such as estimation and simulations need a variogram value for all possible distances and directions (Gringarten and Deutsch, 2001). This calls for variogram modeling which is defined as fitting a curve to experimental variogram values. Any function for a variogram model must be positive definite as the variogram models must be a valid measure of distance (Armstrong and Jabin, 1981). The most common variogram models are spherical, exponential, and gaussian models. The *spherical model* has a simple polynomial expression and its shape matches well with what is often observed: an almost linear growth up to a certain distance and then a stabilization. The spherical model is expressed by:

$$\gamma(|\mathbf{h}|) = c \cdot \text{Sph}_a(|\mathbf{h}|) = \begin{cases} c \cdot \left[1.5 \frac{|\mathbf{h}|}{a} - 0.5 \left(\frac{|\mathbf{h}|}{a} \right)^3 \right] & \text{if } |\mathbf{h}| \leq a \\ c & \text{otherwise} \end{cases} \quad (3.30)$$

where $\mathbf{h}$ is the lag distance, a is the range, and c is the sill value or covariance contribution. The *exponential model* is similar to the spherical, except that it rises more steeply and reaches the sill asymptotically. The practical range is accepted as the distance that corresponds to 95% of the sill value in this function. This model is expressed by:

$$\gamma(|\mathbf{h}|) = c. \text{Exp}_a(|\mathbf{h}|) = c. \left[1 - e^{-\frac{3|\mathbf{h}|}{a}} \right] \quad (3.31)$$

The *Gaussian model* exhibits a parabolic rather than linear behavior at the origin, which implies more short-scale continuity. It is suitable for slowly-varying variables since the increase in variance is very gradual with distance, such as thicknesses. This model is expressed by:

$$\gamma(|\mathbf{h}|) = c. \text{Gauss}_a(|\mathbf{h}|) = c. \left[1 - e^{-\frac{3|\mathbf{h}|^2}{a^2}} \right] \quad (3.32)$$

The comparison of the most commonly used variogram models is illustrated in Figure 3.22.

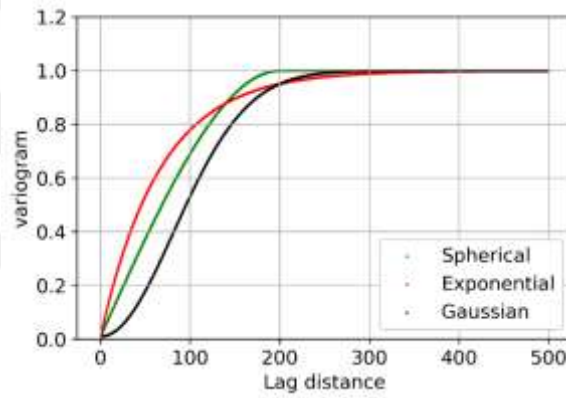


Figure 3.22. Comparison of the variogram models. A standardized sill of 1 is used in the illustration.

In practice, a nested model which is the sum of two or more simple variograms can be defined. The use of nested structures allows the capture of more complex patterns of variability. One of the most common nested models, a spherical model with nugget effect is written as in the following equation:

$$\begin{aligned} \gamma(\mathbf{h}) &= \gamma_1(\mathbf{h}) + \gamma_2(\mathbf{h}) \\ \gamma_1(\mathbf{h}) &= c. \text{Sph}_a(|\mathbf{h}|) = \begin{cases} c. \left[1.5 \frac{|\mathbf{h}|}{a} - 0.5 \left(\frac{|\mathbf{h}|}{a} \right)^3 \right] & \text{if } |\mathbf{h}| \leq a \\ c & \text{otherwise} \end{cases} \\ \gamma_2(\mathbf{h}) &= c_0 \end{aligned} \quad (3.33)$$

An example of a nested model which consists of three simple models (two spherical and one nugget model) is illustrated in Figure 3.23.

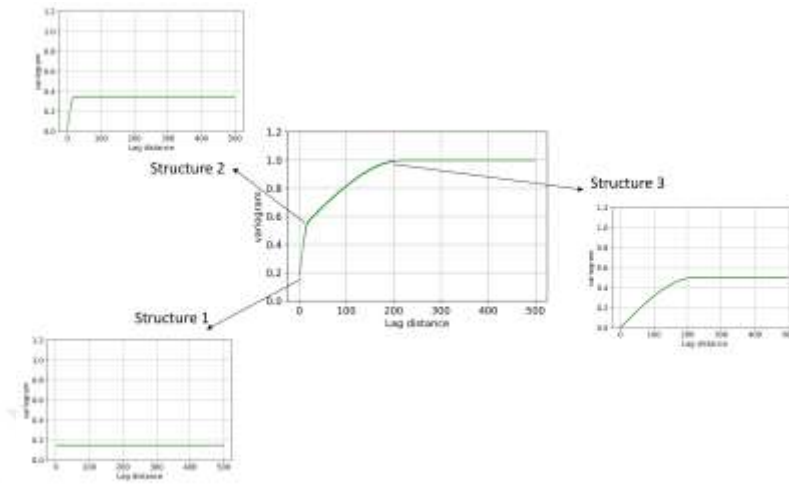


Figure 3.23. An example of the nested model

3.7.4. Geostatistical estimation

Geostatistical methods make use of the variogram or covariance, which captures the spatial variability, to estimate the values at unsampled locations. They are generically called kriging which is known as Best Linear Unbiased Estimator (Isaaks and Srivastava 1989). Here, “best” means minimized estimation variance, which is calculated from the expected squared error between the estimated value and the true (unknown) value (Armstrong, 1998; Olea, 1991). Kriging is considered as an exact method since it returns the original data values at their locations. A decision of second-order stationarity must be made for this method to be applied. The kriging estimate is constructed as a linear combination of the neighboring data (Journel and Huijbregts, 1978). There are numerous flavors of kriging such as SK, OK, kriging with a trend model or universal kriging, KED, factorial kriging. However, only the most common models, SK and OK, are discussed in this thesis.

3.7.4.1. Simple kriging

The SK is a linear estimator and assumes that the mean m of the regionalized variable Z is known and constant (location independent) throughout the domain D (Deutsch and

Journal, 1998). The mean m is explicitly calculated as the average of the observed data and given a priori in the estimator. SK estimate at the unsampled location $\mathbf{u}_0$ is written as:

$$\begin{aligned} z_{SK}^*(\mathbf{u}_0) - m &= \sum_{i=1}^n \lambda_i(\mathbf{u}) \cdot [z(\mathbf{u}_i) - m] \\ z_{SK}^*(\mathbf{u}_0) &= \sum_{i=1}^n \lambda_i(\mathbf{u}) \cdot z(\mathbf{u}_i) + [1 - \sum_{i=1}^n \lambda_i(\mathbf{u})] \cdot m \\ &\quad \forall \mathbf{u}, \mathbf{u}_0, \mathbf{u}_i \in D \end{aligned} \quad (3.34)$$

where $z_{SK}^*(\mathbf{u}_0)$ is the SK estimate at the location of $\mathbf{u}_0$, $[z(\mathbf{u}_i) - m]$ is the residual (also shown as $\{y(\mathbf{u}_i), i = 1, \dots, n\}$) in which $\{z(\mathbf{u}_i), i = 1, \dots, n\}$ describes a set of n nearby relevant samples, $\{\lambda_i(\mathbf{u}), i = 1, \dots, n\}$ represent the kriging weights, and m is the mean value of the dataset. As with all kriging methods, the SK estimator should be unbiased, that is, $E\{Z_{SK}^*(\mathbf{u}_0)\} = E\{Z(\mathbf{u}_0)\}$ given the MSE criterion. Since the mean of error is equal to zero, it is already unbiased by definition. The proof of unbiasedness is given in Equation 3.35, which also demonstrates that the SK does not constrain the weights.

$$\begin{aligned} z^*(\mathbf{u}_0) &= \sum_{i=1}^n \lambda_i \cdot z(\mathbf{u}_i) + \lambda_0 = \lambda_1 z(\mathbf{u}_1) + \lambda_2 z(\mathbf{u}_2) + \dots + \lambda_n z(\mathbf{u}_n) + \lambda_0 \quad (3.35) \\ E\{Z^*(\mathbf{u}_0) - Z(\mathbf{u}_0)\} &= 0 \\ E\{Z^*(\mathbf{u}_0) - Z(\mathbf{u}_0)\} &= \lambda_1 z(\mathbf{u}_1) + \lambda_2 z(\mathbf{u}_2) + \dots + \lambda_n z(\mathbf{u}_n) + \lambda_0 - Z(\mathbf{u}_0) \\ &= \lambda_1 E\{z(\mathbf{u}_1)\} + \lambda_2 E\{z(\mathbf{u}_2)\} + \dots + \lambda_n E\{z(\mathbf{u}_n)\} + \lambda_0 - E\{Z(\mathbf{u}_0)\} \\ &= \lambda_1 m + \lambda_2 m + \dots + \lambda_n m + \lambda_0 - m \\ &= m(\lambda_1 + \lambda_2 + \dots + \lambda_n - 1) + \lambda_0 = 0 \\ \lambda_0 &= m(1 - \lambda_1 - \lambda_2 - \dots - \lambda_n) \\ \lambda_0 &= m \left(1 - \sum_{i=1}^n \lambda_i \right) \end{aligned}$$

The SK weights are determined based on minimizing the expected squared difference between the estimated value and the true value at an unsampled location of $\mathbf{u}_0$, that is, $\sigma_E^2(\mathbf{u}_0) = E\{(Y_{SK}^*(\mathbf{u}_0) - Y(\mathbf{u}_0))^2\}$. The error variance, σ_E^2 , is expressed as (Hohn, 2000; Rossi and Deutsch, 2013):

$$\sigma_E^2(\mathbf{u}_0) = E \left\{ \left(Y^*(\mathbf{u}_0) \right)^2 \right\} - 2 \cdot E \{ Y^*(\mathbf{u}_0) \cdot Y(\mathbf{u}_0) \} + \left\{ \left(Y(\mathbf{u}_0) \right)^2 \right\} \quad (3.36)$$

and can be described as a linear combination of covariance values of the residuals:

$$\begin{aligned} & \sum_{i=1}^n \sum_{j=1}^n \lambda_i \lambda_j E \{ Y(\mathbf{u}_i) \cdot Y(\mathbf{u}_j) \} - 2 \cdot \sum_{i=1}^n \lambda_i E \{ Y(\mathbf{u}) \cdot Y(\mathbf{u}_i) \} + C(0) \\ &= \sum_{i=1}^n \sum_{j=1}^n \lambda_i \lambda_j C(\mathbf{u}_i, \mathbf{u}_j) - 2 \cdot \sum_{i=1}^n \lambda_i C(\mathbf{u}_i, \mathbf{u}_0) + C(0) \end{aligned} \quad (3.37)$$

where $\{C(\mathbf{u}_i, \mathbf{u}_j), i = 1, \dots, n\}$ are the covariances among the samples and $\{C(\mathbf{u}_i, \mathbf{u}_0), i = 1, \dots, n\}$ are the covariances between the samples and the location being estimated. The optimal weights are those that minimize the error variance. These weights are determined by setting the partial derivatives of the error variance with respect to the weights and setting them to zero:

$$\frac{\partial \left[\sum_{i=1}^n \sum_{j=1}^n \lambda_i \lambda_j C(\mathbf{u}_i, \mathbf{u}_j) - 2 \cdot \sum_{i=1}^n \lambda_i C(\mathbf{u}_i, \mathbf{u}_0) + C(0) \right]}{\partial \lambda_i} = 0 \quad i = 1, \dots, n \quad (3.38)$$

The result of Equation 3.38 gives the following system of equations known as normal equations or SK system (Journel and Huijbregts, 1978; Luenberger, 1997):

$$\sum_{j=1}^n \lambda_j C(\mathbf{u}_i, \mathbf{u}_j) = C(\mathbf{u}_i, \mathbf{u}_0) \quad \forall \mathbf{u}_0, \mathbf{u}_i, \mathbf{u}_j \in D \quad (3.39)$$

The SK system can also be defined in a matrix form:

$$\begin{bmatrix} C(\mathbf{u}_1, \mathbf{u}_1) & \cdots & C(\mathbf{u}_n, \mathbf{u}_1) \\ \vdots & \ddots & \vdots \\ C(\mathbf{u}_1, \mathbf{u}_n) & \cdots & C(\mathbf{u}_n, \mathbf{u}_n) \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_n \end{bmatrix} = \begin{bmatrix} C(\mathbf{u}_1, \mathbf{u}_0) \\ \vdots \\ C(\mathbf{u}_n, \mathbf{u}_0) \end{bmatrix} \quad \forall \mathbf{u}_0, \mathbf{u}_i, \mathbf{u}_j \in D \quad (3.40)$$

The SK variance $\sigma_{SK}^2(\mathbf{u})$ that minimizes the error variance can be calculated by:

$$\sigma_{SK}^2(\mathbf{u}_0) = C(0) - \sum_{i=1}^n \lambda_i(\mathbf{u}) C(\mathbf{u}_i, \mathbf{u}_0) \quad \forall \mathbf{u}, \mathbf{u}_0, \mathbf{u}_i \in D \quad (3.41)$$

where $C(0)$ is the stationary variance of the data. The kriging variance is independent of the true sample values; it depends only on the covariance values.

3.7.4.2. Ordinary kriging

The OK estimator assumes that the mean m is unknown and implicitly estimated as a constant value within each search neighborhood (Deutsch and Journel, 1998). Unlike the SK estimator, a constraint that the weights must sum to one is required to ensure unbiasedness of the estimate (Goovaerts, 1997; Wackernagel, 2013). The OK estimator explains the locally varying mean by limiting the study area of stationarity to within the neighborhood centered on the location being estimated (Goovaerts, 1997). This estimator is robust if enough data are available (Journel and Rossi, 1989). The OK estimation at an unsampled location $\mathbf{u}_0$ is written as:

$$\begin{aligned} z_{OK}^*(\mathbf{u}_0) - m &= \sum_{i=1}^n \lambda_i(\mathbf{u}) \cdot [z(\mathbf{u}_i) - m] \\ z_{OK}^*(\mathbf{u}_0) &= \sum_{i=1}^n \lambda_i(\mathbf{u}) z(\mathbf{u}_i) + \left[1 - \sum_{i=1}^n \lambda_i(\mathbf{u}) \right] \cdot m \end{aligned} \quad (3.42)$$

$\forall \mathbf{u}, \mathbf{u}_0, \mathbf{u}_i \in D$

OK estimator constrains the sum of weights thus the mean m in Equation 3.42 is not used in the estimate; that is,

$$1 - \sum_{i=1}^n \lambda_i(\mathbf{u}) = 0, \forall \mathbf{u} \in D \quad (3.43)$$

To obtain optimal weights and derive the OK system of equations which is also a system of normal equations, the Lagrange method of multiplier is used. This method calls

for the Lagrange parameter μ_{OK} that accounts for the constraint on the weights. The system of OK in terms of covariances is given by:

$$\begin{cases} \sum_{j=1}^n \lambda_j(\mathbf{u}) \cdot C(\mathbf{u}_i, \mathbf{u}_j) + \mu_{OK}(\mathbf{u}) = C(\mathbf{u}_i, \mathbf{u}_0) & i = 1, \dots, n \\ \sum_{j=1}^n \lambda_j = 1 \end{cases} \quad (3.44)$$

$$\forall \mathbf{u}, \mathbf{u}_i, \mathbf{u}_j \in D$$

In matrix form:

$$\begin{bmatrix} C(\mathbf{u}_1, \mathbf{u}_1) & \dots & C(\mathbf{u}_n, \mathbf{u}_1) & 1 \\ \vdots & \ddots & \vdots & 1 \\ C(\mathbf{u}_1, \mathbf{u}_n) & \dots & C(\mathbf{u}_n, \mathbf{u}_n) & 1 \\ 1 & \dots & 1 & 0 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_n \\ \mu_{OK} \end{bmatrix} = \begin{bmatrix} C(\mathbf{u}_1, \mathbf{u}_0) \\ \vdots \\ C(\mathbf{u}_n, \mathbf{u}_0) \\ 1 \end{bmatrix} \quad \forall \mathbf{u}_0, \mathbf{u}_i, \mathbf{u}_j \in D \quad (3.45)$$

The OK estimation variance σ_{OK}^2 is given by:

$$\sigma_{OK}^2 = C(0) - \sum_{i=1}^n \lambda_i(\mathbf{u}) \cdot C(\mathbf{u}_i, \mathbf{u}_0) - \mu_{OK}(\mathbf{u}) \quad \forall \mathbf{u}, \mathbf{u}_i \in D \quad (3.46)$$

The OK estimation variance is minimized with the constrain on the weights, which requires a spatial dependence structure in each local search neighborhood.

3.7.5. Geostatistical simulation

Geostatistical conditional simulation is used to produce estimation at each location on a detailed scale. It is similar to kriging in terms of honoring the data. However, the aim of conditional simulation differs from the kriging. Simulation provides a family of equally likely realizations in order to quantify spatial variability and uncertainty (Rossi and Deutsch, 2013). The generated realizations reproduce the data and the statistical characteristics (e.g., histogram and variogram). Kriging, on the other hand, does not reproduce the histogram and variogram. It provides only one estimation and smoothes out the local details of the spatial variability. Kriging also does not permit the calculation of uncertainty. On the other hand,

the average of multiple simulated realizations is very close to kriging. Figure 3.24 shows an example for the comparison of kriging and simulation of spatially distributed data.

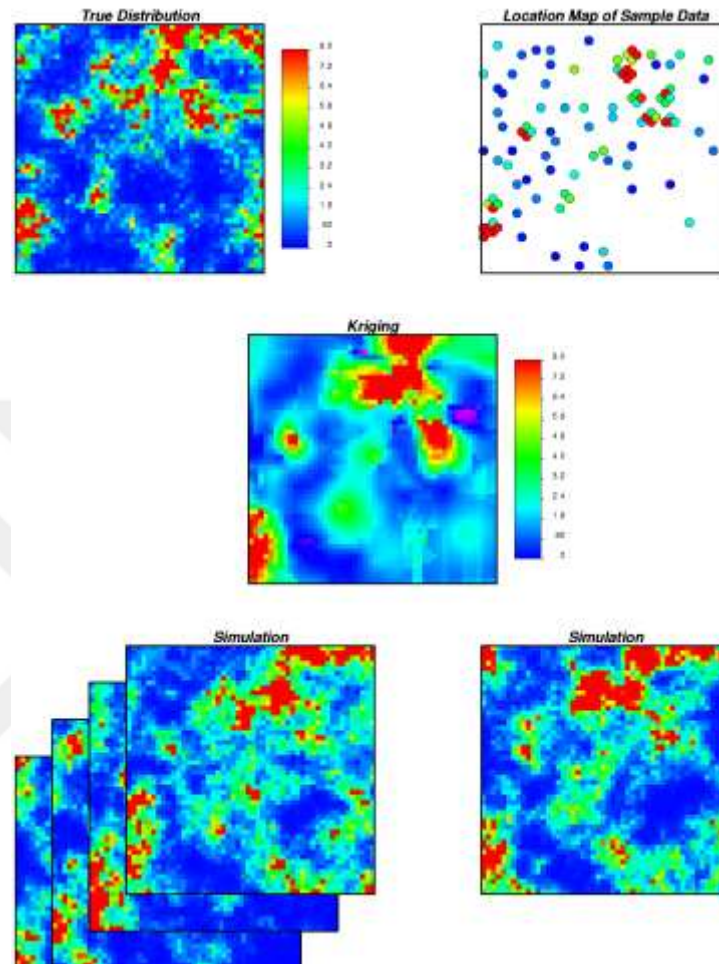


Figure 3.24. The comparison of kriging and geostatistical simulation (after Rossi and Deutsch, 2013).

As it is clear from Figure 3.24 that, for given data, kriging produces a unique estimation, while simulation produces four realizations, where each of them is different. There are many different simulation methods. The most widely used method to create geostatistical simulations is sequential Gaussian simulation (SGS) (Isaaks, 1992). SGS uses the spatial version of Monte Carlo simulation for drawing random values from conditional distributions at every location. Let a continuous variable z with a univariate CDF $\{F_Z(z(\mathbf{u}_i)), i = 1, \dots, n\}$ that demonstrates the entire domain D . The normal score transformation is implemented in order to convert the original variable Z that follows a Gaussian distribution $\{F_Z(z(\mathbf{u}_i)), i = 1, \dots, n\}$ with a mean of m and a standard deviation of

σ to another variable Y that follows a standard normal distribution $\{G_Y(y(\mathbf{u}_i)), i = 1, \dots, n\}$ with a mean of 0 and a standard deviation of 1. The normal distribution is a parametric distribution:

$$y(\mathbf{u}_i) = G_Y^{-1}F_Z(z(\mathbf{u}_i)) \quad \forall \mathbf{u}_i \in D \text{ and } i = 1, \dots, n \quad (3.47)$$

where $\{y(\mathbf{u}_i), i = 1, \dots, n\}$ is the transformed Gaussian variable at n data locations. $G_Y^{-1}(\cdot)$ demonstrates the inverse Gaussian CDF of the random variable Y and $F_Z(\cdot)$ shows the CDF of the original random variable Z . After the normal score transformation, the Gaussian variogram models for the transformed variable are obtained. Then, a random simulation path to be followed for visiting each grid node to be simulated is described. The distribution of possible values is calculated at each grid node and a random sample is drawn from this distribution (Chiles and Definer, 2009). SGS uses SK to build the distribution at a particular location; the estimated SK value is the mean of the conditional distribution and its variance the SK variance. After a location is simulated, SGS moves to another new location, repeats the procedure, and incorporates all previously simulated values as data points (Srivastava, 2013). It is desired to ensure variogram reproduction (Rossi and Deutsch, 2013). Finally, Gaussian realizations are back-transformed to the original distribution:

$$z(\mathbf{u}_i) = F_Z^{-1}G_Y(y(\mathbf{u}_i)) \quad \forall \mathbf{u}_i \in D \text{ and } i = 1, \dots, N \quad (3.48)$$

where all simulation points N are visited. In SGS, multiple realizations can be created by changing the random path and drawing different values from the conditional distributions.

3.7.6. Declustering

In mineral resource estimation, data are rarely collected randomly. Generally, more samples are collected in areas of great interest, for example, high-grade areas (Rossi and Deutsch, 2013). In this situation, the equal-weighted statistics would not be representative of the whole population. Thus, the declustering techniques assign each datum a weight based on its closeness to surrounding data (Deutsch, 2020). The summary statistics and distribution are determined with the weights instead of a constant $1/\text{number of data}$. There are various types of declustering methods, such as *polygonal*, *nearest-neighbor*, and *cell declustering*.

The *polygonal declustering* method assigns each weight proportional to the area of interest of each sample (Isaaks and Srivastava, 1989). It works well only in relatively simple 2D cases with very well-understood geological limits. On the other hand, the *nearest-neighbor declustering* technique looks like the polygonal declustering method but it is applied to a regular grid of blocks or grid nodes. This technique assigns the closest datum of the set being declustered to each block. It is more practical in resource estimation (Rossi and Deutsch, 2013). Finally, the *cell declustering* technique (Deutsch, 1989; Journel, 1983), which has been applied in this thesis, is widely used and robust (Pyrz and Deutsch, 2003).

In cell declustering, firstly volume of interest is divided into a grid of equal volume cells $l = 1, \dots, L$ for each origin of offset. The cell size is nearly the spacing of the data in sparsely sampled regions. Then, the occupied cells L_0 and the number of data in each occupied cell $n_{occ} = 1, \dots, L_0$ are counted. According to the number of data falling in the same cell l , each data is weighted. For instance, the cell declustering weight for data i in cell l is calculated by (Rossi and Deutsch, 2013):

$$w_i = \frac{1}{n_{occ} \cdot L_0} \quad (3.49)$$

In cell declustering, all occupied cells receive the same weight. The weights are greater than 0 and sum to 1. At the last step, the grid is shifted due to the results change according to the origin of the cell network, and resulting weights are averaged over each origin offset. Figure 3.25 shows a simple illustration of the method for one origin offset.

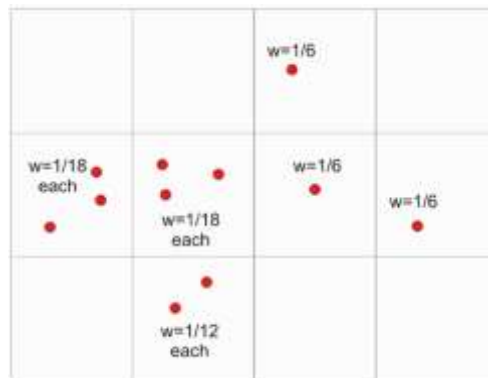


Figure 3.25. Illustration of the cell declustering method (Deutsch, 2020)

In Figure 3.25, the domain is divided into a grid of $L = 12$ cells, with $L_0 = 6$ occupied cells. Thus, the cell weight is $1/6$ and multiple data in a cell share the weight equally. The weights are sensitive to the cell size in this method. When the cell size is set as very small, every sample occupies its own cell and receives an equal weight. When the cell size is set as very large, all samples fall into the same cell and the result is again nearly equal weighting. The common procedure for assigning the cell size that identifies the best weights is plotting the declustered mean of the data versus a range of cell sizes. Then, the size with minimum (data clustered in the high-valued area) or maximum (data clustered in low-valued areas) is chosen. Figure 3.26 shows an example where a clear minimum corresponds to the size of data clusters.

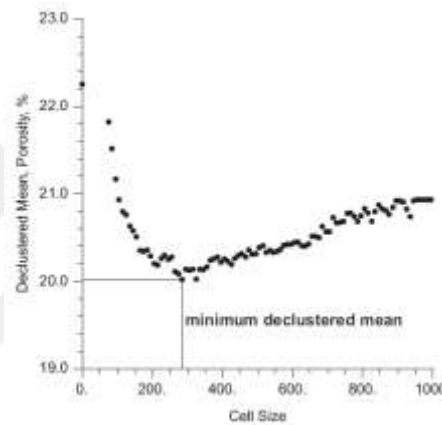


Figure 3.26. The cell declustering mean versus different cell sizes (after Rossi and Deutsch, 2013)

4. MATERIALS AND METHODS

4.1. Introduction

This chapter develops two different methodologies: (1) for managing estimation artifacts in ML models as well as incorporating anisotropy information into the estimation process, and (2) for combining kriging and ML to obtain more accurate, consistent, and unbiased spatial estimates. In particular, the description of the SL model which will be employed for ML modeling in the current research is discussed. This model is based on the stacking of many individual ML models. The additional topics include the main points of the general application of the SL model and kriging for the estimation of the geospatial data. A new application of the partitioning drill hole data into k -folds which is necessary for spatial data is discussed. The details of the sequential quadratic programming which is used in the optimization of the weights of the combined model are covered.

All optimization studies are conducted by using SciPy.Optimize in Python (Virtanen et al., 2020). To produce models and estimates using kriging, pygeostat (2020) which is a Python package in geostatistical modeling is used. Moreover, to fit models and produce estimation using the SL model, the scikit-learn package is employed (Pedregosa et al., 2011).

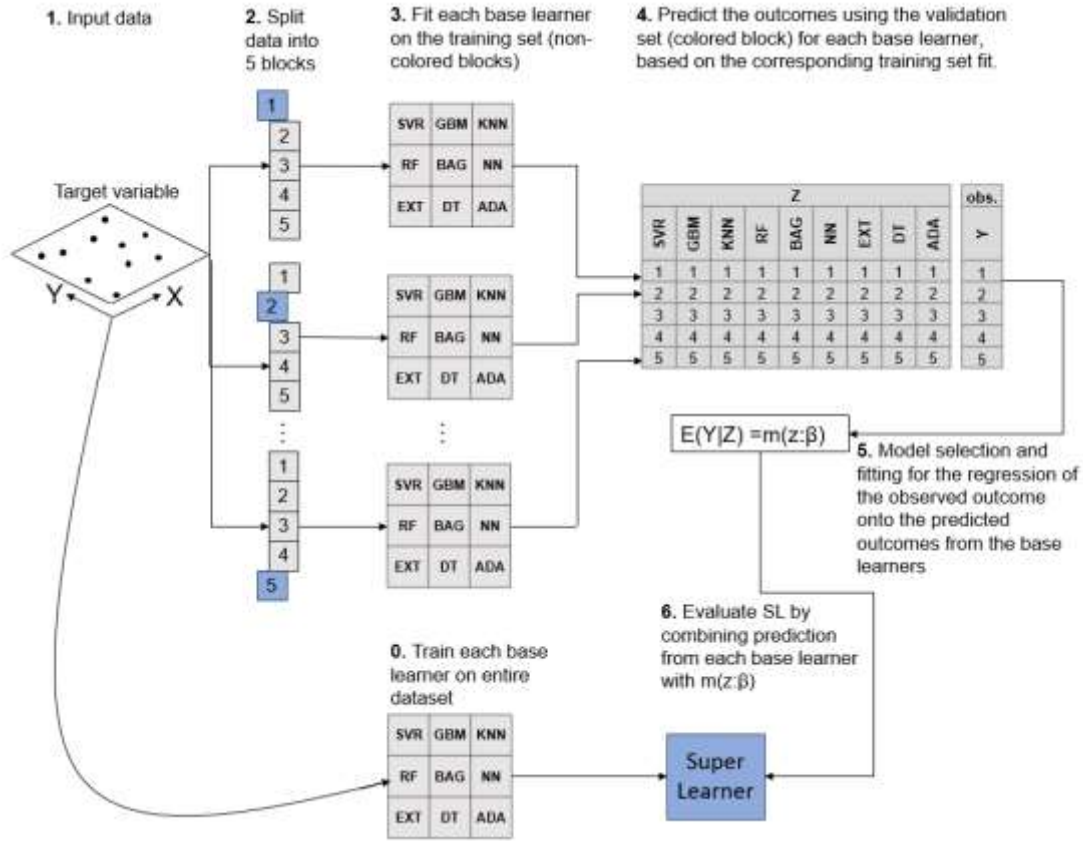
4.2. The Super Learner Model

There are many different types of ML models for an estimation modeling problem. It is impossible to know a priori which ML model will perform best for a given dataset and estimation problem (Van der Laan et al., 2007). Therefore, some researchers have proposed ensemble learners which combine various ML models and have demonstrated better performance over an individual ML model (Breiman, 1996; Freund et al., 1997; Hansen, 1998). The ensemble learners improve the estimation capacity and decrease the variance of the individual ML models (Hastie et al., 2009). The details of the ensemble models have been given in section 3.5.10. However, these learners have the potential to overfit the data and may not be the best way to combine the individual ML models. Recently, Van der Laan

et al. (2007) proposed the SL model that creates the best-weighted combination of individual learners that will enhance the performance of the ensemble.

The SL model forms a set of estimated values based on the optimal weighted combination of a list of estimation models. It is a cross-validation-based ensemble approach, which minimizes a cross-validated loss function for the combination. The choice of the loss function depends on the estimation problem and characteristics of the data. By implementing cross-validation, the problem of having many ML models and not knowing what would be the best models to use can be solved, which helps to avoid overfitting (Naimi and Balzer, 2018).

In an SL model, firstly a k -fold split of the data is defined (typically 5- fold or 10-fold), and all of the different models and model configurations are evaluated on the same split of the data. All out-of-fold estimations are then stored and used to train a meta-model that learns how to combine the estimates by minimizing the mean MSE. The final test statistics are computed from the data that are truly left out from the beginning and are not used in the meta-learning phase. The SL model performs asymptotically as well or better than any candidate model evaluated during the cross-validation (Polley and Van Der Laan, 2010; Van der Laan et al., 2007). It can consider numerous estimation models and benefits from the individual strengths of the best-performing models for the given dataset. The general workflow of the SL model in the current study is illustrated in Figure 4.1.



The SL model is explained in detail here to estimate the conditional expectation, $E[Y | \mathbf{W}]$, where Y is a dependent variable and $\mathbf{W}$ is a vector denoting the set of covariates. The data includes n observations $O = (Y, \mathbf{W})$. In the SL model, the models $(M_1, \dots, M_K)$ are determined first, where K is the number of models. Then, the steps below are followed in sequence:

1. Each model $(M_1, \dots, M_K)$ is trained on all n observations, and estimated values are obtained, that is, an $(n \times K)$ matrix. The matrix is defined by:

$$Z' = \begin{bmatrix} \hat{Y}_{11} & \dots & \hat{Y}_{1K} \\ \vdots & \ddots & \vdots \\ \hat{Y}_{n1} & \dots & \hat{Y}_{nK} \end{bmatrix} \quad (4.1)$$

where $\hat{Y}_{ij}$ with $i = 1, \dots, n$ and $j = 1, \dots, K$ is the element of the matrix Z' denoting the estimated model.

2. The SL model is built on V-fold cross-validation. Thus, data is divided into V equal-sized partitions indexed by $v \in 1, \dots, V$. The validation set $V(v)$ contains observations in partition v , while corresponding training set $T(v)$ comprising all observations except those in the validation set,
3. Each model is trained on $T(v)$ and estimated values are obtained for observations in $V(v)$. It gives an estimation matrix $(n \times K)$ with one estimated value per observation for each candidate learner:

$$Z = \begin{bmatrix} \hat{Y}_{11}^V & \dots & \hat{Y}_{1K}^V \\ \vdots & \ddots & \vdots \\ \hat{Y}_{n1}^V & \dots & \hat{Y}_{nK}^V \end{bmatrix} \quad (4.2)$$

where $\hat{Y}_{ij}^V$ with $i = 1, \dots, n$ and $j = 1, \dots, K$ is the element of the matrix Z denoting the estimated model on validation dataset $V(v)$.

4. A set of weighted combinations of candidate learners indexed by β , $m(z|\beta) = \sum_{k=1}^K \beta_k \hat{Y}_k$ is proposed as the index of the weighted combinations of candidate learners under the constraints of $\beta_k \geq 0 \forall k$ and $\sum_{k=1}^K \beta_k = 1$. This step includes the meta-learning process of the SL model. The contribution of each candidate learner is calculated to the final estimation $(\hat{\beta}_1, \dots, \hat{\beta}_K)$ by defining $\hat{\beta}$ that minimizes the cross-validated loss function:

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^n (Y_i - m(z_i | \beta))^2 \quad (4.3)$$

where, Y_i is the observed data and z_i shows the vector of estimated values $(\hat{Y}_{i1}, \dots, \hat{Y}_{iK})$ for subject i from the i th row of the estimation matrix Z .

5. Finally, SL model estimates are evaluated by using:

$$Y_{SL} = \sum_{k=1}^K \hat{\beta}_k z'_k \quad (4.4)$$

where z'_k is the column k of the Z' estimation matrix obtained in the first step. Appendix-A contains an example for a step-by-step walkthrough of how to build an SL model.

The diversity of the models plays an important role in building a powerful ensemble model. By combining information from diverse modeling approaches, the ensemble models gain more accuracy and robustness. We have selected nine different ML models including k NN, DT, bagging (BAG), RF, EXT, ADA, GBM, SVR, and ANN as base models in the SL modeling process. In this study, Keras Regressor has been used to develop and evaluate the ANN (Brownlee, 2016). It wraps the efficient numerical computation libraries such as Theano and TensorFlow in scikit-learn (Pedregosa et al., 2011), and it defines and trains the ANN. The base models are selected as the standard procedures for the implementation of them are well established and have diverse approaches to solve the estimation problems. Moreover, some of them are also being increasingly used in related geospatial estimation problems (Chapter 2). Moreover, some of these models are accepted as the top data mining algorithms (Wu et al., 2008). In this study, the LR is used as the meta-model (Witten et al., 2016). Wong et al. (2019) have proved that hyper-parameter tuning of candidate ML models in the SL model yields slightly better performance than an SL model that employs default hyper-parameter values. Therefore, in this thesis, the hyper-parameters of each ML model have been tuned through a grid search algorithm that measures the models' cross-validated performance in the training dataset over a specified range of hyper-parameter values.

4.3. Application of the SL Model for the Geospatial Estimation

The main points of the application of the SL model for the geospatial data are summarized as follows:

- The visualization and analysis of the distribution of the true data. It provides important information about the nature of the data.
- Building of input spaces. The coordinates (i.e. Easting, Northing, and Elevation) are used as input variables and the corresponding target variable is used as the output variable.

- Data preprocessing – The data is normalized in order to fit the values in the range [0, 1], where most ML models work optimally (see section 3.4 for more details).
- 5-fold cross-validation splits the data, uses one-fold as a test and the remaining 4 folds as a training set. In this thesis, this step is implemented for cases that have not exhaustive data.
- Training and validation of the SL model.
- Hyper-parameter tuning of individual ML models, whose selected hyper-parameters are listed in Table 3.3, with the grid search algorithm by the 5-fold cross-validation. Appendix-B shows the codes written in Python programming language that implements the grid search algorithm to a 2D dataset example.
- The production of the estimates at every grid node.
- Model evaluation through the performance criteria given in section 3.6.

4.4. Application of Kriging for the Geospatial Estimation

The main points of the application of the kriging for the geospatial data are summarized as follows:

- The visualization and analysis of the distributions of the data.
- Data preprocessing - The datasets are normalized to be compatible with ML modeling.
- 5-fold cross-validation splits the data for training and evaluation. This step is not actually necessary to kriging, it is just implemented to compare the performance of kriging against SL model fold by fold.
- The calculation of the experimental variogram.
- Fitting a positive definite analytical model to the variogram.
- Determine kriging parameters.
- Generating estimates using kriging at every grid node.
- Model evaluation through the performance measures given in section 3.6

4.5. Partitioning Drill Hole Data into k -Folds

An important step of the implementation of a ML is to split the data into training and validation datasets to evaluate the generalization capability of the models, which means how the model works on unseen data (Friedman et al., 2001). A model's generalization capability is measured on a validation dataset independent of the training dataset. The current study makes use of k -fold cross-validation rather than setting aside a validation set. The k -fold cross-validation is more effective in utilizing the information available in the training data (Hastie et al., 2009). In this thesis, Deutsch (2018)'s algorithm is implemented to partition a set of drill hole data into k -folds for model validation. This algorithm considers the drill hole data as *spatially* correlated and designs a closer spacing between partitions and a wider spacing within partitions.

The difference of this algorithm from the cross-validation partitioning method mentioned in section 3.2.4 is that drill holes are assigned to the folds in their entirety. When the samples from the same drill hole are assigned to different folds, it would cause the data to be unreasonably close to one another (data immediately above or below a particular location) during the estimation process. Thus, it leads to optimistic estimations. Figure 4.2 shows an example for a random 5-fold cross-validation in which samples are assigned from the same drill holes.

