İZMİR BAKIRÇAY ÜNİVERSİTESİ

LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ A.B.D

# PERFORMANCE COMPARISON OF CLASSIFICATION ALGORITHMS FOR DETECTING LEVEL-BASED PLAYER CHURN USING IN-GAME DATA

YÜKSEK LİSANS TEZİ

Muhammed Selim TOKER

Danışman: Dr. Öğr. Üyesi Zekeriya Anıl GÜVEN

Ocak 2024

# PERFORMANCE COMPARISON OF CLASSIFICATION ALGORITHMS FOR DETECTING LEVEL-BASED PLAYER CHURN USING IN-GAME DATA

**Yüksek Lisans Tezi**

**Muhammed Selim TOKER İzmir 2024**

# PERFORMANCE COMPARISON OF CLASSIFICATION ALGORITHMS FOR DETECTING LEVEL-BASED PLAYER CHURN USING IN-GAME DATA

**Muhammed Selim TOKER**

**YÜKSEK LİSANS TEZİ**

**Bilgisayar Mühendisliği Anabilim Dalı**
**Danışman: Dr. Öğr. Üyesi Zekeriya Anıl GÜVEN**

**İzmir**
**İzmir Bakırçay Üniversitesi**
**Lisansüstü Eğitim Enstitüsü**
**Ocak 2024**

**FINAL APPROVAL FOR THESIS**

This thesis titled " Performance Comparison of Classification Algorithms for Detecting Level-Based Player Churn Using In-Game Data " has been prepared and submitted by Muhammed Selim TOKER in partial fulfilment of the requirements in "İzmir Bakırçay University Directive on Graduate Education and Examination" for the Degree of Master of Science in Computer Engineering Department has been examined and approved on 22./01/2024

| **Committee Members** | **Title, Name and Surname** | **Signature** |
|---|---|---|
| Member (Supervisor) | Asst. Prof. Dr. Zekeriya Anıl GÜVEN | |
| Member | Asst. Prof. Dr. Okan BURSA | |
| Member | Asst. Prof. Dr. Emine SEZER | |
| Member | | |
| Member | | |

Prof. Dr. Özge TÜZÜN ÖZMEN
Director of Graduate Education Institute

# ÖZET

OYUN İÇİ VERİLERİ KULLANARAK SEVİYE BAZLI OYUNCU KAYBINI TESPİT ETMEDE SINIFLANDIRMA ALGORİTMALARININ PERFORMANSLARININ KARŞILŞTIRILMASI

Muhammed Selim TOKER

Bilgisayar Mühendisliği Anabilim Dalı

İzmir Bakırçay Üniversitesi, Lisansüstü Eğitim Enstitüsü, Ocak 2024

Danışman: Dr. Öğr. Üyesi Zekeriya Anıl GÜVEN

Oyuncu kaybı milyarlarca dolarlık oyun endüstrisi için en önemli sorunlardan biridir. Oyuncu kaybı demek gelir kaybı anlamına da gelir. Bu çalışma level-based oyun içi verileri kullanarak oyuncu kaybı yüksek olan bölümleri tahmin etmeye çalışacaktır. Bunun için oyun şirketinden elde edilen verileri veri temizleme, veri indirgeme, yeni çıkarım yapma, dönüştürme gibi gerekli veri ön işleme işlemlerinden geçirip sınıflandırma algoritmaları ile değerlendirilmiştir. Yapılan işlemler adım adım test edilip etkileri tespit edilmiştir. AdaBoost ve Decision Tree algoritmaları ile sırasıyla 0.79 ve 0.78 F1 skoru elde edilmiştir. Elde edilen bulgular ve sonuçlar çalışmanın sonunda paylaşılmıştır.