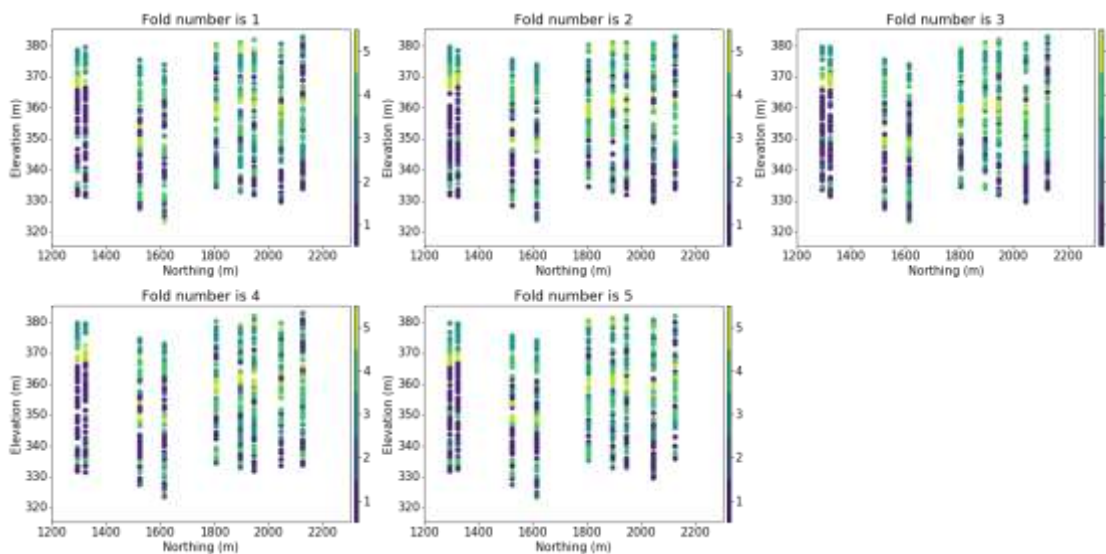


Figure 4.2. An example for a random 5-fold cross-validation

Deutsch's algorithm (2018) does spatial partitioning and considers two important features to obtain uniformity in the partitioning. One of them is that the drill holes from a certain fold should be close to drill holes from another fold. The other one is that the drill holes in the same fold should be far apart. In this algorithm, it is assigned one drill hole into a fold at a time. Firstly, the fold with the fewest data is selected and then a measure of fitness to be assigned to the fold, which is given in Equation 4.5, is used to rank the remaining unassigned drill holes.

$$f_i = \underbrace{-\varepsilon dc_i}_{(I)} + \underbrace{w_{rand} r_i}_{(II)} + \underbrace{w_{same} d_{si}}_{(III)} - \underbrace{w_{diff} dd_i}_{(IV)} \quad i = 1, \dots, n_{remaining} \quad (4.5)$$

where part (I) contains a small number ε and the distance of the drill hole to the centroid of all the drill holes dc_i . In this part, it will be chosen the drill holes near the center of the domain. Part (II) consists of a weight w_{rand} assigned to the random component and the random number r_i assigned to drill hole. This part ensures randomness to avoid any artifacts and to mimic irregular drilling. Part (III) comprises a weight w_{same} assigned to the distance to the closest drill hole in the same fold as being considered and the average close distance d_{si} to the three closest. In this part, it is ensured that the drill holes are far from other drill holes in the same fold. The last part (IV) consists of a weight w_{diff} assigned to the distance to the closest drill hole in a different fold as being considered and again the average close distance dd_i to the three closest. In the latest part, it will be chosen the drill holes close to a different fold.

Figure 4.3 shows a comparison of two-fold partitioning on a regular grid with random (Figure 4.3 (a)) and with the proposed approach (Figure 4.3 (b)). Figure 4.3 (a) demonstrates that the clusters of green and blue points result in a non-uniformity pattern for validation. On the other hand, Figure 4.3 (b) illustrates that the green and blue points are close to each other, which provides more natural nesting of the patterns. The natural “nesting” of the folds is desired to more closely represent the possible data configurations in the final estimate.

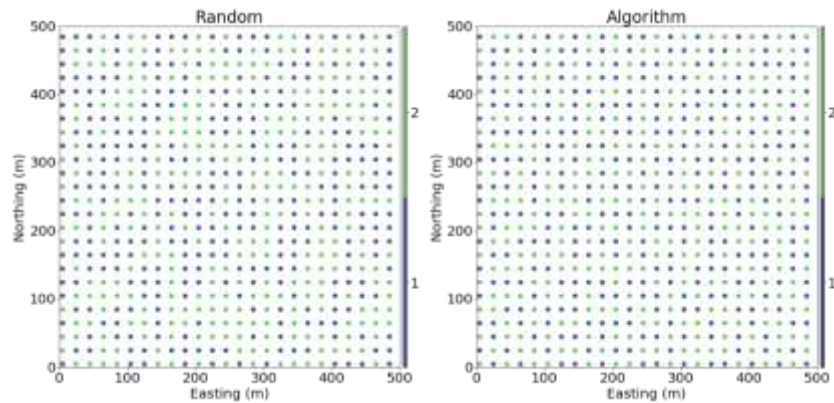


Figure 4.3. Two folds with (a) random and (b) controlling the closeness

The algorithm also considers the spatial arrangement during the partitioning of the data. This property is demonstrated in Figure 4.4 with two-fold partitioning. The proposed data partitioning procedure can give consistent subsets that represent the entire dataset.

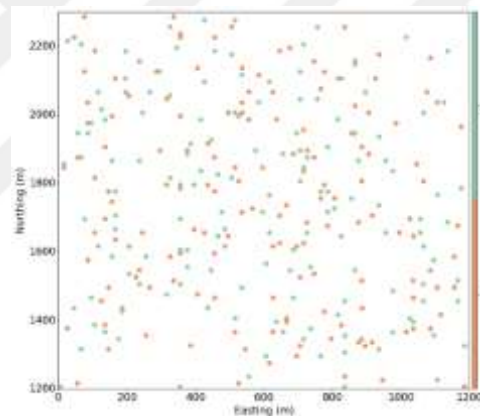


Figure 4.4. Two-fold spatial partitioning

4.6. Managing Estimation Artifacts in Machine Learning Geospatial Estimation

ML models do not consider the spatial locations of the samples during the modeling and ignore the spatial autocorrelation in the data. When an ML model is used for modeling spatial data, it uses coordinates in two- or three-dimensional spaces as predictors, which causes most ML models to be very sensitive to the coordinate. The resulting spatial maps indicate artifacts (i.e. linear or blocky boundaries), which is a significant problem in the practical usefulness of the models. It is nearly impossible to recognize whether these artifacts are due to the available input data or modeling limitations. Moreover, most ML models could

only learn anisotropic features provided that these features are aligning with the coordinate system.

To overcome these significant problems, the ESL model which uses the SL model for ML modeling has been proposed. The main purpose of the ESL model is to provide a tool to manage the estimation artifacts in estimation maps, incorporates anisotropy information into the SL model, and thus, obtain more accurate estimates with the SL model. For this purpose, the coordinate rotation strategy has been used to formulate the estimates such that they are invariant under rotation. A set of rotated coordinates has been used to provide additional spatially relevant predictors to the available features (i.e. Easting, Northing, and Elevation). The proposed methodology obtains information from sample locations and their spatial features and uses this information to assign spatial estimates at unsampled locations. The spatial estimates are then ensembled to obtain the final optimal estimates.

4.6.1. Coordinate system rotation

The simple Cartesian coordinate systems are not conformed to geological formations. It is almost impossible to find a geological formation (i.e. ore body) that is oriented north-south. Thus, it is sometimes worked in a coordinate system that is rotated with the geology to successfully model geological properties (Neufeld and Deutsch, 2004). The most important advantage of using rotated coordinates is to characterize the true anisotropy of the deposit. Figure 4.5 demonstrates an example for rotation to capture the true anisotropy. The gray lines are the apparent anisotropy for the original coordinate system and the blue lines are the true anisotropy for the rotated coordinate system.

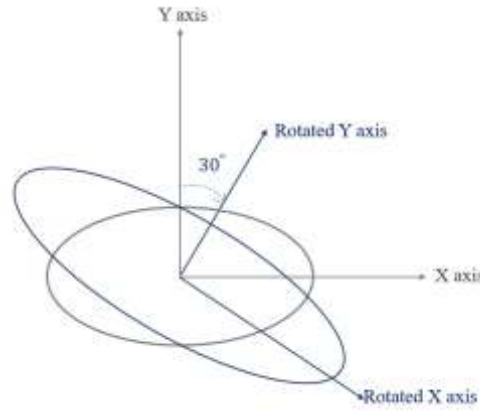


Figure 4.5. An example aims to capture the true anisotropy

In this thesis, the angle rotation method (Deutsch and Journal, 1998) has been implemented to incorporate the rotation into the learning process of the SL model. This rotation method is based on the coordinate system rotation convention used in the GSLIB programs (Deutsch, 2015). It considers the original X direction in the easting, the original Y direction is in northing, and the original Z direction as elevation vertically upward. Then, it aims to rotate the original coordinate system $\{X, Y, Z\}$ to be along strike, down dip, and perpendicular to the structure coordinates $\{X', Y', Z'\}$. An angle α , measured clockwise from the Y-axis, may be used to describe the rotation in 2D orientation. This angle is generally considered as the azimuth or bearing in geologic modeling.

The current study uses azimuth α coordinate rotation to scan the study area at regular intervals to build a rotation. In azimuth rotation in 2D, X and Y axes are rotated in a clockwise angle α and the Z-axis remains unchanged. Equation 4.6 shows the transformation of any point (X, Y) into a rotated point (X', Y') when orienting a new coordinate system.

$$\begin{aligned} X' &= X \cdot \cos \alpha - Y \cdot \sin(\alpha) \\ Y' &= X \cdot \sin \alpha + Y \cdot \cos(\alpha) \end{aligned} \quad (4.6)$$

Equation 4.6 can also be expressed in matrix notation using the following equation:

$$\begin{bmatrix} X' \\ Y' \\ Z \end{bmatrix} = \begin{bmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad (4.7)$$

This transformation is illustrated in Figure 4.6.

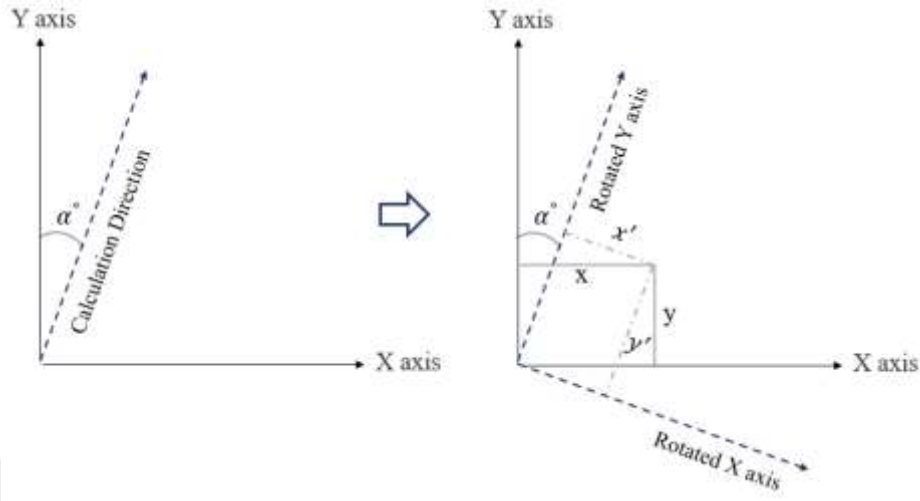


Figure 4.6. Rotation of azimuth in 2D (after Deutsch, 2015)

4.6.2. Ensemble the estimations of individual SL models

The relationship between the input and output characterized by a set of data values with different natures might not be captured sufficiently by a single SL model. For this reason, individual SL models using data from the rotated datasets that are produced from the training dataset are trained, and then estimations of each SL model are ensembled to obtain final optimal estimates. The ensemble strategy generally improves the estimation capacity of ML modeling and maximizes the generalization accuracy.

In this study, a set of rotated training datasets have been constructed by using a single angle α for producing geospatial estimates with just SL model. To select the most suitable value of an angle α , different values from 1° to 15° with an increment of 1° degree have been tested in various 2D and 3D datasets, respectively. It has been observed that when the value of angle α is too small, the computation time increases especially for large datasets while the performance results do not change significantly. Thus, the angle α has been selected as 5° for 2D cases and 10° for the 3D cases. Then, the angles have been changed from 0° to 90° in increments of the α . Therefore, 9 training datasets, at $\alpha = 10^\circ$, and 18 training datasets, at $\alpha = 5^\circ$, have been created for 3D and 2D cases, respectively. Figure 4.7 illustrates how rotated training datasets are created in 5° increments.

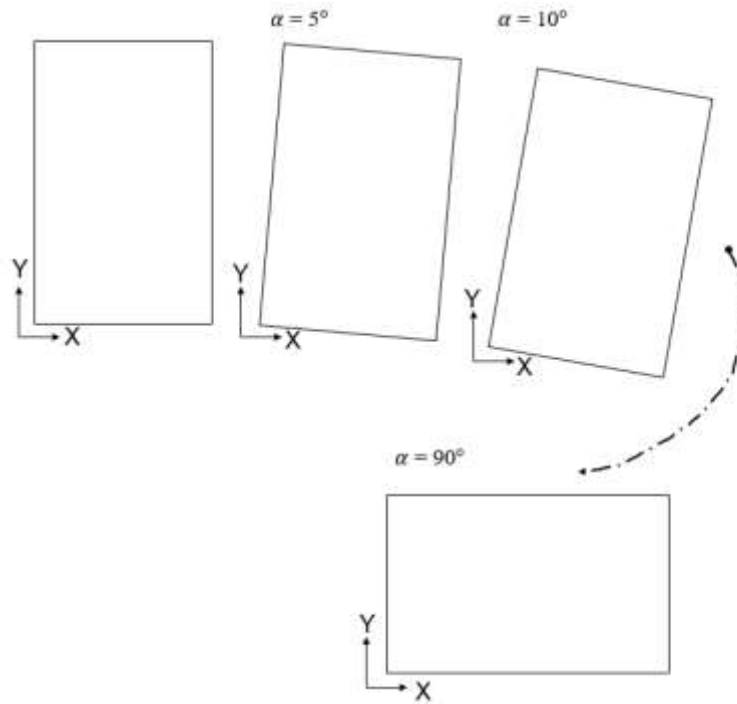


Figure 4.7. The rotated training datasets in 5° increments

The same rotation procedure has been applied for the grid nodes. Then, the SL models have been fitted on the separate rotated training datasets, and these fitted models have been used to estimate the corresponding rotated grid nodes. Finally, all estimation results have been ensembled by averaging, which results in one optimal final estimation map. The flowchart of the proposed ESL model is summarized in Figure 4.8.

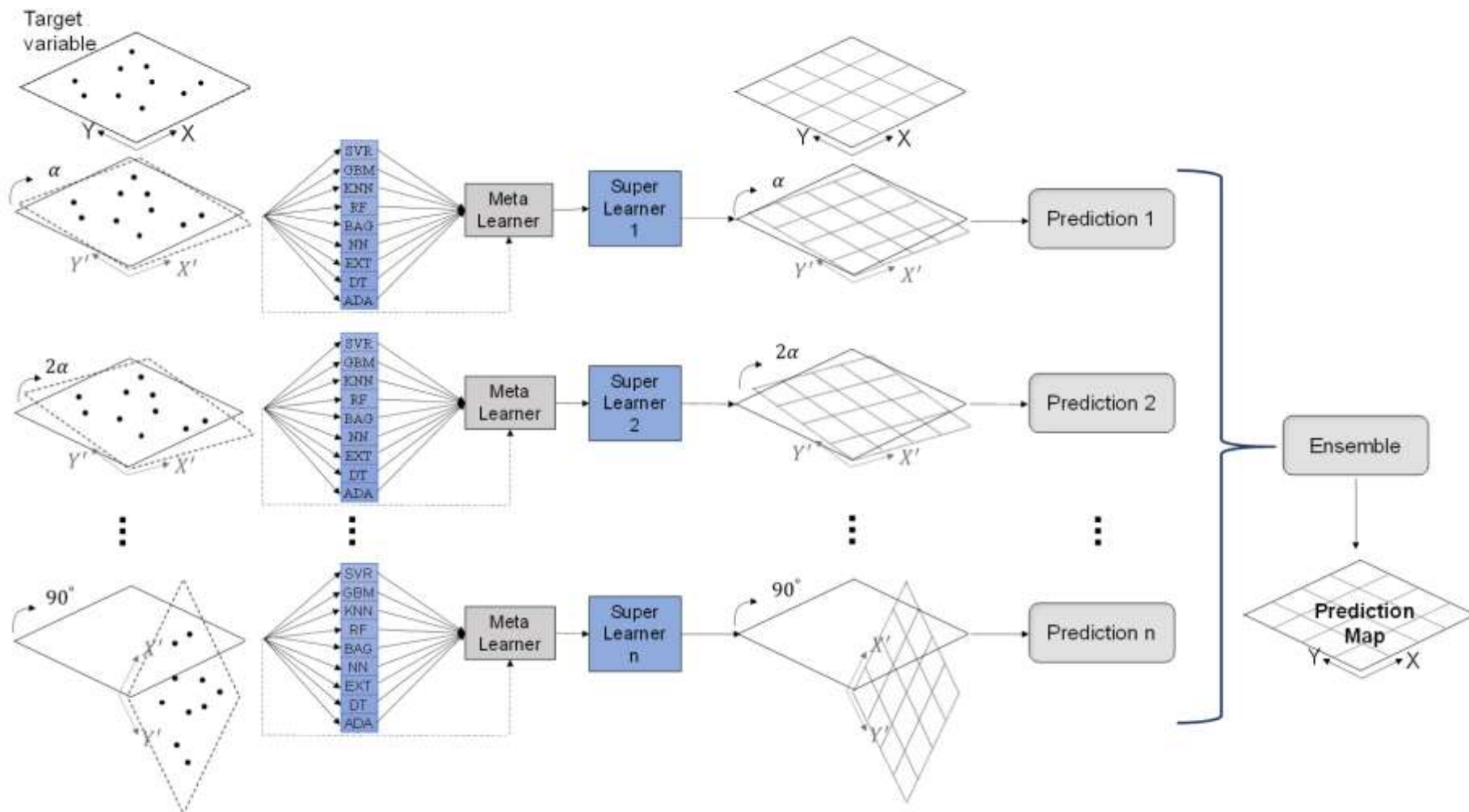


Figure 4.8. Flowchart of the ESL model

4.7. Combination of Machine Learning and Kriging for Geospatial Estimation

The ML and kriging have different pros and cons for a spatial estimation which are listed below:

Geostatistics (Kriging):

- ✓ Kriging is known as the best linear unbiased estimation method.
- ✓ It incorporates spatial variability into the estimation process through the variogram.
- ✓ It reproduces the data at the known locations.
- ✓ It can make simulation, so provides a measure of estimation uncertainty.
- ✗ It requires stationarity assumption (1st and 2nd order stationarity). This implies that the mean and variance of the random variable are constant across the domain. It can be questionable and difficult to verify.
- ✗ It requires the linearity assumption. The relationship between dependent and covariates is assumed to be linear.
- ✗ It assumes that the distribution of the data is normal.
- ✗ It requires expert knowledge for variogram modeling and parameter selection.
- ✗ It may not provide efficient solutions for estimation in complex situations such as in non-Gaussian, non-linear, and non-stationary domains.

ML models (SL model):

- ✗ It considers that the data are independent and identically distributed.
- ✗ The particular relationship between the spatial proximity among observational units and the numeric similarity among their values is ignored.
- ✗ It is not currently capable of simulation.
- ✗ It is based on regression algorithms therefore they do not reproduce observed values at known locations (i.e. exactitude property in kriging). However, data reproduction is an important consideration for real estimation scenarios. The data reproduction at the known locations is illustrated in Figure 4.9.

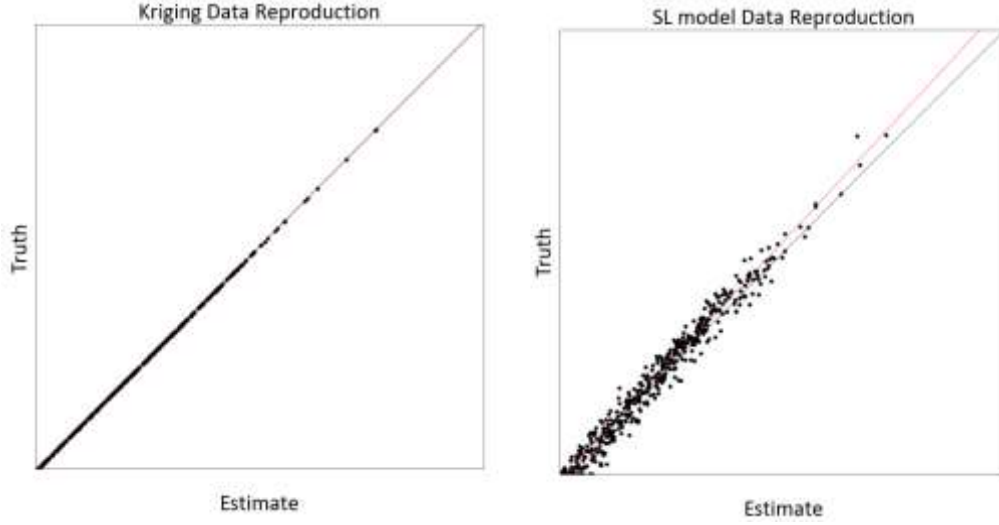


Figure 4.9. Data reproduction for kriging and SL model.

- ✓ It does not require stationarity, linearity, and normality assumptions.
- ✓ There is no need to model the variogram and determine the kriging parameters. Therefore, it requires less expert knowledge and human interaction.
- ✓ It works better in complex domains.

In the current study, the estimates generated by the SL model and kriging are combined by a weighting procedure based on local kriging variance $\sigma_{KR}^2(\mathbf{u})$ (either SK or OK) to mitigate the different shortcomings of each method in spatial estimation, and to increase the accuracy and consistency of the final estimates for some conditions. At each estimation location, assigning weights to both models according to the $\sigma_{KR}^2(\mathbf{u})$ ensures the exact reproduction of measured values at sampled locations, which is of critical importance in terms of many geoscience applications. It also improves the estimation in many situations that may be difficult for only using kriging or only using the SL model.

Considering the kriging, $\sigma_{KR}^2(\mathbf{u})$, which is available for every estimation location throughout the domain, measures how well each grid node in kriging is informed at the time of estimation. The mathematical equation of the $\sigma_{KR}^2(\mathbf{u})$ is given in Equations 3.41 and 3.46 for SK and OK, respectively. In the proposed methodology, $\sigma_{KR}^2(\mathbf{u})$ values have been standardized; that is, the values range between 0 and 1 ($\sigma_{KR}^2(\mathbf{u}) = [0,1]$). As the kriging estimator has exactitude property, the value $\sigma_{KR}^2(\mathbf{u})$ equals 0 at a location where a sample is available. Its value increases up to 1, as the estimation node, moves away from the known

data. During the combination of kriging and SL model, it has been benefiting from this feature of $\sigma_{KR}^2(\mathbf{u})$ and it has been used to assign weights to models. Thus, the ML model is assigned more weights as the estimation node moves away from the known data and the full weight is assigned to the kriging at a sample location. The methodology for combining the SL model and kriging model is summarized in Figure 4.10.



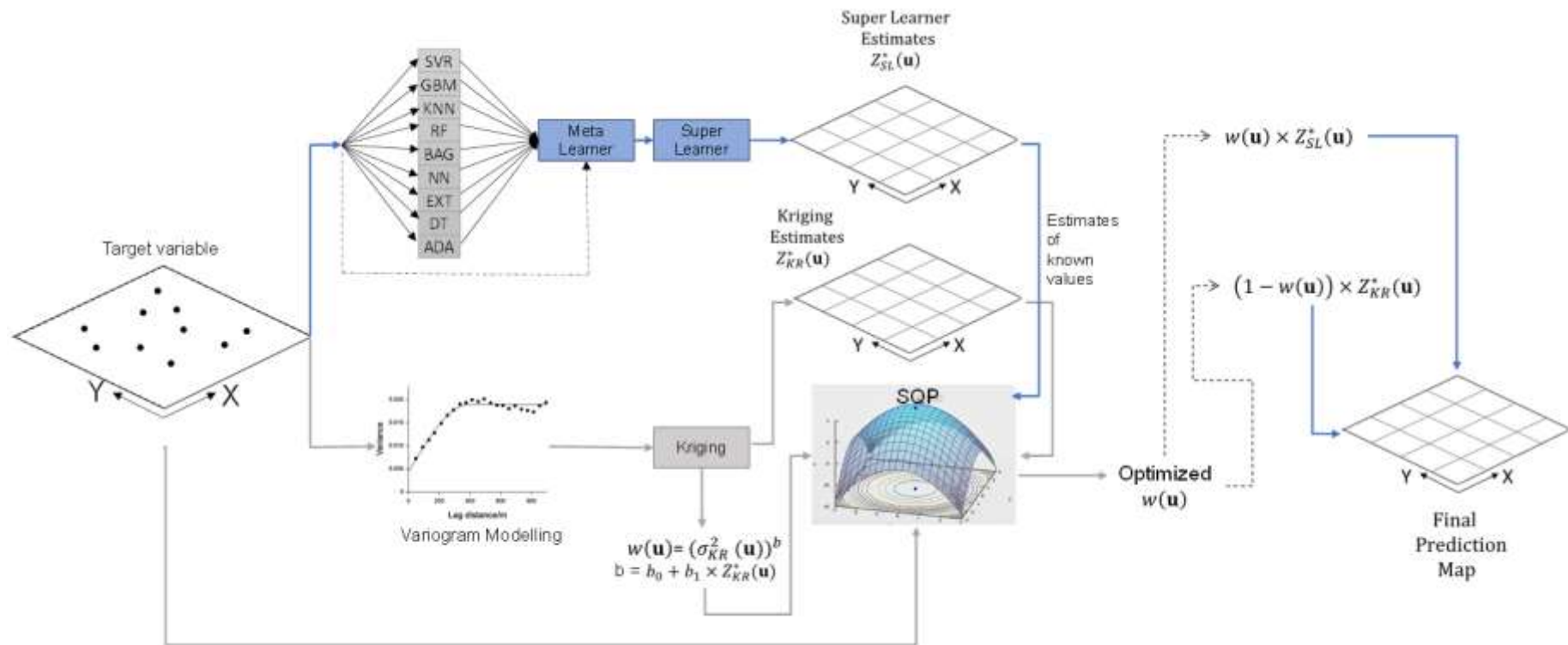


Figure 4.10. The workflow of the combination of ML and kriging for spatial estimation strategy

As can be seen in Figure 4.10, each model (SL and kriging) is used to generate estimates at the available grid nodes by using the same training dataset for the combination, and the estimates generated by them are denoted by $Z_{SL}^*(\mathbf{u})$ and $Z_{KR}^*(\mathbf{u})$ which correspond to the SL model and kriging at the location $\mathbf{u}$, respectively. As for the weighting, the general equations indicating the weighting of the estimations obtained from each model (SL model and kriging) are given in the following equations:

$$Z_{\text{Combined}}^*(\mathbf{u}) = w(\mathbf{u}) \times Z_{SL}^*(\mathbf{u}) + (1 - w(\mathbf{u})) \times Z_{KR}^*(\mathbf{u}) \quad (4.8)$$

$$w(\mathbf{u}) = (\sigma_{KR}^2(\mathbf{u}))^b \quad (4.9)$$

$$b = b_0 + b_1 \times Z_{KR}^*(\mathbf{u}) \quad (4.10)$$

where $Z_{\text{Combined}}^*(\mathbf{u})$ is the estimate generated by the combined model; $w(\mathbf{u})$ is the weight of the SL model; $(1 - w(\mathbf{u}))$ is the weight of kriging and b that includes a linear equation with b_0 , b_1 and $Z_{KR}^*(\mathbf{u})$ is the exponential parameter of the weight. The parameter b regularizes the weight according to the kriging estimate. In this form, when kriging generates a completely inaccurate estimate at an estimation location $\mathbf{u}$ according to the value of the kriging variance (i.e. $\sigma_{KR}^2(\mathbf{u}) = 1$), $Z_{\text{Combined}}^*(\mathbf{u})$ will equal to $Z_{SL}^*(\mathbf{u})$ when kriging generates exactly accurate estimate (i.e. $\sigma_{KR}^2(\mathbf{u}) = 0$), $Z_{\text{Combined}}^*(\mathbf{u})$ will equal to $Z_{KR}^*(\mathbf{u})$.

The parameter b must be optimized according to the true data, $\sigma_{KR}^2(\mathbf{u})$, $Z_{KR}^*(\mathbf{u})$, and $Z_{SL}^*(\mathbf{u})$. This optimization problem is solved with sequential (or successive) quadratic programming (SQP) in this study. The details of SQP will be given in section 4.7.1. In the proposed method, the problem formulation, which is given in Equation 4.11, is the minimization of RMSE that is calculated between true data (Z_{true}) and combined estimates (Z_{Combined}^*) at estimation points ($\mathbf{u}$).

$$\text{Min} \sqrt{\sum_{i=1}^N \frac{(Z_{\text{true}}(\mathbf{u}) - (\sigma_{\text{KR}}^2(\mathbf{u}))^{\mathbf{b}_0 + \mathbf{b}_1 \times Z_{\text{KR}}^*(\mathbf{u})} \times Z_{\text{SL}}^*(\mathbf{u}) + (1 - (\sigma_{\text{KR}}^2(\mathbf{u}))^{\mathbf{b}_0 + \mathbf{b}_1 \times Z_{\text{KR}}^*(\mathbf{u})}) \times Z_{\text{KR}}^*(\mathbf{u})))^2}{N}} \quad (4.11)$$

Subject to: $a_1 \leq b_0 \leq c_1$

$a_2 \leq b_1 \leq c_2$

The objective variables of the optimization function that include $\mathbf{b}_0$ and $\mathbf{b}_1$ are constrained as shown in Equation 4.11, where a and c is upper and lower limits of the objective variables. The initial value doesn't need to be feasible in SQP (Boggs and Tolle, 1995), thus the starting values of b_0 and b_1 have been initialed with $[1, 0]$. The codes written in Python programming language used to optimize parameter b through SQP can be found in Appendix-C.

4.7.1. Sequential quadratic programming

4.7.1.1. Quadratic programming

Quadratic Programming (QP) is the special class of nonlinear optimization problems in which the objective function is quadratic and all the constraints are linear (Frank and Wolfe, 1956). If the objective function is quadratic, it means that it can contain squared terms, or terms that are the product of two variables, or linear terms. The general mathematical formulation of a QP is described by (Yang, 2016):

$$\begin{aligned} f(\mathbf{x}) &= \frac{1}{2} \mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{b}^T \mathbf{x} \\ \text{subject to: } \mathbf{A} \mathbf{x} &\leq \mathbf{d} \\ \mathbf{x} &\geq 0 \end{aligned} \quad (4.12)$$

where $\mathbf{x} = (x_1, x_2, \dots, x_n)^T$ is an n -dimensional vector, $\mathbf{b}$ is the constant vector. Here, $\mathbf{Q} = [q_{ij}]_{n \times n}$ is a symmetric positive semi-definite $n \times n$ matrix, $\mathbf{A} = [a_{ij}]_{m \times n}$ is the usual $m \times n$ matrix for the linear constraints in which m is the number of inequality constraints and $\mathbf{d}$ is the value of the constraint. The constraints can be incorporated using the method of Lagrange multipliers and Karush–Kuhn–Tucker (KKT) conditions.

The Lagrange multipliers is a strategy to determine the local maxima or minima of a function subject to equality constraints (Hoffmann et al., 1989). It is based on an understanding that if there is a single equality constraint, then a local optimum point for the model is obtained when the gradient of the constraint at that point is *parallel* to the gradient of the objective function at that same point (Chinneck, 2006). Figure 4.11 illustrates the basic idea of Lagrange multipliers for a maximization objective.

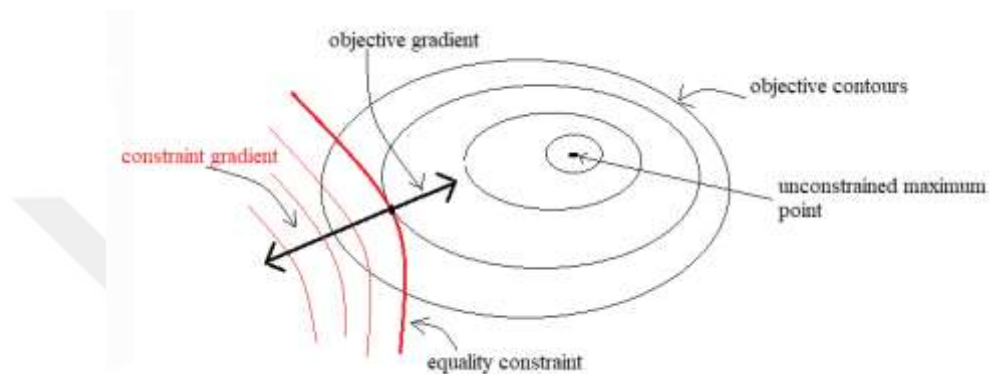


Figure 4.11. Parallel gradients in the Lagrange method (after Chinneck, 2006).

Figure 4.11 demonstrates that the gradient of the objective function is parallel to the gradient of the equality constraint. In other words, these two gradients are in the opposite direction. They are in balance at a local optimum point and they can have different lengths. The relationship between them is stated by the Lagrange multiplier factor, which sets for direction and length. This factor can be positive or negative for an equality constraint.

The Lagrange multipliers are generalized by the KKT conditions, which concern the requirement for a solution to be optimal in nonlinear programming. The Lagrange multipliers give linear functions because they are the derivatives of a quadratic function. Therefore, there is an almost completely linear system to solve to find a solution to this nonlinear problem. The linear constraints form a convex set. If a convex objective is minimized or a concave objective is maximized, subject to a convex set of constraints, then a point satisfying the KKT conditions must be the global optimum point. The KKT approach allows inequality conditions while the Lagrange multipliers allow only equality constraints (Murty and Yu, 1988). The generalized Lagrangian is described as:

$$L = \frac{1}{2} \mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{b}^T \mathbf{x} + \lambda^T (\mathbf{A} \mathbf{x} - \mathbf{d}) - \mu^T \mathbf{x} \quad (4.13)$$

Where L is the Lagrangian function, $\lambda = (\lambda_1, \dots, \lambda_p)$ is the vector of Lagrange multipliers, and $\mu = (\mu_1, \dots, \mu_p)$ denotes the vector of KKT multipliers, respectively. The KKT conditions can be written as:

- Lagrange multiplier nonnegativity condition:

$$\begin{aligned} \lambda &\geq 0 \\ \mu &\geq 0 \end{aligned} \quad (4.14)$$

- Stationary (or parallel gradients) conditions:

$$\nabla_x L = \mathbf{Q} \mathbf{x} + \mathbf{b} + \mathbf{A}^T \lambda - \mu = 0 \quad (4.15)$$

where ∇ is the gradient function.

- Complementary condition:

$$\begin{aligned} \lambda^T (\mathbf{A} - \mathbf{d}) &= 0 \\ \mu^T \mathbf{x} &= 0 \end{aligned} \quad (4.16)$$

From the quadratic objective f , it can be written:

$$\nabla^2 f = \mathbf{Q} \quad (4.17)$$

When $\mathbf{Q}$ is positive semi-definite, the global minimum will be the solution given by the stationary condition.

4.7.1.2. Sequential quadratic programming

Sequential Quadratic Programming (SQP) is one of the most popular methods to solve the constrained nonlinear optimization problems (Nocedal and Wright, 2006). It is based on a profound theoretical foundation and provides robust tools for the solution of large-scale problems (Boggs and Tolle, 1995; Boggs and Tolle, 2000). SQP is defined as follows:

$$\begin{aligned} &\text{minimize } f(\mathbf{x}) \\ &\text{subject to } h_i(\mathbf{x}) = 0 \quad (i = 1, \dots, p), \\ &\quad \quad \quad g_j(\mathbf{x}) \leq 0 \quad (j = 1, \dots, q). \end{aligned} \tag{4.18}$$

where $f(\mathbf{x})$ is the objective function and $h_i(\mathbf{x})$ and $g_j(\mathbf{x})$ are the equality and inequality constrain functions, respectively. SQP uses a *quasi-Newton* method to approximate the computationally extensive full Hessian matrix, which is a square matrix of second-order partial derivatives of a scalar field. Then, this generates a QP subproblem at each iteration. The solution to this QP subproblem could be used for defining the search direction and subsequent trial solution. In this section, firstly, the Newton and quasi-Newton methods are briefly discussed for a clear explanation of the SQP method (Lee and Leyffer, 2011).

Newton's method is an iterative procedure and applied to find the zeros (or roots) of the first derivative $f'(\mathbf{x})$ for a given objective function $f(\mathbf{x})$. There is the Taylor expansion around a known point $\mathbf{x}_n$ (with $\Delta\mathbf{x} = \mathbf{x} - \mathbf{x}_n$) for a given continuously differentiable function $f(\mathbf{x})$:

$$f(\mathbf{x}) = f(\mathbf{x}_n) + (\nabla f(\mathbf{x}_n))^T \Delta\mathbf{x} + \frac{1}{2} \Delta\mathbf{x}^T \nabla^2 f(\mathbf{x}_n) \Delta\mathbf{x} \tag{4.19}$$

The Equation 4.19 is minimized near a critical point when $\Delta\mathbf{x}$ is the solution of the following equation:

$$\nabla f(\mathbf{x}_n) + \nabla^2 f(\mathbf{x}_n) \Delta\mathbf{x} = \mathbf{0} \tag{4.20}$$

Equation 4.20 leads to:

$$\mathbf{x} = \mathbf{x}_n - \mathbf{H}^{-1} \nabla f(\mathbf{x}_n) \quad (4.21)$$

where $\mathbf{H} = \nabla^2 f(\mathbf{x}_n)$ is the Hessian, which is a $n \times n$ square matrix. Newton's method requires the Hessian matrix to search for roots. The Hessian matrix of a function f is usually defined and arranged as follows:

$$\mathbf{H}f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 x_n} \\ \frac{\partial^2 f}{\partial x_2 x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n x_1} & \frac{\partial^2 f}{\partial x_n x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix} \quad (4.22)$$

If the iteration process begins with the initial vector $x^{(0)}$, which is an initial guess, then the formula of Newton's iteration for the n th iteration is described as (Yang, 2016):

$$\mathbf{x}^{(n+1)} = \mathbf{x}^{(n)} - \mathbf{H}^{-1}(\mathbf{x}^{(n)}) \nabla f(\mathbf{x}^{(n)}) \quad (4.23)$$

The calculation of the Hessian matrix for second derivatives can be too expensive to compute at every iteration. Thus, the *quasi-Newton method* can be a good alternative in which an identity matrix $\mathbf{I}$ is used to approximate the Hessian:

$$\mathbf{x}^{(n+1)} = \mathbf{x}^n - \alpha \mathbf{I} \nabla f(\mathbf{x}^{(n)}) \quad (4.24)$$

where $\mathbf{I}$ is the identity matrix and $\alpha \in (0,1)$ is a step size, respectively.

A basic SQP given in Equation 4.18 defines an appropriate search direction $\mathbf{d}_k$, using the Taylor expansions at an iterate $\mathbf{x}_k$, as a solution to the QP subproblem:

$$\begin{aligned}
& \text{minimize } q_k(\mathbf{d}) = f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k)^T \mathbf{d} + \frac{1}{2} \mathbf{d}^T \nabla^2 L(\mathbf{x}_k) \mathbf{d} \\
& \text{subject to } \quad h_i \mathbf{x}_k + \nabla h_i(\mathbf{x}_k)^T \mathbf{d} = 0 \quad (i = 1, \dots, p) \\
& \quad \quad \quad g_j \mathbf{x}_k + \nabla g_j(\mathbf{x}_k)^T \mathbf{d} \leq 0 \quad (j = 1, \dots, q)
\end{aligned} \tag{4.25}$$

where $q_k(\mathbf{d})$ is quadratic approximation generated for the SQP method and the L is:

$$\begin{aligned}
L(\mathbf{x}) &= f(\mathbf{x}) + \sum_{i=1}^p \lambda_i h_i(\mathbf{x}) + \sum_{j=1}^q \mu_j g_j(\mathbf{x}) \\
&= f(\mathbf{x}) + \lambda^T h(\mathbf{x}) + \mu^T g(\mathbf{x})
\end{aligned} \tag{4.26}$$

where $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_p)^T$ is an p dimensional vector of Lagrange multipliers and $\mu = (\mu_1, \mu_2, \dots, \mu_q)^T$ is q dimensional vector of KKT multipliers. Moreover, $h = (h_1(\mathbf{x}), h_2(\mathbf{x}), \dots, h_p(\mathbf{x}))^T$ and $g = (g_1(\mathbf{x}), g_2(\mathbf{x}), \dots, g_n(\mathbf{x}))^T$. The SQP method (Equation 4.25) requires to approximate Hessian $\nabla^2 L(\mathbf{x}_k)$. This matrix is generally replaced with the standard Broydon-Fletcher-Goldfarbo-Shanno approximation of the Hessian, $\mathbf{B}_k$, which uses a quasi-Newton scheme and is updated at each iteration:

$$\mathbf{B}_{k+1} = \mathbf{B}_k - \frac{\mathbf{B}_k \mathbf{d}_k \mathbf{d}_k^T \mathbf{B}_k^T}{\mathbf{d}_k^T \mathbf{B}_k \mathbf{d}_k} + \frac{\mathbf{y}_k \mathbf{y}_k^T}{\mathbf{y}_k^T \mathbf{d}_k} \tag{4.27}$$

where,

$$\mathbf{d}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$$

$$\mathbf{y}_k = \nabla L(\mathbf{x}_{k+1}) - \nabla L(\mathbf{x}_k)$$

The convergence properties of the fundamental SQP method can be improved by using a line search method for minimization the penalty function:

$$\phi(\mathbf{x}) = f(\mathbf{x}) + \rho \left[|h_i(\mathbf{x})| + \sum_{j=1}^q \max\{0, g_j(\mathbf{x})\} \right] \tag{4.28}$$

where $\phi(\mathbf{x})$ denotes the penalty function and $\rho > 0$ is the penalty coefficient, respectively. In case the objective and constraint function satisfy certain conditions, then the penalty method converges to the optimal result.

5. RESULTS AND DISCUSSION

5.1. Introduction

This chapter presents the practical implementation of the proposed methods include the ESL model and the combined model as well as validates and compares their estimation performance with kriging and the SL model using public, simulated and real datasets.

5.2. Case Study – 1: Managing Artifacts in Machine Learning Estimation Through Ensemble Rotation

This section shows the practical details of the application of the ESL model in two public datasets including (1) Walker Lake (Isaaks and Srivastava, 1989) and (2) and Swiss Jura datasets (Atteia et al, 1994; Goovaerts, 1997; Webster et al., 1994) to test mainly if modeling artifacts are present in its final estimation maps. The estimation performance of the ESL model is compared with the OK and SL models in terms of R-squared, RMSE, and MAE to prove its validity. Moreover, estimation maps are assessed through visual inspections. The spatial autocorrelation on the residuals is also evaluated to support the understanding of the ESL model results.

5.2.1. Walker Lake case study

The Walker Lake dataset was derived from a digital elevation model from the Walker Lake area in Nevada. The original elevations were transformed to produce a dataset that includes three variables which are continuous V and U, and categorical T (Isaaks and Srivastava, 1989). While the continuous variables could be grade or thickness, the categorical variable could be rock types. The continuous variables, V and U, were defined as the concentration in parts per million (ppm) unit. Walker Lake dataset was sampled with a subset of data composes of 470 samples on a grid of 260m by 300m. This dataset has also an exhaustive dataset with 78,000 measured values distributed regularly on a 1×1 m grid. As the measurement of the variable U is missing at many sample locations, the variable V has

been used in this study. This variable is preferentially sampled at high-grade zones and presents only one population. Figure 5.1 indicates the location maps of the sample and exhaustive datasets for the variable V.

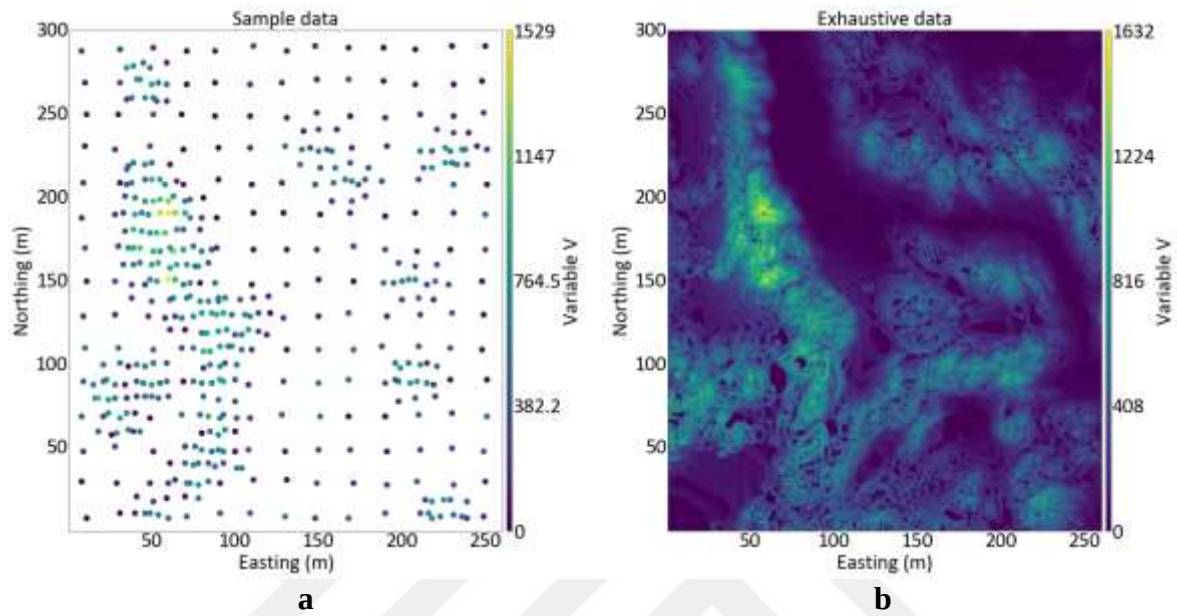


Figure 5.1. (a) the sample and (b) exhaustive datasets for variable V

Figure 5.1 shows that there is a preferential sampling in specified areas. Therefore, the cell declustering method has been applied to represent the sample data distribution more accurately. The study area is divided into cells that take equal weight; every sample is assigned the same weight within the cell (Deutsch, 1989). Figure 5.2 indicates that the optimum cell size is 24 m.

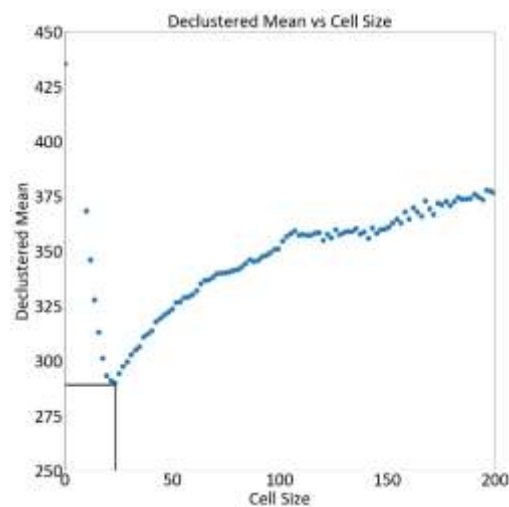


Figure 5.2. Cell size declustering of variable V.

Declustering modifies the weight of the data with regard to their probability in the global distribution. Figure 5.3 shows the histogram plots of the exhaustive and weighted sample datasets for variable V. The p-value of the variable V is less than 0.05 indicates that it does not have a Gaussian distribution (Shapiro and Wilk, 1965). From its histogram plot, the distribution of variable V looks like the lognormal.

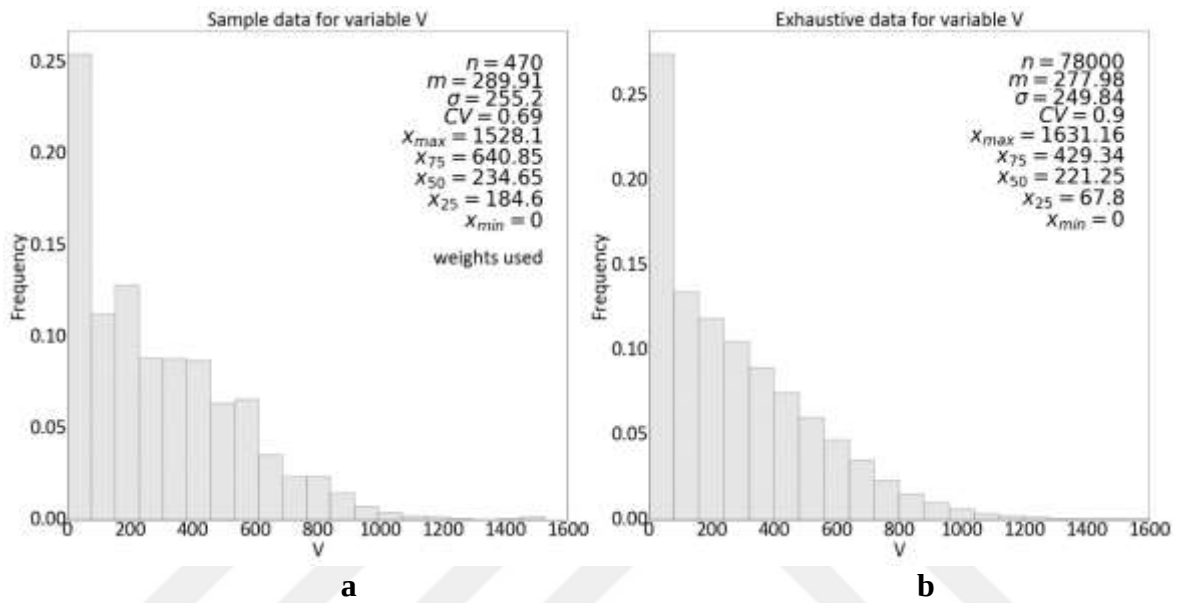


Figure 5.3. Histogram plots of the (a) sample and (b) exhaustive datasets for variable V

There is a clear anisotropy in the Walker Lake dataset, where the major and minor continuity directions are 14° from the West and 76° from the North (Isaaks and Srivastava, 1989). The variograms that consider this anisotropy is demonstrated in Figure 5.4 and Equation 5.1. The variogram model parameters are the same as those in the book 'An Introduction to Applied Geostatistics' (Isaaks and Srivastava, 1989).

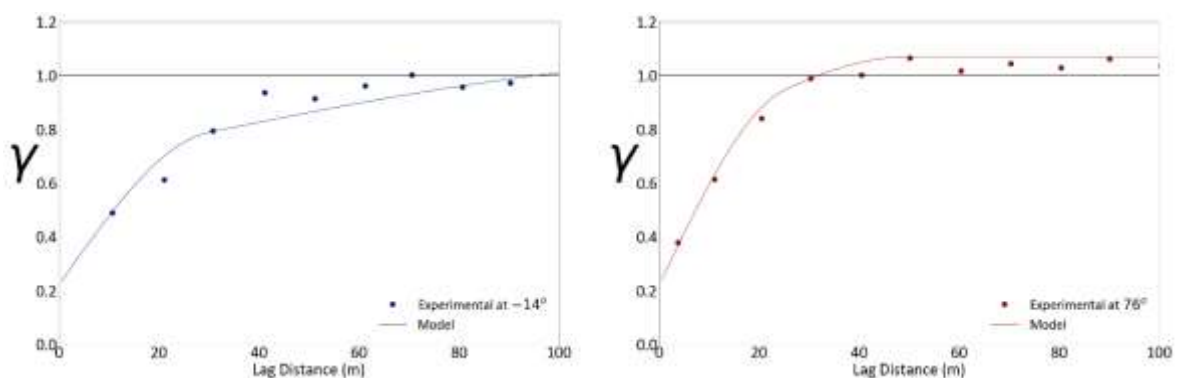


Figure 5.4. Modeled variograms for variable V

$$\gamma(\mathbf{h}) = 0.22_{Nugget} + 0.40 Sph_{a_{major}=150} + 0.45 Sph_{a_{major}=30} \quad (5.1)$$

$$a_{minor}=50 \quad a_{minor}=25$$

$$a_{vertical}=0 \quad a_{vertical}=0$$

The variogram parameters shown in Figure 5.4 and Equation 5.1 have been used to generate estimates with the OK model. For ML studies, the X and Y coordinates of the true data have been considered as input variables and the known variable V has been used as the target variable. Then, this target variable has been normalized to limit the variable to a confident range. As the Walker Lake dataset has an exhaustive dataset, the sampling data has not been split into training and validation datasets. The hyper-parameters of each ML model in the SL model have been tuned by the grid search algorithm with 5-fold cross-validation. It has been defined an initial search space as a grid of hyper-parameter values, which are listed in Table 3.3, for each ML model and evaluated every position in the grid. After the better value in the grid has been found, finer grid searches have been performed in that region, and the best hyper-parameter set that gives the minimum MSE in 5-fold cross-validation has been selected. Table 5.1 shows the optimal parameters of the base ML models.

Table 5.1. The best configuration of hyper-parameters for variable V in Walker Lake dataset.

ML model	Set of Hyper-parameters
kNN	n_neighbors: 3
DT	max_depth: 8
BAG	n_estimators: 2000
RF	n_estimators: 2000
EXT	n_estimators: 2000
ADA	n_estimators: 700
GBM	learning_rate: 0.1, n_estimators: 500, max_depth: 6
SVR	C: 1e3, epsilon: 0.1
NN	hidden_layer_sizes: 4, neurons: 20, batch_size: 20, epoch: 400

The SL model with these optimum hyper-parameters has been then trained firstly with the training set where input features are only X and Y coordinates in order to make a comparison with the ESL model. After this step, the coordinate rotation strategy has been implemented by selecting α as 5° and 18 rotated datasets have been created. The individual SL models have been trained with training datasets that belong to these rotated datasets.

Their estimation results have been obtained over the entire study area (all grid nodes) individually and then ensembled to obtain the final estimation.

Figure 5.5 shows the accuracy plots of the OK, SL, and ESL models, respectively. Their R-squared, RMSE, MAE values and basic statistics of estimations which mean value and the standard deviation (st.dev.) have also been summarized in Table 5.2. As can be seen from Figure 5.5 and Table 5.2 that the estimation performance of the ESL model is much higher than SL model, and produces similar results to the OK model. The overall correlation between OK and ESL maps is high ($r = 0.982$). The ensemble strategy has balanced out the estimation and improved the estimation result. It also appears from Table 5.2 that the mean is overestimated by the OK and SL models, whereas the mean appears underestimated by the ESL model. However, it can be said that all models have reproduced the mean satisfactorily.

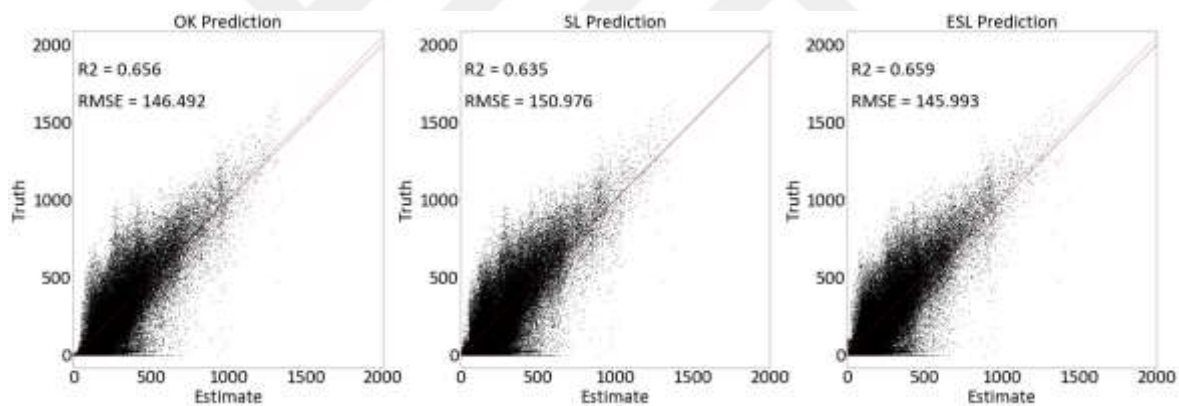


Figure 5.5. Estimation performance of OK, SL, and ESL models for variable V in Walker Lake dataset.

Table 5.2. The comparison of general metrics and the basic statistics of OK, SL, and ESL models for V variable in the Walker Lake dataset

Model	R-Squared	RMSE	MAE	mean	st.dev.
OK	0.656	146.492	110.393	285.005	195.438
SL	0.635	150.976	114.333	282.612	197.360
ESL	0.659	145.993	107.312	268.085	201.066
Truth	-	-	-	277.979	249.845

Figure 5.6 shows the omnidirectional variogram of the raw data and residuals of the ESL model. One can see that the experimental variogram of residuals demonstrates no

spatial autocorrelation. It means that the ESL model can extract structured information and only unpredictable unrelated noise was left (Kanevski et al., 2009).

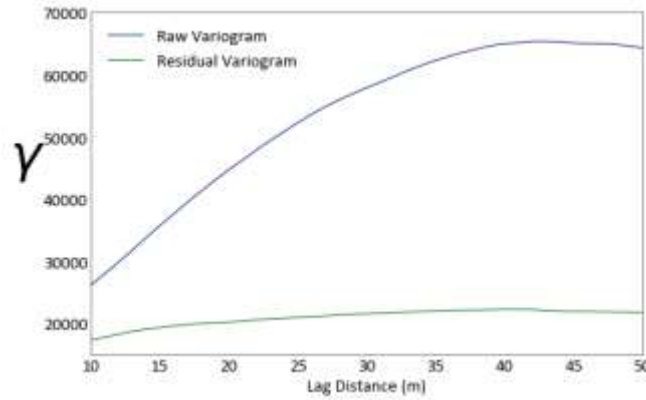


Figure 5.6. Omnidirectional variogram of raw data and the residuals of ESL model estimations

The estimation maps of the OK, SL, and ESL models have been checked in Figure 5.7 to ensure that the estimation models are artifact-free. It is clear from Figure 5.7 that the SL model is again not as good as the estimate of the OK model because of the artifacts. On the other hand, the ESL model manages the artifacts and yields a similar estimation map result to the one obtained by the OK model. For example, linear boundaries in the Northwest section of the SL estimation map have been smoothed out by the ESL model.

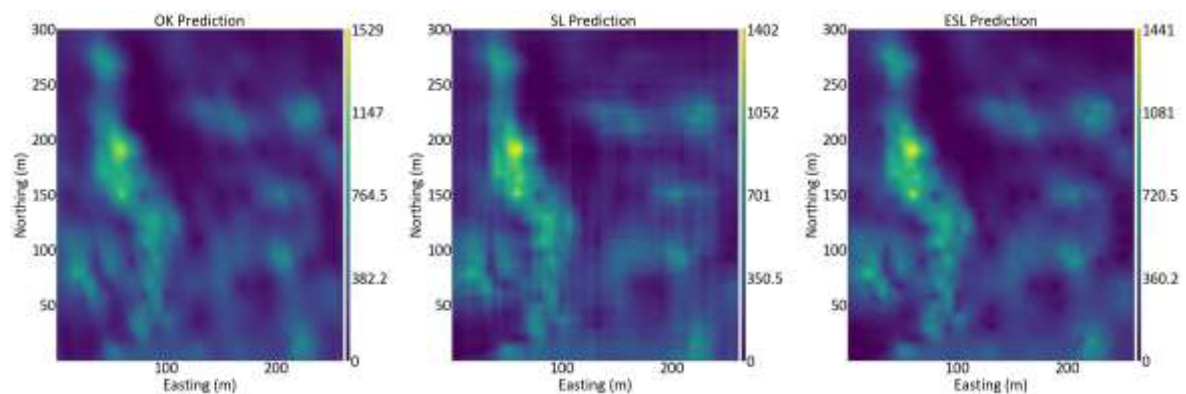


Figure 5.7. Spatial estimation maps of OK, SL, and ESL models for variable V in Walker Lake dataset.

5.2.2. Swiss Jura case study

The Swiss Jura (Atteia et al., 1994; Goovaerts, 1997; Webster et al., 1994) is a two-dimensional dataset and composes of seven heavy metal concentrations (cadmium (Cd), cobalt (Co), chromium (Cr), copper (Cu), nickel (Ni), lead (Pb), and zinc (Zn)) that were measured at 359 locations that cover an area of 14.5 km². 259 random samples were set aside from the dataset to form a training dataset and the remaining 100 samples were used to form a validation dataset (Goovaerts, 1997). Figures 5.8 and 5.9 show the location maps of the training and validation datasets, respectively.

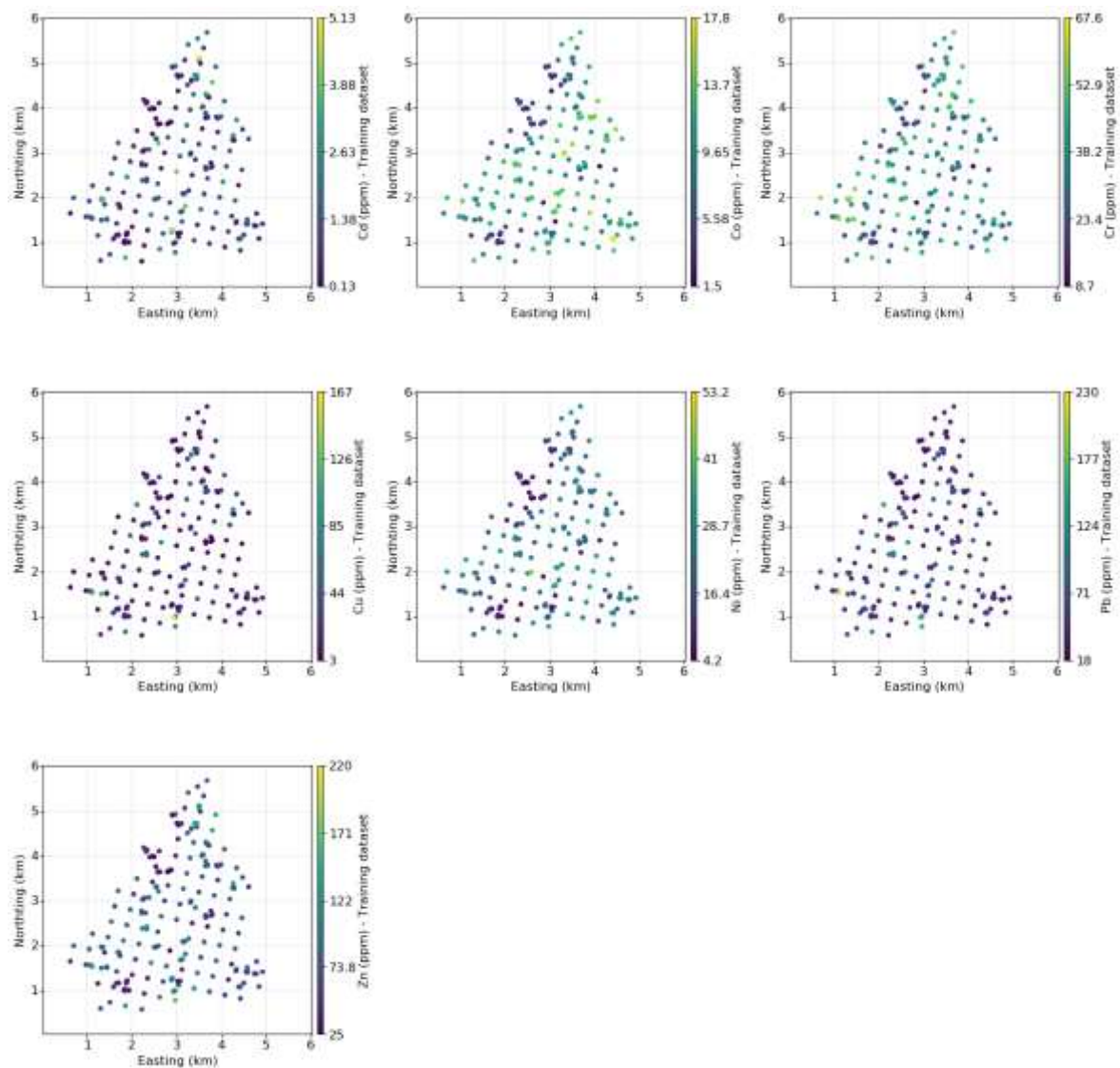


Figure 5.8. The location maps of the continuous variables in the Jura training dataset

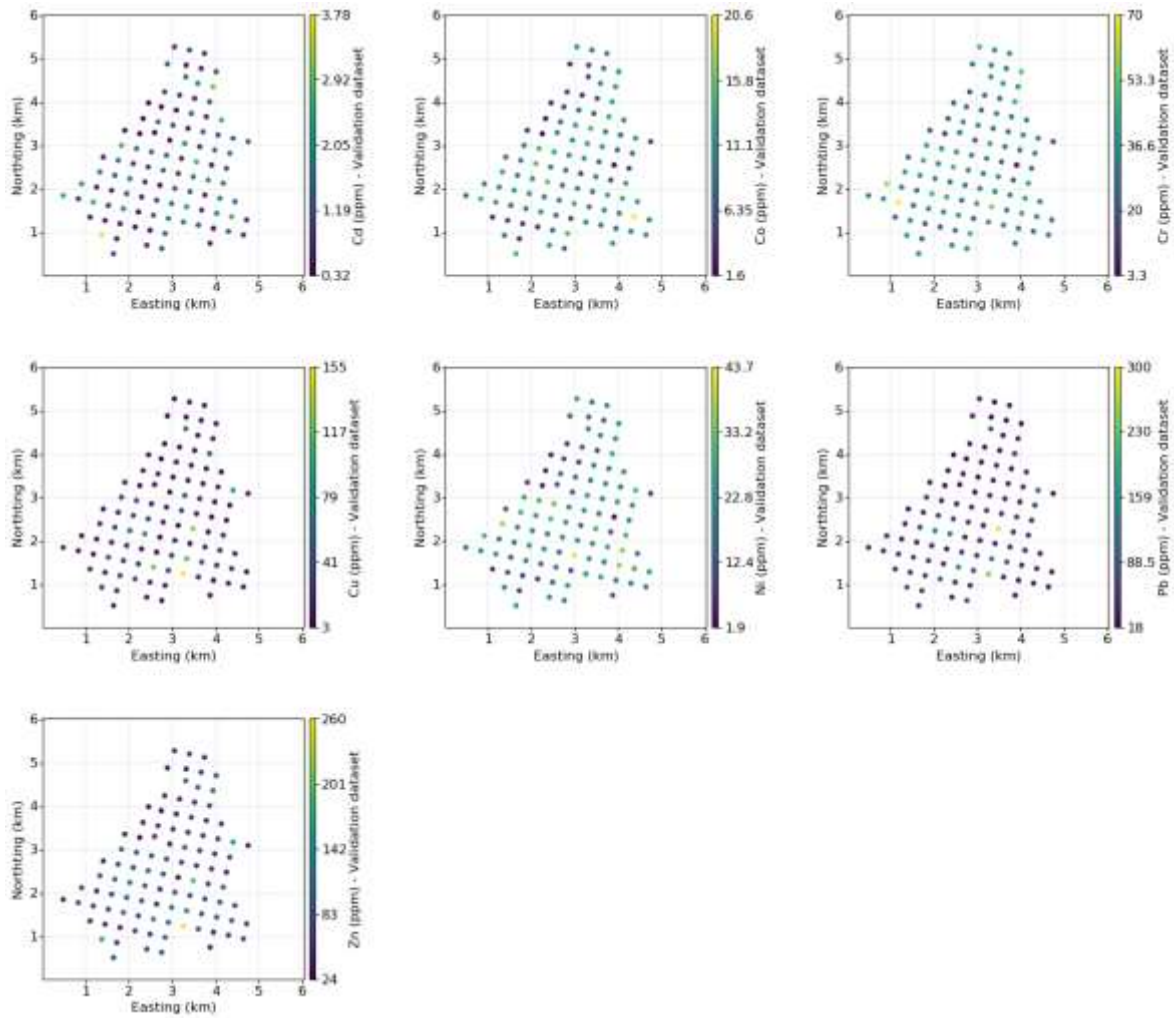


Figure 5.9. The location maps of the continuous variables in the Jura validation dataset

As can be seen in Figures 5.8 and 5.9, the training data are irregularly spaced, while the validation data are sampled at regular intervals. The summary statistics and histograms of the training and validation datasets are shown in Figures 5.10 and 5.11, respectively.

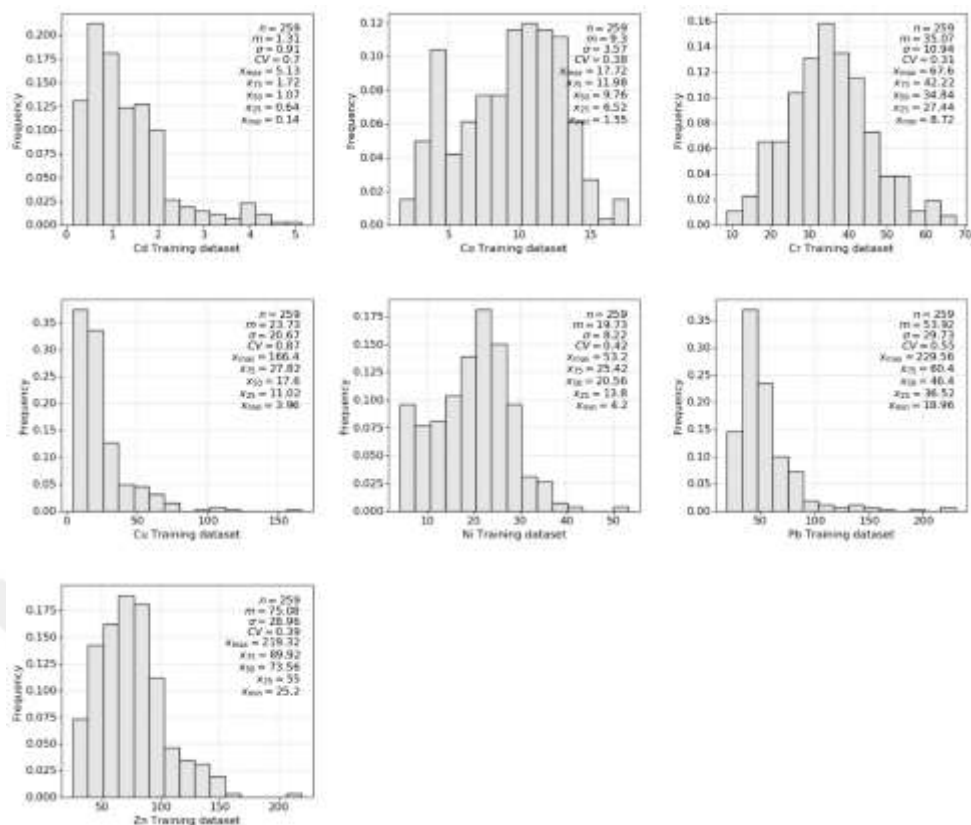


Figure 5.10. The histograms of the continuous variables in the Jura training dataset

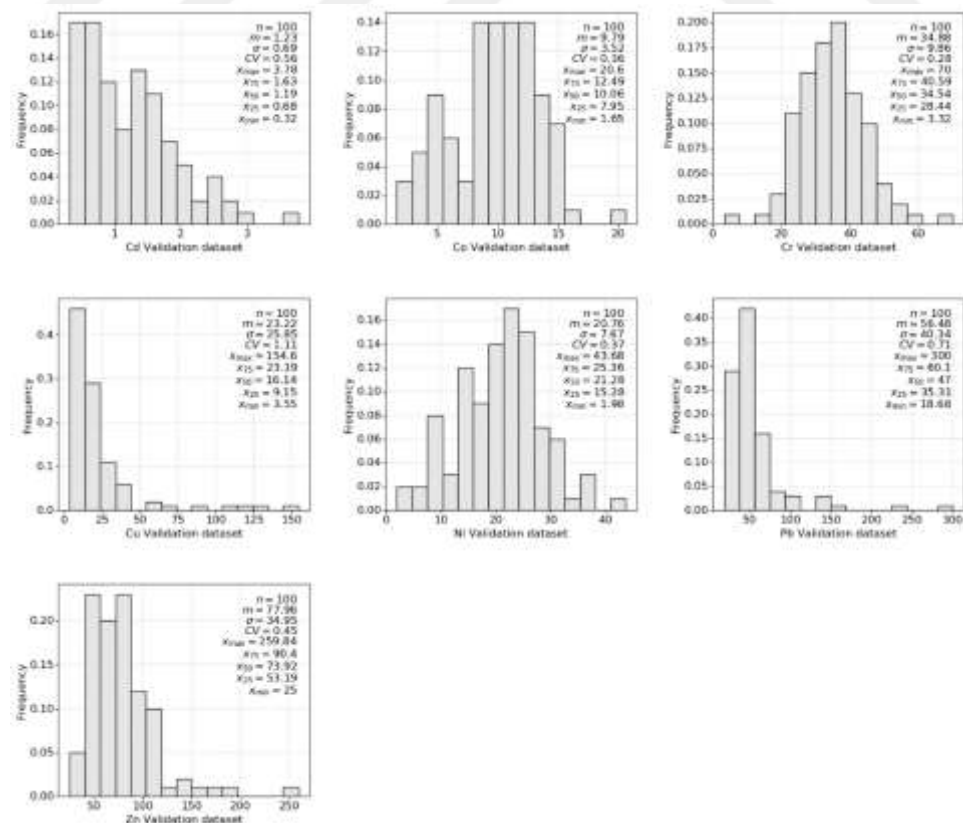


Figure 5.11. The histograms of the continuous variables in the Jura validation dataset

Figures 5.10 and 5.11 demonstrate that the variables have a variety of distributions and skewness. Moreover, it is necessary to check the distribution of each subset for each variable whether they are consistent and representative of the entire dataset (Leuenberger, 2018). In this regard, it is observed that there is a statistical similarity in terms of frequency distributions and basic statistics between training and validation datasets. The training dataset has been also examined in terms of the spatial continuity of each variable. The omnidirectional variograms of the continuous variables in the training dataset are illustrated in Figure 5.12.

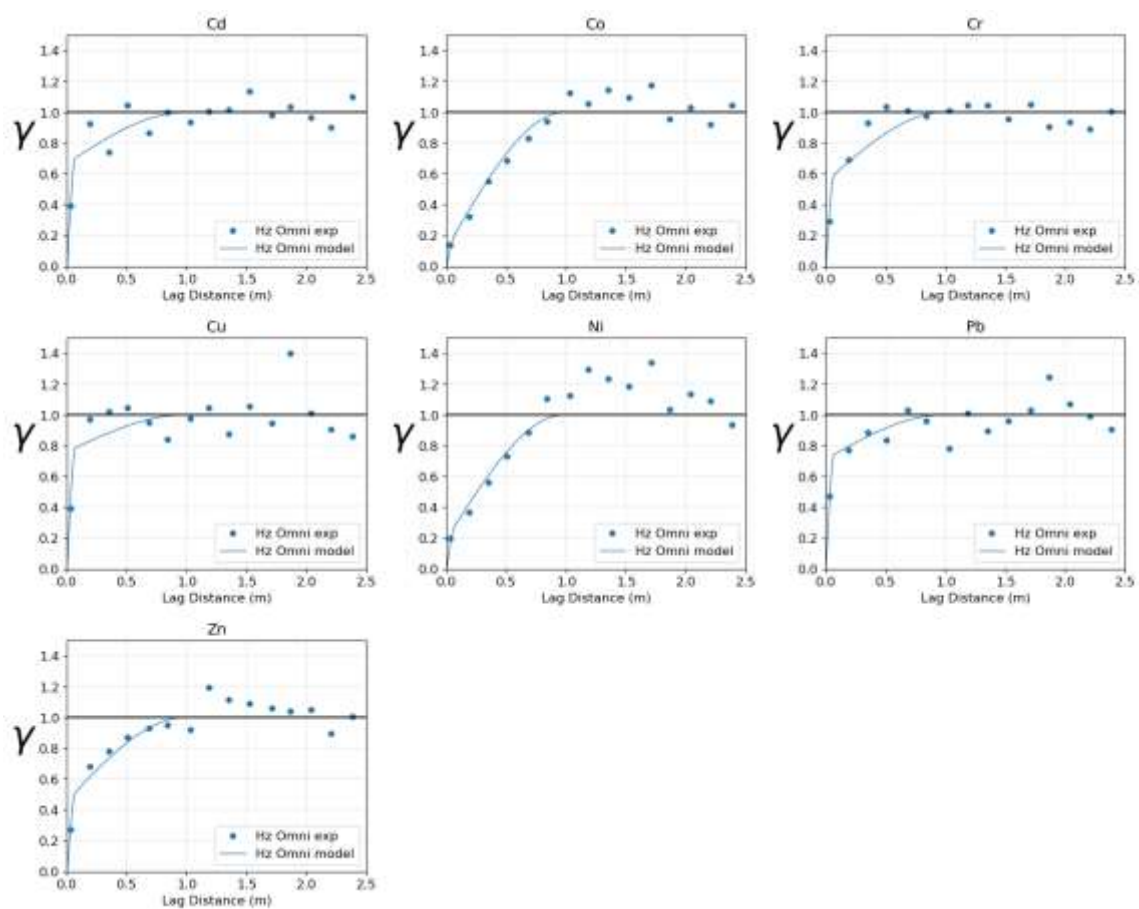


Figure 5.12. The omnidirectional variograms of the continuous variables in the Jura training dataset

As can be seen from Figure 5.12 that the variables have a variety of spatial continuity. In this thesis, all continuous variables (Cd, Co, Cr, Cu, Ni, Pb, and Zn) have been evaluated to examine the performance of the ESL model in different types of distributions and spatial continuity.

OK, SL, and ESL models have made their estimations on a regular grid of 120m×120m. The target variables in this dataset have also been normalized to process the ML models easier, resulting in better precision. The estimates have been generated with the OK model using the variogram model given in Figure 5.12. All procedures to generate estimates with SL and ESL models are the same as the Walker Lake dataset. The individual SL models have been fitted on the 18 rotated training datasets as α has been selected as 5°. Each fitted model has been used to estimate corresponding unknown grid nodes and their estimations have been ensembled by averaging. SL models have produced their estimates using the best hyper-parameters given in Table 5.3. As the hyper-parameters are data-specific, some of the hyper-parameters appear to be different from each other and the previous case study.

Table 5.3. The best configuration of hyper-parameters for the continuous variables in the Jura dataset.

ML model	Set of Hyper-parameters	Cd	Co	Cr	Cu	Ni	Pb	Zn
kNN	n_neighbors	5	5	2	4	2	2	2
DT	max_depth	2	4	6	2	4	8	6
BAG	n_estimators	1000	100	1000	750	100	250	250
RF	n_estimators	500	100	1000	500	250	250	750
EXT	n_estimators	500	1000	1100	750	750	250	100
ADA	n_estimators	250	1000	750	250	750	750	250
GBM	learning_rate	0.01	0.01	0.01	0.01	0.01	0.01	0.01
	n_estimators	4	4	6	4	4	8	6
	max_dept	250	500	250	250	500	500	250
SVR	C	10	10	10	10	10	100	10
	epsilon	0.1	0.1	0.1	0.1	0.1	0.1	0.1
NN	hidden_layer_sizes	2	2	3	4	2	2	2
	neurons	10	25	5	25	25	20	5
	batch_size	80	10	100	10	20	20	80
	epoch	50	100	100	100	100	100	50

The estimation maps generated by OK, SL, and ESL models are presented in Figures 5.13 - 5.15, respectively. As can be seen from Figure 5.14, the SL model shows significant artifacts in its estimations for each variable. On the other hand, the ESL model has managed these artifacts and smoothed out the estimations. For example, blocky artifacts in the Ni estimation map produced by the SL model have been removed by the ESL model. Generally, ESL model has yielded similar estimation maps to the ones generated by the OK model.