**Anahtar Sözcükler:** Oyuncu kaybı; Sınıflandırma; Makine öğrenmesi; Seviye bazlı analiz

# ABSTRACT

PERFORMANCE COMPARISON OF CLASSIFICATION ALGORITHMS FOR DETECTING LEVEL-BASED PLAYER CHURN USING IN-GAME DATA

Muhammed Selim TOKER

Department of Computer Engineering

İzmir Bakırçay University, Graduate Education Institute, January 2024

Supervisor: Asst. Prof. Dr. Zekeriya Anıl GÜVEN

Player churn is one of the most important problems for the multi-billion-dollar gaming industry. Player loss also means loss of revenue. This study will use level-based in-game data to predict levels with high player churn. For this purpose, the data obtained from the game company is subjected to the necessary data preprocessing processes, such as data cleaning, data reduction, feature extraction, transformation, and evaluation with classification algorithms. These processes are tested step by step, and their effects are determined. F1 scores of 0.79 and 0.78 were obtained with the AdaBoost and Decision Tree algorithms, respectively. The findings and results are shared at the end of the study.

**Keywords:** Player churn; Classification; Machine learning; Level-based analysis

.../.../20...

## STATEMENT OF COMPLIANCE WITH ETHICAL PRINCIPLES AND RULES

I hereby truthfully declare that this thesis is an original work prepared by me; that I have behaved in accordance with the scientific ethical principles and rules throughout the stages of preparation, data collection, analysis and presentation of my work; that I have cited the sources of all the data and information that could be obtained within the scope of this study, and included these sources in the references section; and that this study has been scanned for plagiarism with …… scientific plagiarism detection program used by İzmir Bakırçay University, and that "it does not have any plagiarism" whatsoever. I also declare that, if a case contrary to my declaration is detected in my work at any time, I hereby express my consent to all the ethical and legal consequences that are involved.

Muhammed Selim TOKER

## TABLE OF CONTENTS

**Page**

**TITLE** ..... **i**

**FINAL APPROVAL FOR THESIS** ..... **ii**

**ÖZET** ..... **iii**

**ABSTRACT** ..... **iv**

**STATEMENT OF COMPLIANCE WITH ETHICAL PRINCIPLES AND RULES** ..... **v**

**TABLE OF CONTENTS** ..... **vi**

**LIST OF TABLES** ..... **ix**

**LIST OF FIGURES** ..... **xi**

**LIST OF ABBREVIATIONS** ..... **xii**

**1. INTRODUCTION** ..... **1**

**1.1. Related Works** ..... **1**

**1.2. Purpose of Thesis** ..... **3**

**1.3. Thesis Organization** ..... **4**

**2. MACHINE LEARNING** ..... **5**

**2.1. Learning Approaches** ..... **5**

**2.1.1. Supervised learning** ..... **6**

**2.1.2. Unsupervised learning** ..... **6**

**2.2. Machine Learning Steps** ..... **7**

**2.2.1. Data acquisition** ..... **7**

**2.2.2. Data preprocess** ..... **7**

**2.2.2.1. *Cleaning*** ..... **7**

**2.2.2.2. *Transformation*** ..... **8**

**2.2.2.3. *Reduction*** ..... **8**

**2.2.3. Test and train set** ..... **8**

**2.2.4. Model selection** ..... **9**

**2.2.5. Model evaluation** ..... **10**

2.2.5.1. *Confusion matrix* ..... 10
2.2.5.2. *Precision* ..... 10
2.2.5.3. *Recall* ..... 11
2.2.5.4. *Accuracy* ..... 11
2.2.5.5. *F-1 score* ..... 11
3. CLASSIFICATION ALGORITHMS ..... 12
3.1. Decision Trees ..... 12
3.2. Random Forests ..... 13
3.3. Extra Tree Classifier ..... 13
3.4. Naïve Bayes ..... 14
3.5. Logistic Regression ..... 14
3.6. Support Vector Machines ..... 15
3.7. K-Nearest Neighbors Classifier ..... 16
3.8. Dummy Classifier ..... 17
3.9. Ada Boost ..... 17
3.10. Gradient Boosting ..... 18
3.11. XGBoost ..... 19
3.12. Light Gradient Boosting Machine ..... 19
4. METHODOLOGY ..... 20
4.1. Data Collection ..... 21
4.2. Data Preparation ..... 23
4.2.1. Cleaning ..... 23
4.2.2. Feature extraction and deletion ..... 23
4.2.3. Normalization ..... 25
4.2.4. Test and train split ..... 25
4.2.5. Feature selection sets by algorithm ..... 26
4.3. Model Evaluation ..... 26
5. RESULTS AND DISCUSSION ..... 28
5.1. Result of Models ..... 28
5.1.1. Performance of decision tree ..... 28
5.1.2. Performance of random forest ..... 30
5.1.3. Performance of extra tree classifier ..... 32