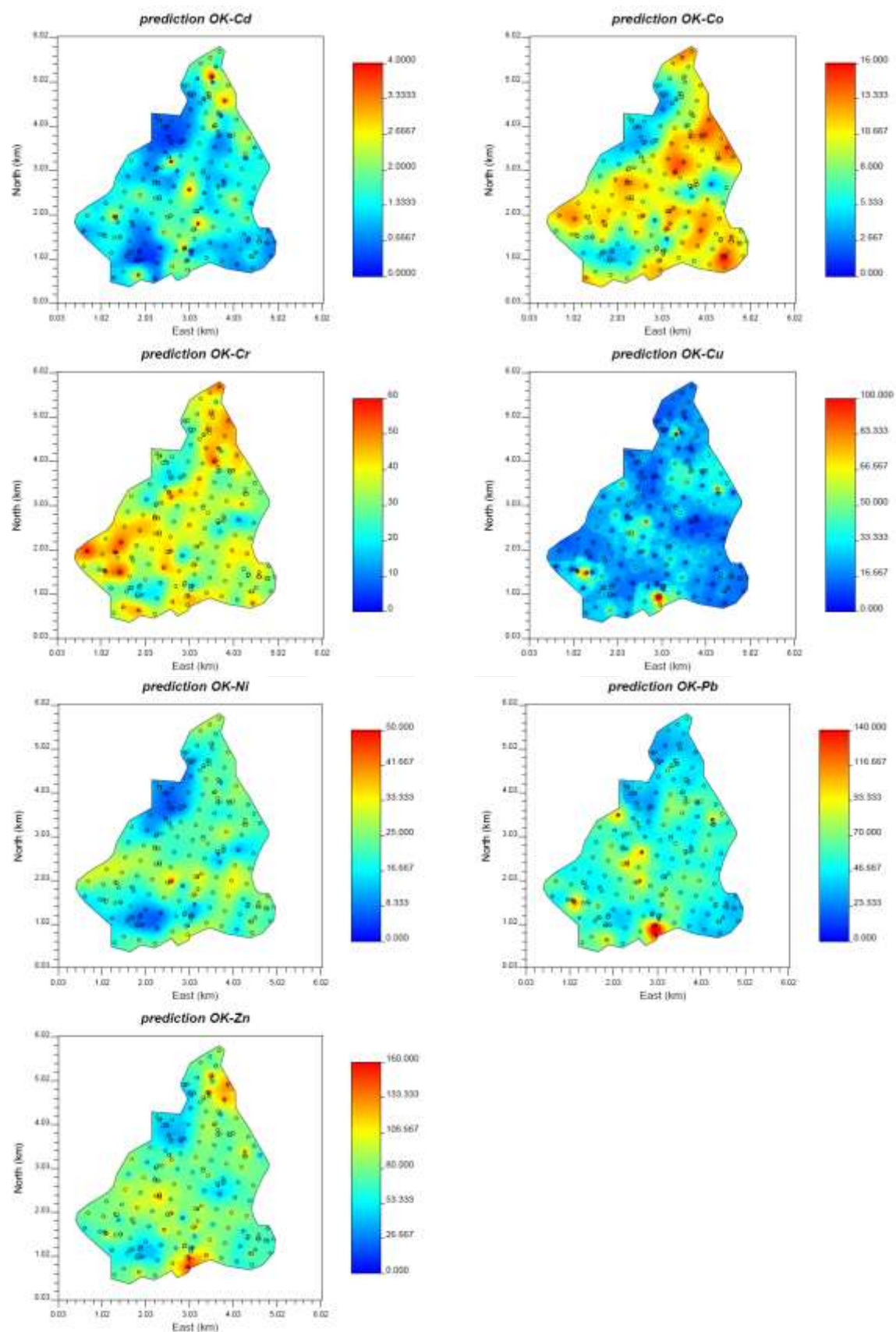


Figure 5.13. OK model spatial estimation maps of the continuous variables in the Jura dataset

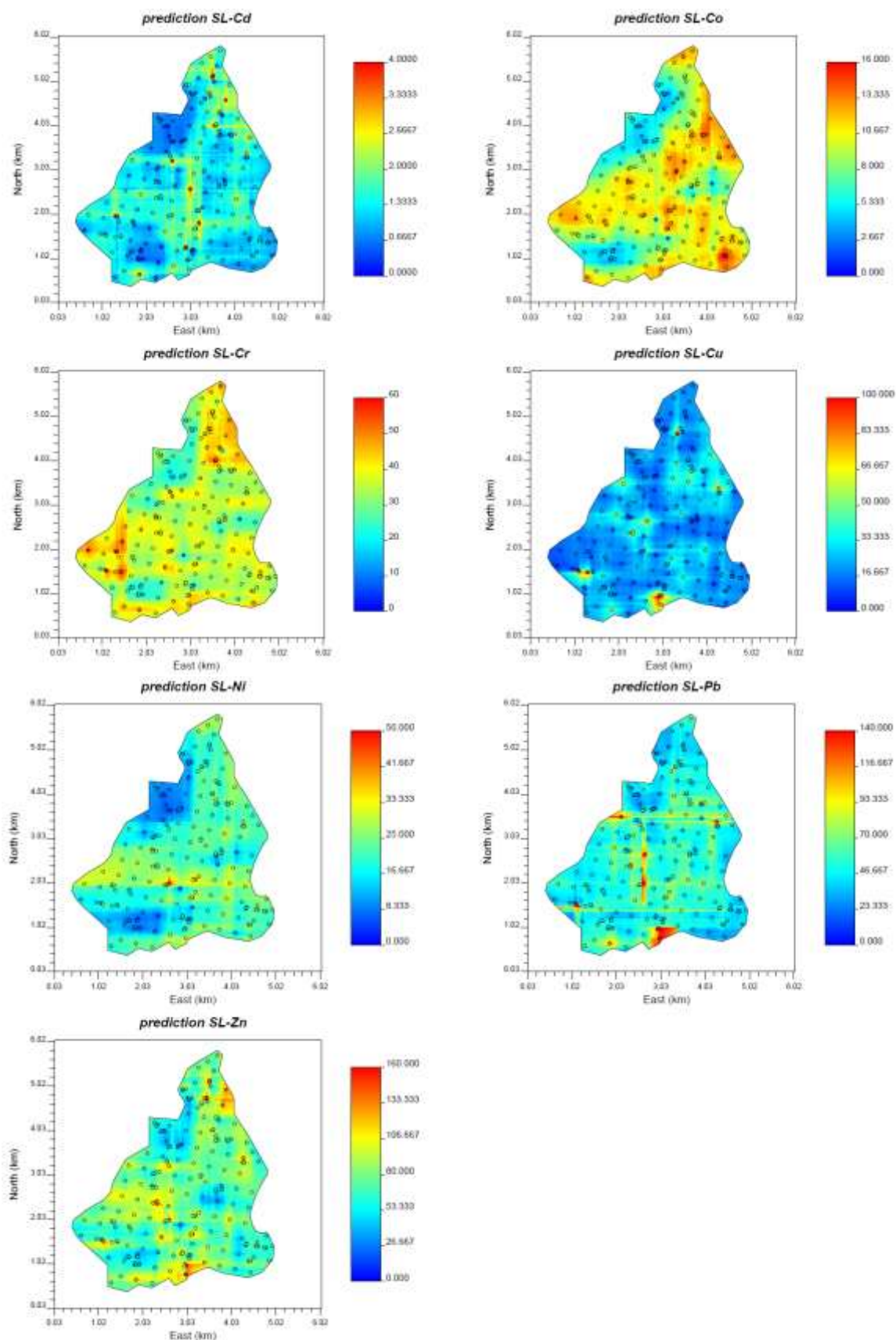


Figure 5.14. SL model spatial estimation maps of the continuous variables in the Jura dataset

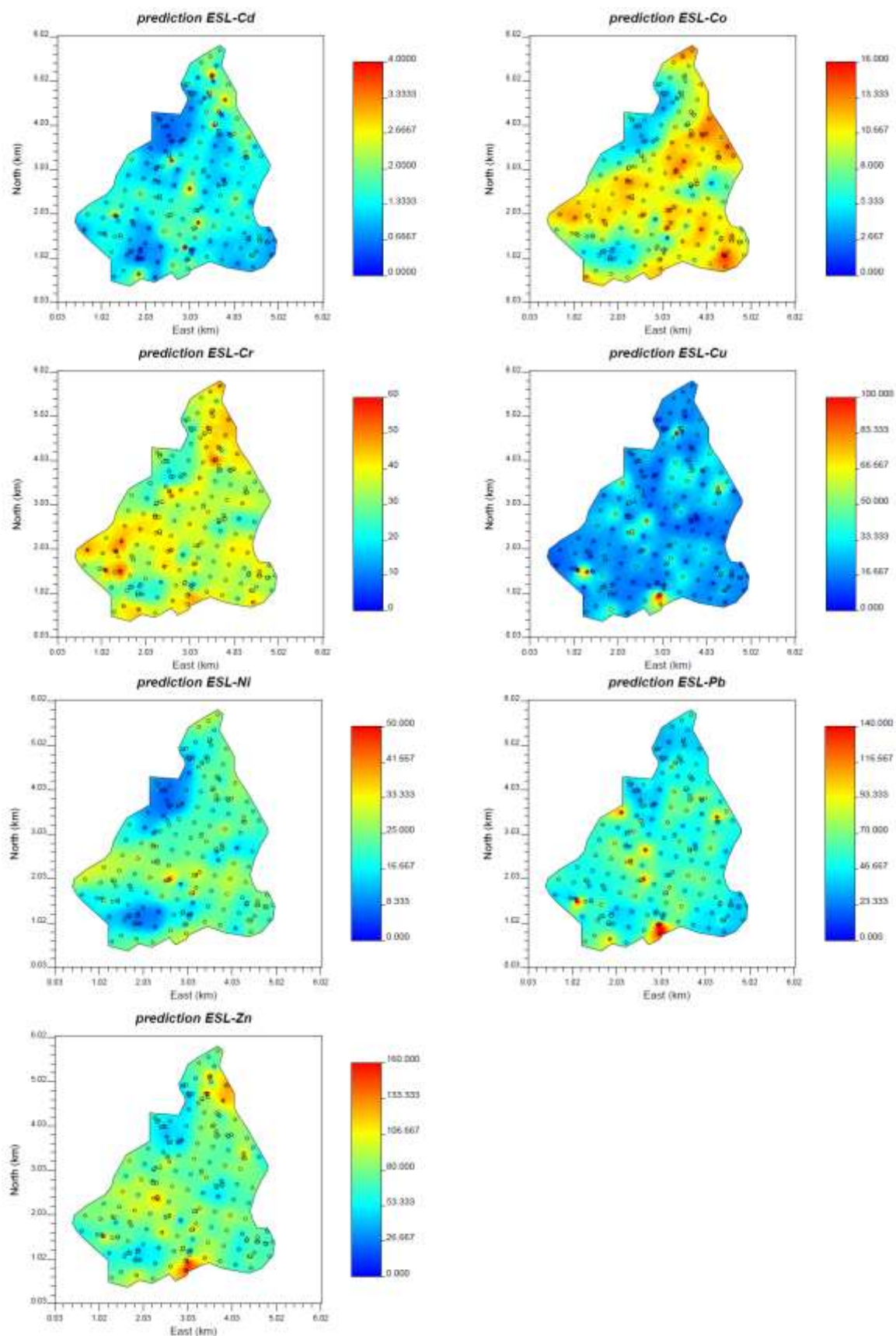


Figure 5.15. ESL model spatial estimation maps of the continuous variables in the Jura dataset

It has been also measured the performances of OK, SL, and ESL models on the validation dataset in terms of R-squared, RMSE, and MAE. The results are demonstrated in Table 5.4. The estimation performance of the ESL model is much higher than the SL model. On the other hand, the ESL model produces as good estimations as the OK model and does so consistently for all target variables. The R-squared values are negative for Cd and Cu in all models. The R-squared is not a mathematical impossibility or the sign of a computer bug. It simply means that the chosen model fits the data really poorly (Cameron and Windmeijer, 1996). Thus, it can be concluded that none of the models are good enough in terms of R-squared.

Table 5.4. The comparison of general metrics of OK, SL, and ESL models for continuous variables in the Jura dataset.

	R-Squared			RMSE			MAE		
	OK	SL	ESL	OK	SL	ESL	OK	SL	ESL
Cd	-0.338	-0.344	-0.136	0.799	0.801	0.736	0.622	0.644	0.584
Co	0.480	0.391	0.487	2.541	2.750	2.523	1.972	2.111	1.975
Cr	0.112	0.077	0.150	9.290	9.470	9.091	7.205	7.140	6.911
Cu	-0.118	-0.063	-0.039	27.336	26.650	26.353	16.388	16.369	16.059
Ni	0.313	0.208	0.326	6.363	6.831	6.302	4.983	5.510	4.830
Pb	0.022	0.018	0.061	39.882	39.979	39.095	22.196	22.723	21.746
Zn	0.044	0.033	0.064	34.182	34.375	33.811	22.262	23.108	22.149

The basic statistics which are the mean value and the st.dev. of the truth and estimations of OK, SL, and ESL models on the validation dataset are summarized in Table 5.5. It can be said that all models have slightly overestimated the mean.

Table 5.5. The basic statistics of OK, SL, and ESL models for continuous variables in the Jura dataset.

	mean						
	Cd	Co	Cr	Cu	Ni	Pb	Zn
OK	1.361	9.445	35.394	24.574	20.722	55.563	76.514
SL	1.423	9.506	35.506	24.634	20.658	54.811	76.599
ESL	1.353	9.463	35.47	24.222	20.627	55.864	77.499
Truth	1.309	9.303	35.07	23.727	19.73	53.917	75.078
	st.dev.						
	Cd	Co	Cr	Cu	Ni	Pb	Zn
OK	0.487	2.419	6.378	10.027	5.651	13.872	15.166
SL	0.407	2.229	5.408	7.852	5.361	12.37	15.24
ESL	0.335	2.18	5.304	7.561	5.192	11.472	13.559
Truth	0.691	3.524	9.86	25.848	7.675	40.337	34.951

The presence of remaining information in the residuals is also assessed as it gives a good measure of the quality of the results. The residuals have been calculated from the difference of original data values from the estimation of the ESL model on the validation dataset. The omnidirectional variograms of the raw data and the residuals for each variable are examined in Figure 5.16.

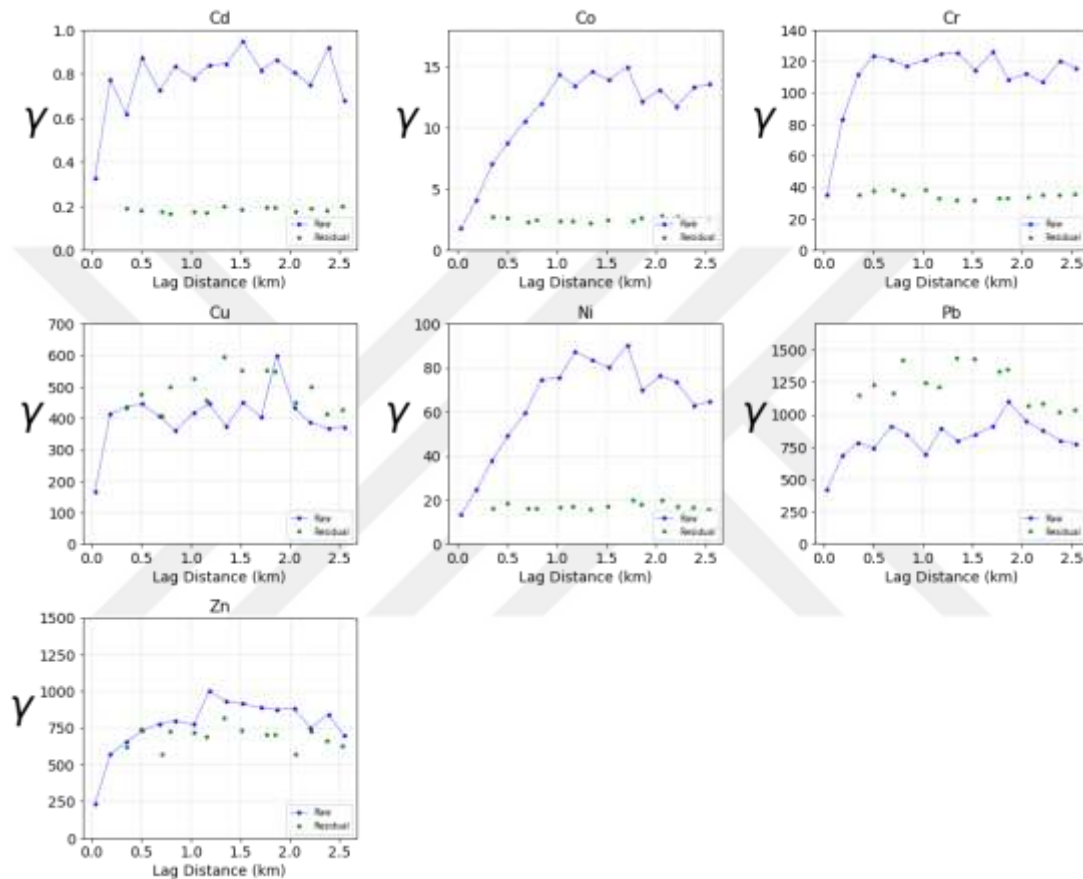


Figure 5.16. The omnidirectional variogram for both raw data and the residuals from the ESL model in the Jura dataset

The residual variograms of the variables including Cd, Co, Cr, and Ni demonstrate the nugget effect, which proves that the ESL model has explained the structure satisfactorily for these variables (Hengl et al., 2018; Leuenberger and Kanevski, 2015). Although there is no clear spatial autocorrelation in the experimental variogram of residuals for the variables including Cu, Pb, and Zn, their sill values are not close to the nugget in raw data. This result is related to the high RMSE values of the estimations of these variables.

5.3. Case Study – 2: Merging Machine Learning and Kriging Estimates

The estimation performance of the SK model (as a baseline kriging model), SL model, and combinations of the SK and SL models (SK+SL) are evaluated for different conditions including nonstationary, non-Gaussian, and Gaussian cases. These case studies are based on 3-D synthetic datasets which have been generated through simulations with known characteristics. Synthetic datasets are chosen as they provide the exhaustive (or underlying reference) data that is used to compare the estimates of the different models directly.

The first case study, which is based on a synthetic dataset created by Samson (2019), demonstrates estimation performance in situations that have complex non-stationary features. The second case study uses simulated data that have a trend, which is also non-Gaussian. The third case study is implemented to demonstrate the performance of the models on highly skewed (i.e. non-Gaussian) lognormal data. The last case study uses another synthetic dataset that is simulated with a Gaussian variogram model. This case study aims to evaluate the performance of the models on the Gaussian distributed data.

The estimation performance of the SK, SL, and SK + SL models are compared with the underlying reference data. In this section, all sample data are used for training because there are gridded exhaustive data to validate the results. When comparing the performance of the SK + SL model vs. SK and SL models; R-squared, RMSE, and MAE are used as a measure of the accuracy of estimation. Moreover, the performance of the proposed method is tested by checking the presence of the spatial structure on the residuals. The estimation maps are also examined.

5.3.1. Non-stationary synthetic data example

The exhaustive non-stationary dataset is generated on a regular grid of $10\text{ m} \times 10\text{ m} \times 10\text{ m}$ over a spatial domain of $500\text{ m} \times 500\text{ m} \times 250\text{ m}$. The exhaustive simulated dataset is represented by 62,500 grid nodes. To generate data values at every grid node, firstly two circles with a radius of 245 m and 95 m have been defined on the simulated grid. Then, the area between these two circles has been assigned values based on the polynomial function

of its spatial locations. Figure 5.17 (a) presents the simulated data that resembles a ring shape where the non-stationarity is reflected in the spatial variation of the data. The simulated data has been sampled at a spacing of 16% of the total distances in the X and Y directions (Samson 2019). Thus, 36 drill holes with 864 samples have been obtained and used as a training dataset while the rest are used for the model validation. The reference realization and sampling location maps are illustrated in Figure 5.17. The histograms for the simulated and sampling data are given in Figure 5.18.

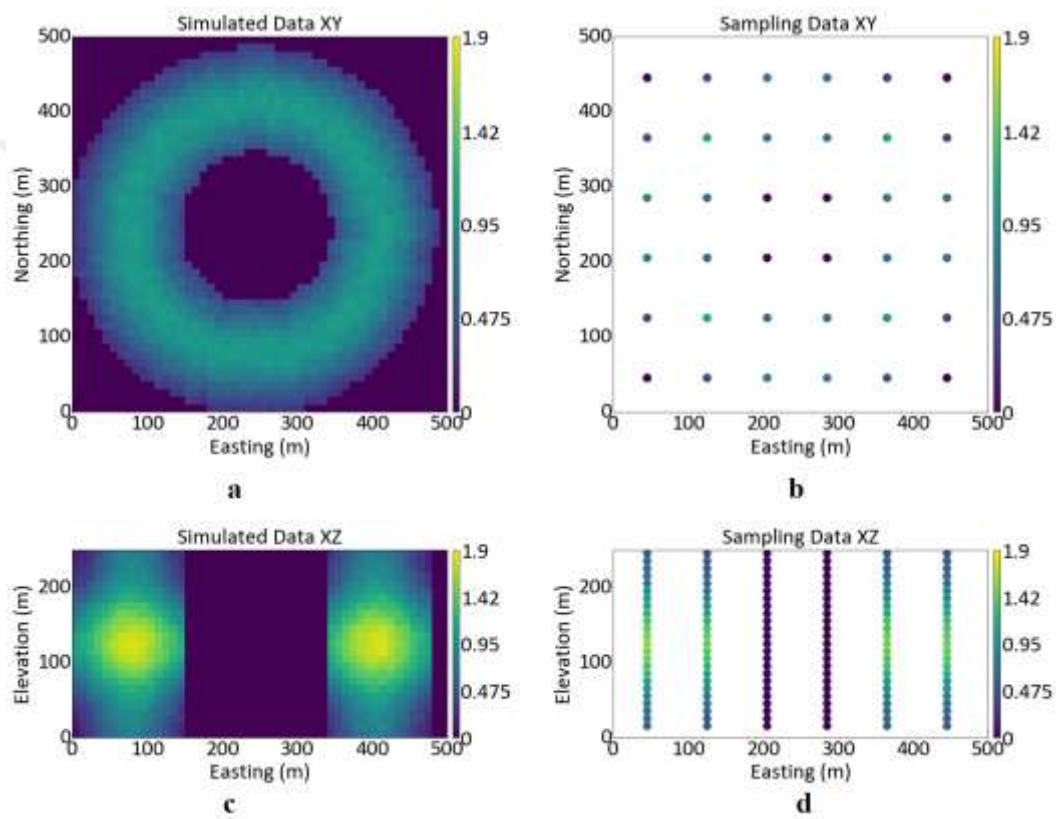


Figure 5.17. (a) Reference realization plan map, (b) sampling plan map at the elevation of 190 m, (c) reference realization, and (d) sampling E-W section map looking from North (non-stationary synthetic data).

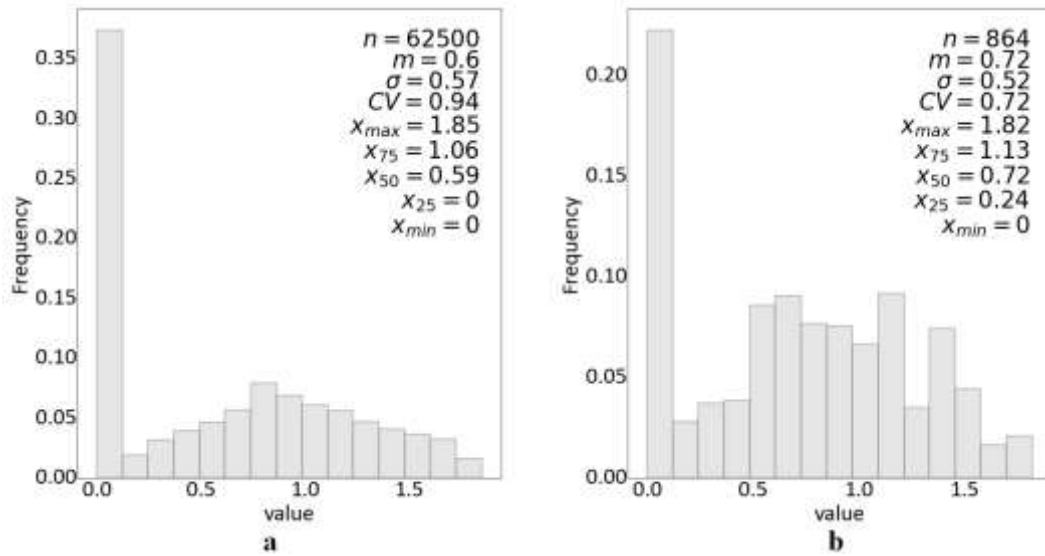


Figure 5.18. Histogram of (a) reference realization and (b) sampling data (non-stationary synthetic data).

The distribution of data from Figure 5.18 illustrates that 0 is the widespread value. The analysis of the variogram map (Figure 5.19) suggests the presence of complex anisotropy indicating non-stationarity on the simulated domain.

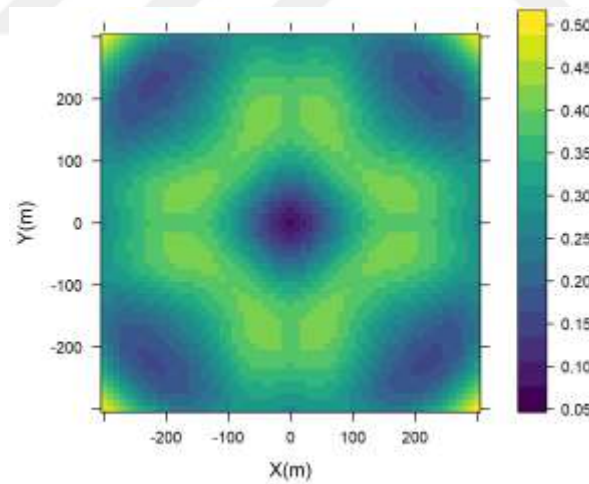


Figure 5.19. Variogram map of the target value for non-stationary synthetic data

To generate estimates by the SL model, the orthogonal coordinates (X, Y, and Z) are used as input variables and their values are used as the target variable. Considering that the different scales of the raw data may negatively impact the performance of some ML models, SK and SL models have been evaluated with normalized values.

To train the SL model, hyper-parameter tuning has been performed using the grid search algorithm with 5-fold cross-validation. The best configuration of the hyper-parameters for each ML model is contained in Table 5.6. The base ML models with the corresponding best hyper-parameter configurations and the meta-model, which is the linear regression, have been stacked through the SL model. The estimates have been generated with the SK and SL models on each grid node using the sample data with the associated variogram model whose parameters are given in Equation 5.2. Finally, the estimation results of both models have been combined according to the proposed weighting procedure based on σ_{SK}^2 . The parameters of weight, b_0 , and b_1 , have been optimized by minimizing the RMSE with SQP.

Table 5.6. The best configuration of hyper-parameters for the nonstationary synthetic data

ML model	Set of Hyper-parameters
kNN	n_neighbors: 2
DT	max_depth: 14
BAG	n_estimators: 3000
RF	n_estimators: 2000
EXT	n_estimators: 2500
ADA	n_estimators: 100
GBM	learning_rate: 0.15, n_estimators: 1500, max_depth = 4
SVR	C:1e4, epsilon: 0.01
NN	hidden_layer_sizes: 2, neurons: 20, batch_size: 10, epoch: 500

$$\gamma(\mathbf{h}) = 0.01_{Nugget} + 0.99Gaus(\mathbf{h}) \begin{matrix} a_{major}=132m \\ a_{minor}=175m \\ a_{vertical}=173m \end{matrix} \quad (5.2)$$

The histograms and the spatial maps of the weights for SK and SL models are shown in Figures 5.20 (a) and 5.20 (b), respectively.

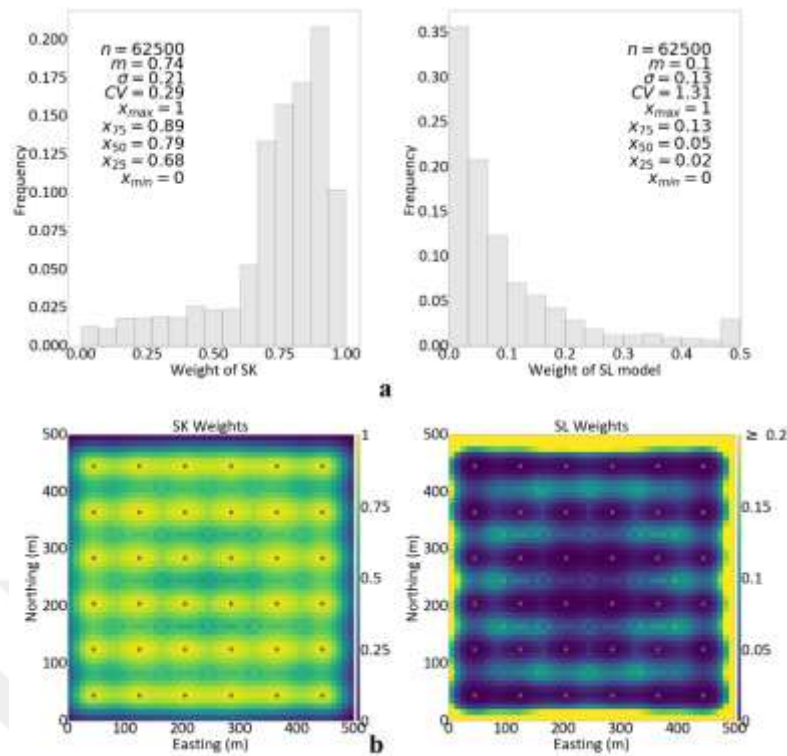


Figure 5.20. (a) The distribution and (b) plan maps of the weights at an elevation of 190 m for SK and the SL models for the nonstationary synthetic data

In Figure 5.20 (a), the histograms clearly indicate how the weights change according to SK and SL models. It appears that the mean of the SK model is larger than that of the SL model; therefore, it can be concluded that kriging is more important during the combination stage. On the other hand, the spatial maps of the weights in Figure 5.20 (b) indicate that the grid nodes close to the data locations, the SK model receives more weight, whereas, at the grid nodes away from data locations, the SL model receives more weight. To verify the solution obtained from the optimization, the contours of the objective function have been plotted. The illustration of objective contours and the feasible solution indicated by the red dot for the nonstationary dataset case are presented in Figure 5.21.

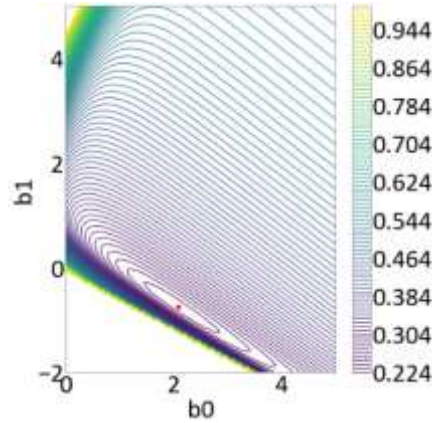


Figure 5.21. Illustration of objective contours and the feasible solution of parameters of weight for the nonstationary synthetic data.

It is clear from Figure 5.21 that $b_0 = 2.08$ and $b_1 = -0.74$ show a clear optimum. It means that QP subproblems in SQP ensure convexity. In this case, b_0 has positively affected the power of weight, while b_1 that determines the coefficient of the kriging estimation ($Z_{KR}^*(\mathbf{u})$) in each grid node has negatively affected the power of weight. Figure 5.22 compares the estimation maps obtained from the SK, SL models, and combinations of SK and SL models (SK+SL). All models indicate similar estimation maps that contain some degree of artifacts.

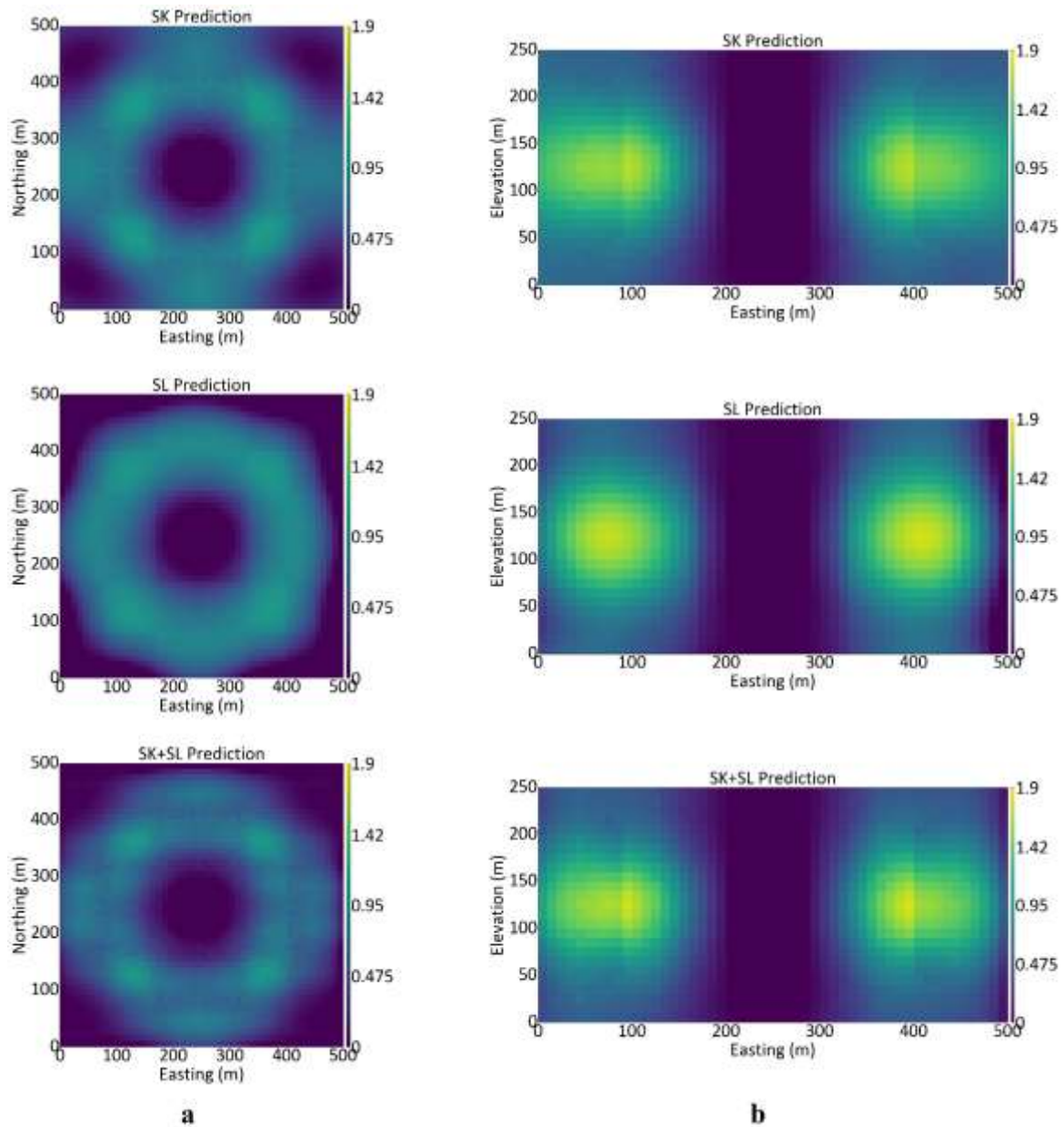


Figure 5.22. Spatial estimation maps of all models in (a) plan view at an elevation of 190 m and (b) E-W section looking from North for the nonstationary synthetic data.

Figure 5.23 and Table 5.7 indicate the estimation performance results of all models considering different criteria. It appears that the SL model produces better estimates than those obtained by the SK model according to R-squared, RMSE, and MAE criteria. This result is reasonable, as kriging is not expected to be optimal when the dataset is nonstationary (Samson, 2019). Moreover, combining these two models allow for data reproduction and capturing nonstationary features, and the SK+SL model has given the best performance of all methods considering the comparison criteria (Table 5.7). In addition, it is clear from Figure 5.23 that the scatter plot of the SK+SL model against the true data shows

slight curvature. It is uncommon that it would be a trade-off bias for improved error. Having said that, from a historical perspective, has been observed in statistical estimation. Basic statistics reproduction such as mean values and st.dev. have also been examined to evaluate the models. It is expected to these statistics are similar considering the data and estimates. Table 5.7 compares the mean and st.dev. of all estimation models and of exhaustive data. The mean is overestimated by the SK model while the SK+SL model produces the most similar mean value with the true data.

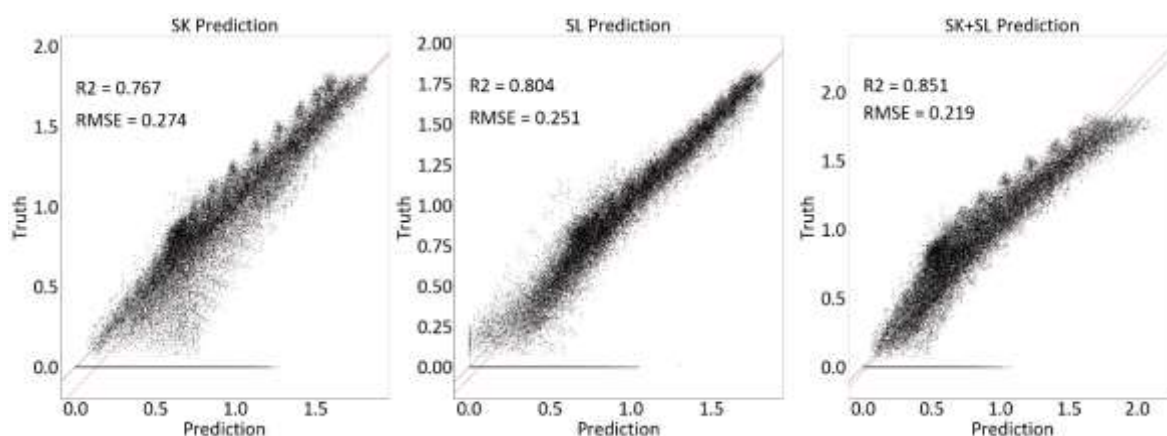


Figure 5.23. Estimation performance of all models for the nonstationary synthetic data.

Table 5.7. The comparison of general metrics and the basic statistics of all models for the nonstationary synthetic data

Model	R-Squared	RMSE	MAE	mean	st.dev.
SK	0.767	0.274	0.182	0.699	0.477
SL	0.804	0.251	0.161	0.643	0.517
SK+SL	0.851	0.219	0.159	0.608	0.502
Truth	-	-	-	0.604	0.567

The 3-D omnidirectional variogram for both the true data and the residuals of the SK+SL model is given in Figure 5.24. The residual variogram does not demonstrate the spatial structure and its sill value is close to the raw data's nugget value, which proves that there is no local spatial bias in estimation.

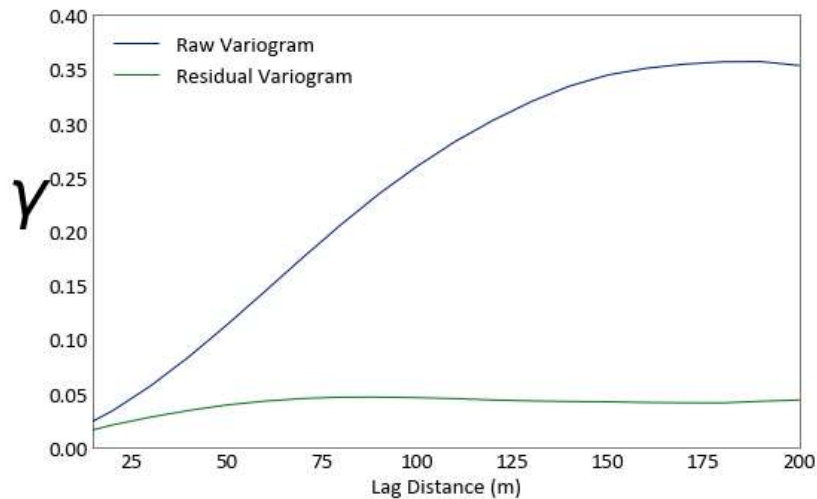


Figure 5.24. Omnidirectional variogram of raw data and SK+SL model residuals for the nonstationary synthetic data.

Figure 5.25 demonstrates the data reproduction for SK, SL, and SK+SL models, respectively. Although the SL model could not reproduce the data exactly, the combination strategy (SK+SL model) allows for data reproduction at the known locations.

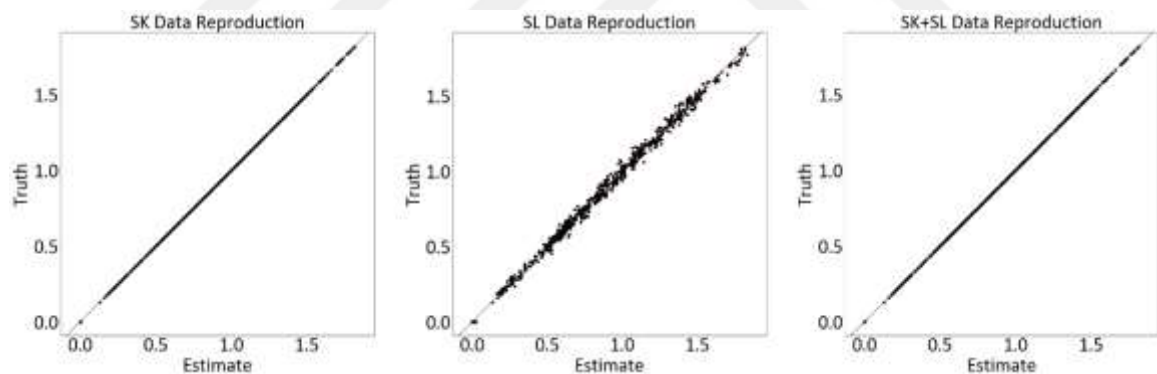


Figure 5.25. Data reproduction for SK, SL, and SK+SL models

5.3.2. Non-Gaussian synthetic data example

The exhaustive non-Gaussian dataset with a trend is generated on the same simulated domain as the previous case. For this case, data values are generated by SGS where an unconditional CDF is obtained by the SK model at every grid node. Then, its distribution is converted to the lognormal distribution where the mean and the st.dev. are equal to 1 and 0.5, respectively. Finally, a trend has been imposed by using the moving window averaging method that computes the three-dimensional trend values using ellipsoid-shaped averaging volumes (Manchuk and Deutsch, 2011). In this method, samples in the windows are

weighted by their distance from the window center using equal weighting scheme. The location map of the exhaustive dataset for this case is shown in Figure 5.26 (a) and (c). A subset from the exhaustive dataset has been chosen at a spacing of 25% of the total distances in the X and Y directions as the sample data that is shown in Figure 5.26 (b) and 4.26 (d) with 16 drill holes and 384 samples.

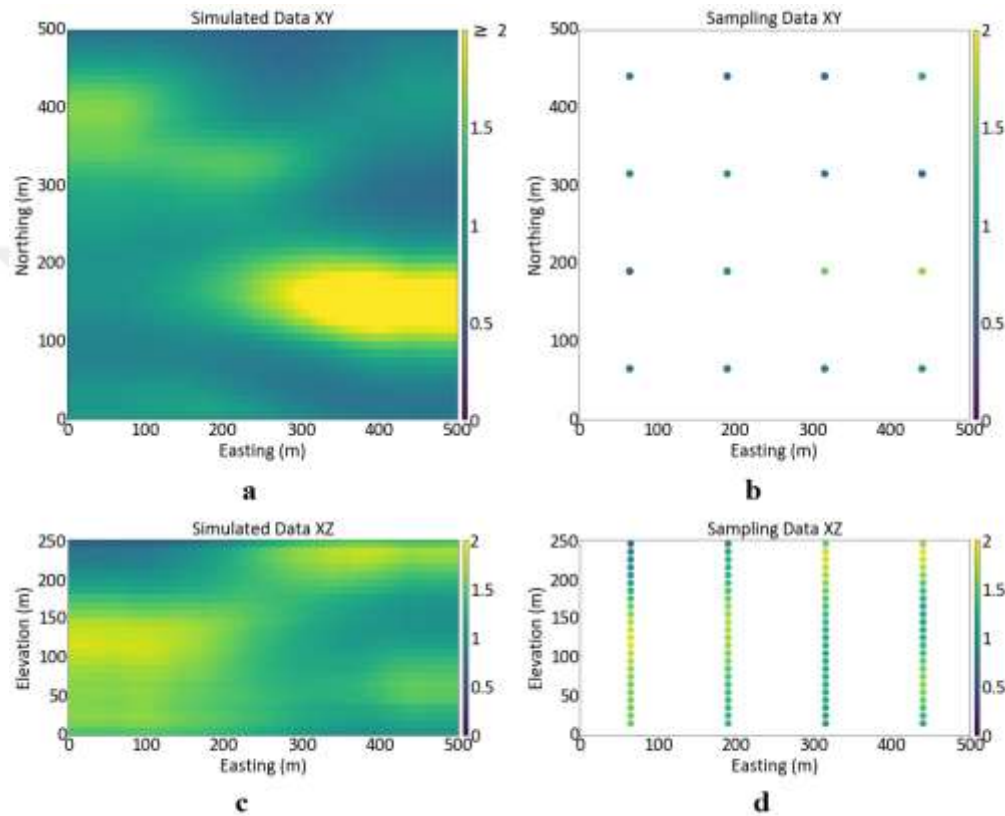


Figure 5.26. (a) Reference realization plan map, (b) sampling plan map at an elevation of 190 m, (c) reference realization, and (d) sampling E-W section map looking from North (non-Gaussian synthetic data).

The histograms of the exhaustive and sample data (Figure 5.27) illustrate that the distribution deviates from Gaussianity (the p-value is less than 0.05). All other procedures to generate estimates with defined models are the same as the ones used in the nonstationary synthetic data example given in Section 5.3.1.

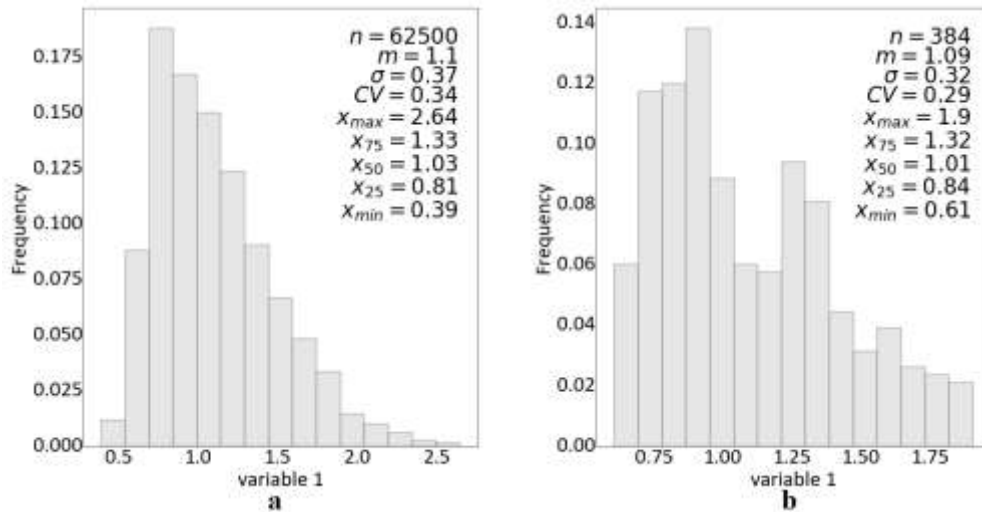


Figure 5.27. Histogram of (a) reference realization (b) sampling data (non-Gaussian synthetic data).

For this case, the best configuration of the hyper-parameters that has been obtained from the grid search algorithm with 5-fold cross-validation is shown in Table 5.8. As the hyper-parameters are data-specific, they could take different values. Therefore, some of the hyper-parameters appear to be different from the ones used in the previous case study.

Table 5.8. The best configuration of hyper-parameters for the non-Gaussian synthetic data

ML model	Set of Hyper-parameters
kNN	n_neighbors: 2
DT	max_depth: 14
BAG	n_estimators: 1200
RF	n_estimators: 1200
EXT	n_estimators: 1400
ADA	n_estimators: 100
GBM	learning_rate: 0.2, n_estimators: 1000, max_depth:4
SVR	C: 1e3, epsilon: 0.01
NN	hidden_layer_sizes: 3, neurons: 20, batch_size: 10, epoch: 500

The estimates are generated with the SK model using the sample data with the associated variogram model that is given in the following equation:

$$\gamma(\mathbf{h}) = 0.01_{Nugget} + 0.99Gaus(\mathbf{h}) \quad \begin{matrix} a_{major}=336m \\ a_{minor}=100m \\ a_{vertical}=150m \end{matrix} \quad (5.3)$$

The estimation map results obtained from SK, SL, and SK+SL models are shown in Figure 5.28. All models produce similar estimation maps. However, it appears that the SL model contains some degree of artifacts. On the other hand, the combined model has shown better estimation maps than that of the SL model in terms of artifacts.

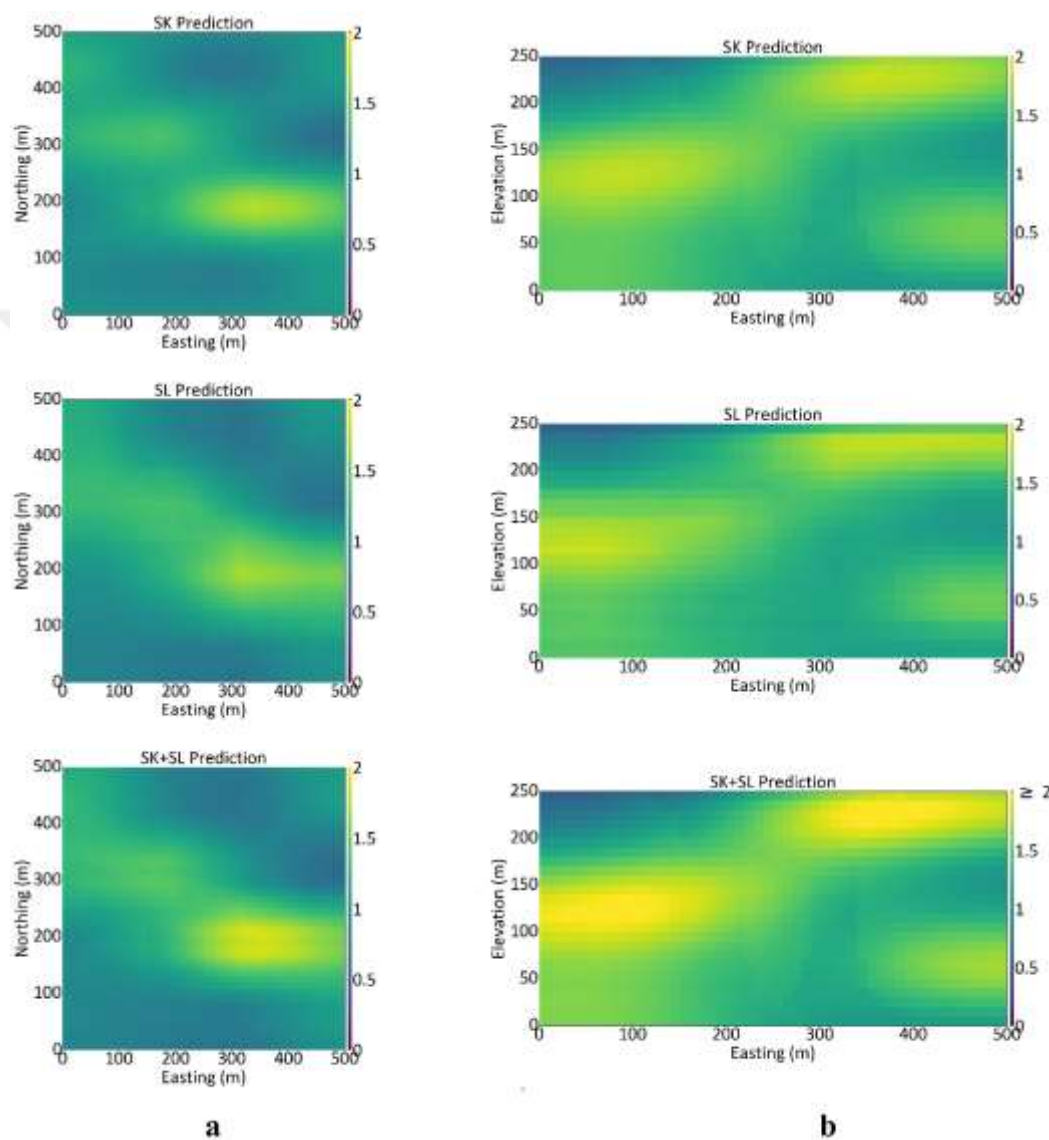


Figure 5.28. Spatial estimation maps of all models in (a) plan view at an elevation of 190 m and (b) E-W section looking from North for the non-Gaussian synthetic data

The results of the estimation performance of all models can be seen in Figure 5.29 and Table 5.9. The SL model provides more accurate estimates than the SK model according to R-squared, RMSE, and MAE criteria. The SK+SL model has produced the best performance in all the performance criteria. Table 5.9 also compares the mean and st.dev. of

all estimation models and of exhaustive data. All estimation models have reproduced the mean similarly.

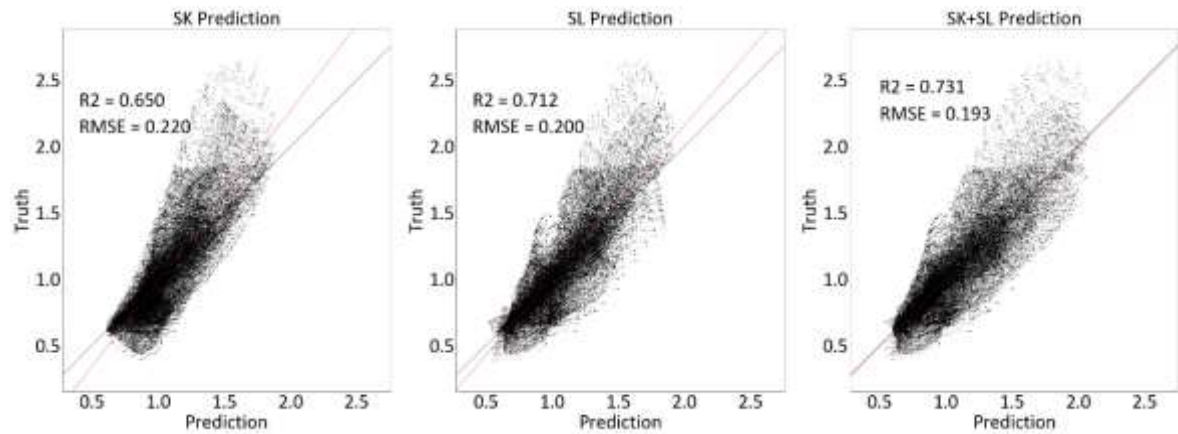


Figure 5.29. Estimation performance of all models for the non-Gaussian synthetic data

Table 5.9. The comparison of general metrics and the basic statistics of all estimation models for non-Gaussian synthetic data

Model	R-Squared	RMSE	MAE	mean	st.dev.
SK	0.650	0.220	0.158	1.109	0.240
SL	0.712	0.200	0.141	1.084	0.276
SK+SL	0.731	0.193	0.136	1.106	0.315
Truth	-	-	-	1.104	0.372

The histograms and the spatial maps of the weights for the SK and SL models can be seen in Figures 5.30 (a) and 5.30 (b), respectively. It appears that the importance of the SK and SL models in the combination is approximately the same. The trend in this dataset had been set in the east-west direction thus the map of the weights also appeared in the direction of the trend.

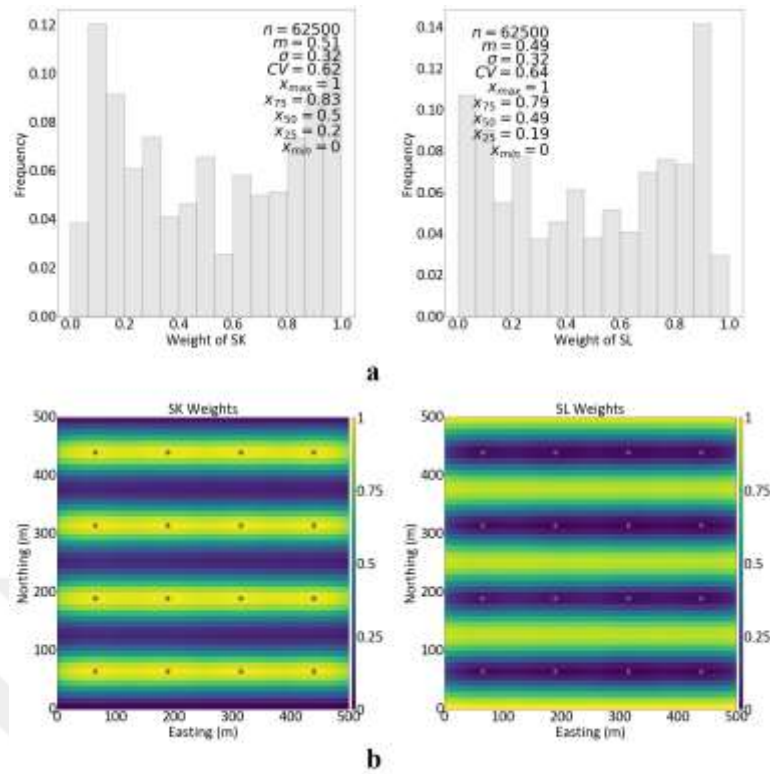


Figure 5.30. (a) The distribution and (b) plan maps of the weights at the elevation of 190 m for SK and the SL models for the non-Gaussian synthetic data

Figure 5.31 shows the diagram of objective contours and the feasible solution indicated by the red dot for the trend case dataset. A unique feasible solution of power of weight b (b_0 is 1.35 and b_1 is -0.33) could have been obtained in this case and b_0 positively affects the power of weight, while b_1 has negative effect on the power of weight.

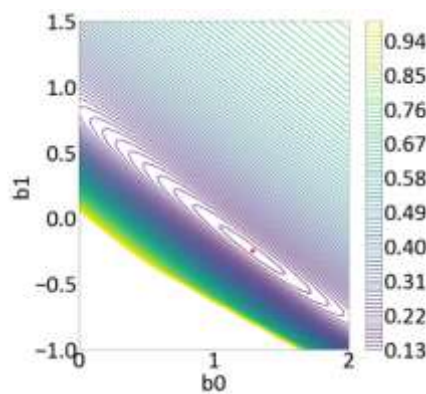


Figure 5.31. Illustration of objective contours and the feasible solution of parameters of weight for the non-Gaussian synthetic data

Finally, Figure 5.32 examines the 3-D omnidirectional variogram for both the true data and the residuals of the SK+SL model. It proves that there is no remaining spatial autocorrelation in the residuals calculated from the SK+SL model. It means that the SK+SL model properly explained the spatial structure in the data.

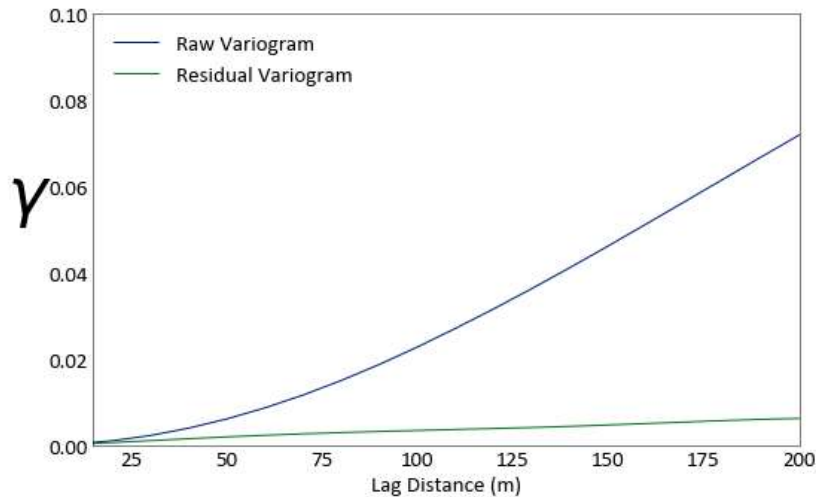


Figure 5.32. Omnidirectional variogram of raw data and SK+SL model residuals for the non-Gaussian synthetic data

5.3.3. Lognormal synthetic data example

The exhaustive lognormal dataset is simulated on a regular grid of $10 \text{ m} \times 10 \text{ m} \times 10 \text{ m}$ over a spatial domain of $500 \text{ m} \times 500 \text{ m} \times 250 \text{ m}$ like previous cases. Data values of this case are generated by the SGS where an unconditional CDF is obtained by the SK model at every grid node. Then, the simulated normally distributed data has been converted to the lognormally distributed data where the mean of 1 and a st.dev. of 2. By the st.dev. value setting high, it is aimed to measure the performance of the models in highly skewed data. The transformed data has been sampled at a spacing of 15% of the total distances in the X and Y directions. Thus, 49 drill holes with 1176 have been drilled from the simulated data and used as the training dataset for the models. Figure 5.33 demonstrates the reference realization and sampling location maps, respectively. The histograms for the simulated and sampling data are also given in Figure 5.34.

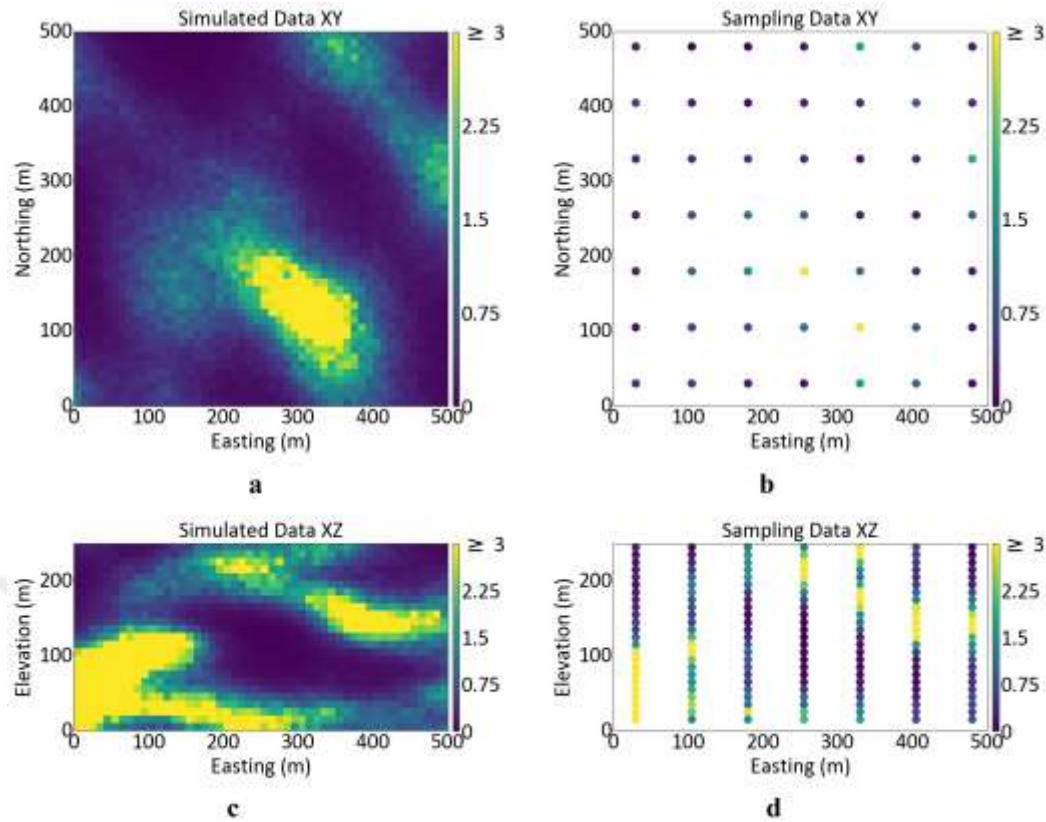


Figure 5.33. (a) Reference realization plan map, (b) sampling plan map at the elevation of 190 m, (c) reference realization, and (d) sampling E-W section map looking from North (lognormal synthetic data).

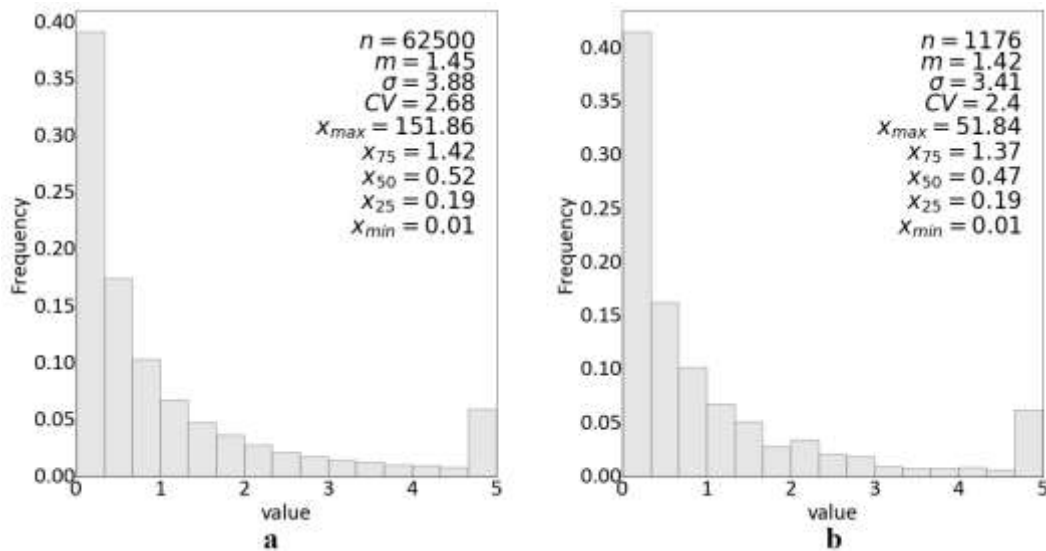


Figure 5.34. Histogram of (a) reference realization (b) sampling data (lognormal synthetic data).

It is clear from Figure 5.34 that the distributions are right-skewed, where the mean is greater than the median as the unusually high scores distort it. SK, SL, and SK+SL models have used the same procedures to generate estimates as the ones used in previous cases. The best configurations of hyper-parameters for each candidate ML model (Table 5.10) have been tuned by the grid search algorithm with 5-fold cross-validation to train the SL model.

Table 5.10. The best configuration of hyper-parameters for the lognormal synthetic data

ML model	Set of Hyper-parameters
kNN	n_neighbors: 8
DT	max_depth: 6
BAG	n_estimators: 2500
RF	n_estimators: 2500
EXT	n_estimators: 2500
ADA	n_estimators: 200
GBM	learning_rate: 0.01, n_estimators: 100, max_depth: 2
SVR	C: 10, epsilon: 0.1
NN	hidden_layer_sizes: 3, neurons: 15, batch_size: 40, epoch: 200

The variogram model given in the following equation has been employed to generate estimates with the SK model:

$$\gamma(\mathbf{h}) = 0.01_{Nugget} + 0.99Gaus(\mathbf{h})_{\substack{a_{major}=300m \\ a_{minor}=177m \\ a_{vertical}=63m}} \quad (5.4)$$

The estimation performance results of SK, SL, and SK+SL models are compared in Table 5.11 and Figure 5.35. The SL model has yielded the worst performance in terms of R-squared, RMSE, and MAE. On the other hand, the best estimation result has been obtained from the SK+SL model on the lognormal dataset. The mean and the st.dev. reproductions are examined in Table 5.11, which compares all estimation models and exhaustive data. All estimation models have underestimated the mean. Nevertheless, the closest mean value to the truth has been reproduced by the SK model.

Table 5.11. The comparison of general metrics and the basic statistics of all models for the lognormal synthetic data

Model	R-Squared	RMSE	MAE	mean	st.dev.
SK	0.850	1.503	0.400	1.422	3.171
SL	0.762	1.893	0.547	1.289	2.743
SK+SL	0.901	1.222	0.381	1.352	3.712
Truth	-	-	-	1.451	3.884

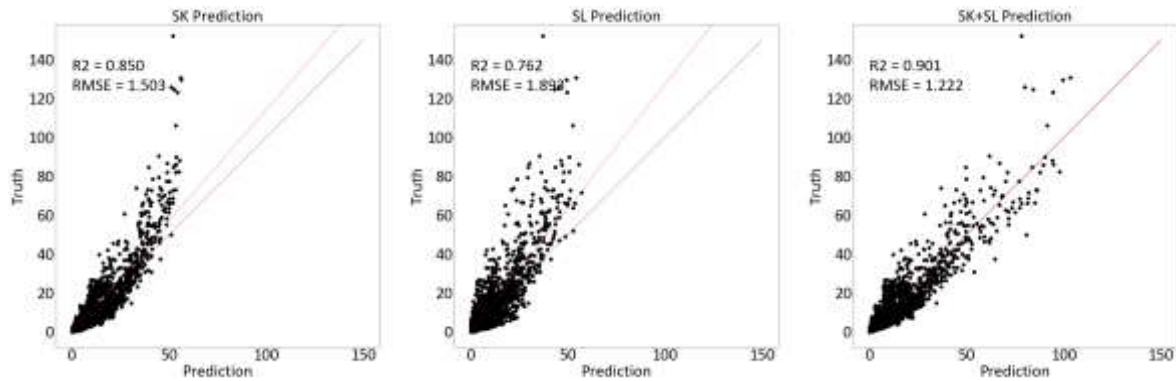


Figure 5.35. Estimation performance of all models for the lognormal synthetic data.

Figure 5.36 indicates the estimation map results obtained from the specified models. The estimation map of the SL model contains significant artifacts. These artifacts could have been eliminated with the combination strategy. The spatial map of the SK + SL model looks like the truth.

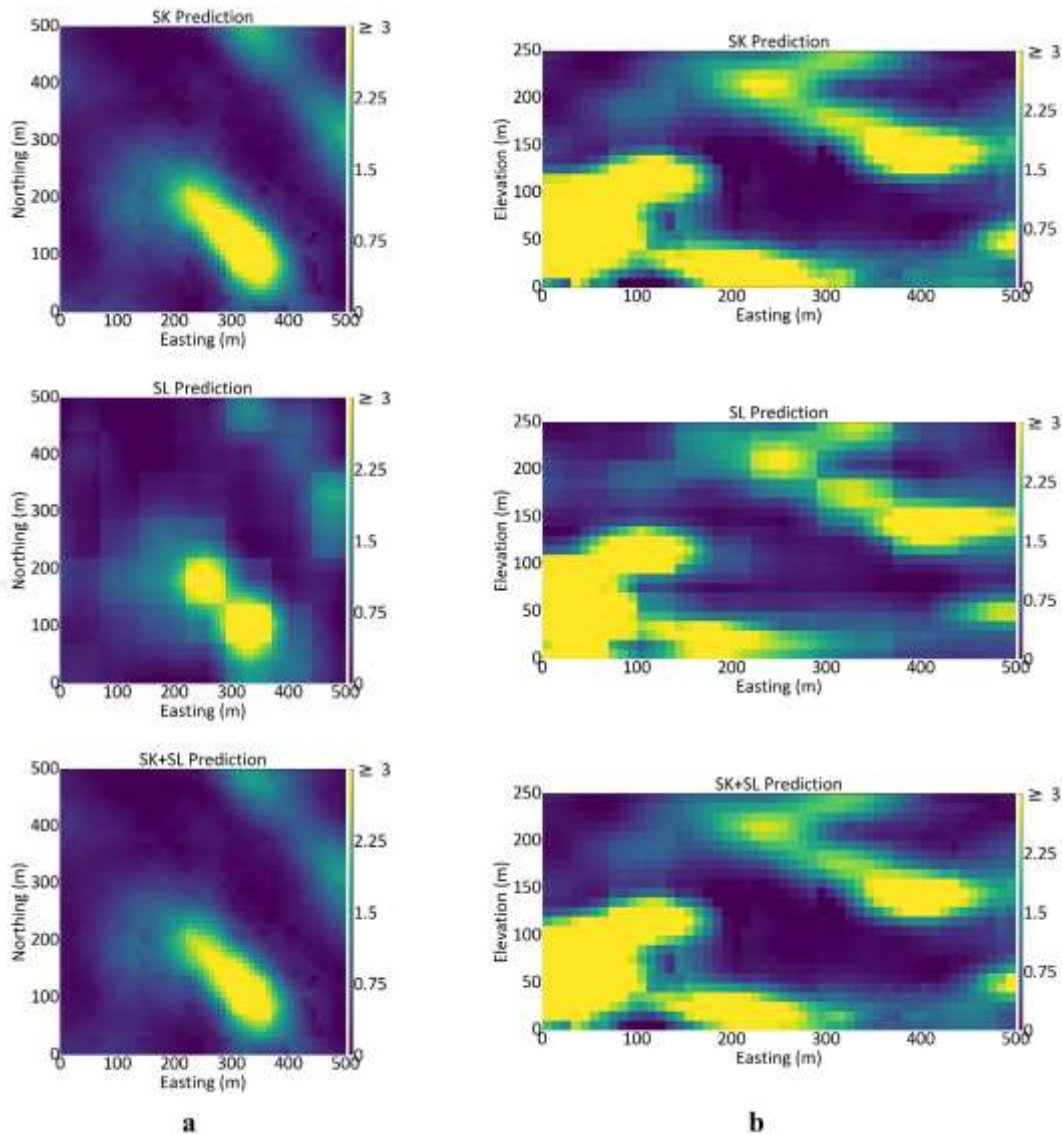


Figure 5.36. Spatial estimation maps of all models in (a) plan view at an elevation of 190 m and (b) E-W section looking from North for the lognormal synthetic data.

Figure 5.37 shows the histograms and the spatial maps of the weights for the SK and SL models, respectively. It can be seen from Figure 5.37 (a) that the distribution of the weight of the SK model is skewed to left. It means that the SK model is more important than the SL model during the combination of the models. The spatial map of the weights in Figure 5.37 (a) indicates, as expected, that while the high weights of the SK model have concentrated on the sampling points and their surroundings, the SL model takes high weight at the locations away from the data.

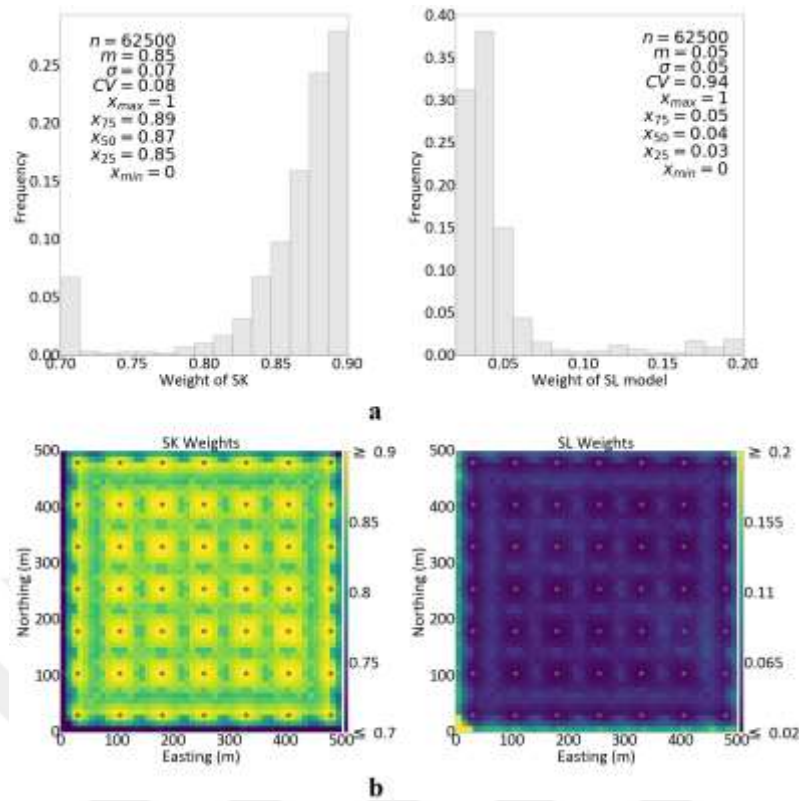


Figure 5.37. (a) The distribution and (b) plan maps of the weights at an elevation of 190 m for SK and the SL models for lognormal synthetic data

The illustration of objective contours for the lognormal dataset is shown in Figure 5.38. The contours are discrete and there is a unique optimum b that includes b_0 (1.42) and b_1 (-0.024). In this case, b_0 has positively affected the power of weight, while b_1 has almost no effect on the power of weight.

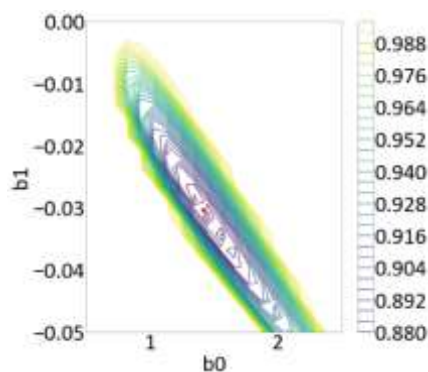


Figure 5.38. Illustration of objective contours and the feasible solution of parameters of weight for the lognormal synthetic data.

Finally, the 3-D omnidirectional variogram for both the true data and the residuals of the SK+SL model is examined in Figure 5.39. The residual variogram does not demonstrate the spatial structure, which proves that there is no local spatial bias in estimation for the examined lognormal dataset.

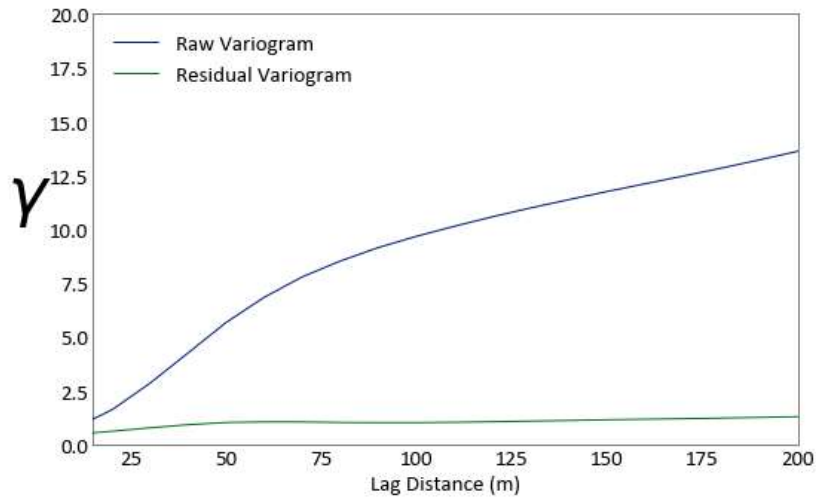


Figure 5.39. Omnidirectional variogram of raw data and SK+SL model residuals for the lognormal synthetic data

5.3.4. Gaussian synthetic data example

The exhaustive Gaussian dataset is produced on the same simulated domain (500 m × 500 m × 250 m) like the ones used in previous cases. Data values of this case are generated by the SGS with the SK model. The simulated realization (underlying reference data) and the results of the sampling are shown in Figure 5.40. The samples are drilled at a spacing of 16% of the total distances in the X and Y directions. The 36 drill holes with 864 samples taken from the simulated realization have been used as a training dataset for both the SK and SL models.