**5.1.4. Performance of naïve bayes** ..... **34**
**5.1.5. Performance of logistic regression** ..... **35**
**5.1.6. Performance of support vector machines** ..... **37**
**5.1.7. Performance of k-nearest neighbors** ..... **38**
**5.1.8. Performance of adaboost** ..... **39**
**5.1.9. Performance of gradient boosting classifier** ..... **41**
**5.1.10. Performance of xgboost** ..... **43**
**5.1.11. Performance of light gradient boosting machine** ..... **45**
**5.2. Comparison of Models** ..... **47**
**5.2.1. Performance impact of preprocessing** ..... **47**
**5.2.2. Performance impact of feature selection and tuning** ..... **48**
**5.2.3. Performance impact of different feature set** ..... **49**
**6. CONCLUSION AND FUTURE WORKS** ..... **51**
**6.1. Future Works** ..... **52**
**7. REFERENCES** ..... **54**

## LIST OF TABLES

**Page**

**Table 4.1.** Game design data before preprocessing ..... 22
**Table 4.2.** Analytic data before preprocessing ..... 22
**Table 4.3.** Analytic data features after extraction ..... 25
**Table 4.4.** Design data features after extraction ..... 25
**Table 4.5.** Final version of the features in the data set ..... 25
**Table 4.6.** Description of data sets ..... 27
**Table 4.7.** Description of feature selection data sets ..... 27
**Table 5.1.** Performance comparison of DT with preprocessed and non-preprocessed data ..... 28
**Table 5.2.** Performance comparison of DT after feature selection ..... 29
**Table 5.3.** Performance comparison of DT after model tuning ..... 29
**Table 5.4..** Performance comparison of the improvement steps with DT ..... 29
**Table 5.5.** Performance comparison of RF with preprocessed and non-preprocessed data ..... 30
**Table 5.6.** Performance comparison of RF after feature selection ..... 31
**Table 5.7.** Performance comparison of RF after model tuning ..... 31
**Table 5.8.** Performance comparison of the improvement steps with RF ..... 31
**Table 5.9.** Performance comparison of ET with preprocessed and non-preprocessed data ..... 32
**Table 5.10.** Performance comparison of ET after feature selection ..... 33
**Table 5.11.** Performance comparison of ET after model tuning ..... 33
**Table 5.12.** Performance comparison of the improvement steps with ET ..... 33
**Table 5.13.** Performance comparison of NB with preprocessed and non-preprocessed data ..... 34
**Table 5.14.** Performance comparison of NB after feature selection ..... 34
**Table 5.15.** Performance comparison of the improvement steps with NB ..... 34
**Table 5.16.** Performance comparison of LR with preprocessed and non-preprocessed data ..... 35
**Table 5.17.** Performance comparison of LR after feature selection ..... 36
**Table 5.18.** Performance comparison of LR after model tuning ..... 36
**Table 5.19.** Performance comparison of the improvement steps with LR ..... 36