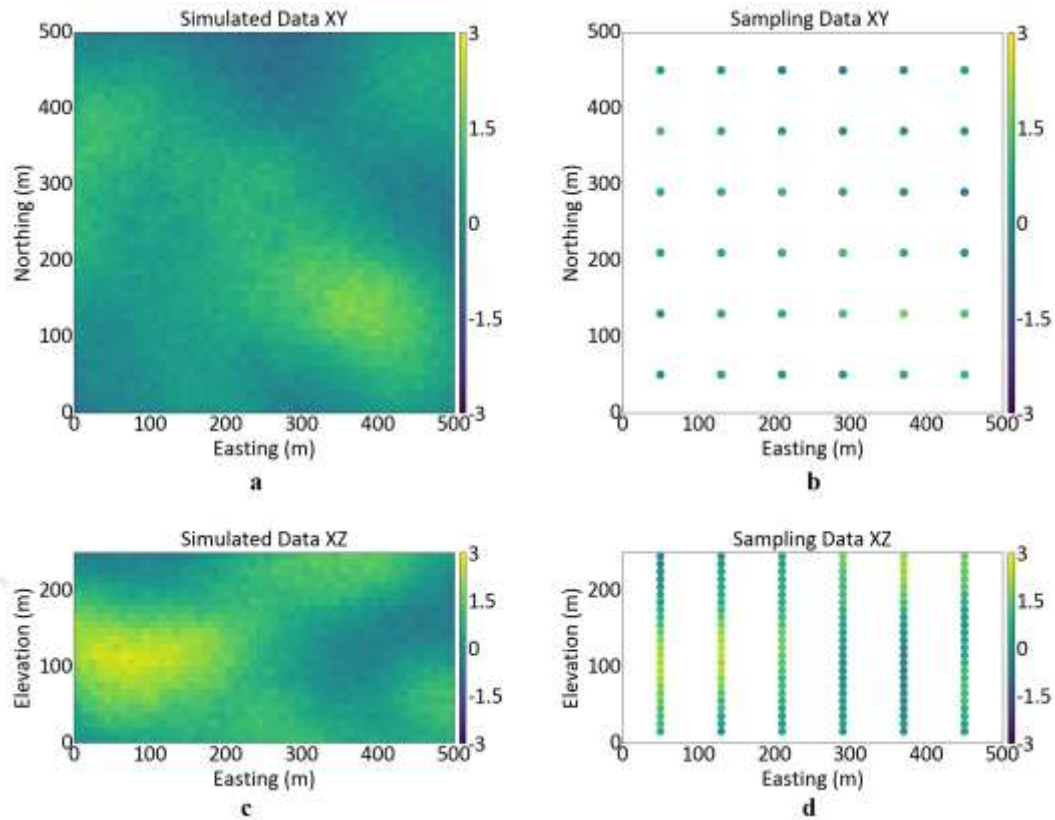


Figure 5.40. (a) Reference realization plan map, (b) sampling plan map at an elevation of 190 m, (c) reference realization, and (d) sampling E-W section map looking from North (Gaussian synthetic data).

The histograms for the simulated and sampled datasets (Figure 5.41) show a symmetric bell-shaped curve, and p-value greater than 0.05, revealing the fact that this synthetic dataset has a strong Gaussian essence in it.

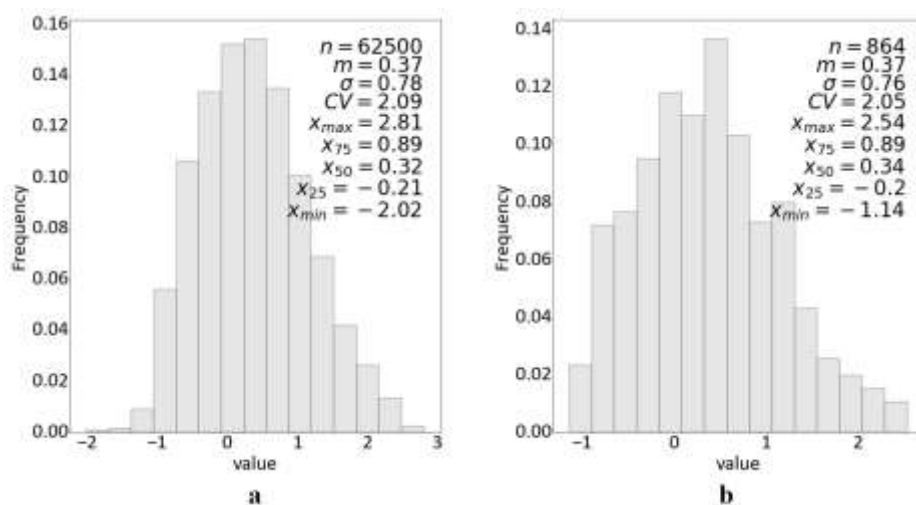


Figure 5.41. Histogram of (a) reference realization (b) sampling data (Gaussian synthetic data).

The workflow to generate estimates with defined models are the same as the ones used in previous examples. The best configuration of the hyper-parameters for each selected ML model, which have been obtained from the grid search algorithm with 5-fold cross-validation, is contained in Table 5.12. After hyper-parameter tuning, the estimates have been generated using the SL model.

Table 5.12. The best configuration of hyper-parameters for the Gaussian synthetic data

ML model	Set of Hyper-parameters
kNN	n_neighbors: 2
DT	max_depth: 12
BAG	n_estimators: 1500
RF	n_estimators: 1000
EXT	n_estimators: 2500
ADA	n_estimators: 800
GBM	learning_rate: 0.1, n_estimators: 1000, max_depth: 4
SVR	C: 1e3, epsilon: 0.01
NN	hidden_layer_sizes: 4, neurons: 15, batch_size: 40, epoch: 400

The variogram model given in Equation 5.5 has been used to generate the estimates with the SK model.

$$\gamma(\mathbf{h}) = 0.01_{Nugget} + 0.99Gaus(\mathbf{h}) \begin{matrix} a_{major}=153m \\ a_{minor}=293m \\ a_{vertical}=122m \end{matrix} \quad (5.5)$$

Figure 5.42 and Table 5.13 indicate the estimation performance results of SK, SL, and SK+SL models. Although the SL model performance result appears to compete with the SK model in terms of R-squared, RMSE, and MAE, any method does not outperform kriging in the case of the Gaussian datasets. The SK+SL model has yielded very similar results to the SK model. Table 5.13 also compares the mean and st.dev. of all estimation models and of exhaustive data. All models reproduce a similar mean with underlying data.

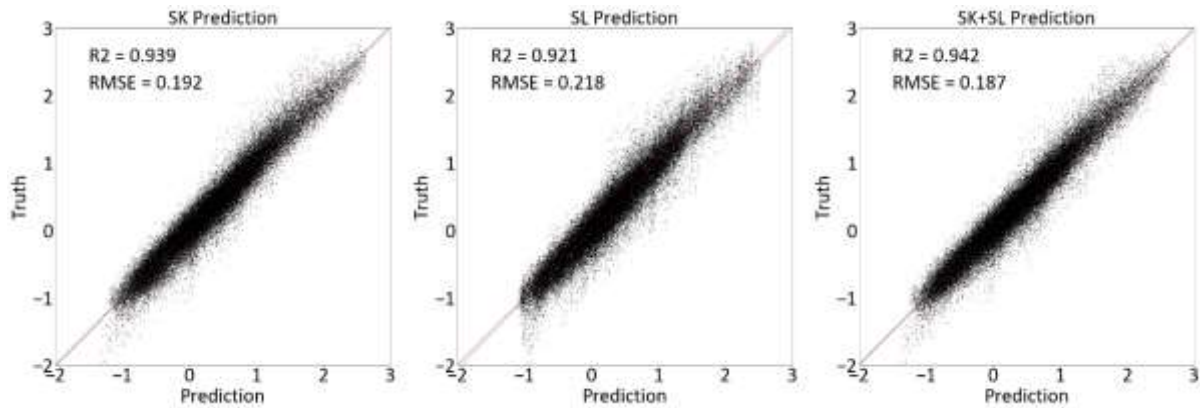


Figure 5.42. Estimation performance of all models for the Gaussian synthetic data.

Table 5.13. The comparison of general metrics and the basic statistics of all models for the Gaussian synthetic data

Model	R-Squared	RMSE	MAE	mean	st.dev.
SK	0.939	0.192	0.147	0.370	0.742
SL	0.921	0.218	0.165	0.365	0.723
SK+SL	0.942	0.187	0.144	0.376	0.761
Truth	-	-	-	0.372	0.776

Figure 5.43 shows the estimation map results obtained from the SK, SL, and SK+SL models. The SL model has not yielded satisfactory visual results as it contains artifacts in its estimation map. On the other hand, the SK and SK+SL models have produced similar estimation maps with each other and the underlying reference data.

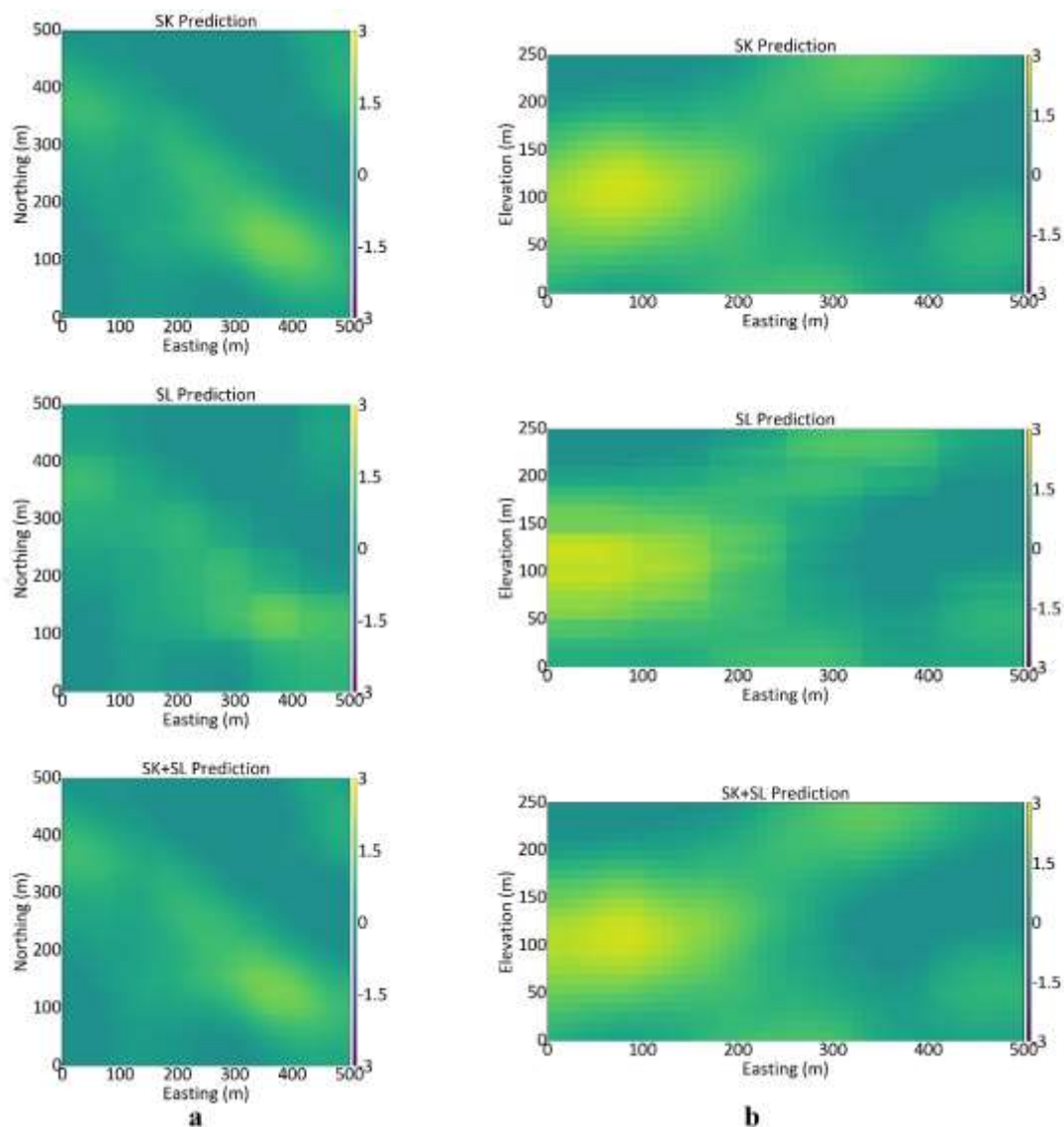


Figure 5.43. Spatial estimation maps of all models in (a) plan view at an elevation of 190 m and (b) E-W section looking from North for the Gaussian synthetic data.

Figure 5.44 shows the histograms and the spatial maps of the weights for the SK and SL models. Histograms show that the SK model dominates the weighting and it will be much more important than the SL model during the combination of the models.

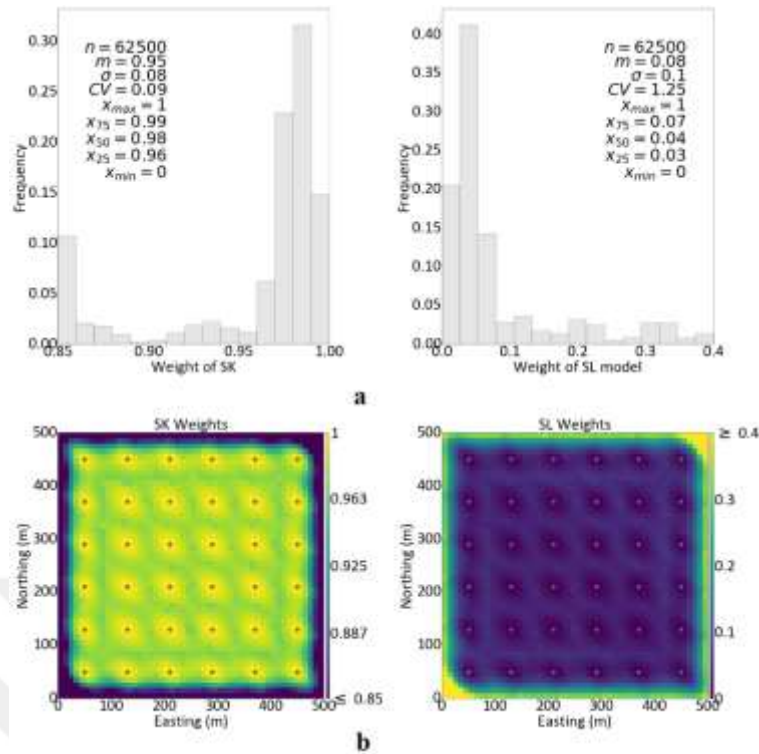


Figure 5.44. (a) The distribution and (b) plan maps of the weights at the elevation of 190 m for SK and the SL models for the Gaussian synthetic data

The illustration of objective contours and a unique feasible solution of parameter b (b_0 is 0.83 and b_1 is 0.04) indicated by the red dot for this dataset are shown in Figure 5.45.

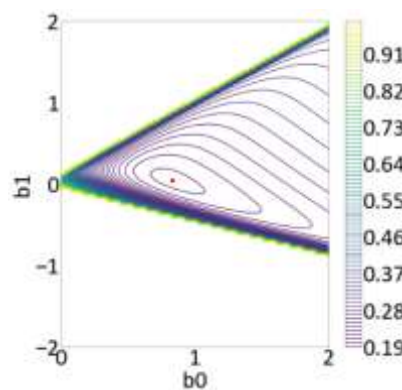


Figure 5.45. Illustration of objective contours and the feasible solution of parameters of weight for the Gaussian synthetic data.

The 3-D omnidirectional variogram for both the exhaustive data and the residuals of the SK+SL model is examined in Figure 5.46. There is no spatial structure in the residuals calculated from the SK+SL model, which shows that structured information has been extracted by the proposed method.

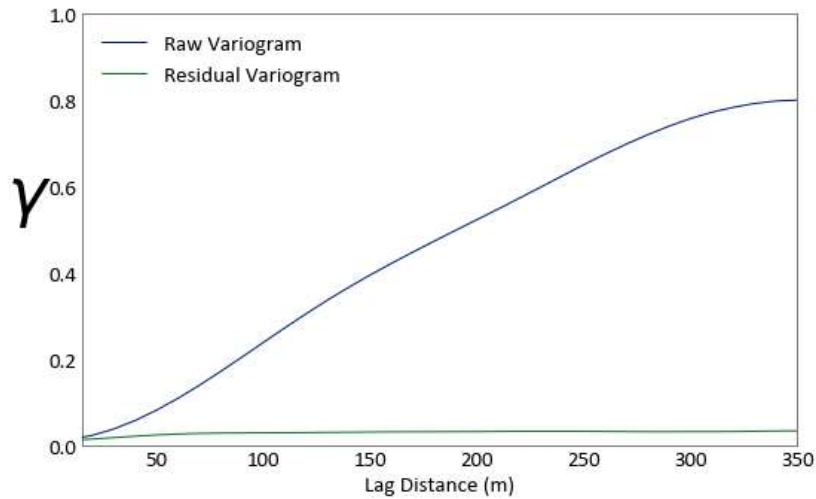


Figure 5.46. Omnidirectional variogram of raw data and SK+SL model residuals for the Gaussian synthetic data.

5.4. Case Study - 3: Implementation of the Methodologies in Mining Cases

This section measures and validates the estimation performance of the proposed methods with real case studies using the datasets from oilsands and the high-grade gold deposits. While the oilsands dataset is examined to demonstrate the practical application of the combined model that includes the OK and SL models, the gold dataset is used for the combination of the SK and ESL models. The estimation results of the proposed methods are compared with kriging using a new k-fold cross-validations approach, which spatially divides the data into k subsets, by refitting the models in each subset. For this comparison, the same performance criteria as the ones used in previous cases are used. In addition to these criteria, the value of the improvement in performance measures is evaluated in terms of practical economic engineering sets.

5.4.1. Oilsands data example

The data are shown in Figures 5.47 (a) and 5.47 (b) contain 280 drill holes with 5808 measurements from an oilsands deposit. The location coordinates range from 500 to 4,000 meters in the East, from 5,000 to 11,000 meters in the North, and from 145 to 295 meters in Elevation with an interval of 3 meters in Elevation.

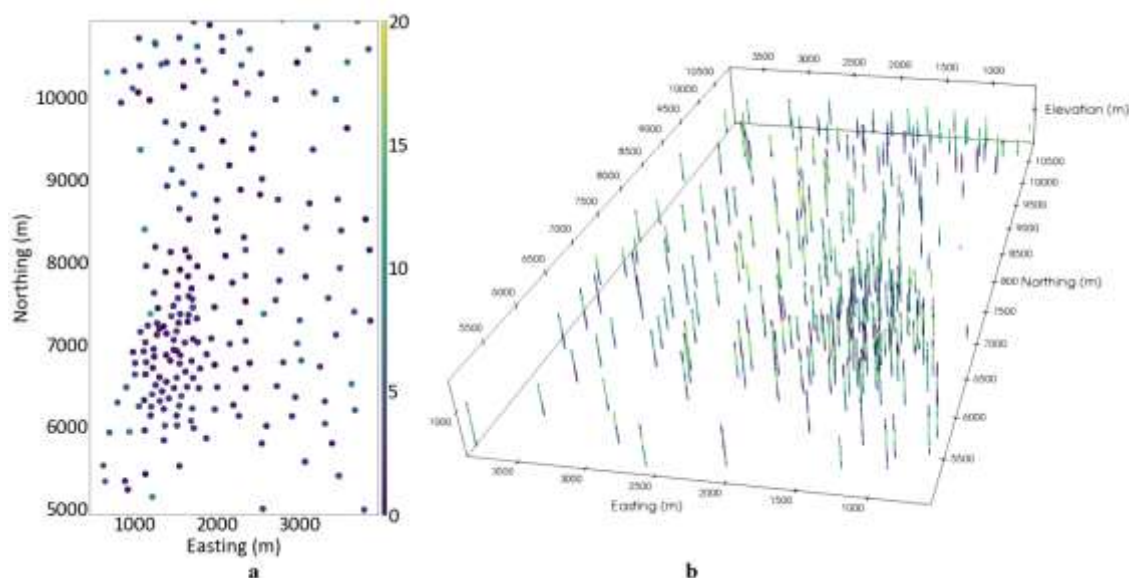


Figure 5.47. (a) Location plan map of Bitumen at an elevation of 235 m, (b) location map in 3-D view

Figure 5.48 shows the experimental horizontal and vertical variogram points and associated analytical models fitted to these variograms. The major and minor continuity directions are 40 and 130 degrees, respectively from the North. It is clear from Figure 5.48 that this deposit shows strong vertical to horizontal anisotropy due to the stratified geology of the oilsands (Deutsch et al., 2014).

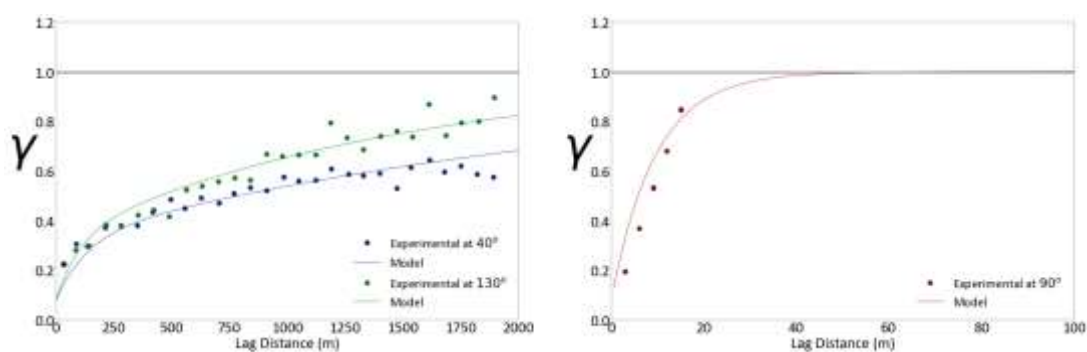


Figure 5.48. Modelled Bitumen variograms.

As can be seen from Figure 5.47, more samples are collected in the South West area thus the equal-weighted statistics could not be representative of the whole population. In this situation, declustering assigns each datum a weight based on its closeness to surrounding data to represent the distribution of the spatial variable. Figure 5.49 (a) demonstrates the plot of the cell size against the declustered mean. It can be seen from Figure 5.49 (a) that there is a clear minimum (at 38.5-meter cell size) that corresponds to the appropriate size of data clusters. The histogram of the declustered data is shown in Figure 5.49 (b) with a mean of 6.93% and an st.dev. of 5.07%.

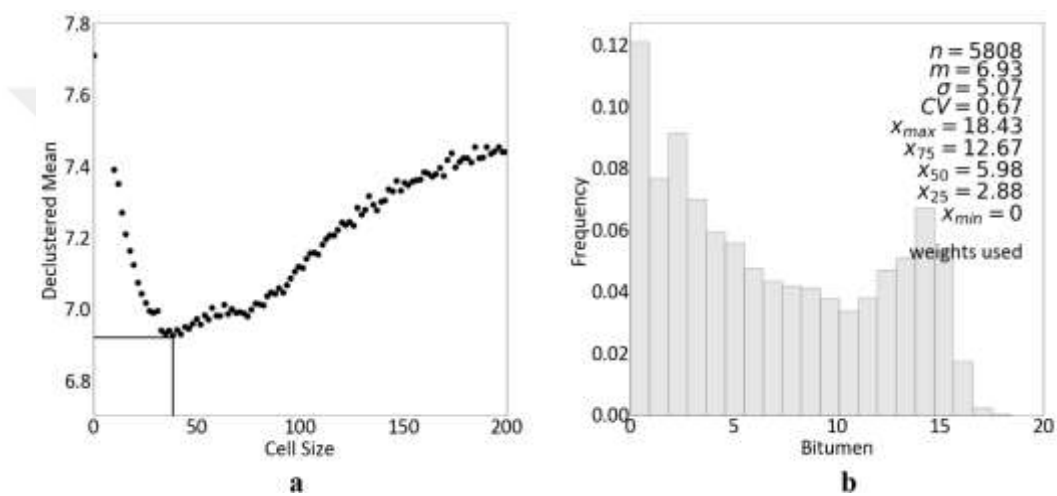


Figure 5.49. (a) Cell size declustering, (b) histogram with declustering of bitumen

In this case study, there is no exhaustive dataset to compare results directly to an underlying truth. Therefore, the algorithm proposed by Deutsch (2018) has been implemented to partition a set of drill hole data into k -folds for model validation. The basic idea of using this algorithm here is to keep aside part of the data from the available dataset and not use them in the training process. This approach enables one to estimate how the model is expected to perform in general. k has been selected as 5 as recommended by Zhang, et al., (1999) and An et al., (2007), thus 5-fold cross-validation has been implemented throughout the study. The original, training and validation sample location maps for the first fold are illustrated in Figure 5.50.

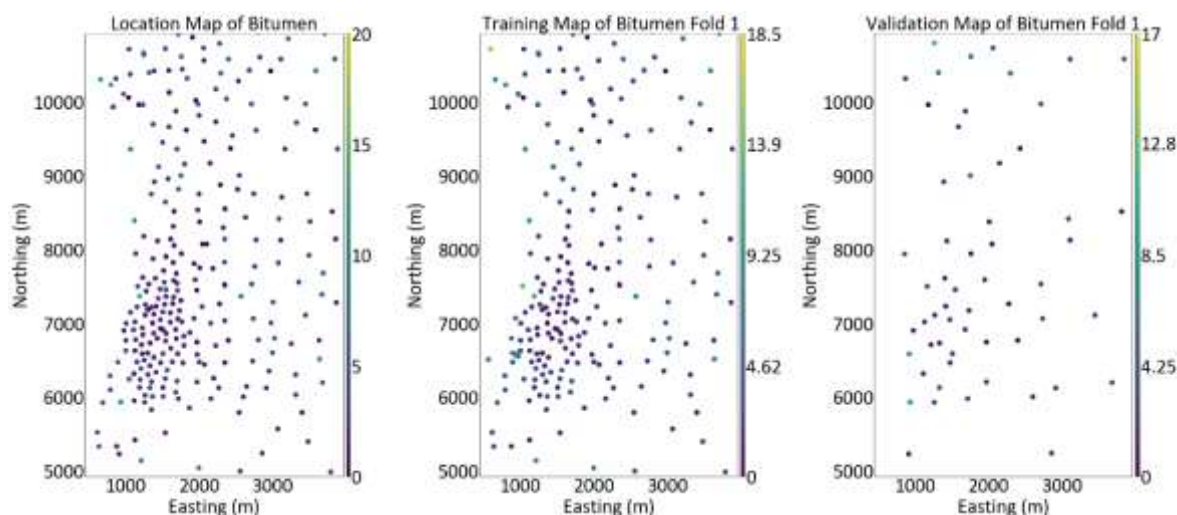


Figure 5.50. Original, training, and validation dataset plan map for the first fold at the elevation of 235 m

For consistent validation, the pertinence of the subsets compared to the original dataset is recommended to be checked (Kanevski, 2013). The statistical similarity of these subsets is verified for the first fold by histogram plots, as shown in Figure 5.51. All of the subsets have similar distributions. If it was not, the model would saturate and underperform for some values. The estimation performance of the OK, SL model, and combinations of OK and SL models (OK + SL) are compared with k -fold validation.

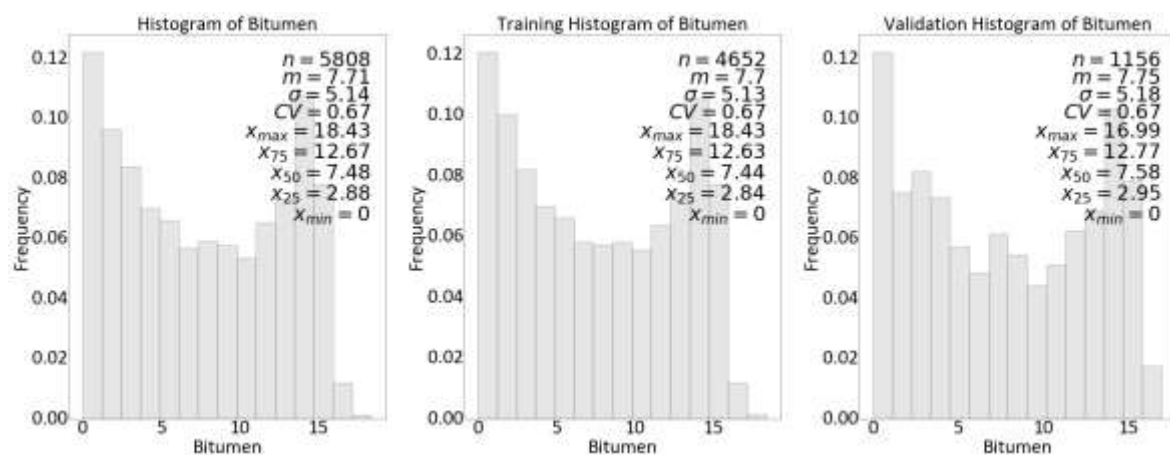


Figure 5.51. The histograms of the original, training, and validation datasets for the first fold

The SL model has been trained with the best hyper-parameter configuration of each ML model shown in Table 5.14 and used to produce estimation on the grid nodes.

Table 5.14. The best configuration of hyper-parameters for the oilsands data

ML model	Set of Hyper-parameters
kNN	n_neighbors: 15
DT	max_depth: 8
BAG	n_estimators: 500
RF	n_estimators: 500
EXT	n_estimators: 100
ADA	n_estimators: 300
GBM	learning_rate: 0.01, n_estimators: 500, max_depth = 4
SVR	C: 1e3, epsilon: 0.1
NN	hidden_layer_sizes: 3, neurons: 10, batch_size: 10, epoch: 300

The variogram model to generate estimates through OK is expressed in the following equation:

$$\gamma(\mathbf{h}) = 0.08_{Nugget} + 0.25 \text{Exp}(\mathbf{h})_{\substack{a_{major}=400m \\ a_{minor}=300m \\ a_{vertical}=25m}} + 0.67 \text{Exp}(\mathbf{h})_{\substack{a_{major}=8000m \\ a_{minor}=4500m \\ a_{vertical}=30m}} \quad (5.6)$$

After the OK and SL models have been prepared to produce estimation on each grid node, their estimates have been combined by using weights that are optimized on the validation dataset of each fold. Table 5.15 shows the optimized b_0 and b_1 for each fold. b_1 has taken the value approximately equal to zero in each fold. It means that the weight is determined mainly by b_0 .

Table 5.15. The optimized b_0 and b_1 for each fold

	b_0	b_1
Fold 1	0.924	0.001
Fold 2	1.262	-0.022
Fold 3	0.961	0.004
Fold 4	0.938	0.003
Fold 5	0.841	0.015

Figure 5.52 shows the accuracy plots for the estimations of OK, SL and OK+SL models for each fold. Moreover, Table 5.16 summarizes the performance results of each fold in terms of R-squared, RMSE, and MAE criteria. The overall performance is average across all folds.

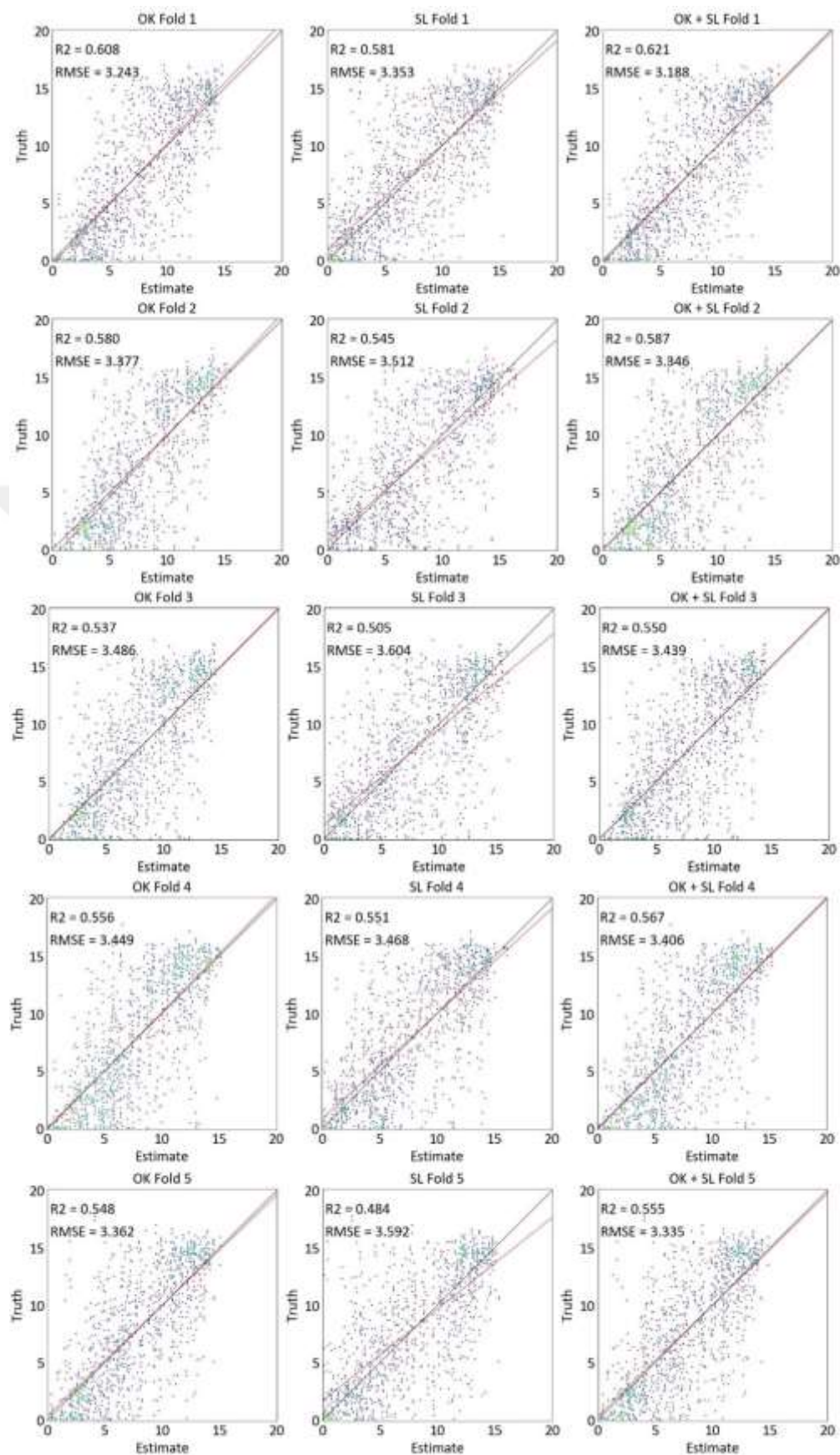


Figure 5.52. Estimation performance of OK, SL, and OK+SL models for each fold of oilsands data

Table 5.16. The comparison of general metrics of all estimation models for oilsands data

	R-Squared			RMSE			MAE		
	OK	SL	OK+SL	OK	SL	OK+SL	OK	SL	OK+SL
Fold 1	0.608	0.581	0.621	3.243	3.353	3.188	2.528	2.483	2.453
Fold 2	0.580	0.545	0.587	3.377	3.512	3.346	2.611	2.600	2.539
Fold 3	0.537	0.505	0.550	3.486	3.604	3.439	2.671	2.636	2.623
Fold 4	0.556	0.551	0.567	3.449	3.468	3.406	2.646	2.611	2.603
Fold 5	0.548	0.484	0.555	3.362	3.592	3.335	2.532	2.569	2.515
Overall	0.566	0.533	0.576	3.384	3.507	3.344	2.597	2.579	2.546

It is clear from Table 5.16 that the best performance is obtained from the OK+SL model, with an R-squared of 0.576, RMSE of 3.344, and MAE of 2.546. SL model yields the worst performance with an R-squared of 0.533 and an RMSE of 3.507. The performance of the OK model is not as good as the performance of the OK+SL model (R-squared of 0.566, RMSE of 3.384, and MAE of 2.597). The mean and st.dev. of all estimation models and of data in each fold are compared in Table 5.17. The mean is slightly overestimated from the OK model and slightly underestimated from the SL model; however, the mean appears exactly reproduced by the OK+SL model.

Table 5.17. The comparison of the basic statistics of all estimation models and of oilsands data in each fold

	mean				st.dev.			
	<i>Truth</i>	OK	SL	OK+SL	<i>Truth</i>	OK	SL	OK+SL
Fold 1	7.754	7.697	7.448	7.796	5.179	3.839	4.382	4.000
Fold 2	7.454	7.691	7.639	7.501	5.209	3.758	4.440	3.991
Fold 3	7.700	7.707	7.782	7.687	5.125	3.716	4.470	3.783
Fold 4	7.847	7.788	7.653	7.854	5.177	3.779	4.226	3.850
Fold 5	7.757	7.675	7.607	7.674	5.002	3.852	4.510	3.843
Overall	7.702	7.712	7.626	7.702	5.138	3.789	4.406	3.893

Given the different approaches, the overall spatial estimation maps by the OK and SL models are similar (Figure 5.53). However, it can be seen from Figure 5.53 that the SL model has not yielded satisfactory visual results and contains artifacts (linear boundaries) in its estimation map. On the other hand, the combination of the OK and SL models has improved and dramatically reduced the degree of artifacts in the estimation maps.

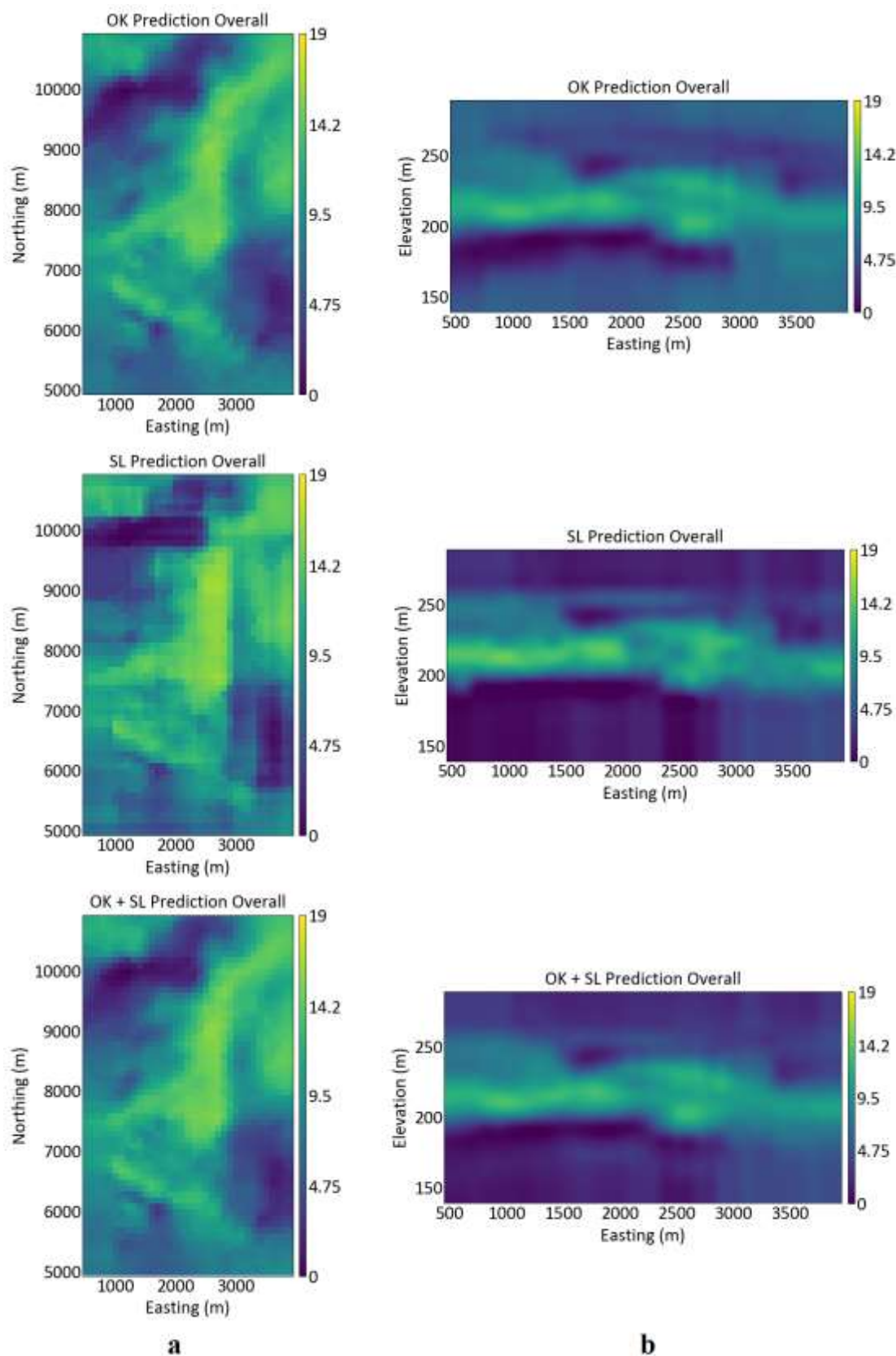


Figure 5.53. Spatial estimation maps of all models in (a) plan view at an elevation of 235 m and (b) E-W section looking from North for the oilsands data

The presence of the structure (remaining information) in the residuals has also been tested to understand the quality of the OK+SL model. The variogram of the raw data and the cross-validation residuals have been analyzed in Figure 5.54. There is no remaining spatial structure in the residuals, which means that the proposed method extracted all structured information properly for this case.