**Table 5.20.** Performance comparison of SVM with preprocessed and non-preprocessed data....37
**Table 5.21.** Performance comparison of SVM after feature selection....38
**Table 5.22.** Performance comparison of SVM after model tuning....38
**Table 5.23.** Performance comparison of KNN with preprocessed and non-preprocessed data....38
**Table 5.24.** Performance comparison of KNN after feature selection....39
**Table 5.25.** Performance comparison of the improvement steps with KNN....39
**Table 5.26.** Performance comparison of ADA with preprocessed and non-preprocessed data....40
**Table 5.27.** Performance comparison of ADA after feature selection....40
**Table 5.28.** Performance comparison of ADA after model tuning....41
**Table 5.29.** Performance comparison of the improvement steps with ADA....41
**Table 5.30.** Performance comparison of GBC with preprocessed and non-preprocessed data....42
**Table 5.31.** Performance comparison of GBC after feature selection....42
**Table 5.32.** Performance comparison of GBC after model tuning....43
**Table 5.33.** Performance comparison of the improvement steps with GBC....43
**Table 5.34.** Performance comparison of XGB with preprocessed and non-preprocessed data....44
**Table 5.35.** Performance comparison of XGB after feature selection....44
**Table 5.36.** Performance comparison of XGB after model tuning....45
**Table 5.37.** Performance comparison of the improvement steps with XGB....45
**Table 5.38.** Performance comparison of LGBM with preprocessed and non-preprocessed data....46
**Table 5.39.** Performance comparison of LGBM after feature selection....46
**Table 5.40.** Performance comparison of LGBM after model tuning....47
**Table 5.41.** Performance comparison of the improvement steps with LGBM....47
**Table 5.42.** Impact of preprocessing to model performance....48
**Table 5.43.** Impact of tuned with feature selection to model performance....48
**Table 5.44.** Best performed feature sets on algorithm....49

## LIST OF FIGURES

**Page**

**Figure 2.1.** Machine learning approaches and usage areas ..... 5
**Figure 2.2.** The difference between supervised and unsupervised learning ..... 6
**Figure 2.3.** Effect of standardization and normalization on data ..... 8
**Figure 2.4.** Machine learning algorithms ..... 9
**Figure 2.5.** Confusion matrix ..... 10
**Figure 3.1.** Decision tree demonstration ..... 12
**Figure 3.2.** Random forest demonstration ..... 13
**Figure 3.3.** Comparison of linear and logistic regression ..... 15
**Figure 3.4.** Support vector machine demonstration ..... 16
**Figure 3.5.** K-Nearest Neighbors demonstration ..... 17
**Figure 3.6.** Example of AdaBoost ..... 18
**Figure 3.7.** Differences between level-wise and leaf-wise ..... 19
**Figure 4.1.** Explanation of general methodology of this work ..... 20
**Figure 4.2.** Screenshots from game ..... 21
**Figure 5.1.** Most important features according to DT ..... 28
**Figure 5.2.** Most important features according to RF ..... 30
**Figure 5.3.** Most important features according to ET ..... 32
**Figure 5.4.** Most important features according to LR ..... 35
**Figure 5.5.** Most important features according to SVM ..... 37
**Figure 5.6.** Most important features according to ADA ..... 40
**Figure 5.7.** Most important features according to GBC ..... 42
**Figure 5.8.** Most important features according to XGB ..... 44
**Figure 5.9.** Most important features according to LGBM ..... 46
**Figure 5.10.** Best two algorithms and result ..... 50

## LIST OF ABBREVIATIONS

ADA : AdaBoost

AI : Artificial Intelligence

DT : Decision Tree

ET : Extra Tree Classifier

FN : False Negative

FP : False Positive

GBM : Gradient Boosting Machine

KNN : K-Nearest Neighbors

LGBM : Light Gradient Boosting Machine

LR : Logistic Regression

ML : Machine Learning

NB : Naïve Bayes Classifier

RF : Random Forests

SVM : Support Vector Machine

TN : True Negative

TP : True Positive

USA : United States of America

XGB : XGBoost

# 1. INTRODUCTION

People have been playing games for thousands of years. They have been used to entertain, teach, and challenge. Games can be played alone or in groups, and they are often a source of social interaction.