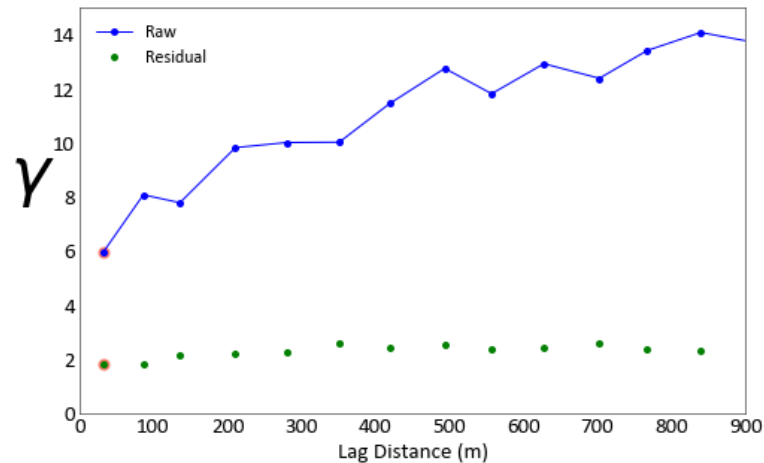


Figure 5.54. The variogram of raw data and cross-validation residuals

In order to emphasize how significant the increase in R-squared and decrease in RMSE and MAE in practical economic engineering sets, a new procedure has been followed. This procedure aims to quantify the improvement in practical units. The units here could be a currency but this would require so many economic assumptions. Therefore, the number of drill holes (DHs) required for the same model performance can be fair units, as people appreciate the drilling and sampling costs. Therefore, the OK and OK+SL models have been applied with m and n numbers of DHs, respectively where $n < m$. The difference between m and n ($m-n$) is somehow equal to the value of using the combined (OK+SL) model. Some DHs (29 DHs with 574 samples) are kept back randomly as blind test data to compare the performance of the OK model with more data (m) and the OK+SL model with fewer data (n). Then, the estimates have been obtained from the OK model on the new dataset. To use fewer DHs and samples for the OK+SL model, a part of DHs has been randomly discarded again (23 DHs with 467 samples). Then, the OK+SL model has been performed on this reduced number of DHs to generate estimates. Finally, the performance of each model has been compared on the blind data. Figure 5.55 demonstrates the blind test data, second discarded data, and remaining data that have been used by the OK+SL model.

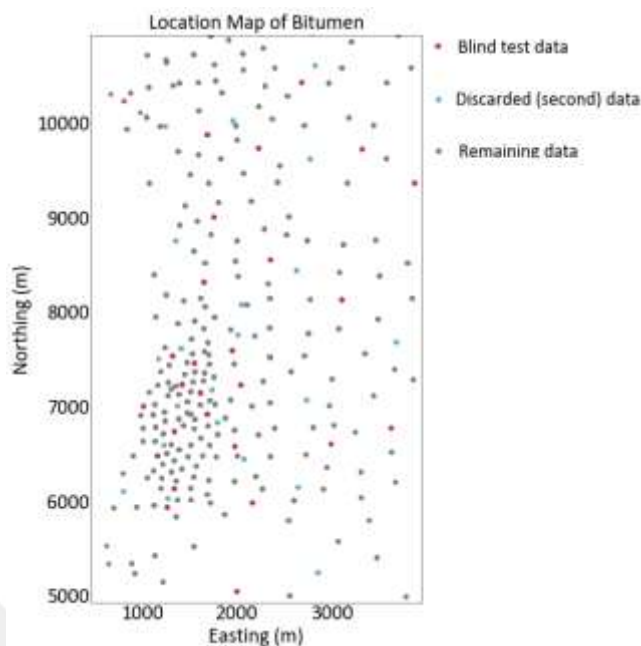


Figure 5.55. The dataset division for the performance comparison of models in different numbers of DHs for the oilsands data

A comparison of the OK and the OK+SL models is illustrated in Figure 5.56 and Table 5.18, respectively. The results indicate that although the OK+SL model effectively has fewer drill holes, it has yielded similar performance quality to the one observed in the OK case. Thus, the improvement by using the OK+SL model in R-squared, RMSE, and MAE could be considered a significant improvement in the estimation.

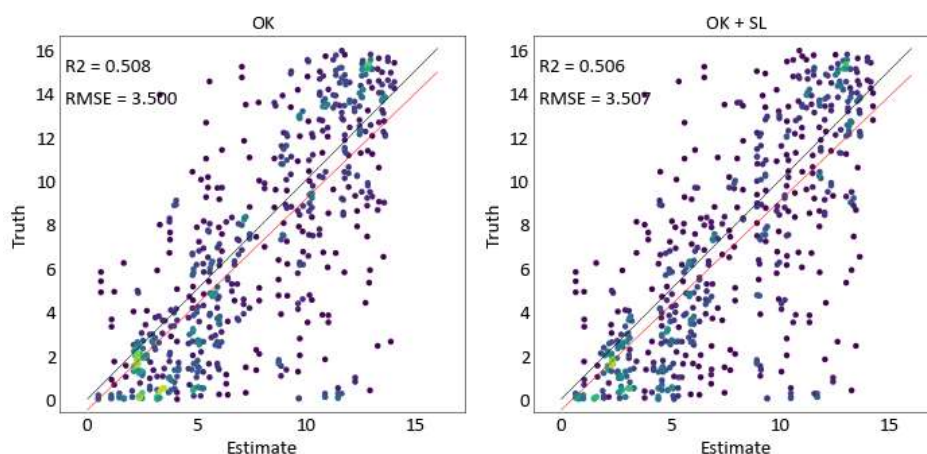


Figure 5.56. The comparison of OK and OK+SL models in different numbers of DHs for oilsands data

Table 5.18. The comparison of OK and OK+SL models in different numbers of DHs for oilsands data

	OK 251 DHs 5216 Samples	OK+SL 228 DHs 4749 Samples
R-squared	0.508	0.506
RMSE	3.500	3.507
MAE	2.633	2.626

5.4.2. Gold data example

The dataset under study was collected from a high-grade gold deposit (Vasylchuk and Deutsch, 2017). The drill holes are located within a $4,190 \times 6,480 \text{ m}^2$ area and the elevation ranges from 110 m to 290 m with the sample length of 1 m down. Figure 5.57 shows that the location map of the data contains 322 drill holes with 30,774 measurements. It is clear from Figure 5.57 that the drill holes are located in an irregular configuration.

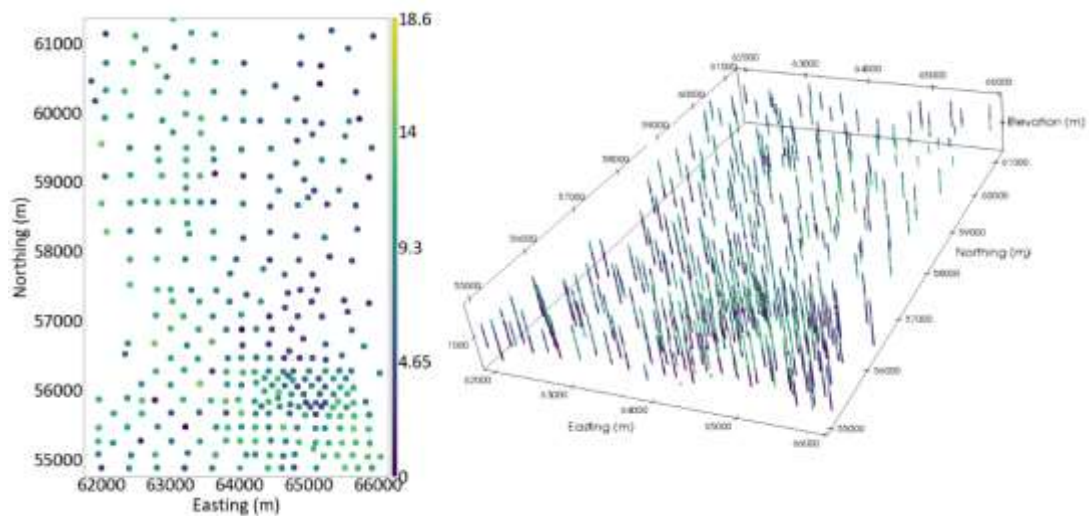


Figure 5.57. (a) Location plan map of gold at an elevation of 210 m, (b) location map in 3-D view

For this dataset, the declustering method has not been implemented as there is no clear appropriate cell size to assign weights to the samples, as shown in Figure 5.58 (a). The histogram of the unweighted gold variable is demonstrated in Figure 5.58 (b). The gold grade values range from 0 g/t to 19.52 g/t with a mean of 6.36 g/t and a st.dev. of 5.42 g/t. 0 g/t is the widespread grade and the high-grade ore is located mostly in narrow veins intermixed with barren rock (Vasylchuk and Deutsch, 2017).

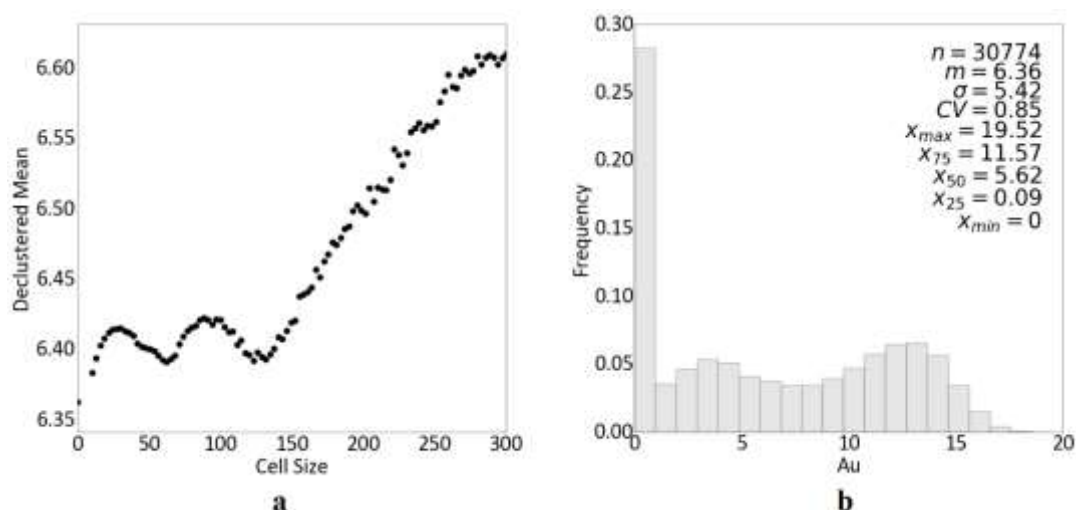


Figure 5.58. (a) Cell size declustering, (b) histogram with declustering of the gold variable

The spatial continuity of the gold grade has been revealed through variogram modeling. Figure 5.59 shows the experimental major, minor and vertical variogram points and associated analytical models fitted to these variograms. The major and minor continuity directions are 135° and 45° from the North, respectively. The variograms have a modest nugget effect due to variations at small scales.

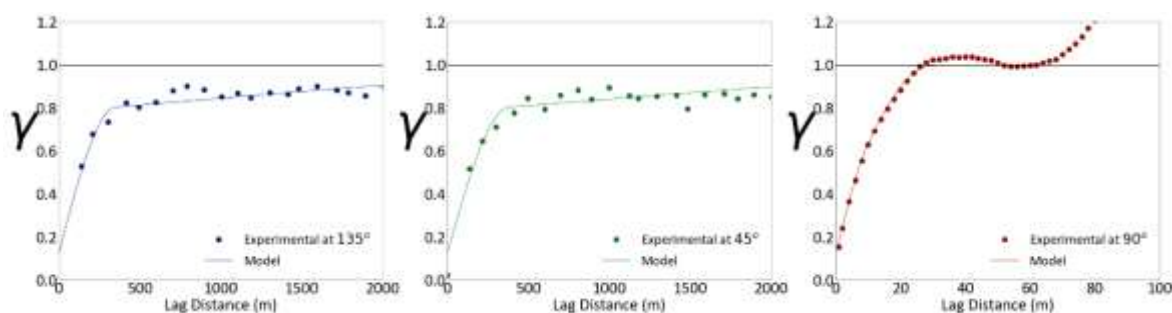


Figure 5.59. Variograms of the gold variable in three directions

The optimal parameters of the base ML models inside the SL model are shown in Table 5.19. An SL model with these optimum hyper-parameters has been trained firstly with the initial training set. Then, 9 rotated datasets have been created by selecting α as 10° in the coordinate rotation strategy. The individual SL models have been trained with these rotated training datasets. After this step, the estimates have been generated on the corresponding grid nodes individually and then ensembled to obtain the final estimates for the ESL model.

Table 5.19. The best configuration of hyper-parameters for the gold data

ML model	Set of Hyper-parameters
kNN	n_neighbors: 15
DT	max_depth: 16
BAG	n_estimators: 1000
RF	n_estimators:1000
EXT	n_estimators: 1000
ADA	n_estimators: 300
GBM	learning_rate: 0.01, n_estimators: 700, max_depth: 6
SVR	C: 1e3, epsilon: 0.1
NN	hidden_layer_sizes: 4, neurons: 15, batch_size: 80, epoch: 400

To obtain estimates from SK, the variogram model is expressed in the following equation:

$$\gamma(\mathbf{h}) = 0.15_{Nugget} + 0.44Exp(\mathbf{h})_{\substack{a_{major}=250m \\ a_{minor}=200m \\ a_{vertical}=10m}} + 0.2Exp(\mathbf{h})_{\substack{a_{major}=600m \\ a_{minor}=600m \\ a_{vertical}=10m}} + 0.21Exp(\mathbf{h})_{\substack{a_{major}=2250m \\ a_{minor}=500m \\ a_{vertical}=10m}} \quad (5.7)$$

For the performance evaluation of SK, SL, ESL, and SK+ESL models, k -fold cross-validation has been made by refitting the models to each subset as the ones used in the previous case study. The k -fold validation process has been repeated until each fold of the 5-folds is used as the validation set. Figure 5.60 shows the original, training and validation sample location maps only for the first fold.

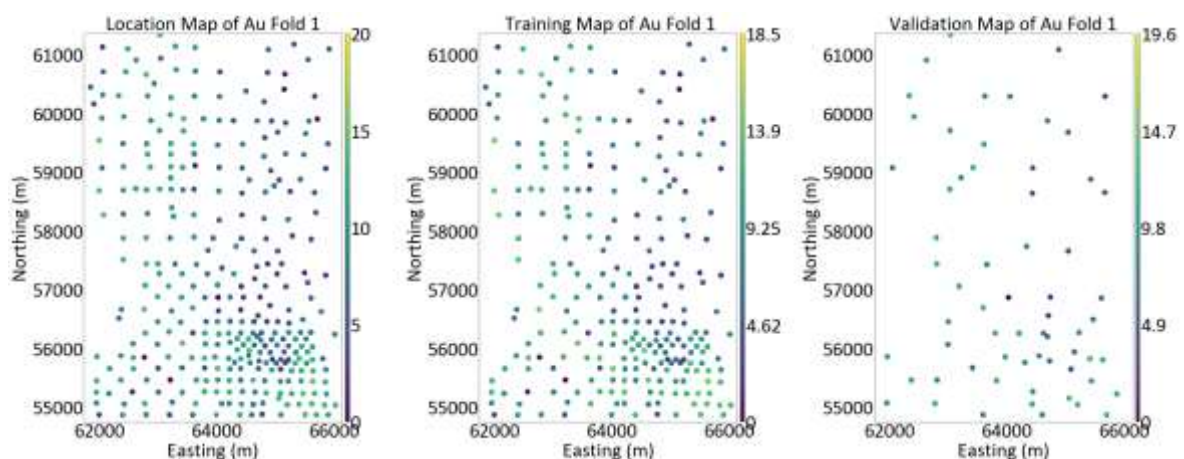


Figure 5.60. Original, training, and validation datasets plan maps for the first fold at the elevation of 210 m

The statistical similarity of these datasets is verified by comparing frequency distribution through histogram plots (Figure 5.61).

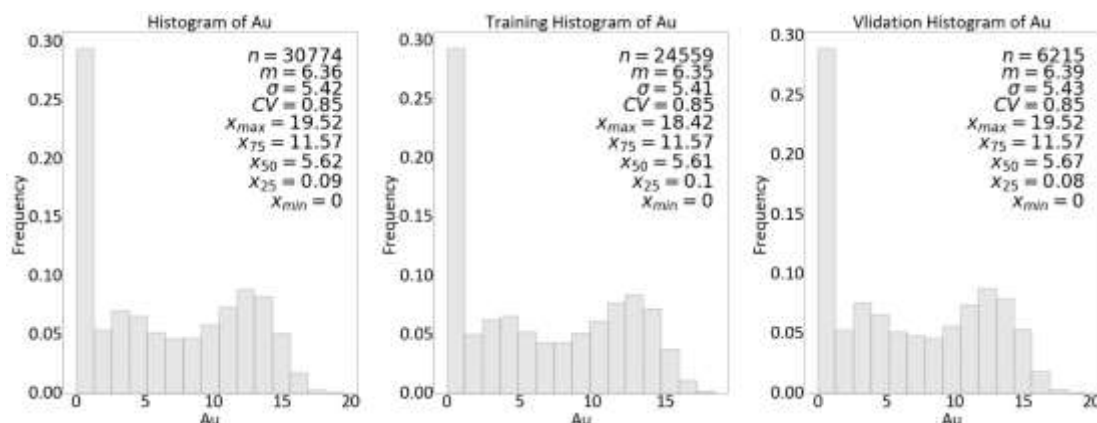


Figure 5.61. The histograms of the original, training, and validation datasets for the first fold

To combine the SK and ESL models estimates at all grid nodes, the parameter b has been optimized on each fold's validation dataset through the SQP. The optimized values of b_0 and b_1 are contained in Table 5.20. b_1 has taken the value approximately equal to zero like in the previous case for each fold. It means that b_1 has a minor effect on the weight.

Table 5.20. The optimized b_0 and b_1 for each fold

	b_0	b_1
Fold 1	1.1392	-0.009
Fold 2	1.0087	9.26e-05
Fold 3	1.1706	-0.0149
Fold 4	1.356	-0.0336
Fold 5	0.7656	0.0234

The overall spatial estimation maps of SK, SL, ESL, and SK+SL models are compared in Figure 5.62.

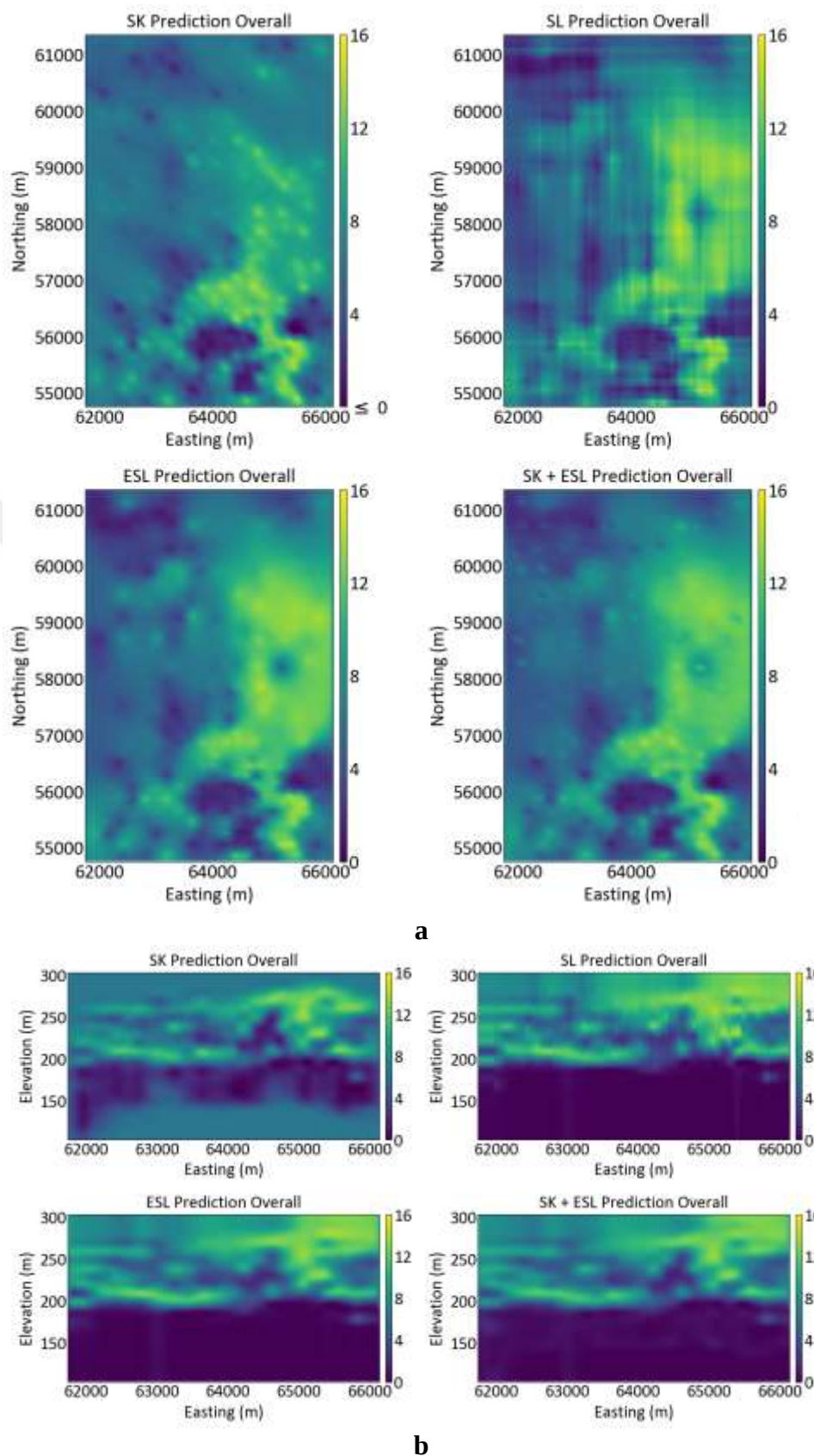


Figure 5.62. Spatial estimation maps of all models in (a) plan view at an elevation of 210 m and (b) E-W section looking from North for the gold data

It is clear from the results shown in Figure 5.62 that the SL model yields unpractical linear boundaries, while the ESL model removes these artifacts in its resulting estimation map. The performance results of the SK, SL, ESL, and SK+ESL models in terms of R-squared, RMSE, and MAE criteria are summarized in Table 5.21. SK+ESL model yields the best performance, with an R-squared of 0.387, RMSE of 4.241, and MAE of 3.250. The worst performance is obtained from the SL model with an R-squared of 0.332 and RMSE of 4.427. ESL, on the other hand, indicates better performance than those of the SK and SL models (R-squared of 0.372, RMSE of 4.292, and MAE of 3.268).

Table 5.21. The comparison of general metrics of all estimation models for the gold data

	R-Squared				RMSE				MAE			
	SK	SL	ESL	SK+ESL	SK	SL	ESL	SK+ESL	SK	SL	ESL	SK+ESL
Fold 1	0.327	0.310	0.354	0.368	4.454	4.507	4.363	4.313	3.708	3.401	3.330	3.307
Fold 2	0.366	0.368	0.393	0.405	4.324	4.315	4.231	4.188	3.563	3.223	3.224	3.195
Fold 3	0.322	0.307	0.356	0.367	4.450	4.498	4.337	4.298	3.698	3.394	3.339	3.327
Fold 4	0.375	0.333	0.382	0.401	4.278	4.420	4.254	4.188	3.539	3.275	3.174	3.189
Fold 5	0.348	0.340	0.374	0.392	4.364	4.392	4.276	4.215	3.619	3.294	3.275	3.235
Overall	0.347	0.332	0.372	0.387	4.374	4.427	4.292	4.241	3.625	3.317	3.268	3.250

The basic statistics of data and of estimates in each fold are contained in Table 5.22. It appears that the mean is slightly overestimated by SK, SL, and ESL models, whereas the mean appears slightly underestimated by the SK+ESL model. It can be said that all models have reproduced the mean satisfactorily.

Table 5.22. The comparison of the basic statistics of all estimation models and of gold data in each fold.

	mean					st.dev.				
	Truth	SK	SL	ESL	SK+ESL	Truth	SK	SL	ESL	SK+ESL
Fold 1	6.388	6.521	6.557	6.537	6.351	5.427	2.640	3.777	3.672	3.436
Fold 2	6.289	6.305	6.198	6.290	6.265	5.428	2.762	3.801	3.679	3.479
Fold 3	6.320	6.477	6.385	6.427	6.302	5.403	2.674	3.765	3.594	3.423
Fold 4	6.193	6.321	6.454	6.414	6.177	5.412	2.750	3.920	3.767	3.639
Fold 5	6.616	6.420	6.570	6.515	6.640	5.406	2.682	3.901	3.748	3.488
Overall	6.361	6.408	6.432	6.437	6.347	5.415	2.702	3.833	3.692	3.493

The variogram of the raw data and the cross-validation residuals have been analyzed to quantify the quality of the SK+ESL model. The experimental variogram of cross-validation residuals demonstrates no spatial structure in the residuals. It means that there is no local spatial bias in estimation for this dataset.

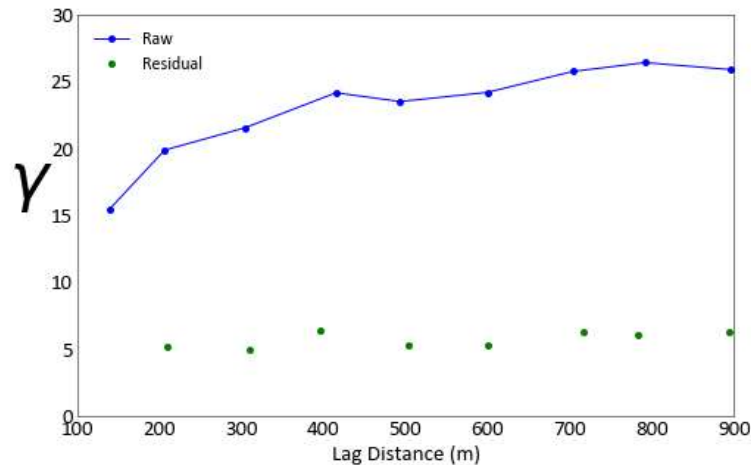


Figure 5.63. The variogram of raw data and cross-validation residuals

To quantify the performance improvement in R-squared, RMSE, and MAE in practical economic engineering sets, the number of DHs has been used as practical units. The SK+ESL model, which yielded the best performance in the high-grade gold deposit dataset, has been applied with fewer DHs and compared with the SK model that considers more DHs. The difference between the number of DHs used by SK and ESL models is somehow equal to the “value” of using the ESL model. For this comparison, some DHs (106 DHs with 10229 samples) have been set aside as blind test data, and the SK model has produced estimates on this new dataset. After that, one more part of DHs (71 DHs with 6800 samples) has been discarded to train the SK+ESL model with fewer DHs and samples. The blind test data, second discarded data, and remaining data that have been used by the SK+ESL model are demonstrated in Figure 5.64.

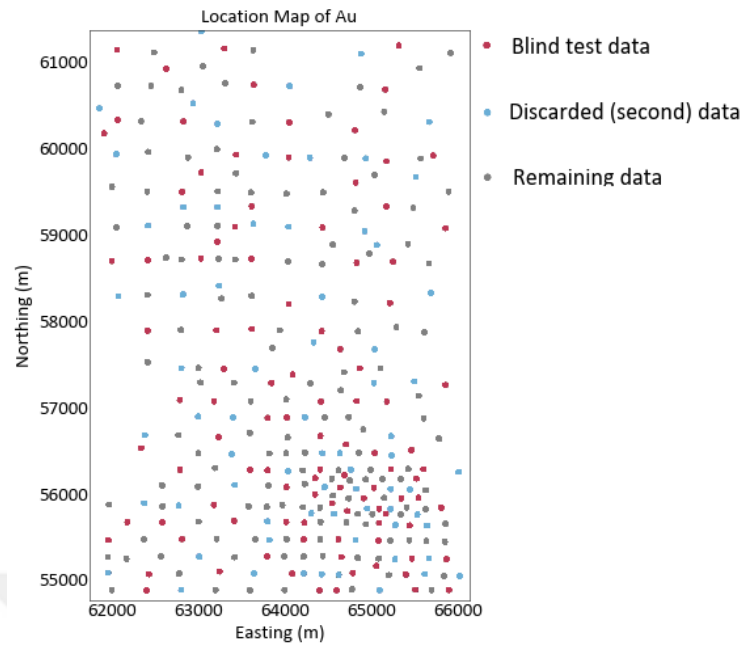


Figure 5.64 The dataset division for the performance comparison of models in the different number of DHs for the gold data

The estimates have been obtained from the SK+ESL model on this reduced number of DHs. Finally, the performances of the SK and SK+ESL models have been compared on the blind test data. Table 5.23 shows the performance results of this comparison and Figure 5.65 demonstrates the scatter plots of the actual vs. estimated values for each model. It can be seen from the results that although the ESL model effectively has fewer drill holes, it showed similar performance quality to the SK model. It appears that the improvement in RMSE, MAE, and R-squared by using the ESL model has a significant contribution to practical economic engineering sets.

Table 5.23. The comparison of SK and SK+ESL models in different numbers of DHs for the gold data

	SK 216 DHs 20545 Samples	SK+ESL 162 DHs 15407 Samples
R-squared	0.329	0.330
RMSE	4.413	4.409
MAE	3.523	3.413

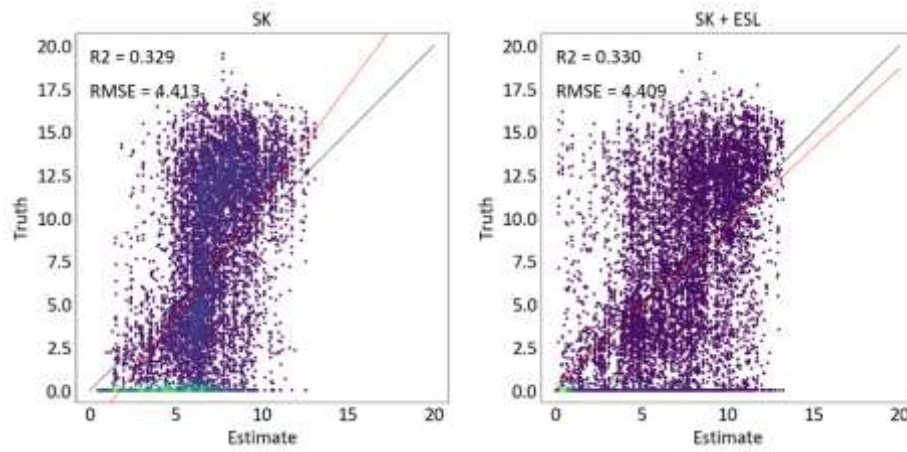


Figure 5.65. The comparison of SK and SK+ESL models in different numbers of DHs for the gold data

6. CONCLUSION AND RECOMMENDATIONS

The use of ML models provides useful tools to generate geospatial estimates at unsampled locations, but they have some disadvantages. For instance, they ignore the spatial proximity among observational units and do not reproduce the data at measured locations. Moreover, most ML models produce visible artifacts in resulting estimation maps, which is a significant problem in the practical usefulness of the models. On the other hand, kriging is known as the best linear unbiased estimation method but it requires some assumptions such as stationarity and linearity. Kriging may not also yield satisfactory estimation performance for complex situations such as non-Gaussian, non-linear, and non-stationary.

In this thesis, ML-based methodologies were developed for the estimation of geospatial variables. The SL model was the ML model explored in this study. This model tries to find an optimal way to combine individual ML models and discover the best trade-off between bias and variance. The performance of the SL model is likely to result in a model that performs better than any ML model it contains. Nine different ML models with different capabilities were evaluated in the SL model. Moreover, an iterative grid search algorithm with 5-fold cross-validation was applied to perform hyper-parameter tuning of the ML models. The performance of the proposed methodologies was measured and compared in terms of R-squared, RMSE, MAE, and reproduction of basic statistics on the given datasets. The estimation maps are visually evaluated for the artifacts. The quality of the proposed methods was also quantified by checking the presence of the spatial structures in the residuals.

In this thesis, firstly, the ESL model to manage visual artifacts in estimation maps as well as incorporating anisotropy information into the estimation process was proposed. In the ESL model, numerous training datasets were produced from the available dataset with coordinate rotation strategy, and the estimates of SL models fitted on these rotated training sets were ensembled to produce optimal final estimates. The ESL model was illustrated in Walker Lake, Swiss Jura public datasets, and a dataset from a high-grade gold deposit to examine its validity for spatial estimation framework. All of the case studies indicated that

the ESL model smoothed out the visual artifacts and yielded reasonable estimation maps. This model performed at least as well as the kriging for Walker Lake and Jura datasets and provided better performance than the ones generated by kriging for the gold dataset. The ESL model yielded much better estimation performance than the SL model in all cases that uses only coordinates as the input features. The results also showed that the ESL model could extract the structured information properly as there was no remaining spatial structure in the residuals. To sum up, the ESL model accomplished three goals: (1) it offered satisfying estimation performance compared to the kriging without the need for expert knowledge, such as variogram modeling, and stationarity assumptions, (2) it provided an improved ML model, and (3) it avoided artifacts in the estimations.

This thesis, secondly, presented a methodology to combine kriging and ML to produce more accurate and consistent estimates in specified situations. For the combination, a function of the $\sigma_{KR}^2(\mathbf{u})$ at all grid points was used as the weight and these weights were optimized using SQP. The combination methodology was evaluated on four different synthetic datasets that include non-stationary, non-Gaussian, lognormal, and Gaussian data. It was observed that the SL model performed better than kriging in the non-stationary and non-Gaussian datasets in terms of R-squared, RMSE, MAE. The kriging had better capability in truly Gaussian and lognormal dataset cases. The combined model (SK+SL), on the other hand, considerably improved the estimation performance results in all cases except the Gaussian dataset. When data is Gaussian or multi Gaussian, kriging is always the best estimation method.

The combined model was also evaluated on data from an oilsands (OK+SL) and gold deposits (SK+ESL) by 5-fold cross-validation to validate the capability of the methodologies in real cases. For both cases, improved models were obtained by combining the strengths of kriging and ML with the weighting approach. While the ML model took the non-stationary, non-Gaussian, and complex features into the estimation, kriging ensured data reproduction and unbiasedness. Moreover, the residual variograms demonstrated the nugget effect, which proved that the combined model could explain the spatial structure properly. Finally, the improvement in performance in terms of R-squared, RMSE, and MAE was quantified in practical units by using a different number of DHs for real cases. The results showed that the combination of the kriging and SL model had a huge contribution to the spatial estimation

in terms of the economy and that although the combined model effectively had fewer drill holes, they yielded similar performance quality to the one obtained from kriging in both of the real datasets. Results of the present analysis are encouraging for further applications of the combined model to produce spatial maps.

6.1. Statement of Contributions

The main contributions of the thesis can be stated as follows:

- The development and adaptation of consistent and coherent methodologies for geospatial data estimation based on ML.
- The application and adaptation of pre-processing, hyper-parameter tuning, and data splitting procedures based on the SL model that is employed for ML modeling.
- The development and investigation of the ESL model, which mainly manages the artifacts in resulting estimation maps.
- The validation of the ESL model on public and real data case studies.
- The development and investigation of the combined model consist of kriging and SL models.
- The analysis of the combined model on different 3-D synthetic and real data case studies dealing with mining data.

6.2. Recommendation for Future Work

This thesis developed mainly the coherent methodologies for the estimation of geospatial data using ML models and applied them to the simulated, public, and real datasets. It was presented two methodologies include (1) an improved ML model called the ESL model that yielded similar results to kriging while not needing the assumption of stationarity, and (2) a combined model that yielded more accurate, consistent, and unbiased spatial

estimation in complex domains. However, the following recommendations for future research are given;

- The ML models do not estimate directly uncertainties around the unknown values. Therefore, additional effort is required to obtain the estimated uncertainty which has critical importance in the evaluation of a geospatial estimation method.
- The proposed methods have been trained against point support data. However, the estimations should also be produced at various block support sizes.
- The grid search algorithm is one of the most commonly used algorithms to tune the values of hyper-parameters. However, it is employed manually and needs an iterative process that makes it labor-intensive. Therefore, it is planned to employ advanced algorithms, such as GA, that select hyper-parameter values automatically and more efficiently.
- Kriging methods such as IK, universal kriging, factorial kriging could be evaluated for different types of problems.
- The direction of anisotropy and anisotropy ratio could be incorporated into the procedure of the ESL model.
- The original coordinate system could be rotated in three directions $\{\mathbf{X}, \mathbf{Y}, \mathbf{Z}\}$ to orient anisotropy in 3D space.

REFERENCES

- Abzalov, M., 2016, Applied mining geology. Springer, p 221.
- Al-Anazi, A., Gates, I., 2010a, On the capability of support vector machines to classify lithology from well logs, *Natural Resources Research*, 19,2, 125-139.
- Al-Anazi, A., Gates, I., 2010b, Support vector regression for porosity prediction in a heterogeneous reservoir: A comparative study, *Computers & Geosciences*, 36,12, 1494-1503.
- Alpaydin, E., 2014, Introduction to Machine Learning. The MIT Press.
- An, S., Liu, W., Venkatesh, S., 2007, Fast cross-validation algorithms for least squares support vector machine and kernel ridge regression, *Pattern Recognition*, 40,8, 2154-2162.
- Armstrong, M., 1998, Basic linear geostatistics. Springer Science & Business Media.
- Armstrong, M., Jabin, R., 1981, Variogram models must be positive-definite, *Journal of the International Association for Mathematical Geology*, 13,5, 455-459.
- Atkinson, P.M., Tate, N.J., 2000, Spatial scale problems and geostatistical solutions: a review, *The Professional Geographer*, 52,4, 607-623.
- Atteia, O., Dubois, J.P., Webster, R., 1994, Geostatistical analysis of soil contamination in the Swiss Jura, *Environmental Pollution*, 86,3, 315-27.
- Awad, M., Khanna, R., 2015, Support vector regression, *Efficient learning machines*. Springer, p. 67-80.
- Badel, M., Angorani, S., Panahi, M.S., 2011, The application of median indicator kriging and neural network in modeling mixed population in an iron ore deposit, *Computers & Geosciences*, 37,4, 530-540.
- Bergstra, J., Bengio, Y., 2012, Random search for hyper-parameter optimization, *The Journal of Machine Learning Research*, 13,1, 281-305.
- Biau, G., Scornet, E., 2016, A random forest guided tour, *Test*, 25,2, 197-227.
- Bishop, C.M., 1995, Neural networks for pattern recognition. Oxford university press.
- Bishop, C.M., 2006, Pattern recognition and machine learning. Springer.