Games have an extensive history, dating back to early radar displays and oscilloscopes of the 1960s and early home video game consoles of the 1970s. From the 1980s to the high-powered processors of today's, video games have come a long way. (Metuarau, 2017) Being that mobile phones are so widely used, and technology is advancing so quickly, games are a part of our everyday lives, and the idea of mobile games has been explored in literature.

Based on the estimations for the year 2022, the video game industry has a market size of around 250 billion dollars. Conversely, mobile games account for 42% of the whole industry and have an estimated market value of almost $100 billion. (Video Game Market, 2022)

Players are seen as the consumers of this gigantic industry. Like any other industry, gaming firms prioritize the acquisition and retention of clients (players). Acquiring players incurs an economic expense for corporations, which they cover by charging non-directionally their consumers. This is often done through the sale of game copies, downloadable content, and in-game purchases. However, the challenge lies in retaining these players and keeping them engaged to generate ongoing revenue. To achieve this, gaming firms invest heavily in creating captivating and immersive experiences, providing regular updates and new content, and fostering a strong sense of community among players. Ultimately, the success of gaming firms depends on their ability to attract and retain a loyal player base, ensuring continued growth and profitability in this ever-expanding industry. Player churn may lead to decreased income and have a detrimental impact on the game's long-term viability. It is crucial for gaming firms to constantly analyze and address the reasons behind player churn. By understanding why players are leaving, firms can make targeted improvements to their games and services. Additionally, offering incentives or rewards to loyal players can help foster a sense of loyalty and reduce churn. Ultimately, by constantly evolving and adapting to the needs and desires of their player base, gaming firms can ensure long-term success in an industry that is constantly evolving.

Hence, player churn is a significant concern for gaming producers. It is strategically important for game firms to identify the factors that lead to player churn and implement measures to minimize it. By doing so, they can ensure the sustainability and success of their game in the competitive market.

### 1.1. Related Works

When the studies are analyzed, there is a study on predicting player churn. These were done using data such as player behavior and demographics. Rio and friends approached the problem of player churn and conversion with the Time Series method. It aims to categorize players into different groups, aims to acquire them again, and conducts user-based studies (Rio et al.,2021).

In another study by Sifa aims to player behavior was modeled using the Artificial Neural Networks technique. They tried to predict player churn with the Echo State Network approach (Sifa, 2021).

Yang and friends prevent a model based on playtime is planned to be developed. Using the concepts of entropy and cross-entropy, game regularity is detected, and feature extraction is performed (Yang et al., 2022).

In other study Kim and friends develop a standard churn analysis process for casual games, focusing on new players and defining churn using observation and prediction periods. Machine learning and deep learning models are used in this work, which suggests that a few well-chosen features are sufficient for decision-making (Kim et al., 2017).

Rothmeier and friends worked on churn detection with eight machine learning algorithms and four labeling strategies. The ideal approach was random forests with sliding windows, which produced 97% prediction accuracy (Rothmeier, 2020).

Another study presents a survival ensemble model for mobile social games, focusing on churn prediction. The model predicts player churn probability as a function of time, allowing for differentiation of loyalty profiles and risk factors. The results of the work show that this model significantly improves the accuracy and robustness of traditional analyses (Periáñez et al., 2016).

Runge and friends investigated the potential of predicting churn in casual social games to increase lifetime and revenue contributions. It compares four classification

algorithms and implements a hidden Markov model to address temporal dynamics. The study found that a neural network achieves the best prediction performance. An A/B test on one game showed that contacting players before the predicted churn event improved communication and didn't significantly impact player churn rate or monetization. Cross-linking may be a more effective measure to deal with churning players as a result of this study (Runge et al., 2014).

When the studies are examined, many methods have been applied to predict player churn. In all of these, player-based churn has been analyzed. In this study, a level-based solution is proposed.

## 1.2. Purpose of Thesis

This thesis aims to use Machine Learning and Classification Algorithms to detect level-based player churn using in-game data, to determine the factors that most affect player churn, to determine the best performing algorithms, to analyze the results and to report the findings. This study is based on the following basic hypotheses.

- Machine Learning and Classification Algorithms based classification algorithms can be effective in detecting level-based player churn.
- Different Classification Algorithms may differ in their performance in player loss detection.
- The best performing algorithm may be more effective in making recommendations to reduce player churn.

The main research questions of this thesis are as follows:

- With the help of in-game data, can classification algorithms predict level-based player attrition?
- Considering the in-game data, do different classification algorithms differ in terms of their performance in detecting level-based player loss?

The main scope and limitations of this thesis are as follows:

- The data to be used in the study includes in-game data. In-game data includes the design details of each level and level-based interactions with the game (how many seconds it takes to finish, level difficulty, in-game economy, etc.). It does not contain user demographics, device information, personal information or any person-based data.

- In the study, it is planned to detect level-based player loss. It is not interested in which player is lost and when. It aims to predict at which level player churn will occur.

It plans to use Machine Learning based classification algorithms to detect level-based player churn. It aims to compare the algorithms used and determine the most performant one for level-based player churn prediction.

### 1.3. Thesis Organization

This study focuses on detecting player churn using in-game data. To this end, Section 1 explains the importance and rationale for the study and summarizes other work in this area. Section 2 provides a brief introduction to machine learning and describes the methods used. In Section 3, the general working logic of the algorithms used to solve the classification problem is discussed. In Section 4, the details of the implementation phase are mentioned. Classification processes were carried out for each model and set. In the last section, the performances are compared and evaluated. Conclusions and findings are presented.

## 2. MACHINE LEARNING

Machine Learning (ML) is a discipline within computer science and Artificial Intelligence (AI)that concentrates on creating algorithms and models that empower computers to enhance their performance autonomously by using experience. This entails the use of statistical methodologies to assist computers in acquiring knowledge from data and making predictions or choices, without the need for explicit programming to carry out the job. The topic of machine learning has garnered considerable interest owing to its potential applications in many fields such as healthcare, finance, and autonomous systems. Consequently, it has emerged as a field that is now being extensively studied and developed, with many trends, viewpoints, and possibilities being investigated (Jordan and Mitchell, 2015).

### 2.1. Learning Approaches

Machine learning involves the use of various techniques such as supervised learning, unsupervised learning, and reinforcement learning to learn from data and make predictions or decisions based on that learning.

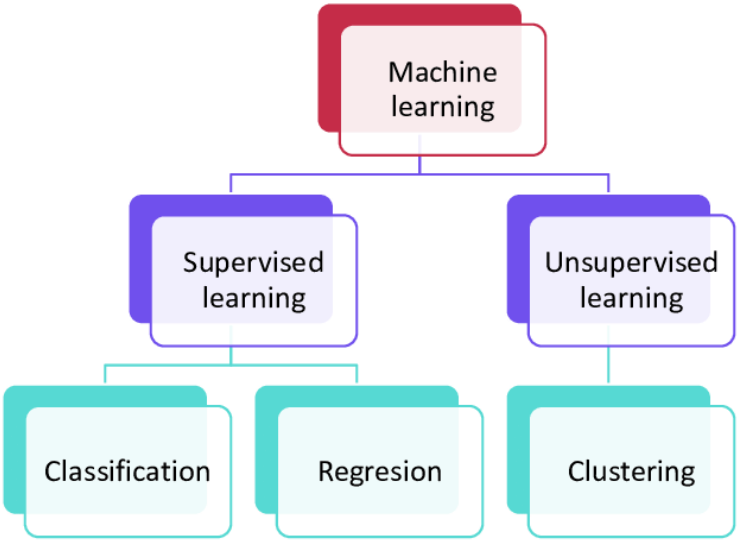


**Figure 2.1.** Machine learning approaches and usage areas (Vitola et al., 2017)

### 2.1.1. Supervised learning

Supervised learning is a key principle in machine learning, whereby an algorithm acquires knowledge from labeled training data and then generates predictions on novel, unobserved data. The training data is labeled and comprises pairings of input and output. The algorithm acquires the ability to associate the input with the outcome by using this training data (Bzdok et al., 2018).