REFERENCES (continued)

- Boggs, P.T., Tolle, J.W., 1995, Sequential quadratic programming, *Acta numerica*, 4,1, 1-51.
- Boggs, P.T., Tolle, J.W., 2000, Sequential quadratic programming for large-scale nonlinear optimization, *Journal of computational and applied mathematics*, 124,1-2, 123-137.
- Bose, B., 2020, Performance Metrics: Regression Model. *Analytics Vidhya*, <https://medium.com/analytics-vidhya/performance-metrics-regression-model-69f68a18504f>, Accessed: 31.01.2021.
- Breiman, L., 1996, Bagging predictors, *Machine learning*, 24,2, 123-140.
- Breiman, L., 1998, Arcing classifier (with discussion and a rejoinder by the author), *The annals of statistics*, 26,3, 801-849.
- Breiman, L., 2001, Random forests, *Machine learning*, 45,1, 5-32.
- Breiman, L., Friedman, J., Stone, C.J., Olshen, R.A., 1984, *Classification and regression trees*. CRC press.
- Brownlee, J., 2016, *Deep learning with Python: develop deep learning models on Theano and TensorFlow using Keras*. Machine Learning Mastery.
- Cameron, A.C., Windmeijer, F.A., 1996, R-squared measures for count data regression models with applications to health-care utilization, *Journal of Business Economic Statistics*, 14,2, 209-220.
- Chatterjee, S., Bandopadhyay, S., Machuca, D., 2010, Ore grade prediction using a genetic algorithm and clustering based ensemble neural network model, *Mathematical Geosciences*, 42,3, 309-326.
- Chatterjee, S., Bandopadhyay, S., 2011, Goodnews Bay Platinum resource estimation using least squares support vector regression with selection of input space dimension and hyperparameters, *Natural Resources Research*, 20,2, 117-129.
- Chatterjee, S., Hadi, A.S., 2015, *Regression analysis by example*. John Wiley & Sons.
- Cheng, B., Titterton, D.M., 1994, Neural networks: A review from a statistical perspective, *Statistical science*, 2-30.
- Chiles, J.-P., Delfiner, P., 2009, *Geostatistics: modeling spatial uncertainty*, 497. John Wiley & Sons.

REFERENCES (continued)

- Chinneck, J.W., 2006, Practical optimization: a gentle introduction, Systems and Computer Engineering, Carleton University, Ottawa, 11.
- Conway, D., White, J., 2012, Machine learning for hackers. O'Reilly Media, Inc.
- Cortes, C., Vapnik, V., 1995, Support-vector networks, Machine learning, 20,3, 273-297.
- Cover, T., Hart, P., 1967, Nearest neighbor pattern classification, IEEE transactions on information theory, 13,1, 21-27.
- Dangeti, P., 2017, Statistics for machine learning. Packt Publishing Ltd.
- David, M., 1977, Geostatistical ore reserve estimation, 2. Elsevier Science, p 364.
- Deutsch, C.V., 1989, DECLUS: a FORTRAN 77 program for determining optimum spatial declustering weights, Computers & Geosciences, 15,3, 325-332.
- Deutsch, C.V., 2003, Fundamental of Geostatistics: Principles and Hands on Practice Short Course, University of Alberta, Edmonton
- Deutsch, C.V., 2018, Partitioning Drill Hole Data into K Folds, CCG Annual Report 20.
- Deutsch, C.V., 2020, Cell declustering parameter selection, Geostatistics Lessons. Retrieved from <http://www.geostatisticslessons.com/lessons/celldeclustering>, Accessed:15.11.2020.
- Deutsch, C.V., Journel, A.G., 1998, GSLIB: Geostatistical software library and user's guide. Oxford University Press., New York, NY.
- Deutsch, J., Szymanski, J., Deutsch, C., 2014, Checks and measures of performance for kriging estimates, Journal of the Southern African Institute of Mining and Metallurgy, 114,3, 223-223.
- Deutsch, M., 2015, The Angle Specification for GSLIB Software, Geostatistics Lessons. Retrieved from <http://www.geostatisticslessons.com/lessons/celldeclustering>, Accessed:15.10.2020.
- Doan, T., Kalita, J., 2015, Selecting machine learning algorithms using regression models, 2015 IEEE International Conference on Data Mining Workshop (ICDMW). IEEE, p. 1498-1505.

REFERENCES (continued)

- Drucker, H., Burges, C.J., Kaufman, L., Smola, A.J., Vapnik, V., 1997, Support vector regression machines, *Advances in neural information processing systems*, p. 155-161.
- Dutta, S., Bandopadhyay, S., Ganguli, R., Misra, D., 2010, Machine learning algorithms and their application to ore reserve estimation of sparse and imprecise data, *Journal of Intelligent Learning Systems Applications*, 2,02, 86.
- Efron, B., Tibshirani, R., 1986, Bootstrap methods for standard errors, confidence intervals, and other measures of statistical accuracy, *Statistical science*, 54-75.
- Fernandes, L., 2020, Evaluate the Performance of a Regression Model. *openclassrooms*, <https://openclassrooms.com/en/courses/6401081-improve-the-performance-of-a-machine-learning-model/6519016-evaluate-the-performance-of-a-regression-model>, Accessed: 02.03.2021.
- Fouedjio, F., Klump, J., 2019, Exploring prediction uncertainty of spatial data in geostatistical and machine learning approaches, *Environmental Earth Sciences*, 78,1, 38.
- Frank, M., Wolfe, P., 1956, An algorithm for quadratic programming, *Naval research logistics quarterly*, 3,1-2, 95-110.
- Freund, Y., Schapire, R.E., 1997, A decision-theoretic generalization of on-line learning and an application to boosting, *Journal of computer and system sciences*, 55,1, 119-139.
- Freund, Y., Schapire, R.E., Singer, Y., Warmuth, M.K., 1997, Using and combining predictors that specialize, *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, p. 334-343.
- Friedman, J., Hastie, T., Tibshirani, R., 2000, Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors), *The annals of statistics*, 28,2, 337-407.
- Friedman, J., Hastie, T., Tibshirani, R., 2001, *The elements of statistical learning*. vol. 1 Springer series in statistics, New York.
- Friedman, J.H., 2001, Greedy function approximation: a gradient boosting machine, *Annals of statistics*, 1189-1232.
- Geurts, P., Ernst, D., Wehenkel, L., 2006, Extremely randomized trees, *Machine learning*, 63,1, 3-42.

REFERENCES (continued)

- Goovaerts, P., 1997, Geostatistics for natural resources evaluation. Oxford University Press on Demand.
- Goswami, A.D., Mishra, M.K., Patra, D., 2016, Adapting pattern recognition approach for uncertainty assessment in the geologic resource estimation for Indian iron ore mines, 2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES). IEEE, pp. 1816-1821.
- Gringarten, E., Deutsch, C.V., 2001, Teacher's aide variogram interpretation and modeling, *Mathematical Geology*, 33,4, 507-534.
- Hackeling, G., 2017, Mastering Machine Learning with scikit-learn. Packt Publishing Ltd.
- Hansen, J.V., 1998, Combining predictors: Some old methods and a new method, *CiteSeerx*.
- Harrington, P., 2012, Machine learning in action. Manning Publications Co.
- Hastie, T., Tibshirani, R., Friedman, J., 2009, The elements of statistical learning: data mining, inference, and prediction. Springer Science & Business Media.
- Haykin, S.S., 2009, Neural networks and learning machines. New York: Prentice Hall.
- Hengl, T., Nussbaum, M., Wright, M.N., Heuvelink, G.B., Gräler, B., 2018, Random forest as a generic framework for predictive modeling of spatial and spatio-temporal variables, *PeerJ*, 6, e5518.
- Hoffmann, L.D., Bradley, G.L., Rosen, K.H., 1989, Calculus for business, economics, and the social and life sciences. McGraw-Hill.
- Hohn, M.E., 2000, Geostatistics and Petroleum Geology. Taylor & Francis Group.
- Isaaks, E.H., Srivastava, M.R., 1989, Applied geostatistics, 551.72 ISA.
- Isaaks, E.H., 1992, The application of Monte Carlo methods to the analysis of spatially correlated data. Ph.D. Thesis, Stanford University, p 213.
- Jahangiri, M., Ghavami Riabi, S.R., Tokhmechi, B., 2018, Estimation of geochemical elements using a hybrid neural network-Gustafson-Kessel algorithm, *Journal of Mining Environmental Modelling & Software*, 9,2, 499-511.

REFERENCES (continued)

- Jalloh, A.B., Kyuro, S., Jalloh, Y., Barrie, A.K., 2016, Integrating artificial neural networks and geostatistics for optimum 3D geological block modeling in mineral reserve estimation: A case study, *International Journal of Mining Science Technology*, 26,4, 581-585.
- James, G., Witten, D., Hastie, T., Tibshirani, R., 2013, *An introduction to statistical learning*, 112. Springer.
- Jafrasteh, B., Fathianpour, N., Suárez, A., 2018, Comparison of machine learning methods for copper ore grade estimation, *Computational Geosciences*, 22,5, 1371-1388.
- Journel, A.G., 1983, Nonparametric estimation of spatial distributions, *Journal of the International Association for Mathematical Geology*, 15,3, 445-468.
- Journel, A.G., Huijbregts, C.J., 1978, *Mining geostatistics*, 600. Academic press London.
- Journel, A.G., Rossi, M., 1989, When do we need a trend model in kriging?, *Mathematical Geology*, 21,7, 715-739.
- Kanevski, M. et al., 2002, Support vector machines for classification and mapping of reservoir data, *Soft Computing for Reservoir Characterization and Modeling*. Springer, pp. 531-558.
- Kanevski, M., 2013, *Advanced mapping of environmental data*. John Wiley & Sons.
- Kanevski, M., Pozdnoukhov, A., Pozdnukhov, A., Timonin, V., 2009, *Machine learning for spatial environmental data: theory, applications, and software*. EPFL press.
- Kapageridis, I.K., 1999, *Application of artificial neural network systems to grade estimation from exploration data*, University of Nottingham.
- Kapageridis, I.K., 2005, Input space configuration effects in neural network-based grade estimation, *Computers Geosciences*, 31,6, 704-717.
- Kaplan, U.E., Topal, E., 2020, A New Ore Grade Estimation Using Combine Machine Learning Algorithms, *Minerals*, 10,10, 847.
- Kıvrak, H., 2018, g-ROKY: Maden Kaynak Kestiriminde Homojen Ortam Modellenmesine Yönelik Yeni Bir Yöntemin Geliştirilmesi, *Doctoral Thesis*, Hacettepe Üniversitesi, Ankara.
- Kohavi, R., 1995, A study of cross-validation and bootstrap for accuracy estimation and model selection, *Ijcai*. Montreal, Canada, p. 1137-1145.

REFERENCES (continued)

- Koike, K., Matsuda, S., Suzuki, T., Ohmi, M., 2002, Neural network-based estimation of principal metal contents in the Hokuroku district, northern Japan, for exploring Kuroko-type deposits, *Natural Resources Research*, 11,2, 135-156.
- Krige, D.G., 1951, A statistical approach to some basic mine valuation problems on the Witwatersrand, *Journal of the Southern African Institute of Mining and Metallurgy*, 52,6, 119-139.
- Kuhn, M., Johnson, K., 2013, *Applied predictive modeling*, 26. Springer.
- Kuhn, M., Johnson, K., 2019, *Feature engineering and selection: A practical approach for predictive models*. CRC Press.
- Kvålseth, T.O., 1985, Cautionary note about R², *The American Statistician*, 39,4, 279-285.
- Larose, D.T., Larose, C.D., 2014, *Discovering knowledge in data: an introduction to data mining*, 4. John Wiley & Sons.
- Lavesson, N., Davidsson, P., 2006, Quantifying the impact of learning algorithm parameter tuning, *AAAI*, p. 395-400.
- Lee, J., Leyffer, S., 2011, *Mixed integer nonlinear programming*, 154. Springer Science & Business Media.
- Leuenberger, M., 2018, *Environmental Data Mining using Machine Learning Algorithms: Methodological Developments and Case Studies*, Ph.D. Thesis, Université de Lausanne, Faculté des géosciences et de l'environnement, 143 p.
- Leuenberger, M., Kanevski, M., 2015, Extreme Learning Machines for spatial environmental data, *Computers & Geosciences*, 85, 64-73.
- Li, J., Heap, A.D., Potter, A., Daniell, J.J., 2011, Application of machine learning methods to spatial interpolation of environmental variables, *Environmental Modelling Software*, 26,12, 1647-1659.
- Li, X.-l., Li, L.-h., Zhang, B.-l., Guo, Q.-j., 2013, Hybrid self-adaptive learning based particle swarm optimization and support vector regression model for grade estimation, *Neurocomputing*, 118, 179-190.
- Li, X.-l., Xie, Y.-l., Guo, Q.-j., Li, L.-h., 2010, Adaptive ore grade estimation method for the mineral deposit evaluation, *Mathematical Computer Modelling*, 52,11-12, 1947-1956.

REFERENCES (continued)

- Liashchynskiy, P., Liashchynskiy, P., 2019, Grid search, random search, genetic algorithm: A big comparison for nas, arXiv:1912.06059.
- Loterman, G., Brown, I., Martens, D., Mues, C., Baesens, B., 2012, Benchmarking regression algorithms for loss given default modeling, *International Journal of Forecasting*, 28,1, 161-170.
- Luenberger, D.G., 1997, *Optimization by vector space methods*. John Wiley & Sons.
- Mahmoudabadi, H., Izadi, M., Menhaj, M.B., 2009, A hybrid method for grade estimation using genetic algorithm and neural networks, *Computational Geosciences*, 13,1, 91-101.
- Manchuk, J.G., Deutsch, C.V., 2011, A Short Note on Trend Modeling using Moving Windows, Centre for Computational Geostatistics, University of Alberta, Edmonton, Canada, CCG Paper 2011-403.
- Matias, J., Vaamonde, A., Taboada, J., González-Manteiga, W., 2004, Comparison of kriging and neural networks with application to the exploitation of a slate mine, *Mathematical Geology*, 36,4, 463-486.
- Matheron, G., 1963, Principles of geostatistics, *Economic geology*, 58,8, 1246-1266.
- Matheron, G., 1971, *The Theory of Regionalized Variables and its Applications*. vol. 5, Paris: École National Supérieure des Mines, 211.
- Matin, S.S., Chelgani, S.C., 2016, Estimation of coal gross calorific value based on various analyses by random forest method, *Fuel*, 177, 274-278.
- Mitchell, T., 1996, *Machine Learning*. McGraw-Hill Science, p 432.
- Molinaro, A.M., Simon, R., Pfeiffer, R.M., 2005, Prediction error estimation: a comparison of resampling methods, *Bioinformatics*, 21,15, 3301-7.
- Murty, K.G., Yu, F.-T., 1988, Linear complementarity, linear and nonlinear programming, 3. Citeseer.
- Müller, A., Guido, S., 2016, *Introduction to machine learning with python*. O'Reilly Media, Incorporated.
- Naimi, A.I., Balzer, L.B., 2018, Stacked generalization: an introduction to super learning, *European journal of epidemiology*, 33,5, 459-464.

REFERENCES (continued)

- Neufeld, C., Deutsch, C.V., 2004, Angle Rotations in GSLIB, CCG Paper, 2004.
- Noble, W.S., 2006, What is a support vector machine?, *Nature biotechnology*, 24,12, 1565-1567.
- Nocedal, J., Wright, S.J., 2006, Sequential quadratic programming, *Numerical optimization*, 529-562.
- Olea, R.A., 1991, *Geostatistical glossary and multilingual dictionary*. Oxford University Press on Demand.
- Olea, R.A., 2006, A six-step practical approach to semivariogram modeling, *Stochastic Environmental Research and Risk Assessment*, 20,5, 307-318.
- Pedregosa, F. et al., 2011, Scikit-learn: Machine learning in Python, *the Journal of machine Learning research*, 12, 2825-2830.
- Picard, R.R., Berk, K.N., 1990, Data splitting, *The American Statistician*, 44,2, 140-147.
- Polley, E.C., Van Der Laan, M.J., 2010, Super learner in prediction, *Berkeley Division of Biostatistics Working Paper Series*.
- Probst, P., Boulesteix, A.-L., Bischl, B., 2019, Tunability: Importance of Hyperparameters of Machine Learning Algorithms, *J. Mach. Learn. Res.*, 20,53, 1-32.
- pygeostat, 2020, <http://www.ccgliberta.com/pygeostat>, Accessed: 01.12.2020.
- Pyrz, M.J., Deutsch, C.V., 2003, Declustering and debiasing, *Newsletter*, 19.
- Pyrz, M.J., Deutsch, C.V., 2014, *Geostatistical reservoir modeling*. Oxford University Press, New York.
- Quinlan, J.R., 1986, Induction of decision trees, *Machine learning*, 1,1, 81-106.
- Rendu, J., Readdy, L., 1982, Geology and the semivariogram. A critical relationship, *APCOM*, p. 771-783.
- Ridgeway, G., 2007, *Generalized Boosted Models: A guide to the gbm package*, Update, 1,1, 2007.
- Rijmenam, M.V., 2019, 7 Steps to Machine Learning: How to Prepare for an Automated Future, *DataSeries*, <https://medium.com/dataseries/7-steps-to-machine-learning-how-to-prepare-for-an-automated-future-78c7918cb35d>, Accessed:05.02.2021.

REFERENCES (continued)

- Rossi, M.E., Deutsch, C.V., 2013, Mineral resource estimation. Springer Science & Business Media.
- Rumelhart, D.E., Hinton, G.E., Williams, R.J., 1986, Learning representations by back-propagating errors, *nature*, 323,6088, 533-536.
- Russell, S., Norvig, P., 2002, Artificial intelligence: a modern approach.
- Samanta, B., Bandopadhyay, S., Ganguli, R., Dutta, S., 2004, Sparse data division using data segmentation and Kohonen network for neural network and geostatistical ore grade modeling in Nome offshore placer deposit, *Natural resources research*, 13,3, 189-200.
- Samanta, B., Ganguli, R., Bandopadhyay, S., 2005, Comparing the predictive performance of neural networks with ordinary kriging in a bauxite deposit, *Mining Technology*, 114,3, 129-139.
- Sagi, O., Rokach, L., 2018, Ensemble learning: A survey, *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8,4, e1249.
- Samson, M.J., 2019, Mineral Resource Estimates with Machine Learning and Geostatistics, University of Alberta, Edmonton, Canada.
- Samson, M., Deutsch, C.V., 2021, A Hybrid Estimation Technique Using Elliptical Radial Basis Neural Networks and Cokriging, *Mathematical Geosciences*.
- Samui, P., Sitharam, T., 2010, Applicability of statistical learning algorithms for spatial variability of rock depth, *Mathematical Geosciences*, 42,4, 433-446.
- Schapire, R.E., 2013, Explaining adaboost, *Empirical inference*. Springer, p. 37-52.
- Shalev-Shwartz, S., Ben-David, S., 2014, Understanding machine learning: From theory to algorithms. Cambridge University Press.
- Shapiro, S.S., Wilk, M.B., 1965, An analysis of variance test for normality (complete samples), *Biometrika*, 52,3/4, 591-611.
- Smola, A.J., Schölkopf, B., 2004, A tutorial on support vector regression, *Statistics computing*, 14,3, 199-222.
- Srivastava, R.M., 2013, Geostatistics: a toolkit for data analysis, spatial prediction and risk management in the coal industry, *International Journal of Coal Geology*, 112, 2-13.

REFERENCES (continued)

- Tahmasebi, P., Hezarkhani, A., 2011, Application of a modular feedforward neural network for grade estimation, *Natural resources research*, 20,1, 25-32.
- Van der Laan, M.J., Polley, E.C., Hubbard, A.E., 2007, Super learner, *Stat Appl Genet Mol Biol*, 6,1, Article25.
- Vapnik, V., 1998, *Statistical learning theory*. Wiley, New York, John Wiley and Sons.
- Vapnik, V., 2013, *The nature of statistical learning theory*. Springer science & business media.
- Vasylchuk, Y.V., Deutsch, C.V., 2017, A Dataset for Testing Geostatistical Algorithms (Drillholes), *CCG Annual Report 19*, Paper 405.
- Virtanen, P. et al., 2020, SciPy 1.0: fundamental algorithms for scientific computing in Python, *Nature methods*, 17,3, 261-272.
- Wackernagel, H., 2013, *Multivariate geostatistics: an introduction with applications*. Springer Science & Business Media.
- Wang, H., Guan, Y., Reich, B., 2019, Nearest-Neighbor Neural Networks for Geostatistics, *International Conference on Data Mining Workshops (ICDMW)*. IEEE, Beijing, China, p. 196-205.
- Webster, R., Atteia, O., Dubois, J.P., 1994, Coregionalization of trace metals in the soil in the Swiss Jura, *European Journal of Soil Science*, 45,2, 205-218.
- Weerts, H.J., Mueller, A.C., Vanschoren, J., 2020, Importance of tuning hyperparameters of machine learning algorithms, *arXiv*.
- Witten, I., Frank, E., Hall, M., Pal, C., 2016, *Data mining: practical machine learning tools and techniques*. Todd Green.
- Witten, I.H., Frank, E., 2002, *Data mining: practical machine learning tools and techniques with Java implementations*, *Acm Sigmod Record*, 31,1, 76-77.
- Wolpert, D.H., 1992, Stacked generalization, *Neural networks*, 5,2, 241-259.
- Wong, J., Manderson, T., Abrahamowicz, M., Buckeridge, D.L., Tamblyn, R., 2019, Can hyperparameter tuning improve the performance of a super learner?: a case study, *Epidemiology*, 30,4, 521.
- Wu, X., Zhou, Y., 1993, Reserve estimation using neural network techniques, *Computers & Geosciences*, 19,4, 567-575.

REFERENCES (continued)

- Wu, X. et al., 2008, Top 10 algorithms in data mining, *Knowledge and information systems*, 14,1, 1-37.
- Yamamoto, J.K., 2000, An alternative measure of the reliability of ordinary kriging estimates, *Mathematical Geology*, 32,4, 489-509.
- Yang, X.-S., 2016, *Engineering mathematics with examples and applications*. Academic Press.
- Yufeng, G., 2017, *The 7 Steps of Machine Learning*, Google Cloud, <https://towardsdatascience.com/the-7-steps-of-machine-learning-2877d7e5548e>, Accessed:12.03.2021.
- Zhang, G., Hu, M.Y., Patuwo, B.E., Indro, D.C., 1999, Artificial neural networks in bankruptcy prediction: General framework and cross-validation analysis, *European journal of operational research*, 116,1, 16-32.
- Zhang, X., Song, S., Li, J., Wu, C., 2013, Robust LS-SVM regression for ore grade estimation in a seafloor hydrothermal sulphide deposit, *Acta Oceanologica Sinica*, 32,8, 16-25.
- Zhou, Z.-H., 2012, *Ensemble methods: foundations and algorithms*. Chapman and Hall/CRC.

APPENDIX

Appendix – A. An implementation example of the SL model

Appendix - B. The Grid Search Algorithm Python Codes

Appendix - C. The SQP Codes



APPENDIX – A

An implementation example of the SL model

This appendix explains the SL model step-by-step with an example using a standard ML dataset (<https://raw.githubusercontent.com/jbrownlee/Datasets/master/housing.csv>), called housing dataset, comprising 506 samples with 13 numerical features and one output. In this example, three different ML models, which are linear regression, kNN, and BAG, are used as base models.

1. Import dataset

A continuous output, Y , and 13 features, from X_1 to X_{13} , are important from the housing dataset.

Split the data so that 80% is used for training the model and 20% is held back to evaluate the final SL and base models.

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	Y
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.456	6.998	45.8	6.0622	3	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.456	7.147	54.2	6.0622	3	222.0	18.7	396.90	5.33	36.2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
501	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273.0	21.0	391.99	9.67	22.4
502	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2675	1	273.0	21.0	396.90	9.08	20.6
503	0.06076	0.0	11.93	0	0.573	6.978	91.0	2.1676	1	273.0	21.0	396.90	5.64	23.9
504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273.0	21.0	393.45	6.48	22.0
505	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273.0	21.0	396.90	7.88	11.9

506 rows × 14 columns

Training:

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	Y
0	8.44405	0.0	18.10	0.0	0.584	6.425	74.8	2.2004	24.0	666.0	20.2	97.96	12.03	16.1
1	0.30347	0.0	7.38	0.0	0.493	6.312	28.9	5.4159	5.0	287.0	19.8	396.90	6.15	23.0
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7
3	8.20058	0.0	18.10	0.0	0.713	5.936	80.3	2.7792	24.0	666.0	20.2	3.50	16.94	13.6
4	15.87440	0.0	18.10	0.0	0.671	6.545	99.1	1.5192	24.0	666.0	20.2	396.90	21.08	10.9
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
399	0.12083	0.0	2.89	0.0	0.445	8.069	76.0	3.4952	2.0	276.0	19.0	396.90	4.21	38.7
400	0.62739	0.0	8.14	0.0	0.538	5.834	66.5	4.4966	4.0	307.0	21.0	395.62	8.47	19.9
401	0.09164	0.0	10.81	0.0	0.413	6.065	7.8	5.2873	4.0	305.0	19.2	390.91	5.52	22.8
402	4.83567	0.0	18.10	0.0	0.583	5.905	53.2	3.1523	24.0	666.0	20.2	388.22	11.45	20.6
403	0.01311	60.0	1.22	0.0	0.403	7.246	21.9	6.6966	5.0	226.0	17.9	395.93	4.81	35.4

404 rows × 14 columns

Testing:

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	Y
0	8.98296	0.0	18.10	1.0	0.770	6.212	97.4	2.1222	24.0	666.0	20.2	377.73	17.60	17.8
1	0.09068	45.0	3.44	0.0	0.437	6.951	21.5	6.4798	5.0	398.0	15.2	377.68	5.10	37.0
2	0.07151	0.0	4.49	0.0	0.449	6.121	56.8	3.7479	3.0	247.0	18.5	395.15	8.44	22.2
3	0.08829	12.5	7.87	0.0	0.524	6.012	86.6	5.6605	5.0	311.0	15.2	396.60	12.43	22.9
4	5.69175	0.0	18.10	0.0	0.583	6.114	79.8	3.5459	24.0	666.0	20.2	392.68	14.98	19.1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
97	0.03502	80.0	4.95	0.0	0.411	6.861	27.9	5.1167	4.0	245.0	19.2	396.90	3.33	28.5
98	0.19133	22.0	5.95	0.0	0.431	5.605	70.2	7.6549	7.0	330.0	19.1	388.13	18.46	18.5
99	0.04680	52.5	5.32	0.0	0.405	6.315	45.6	7.3172	6.0	293.0	18.6	396.90	7.60	22.3
100	9.91655	0.0	18.10	0.0	0.693	5.852	77.8	1.5004	24.0	666.0	20.2	338.16	29.97	6.3
101	0.32543	0.0	21.89	0.0	0.624	6.431	98.8	1.8125	4.0	437.0	21.2	396.90	15.39	18.0

102 rows × 14 columns

2. Split training set into 5 folds Data is split into 5 folds	output Predictors (variables, features) 1 2 3 4 5
3. Fit base models for the first cross-validation- fold All base models are trained on the cross-validation folds except the last one, which is used as a validation dataset. Note: Each fold serves as the validation set one time.	LR (1 2 3 4) kNN (1 2 3 4) BAG (1 2 3 4)
4. Obtain estimations for the first cross-validation fold The validation data, the last fold, is got used to obtain the first set of cross-validated estimations. There are 404 samples, that give three vectors of length 84: a set of predictions for each base model.	Linear Regression kNN BAG 320 29.333501 22.02 25.88 321 13.321766 12.34 11.16 322 20.329717 21.10 19.92 323 16.857586 21.10 18.48 324 43.434356 41.54 48.18 399 35.707520 23.66 41.77 400 19.304630 21.28 20.66 401 24.280019 30.08 21.87 402 21.879824 21.50 22.02 403 30.611603 41.54 33.56 84 rows × 3 columns 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
5. Obtain cross-validation estimations for the entire data set The base models are fit on every fold so that 404 estimations are obtained, one for every data in samples.	Linear Regression kNN BAG 0 22.825575 12.64 19.06 1 23.669503 27.92 24.59 2 30.091829 21.14 34.23 3 12.787078 11.56 16.44 4 20.091555 11.78 13.50 399 35.707520 23.66 41.77 400 19.304630 21.28 20.66 401 24.280019 30.08 21.87 402 21.879824 21.50 22.02 403 30.611603 41.54 33.56 404 rows × 3 columns 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5

6. Select and calculate the loss function of interest through the meta learner.

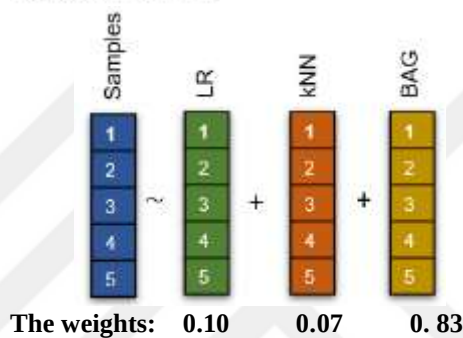
Meta learner takes information from all of the base models and tries to minimize a final loss function (MSE) between true output (blue column) and cross-validation estimates to combine the base models.

Meta learner is selected as LR whose features are estimations of the base models and output is the true output.

This meta learner provides the coefficients, or weights, to apply to each of the base models.

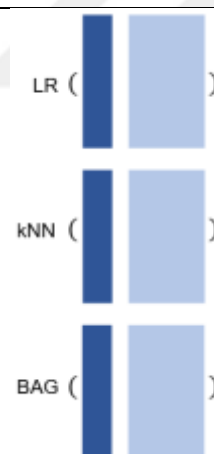
	Samples	Linear Regression	kNN	BAG
0	16.1	22.825575	12.64	19.06
1	23.0	23.669503	27.92	24.59
2	34.7	30.091829	21.14	34.23
3	13.5	12.787078	11.56	16.44
4	10.9	20.091555	11.78	13.50
...	...	...	...	...
399	38.7	35.707520	23.66	41.77
400	19.9	19.304630	21.28	20.66
401	22.8	24.280019	30.08	21.87
402	20.6	21.879824	21.50	22.02
403	35.4	30.611603	41.54	33.56

404 rows × 4 columns



7. Fit base models on the entire data set

To implement it and produce final estimates, it is needed to fit the base models on the full data set.



8. Obtain estimates from each base model on the entire data set



9. Use meta learner fit to weight base models The estimates from the full data set are input to the meta learner, where the output will be a final estimation for Y.	meta learner (+ +) >> estimate output <table><thead><tr><th colspan="2">SL Estimates</th></tr></thead><tbody><tr><td>0</td><td>19.020193</td></tr><tr><td>1</td><td>23.689972</td></tr><tr><td>2</td><td>35.438415</td></tr><tr><td>3</td><td>12.617502</td></tr><tr><td>4</td><td>11.115877</td></tr><tr><td>...</td><td>...</td></tr><tr><td>399</td><td>40.453412</td></tr><tr><td>400</td><td>20.065355</td></tr><tr><td>401</td><td>23.976392</td></tr><tr><td>402</td><td>21.109390</td></tr><tr><td>403</td><td>36.133626</td></tr></tbody></table> 404 rows × 1 columns			SL Estimates		0	19.020193	1	23.689972	2	35.438415	3	12.617502	4	11.115877	...	...	399	40.453412	400	20.065355	401	23.976392	402	21.109390	403	36.133626												
SL Estimates																																							
0	19.020193																																						
1	23.689972																																						
2	35.438415																																						
3	12.617502																																						
4	11.115877																																						
...	...																																						
399	40.453412																																						
400	20.065355																																						
401	23.976392																																						
402	21.109390																																						
403	36.133626																																						
10. Obtain estimates on new data Step 8 and Step 9 are modified to accommodate any new sample(s). To estimate new data which can be test dataset (20% part of the original dataset): - Use the fits from each base model to obtain the base model estimates for the new sample(s). - Plug those base model estimates into the meta learner fit.	<table><thead><tr><th></th><th>Test output</th><th>SL Estimates</th></tr></thead><tbody><tr><td>0</td><td>17.8</td><td>13.944209</td></tr><tr><td>1</td><td>37.0</td><td>30.524815</td></tr><tr><td>2</td><td>22.2</td><td>23.242692</td></tr><tr><td>3</td><td>22.9</td><td>20.219870</td></tr><tr><td>4</td><td>19.1</td><td>19.218799</td></tr><tr><td>...</td><td>...</td><td>...</td></tr><tr><td>97</td><td>28.5</td><td>33.805538</td></tr><tr><td>98</td><td>18.5</td><td>19.238638</td></tr><tr><td>99</td><td>22.3</td><td>24.287669</td></tr><tr><td>100</td><td>6.3</td><td>9.831798</td></tr><tr><td>101</td><td>18.0</td><td>17.317423</td></tr></tbody></table> 102 rows × 2 columns				Test output	SL Estimates	0	17.8	13.944209	1	37.0	30.524815	2	22.2	23.242692	3	22.9	20.219870	4	19.1	19.218799	...	...	...	97	28.5	33.805538	98	18.5	19.238638	99	22.3	24.287669	100	6.3	9.831798	101	18.0	17.317423
	Test output	SL Estimates																																					
0	17.8	13.944209																																					
1	37.0	30.524815																																					
2	22.2	23.242692																																					
3	22.9	20.219870																																					
4	19.1	19.218799																																					
...	...	...																																					
97	28.5	33.805538																																					
98	18.5	19.238638																																					
99	22.3	24.287669																																					
100	6.3	9.831798																																					
101	18.0	17.317423																																					
11. Performance Evaluation	LR kNN BAG SL model	RMSE	3.912 5.716 3.139 2.932																																				

APPENDIX – B

The Grid Search Algorithm Python Codes

This appendix provides the grid search algorithm codes written in Python programming language using related packages and by following steps which are presented in Section 3.3.1.

```
1 # Import Required Packages
2 import numpy as np
3 import pandas as pd
4 from sklearn.model_selection import GridSearchCV
5 from sklearn.neighbors import KNeighborsRegressor
6 from sklearn.tree import DecisionTreeRegressor
7 from sklearn.svm import SVR
8 from sklearn.ensemble import GradientBoostingRegressor
9 from sklearn.ensemble import BaggingRegressor
10 from sklearn.ensemble import RandomForestRegressor
11 from sklearn.ensemble import ExtraTreesRegressor
12 from sklearn.model_selection import cross_val_score
13 from keras.models import Sequential
14 from keras.layers import Dense
15 from keras.wrappers.scikit_learn import KerasRegressor
16 from sklearn.ensemble import AdaBoostRegressor
17 from sklearn.ensemble import ExtraTreesRegressor
18 from sklearn.model_selection import RepeatedKFold

1 # Function to create model, required for KerasRegressor
2 def wider_model(optimizer='adam', activation = 'sigmoid',neurons=1, hidden_layers=1):
3     # create model
4     model = Sequential()
5     model.add(Dense(neurons, input_dim=2, kernel_initializer='normal', activation='relu'))
6
7     for i in range(hidden_layers):
8         # Add one hidden layer
9         model.add(Dense(neurons, activation=activation))
10
11     model.add(Dense(1, kernel_initializer='normal')) # output layer
12     # Compile model
13     model.compile(loss='mean_squared_error', optimizer='adam')
14     return model
15
16 # fix random seed for reproducibility
17 seed = 7
18 np.random.seed(seed)
19
20 # Load normalized dataset
21 data = np.asarray(testflv.data)
22
23 # split into input (X) and output (Y) variables
24 X = data[:,0:2] # This example for 2D dataset
25 y = data[:,2:3]
26 y = y.ravel()
27
28 # create model
29 model_ANN = KerasRegressor(build_fn=wider_model, verbose=0, epochs=200,batch_size=60)
30
```

```

30
31 # define the grid search parameters for just ANN
32 neurons = [5,10,15,20,25,30]
33 hidden_layers = [2,3,4]
34 batch_size = [10,20,40,60,80,100]
35 epochs = [50,100,200,300,400,500]
36 param_grid = dict(neurons=neurons, hidden_layers=hidden_layers, batch_size=batch_size, epochs=epochs)
37
38 # define split of data
39 kfold = RepeatedKFold(n_splits=5, n_repeats=5, random_state=seed)
40
41 grid = GridSearchCV(estimator=model ANN, param_grid=param_grid, n_jobs=-1, cv=kfold)
42 grid_result = grid.fit(X, y)
43
44 # summarize results
45 print("Best ANN hyperparameters: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
46 means = grid_result.cv_results_['mean_test_score']
47 stds = grid_result.cv_results_['std_test_score']
48 params = grid_result.cv_results_['params']
49 for mean, stdev, param in zip(means, stds, params):
50     print("%f (%f) with: %r" % (mean, stdev, param))

```

```

1 ## Hyperparameter Tuning with Grid Search- All models except NN
2 scoring = 'neg_mean_squared_error'
3
4 # fix random seed for reproducibility
5 seed = 7
6 np.random.seed(seed)
7
8 modelnames = [ 'KNN', 'DT', 'BAG', 'RF', 'EXT', 'ADA', 'GBM', 'SVR' ]
9
10 models_to_run = [KNeighborsRegressor(),
11                  DecisionTreeRegressor(),
12                  BaggingRegressor(random_state=seed),
13                  RandomForestRegressor(random_state=seed),
14                  ExtraTreesRegressor(random_state=seed),
15                  AdaBoostRegressor(),
16                  GradientBoostingRegressor(random_state=seed),
17                  SVR(kernel='rbf', gamma='scale')]
18
19 models_param_grid = [ { # 1st param grid, corresponding to KNN
20                        'n_neighbors': [2,3,4,5]
21                      },
22                      { # 2nd param grid, corresponding to DT
23                        'max_depth' : [2,4,6,8,10]
24                      },
25                      { # 3rd param grid, corresponding to BAG
26                        'n_estimators' : [1000,1500,2000,2500]
27                      },
28                      { # 4th param grid, corresponding to RF
29                        'n_estimators': [1000,1500,2000,2500]
30                      },
31                      { # 5th param grid, corresponding to EXT
32                        'n_estimators': [1000,1500,2000,2500]
33                      },
34                      { # 6th param grid, corresponding to ADA
35                        'n_estimators' : [1000,1500,2000,2500]
36                      },
37                      { # 7th param grid, corresponding to GBM
38                        'n_estimators': [250,500,750,1000],
39                        'learning_rate':[0.01, 0.1, 0.2],
40                        'max_depth' : [4,6,8]
41                      },
42                      { # 8th param grid, corresponding to SVR
43                        'C': [1, 10, 100, 1000],
44                        'epsilon': [0.01, 0.1, 1]
45                      }
46                    ]
47
48
49
50 # define split of data
51 kfold = RepeatedKFold(n_splits=5, n_repeats=5, random_state=seed)
52
53 # Tuning
54 for (i,model), j in zip(enumerate(models_to_run), modelnames):
55     grid = GridSearchCV(model, param_grid=models_param_grid[i], scoring=scoring, cv=kfold)
56     grid_result = grid.fit(X, y)
57     print("Best {} hyperparameters: %f using %s".format(j) % (grid_result.best_score_, grid_result.best_params_))

```

APPENDIX – C

The SQP Python Codes

This appendix describes the codes developed for the implementation of the SQP that optimizes the weights of the presented combination method.

```
1 # import required package
2 from scipy.optimize import minimize
3
4 #define optimization function
5 def objective(weights):
6     b0, b1 = weights[0], weights[1]
7     for weight in zip(weights):
8         final_est = (W ** ((b0 * b1 * SK))) * SL_est + ((1-W) ** ((b0 + b1 * SK))) * SK_est
9     return sqrt(mean_squared_error(y_all, final_prediction))
10
11 starting_values = [0, 0]
12
13 #the weights are bound between
14 bounds = [(-3,3),(-2,2)]
15
16 res = minimize(objective, starting_values, method='SLSQP', bounds=bounds,options={'disp': True, 'ftol':1e-30})
17
18 print('Ensemble Score: {best_score}'.format(best_score=res['fun']))
19 print('Best Weights: {weights}'.format(weights=res['x']))
```



```
1 gs.set_style('pt40')
2
3 # plot the function f(x)
4 x = np.linspace(-3, 3, 100)
5 y = np.linspace(-2, 2, 100)
6
7 X, Y = np.meshgrid(x, y)
8 Z = np.zeros((x.size, y.size))
9 for i in range(x.size):
10     for j in range(y.size):
11         Z[i, j] = objective((x[i], y[j]))
12         if Z[i, j] >= 1:
13             Z[i, j] = 1
14         else:
15             Z[i, j] = Z[i, j]
16
17 fig= plt.figure(figsize=(10,10))
18
19 plt.contour(X, Y, Z.T, 100);
20 plt.colorbar()
21 plt.scatter(res.x[0], res.x[1], c='red',alpha=1)
22 plt.xlabel('b0')
23 plt.ylabel('b1')
24
25 plt.savefig('./Figures/weightsopt.png', bbox_inches = 'tight', dpi = 100)
26 plt.tight_layout()
```