Supervised learning is an algorithm aiming to derive generalizations from the provided training data to generate precise predictions on unfamiliar data (Seeger, 2017). Supervised learning has been extensively researched and used in many fields such as computer science, artificial intelligence, and healthcare. It has been used for purposes such as categorization, estimation, and forecasting (Shi and Zhang, 2011).

### 2.1.2. Unsupervised learning

Unsupervised learning algorithms are a category of machine learning methods that seek to derive patterns or information from incoming data without explicit direction or labeled results. These algorithms are especially valuable for analyzing extensive and unstructured datasets, since they possess the ability to automatically detect hidden structures or connections within the data (Liu et al., 2022).

An important benefit of unsupervised learning is its capacity to extract complex associations from observed data, without requiring knowledge of target classes (Abdel-Basset et al., 2021). Due to its practicality and performances, unsupervised learning possess many applications in several domains, including computer science, vision research, genetics, and image analysis.

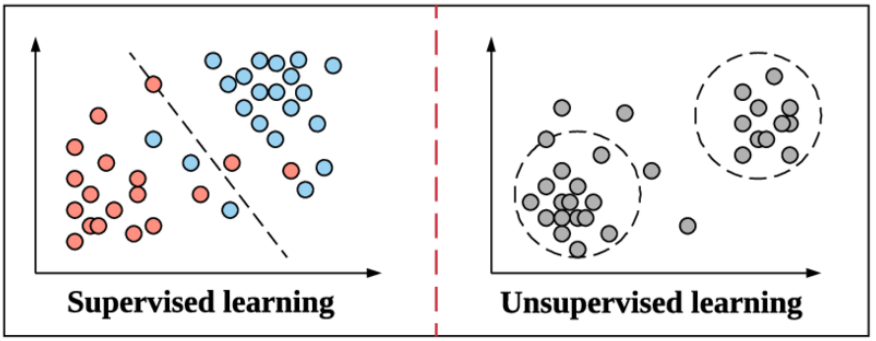


**Figure 2.2.** The difference between supervised and unsupervised learning (Su et al., 2019)

## 2.2. Machine Learning Steps

To create a successful model, major steps of machine learning involve data acquisition, preparation, processing, modeling, and prediction. These steps are facilitated by the availability of large datasets, advances in computational power, and the application of various machine learning techniques such as supervised and unsupervised learning models.

### 2.2.1. Data acquisition

Data acquisition for machine learning is an essential stage in the training and development of precise and dependable models. The method entails collecting pertinent and top-notch data from diverse sources to guarantee the machine learning algorithms can proficiently acquire knowledge and provide accurate forecasts. (Oh et al., 2020).

### 2.2.2. Data preprocess

Data preprocessing in machine learning refers to the first stage of converting raw data into a format that is appropriate for machine learning algorithms. The input data quality has a vital role in the accuracy and efficacy of machine learning models, making it a critical stage in the machine learning pipeline. (Abram and McCloskey, 2022; Winck et al., 2013).

The procedure entails many phases, including cleaning, normalization, and feature selection, to guarantee that the data is in a consistent and useful format. These steps also aim to enhance the quality of the data, hence improving the performance of the model (Shah et al., 2022).

#### *2.2.2.1. Cleaning*

Incomplete, inaccurate, outlier or duplicate data may adversely affect the decision-making process of the trained model. To improve the quality of the data set, faulty data should be deleted, or necessary adjustments should be made. Generally, duplicate data, irrelevant data and outliers are removed from the data set. Missing features should either be deleted or filled with the mean or median of the values in other records.

#### *2.2.2.2. Transformation*

Machine learning algorithms make predictions using numerical variables and perform better when these variables are scaled to a standard range.

Since fractional and multi-digit numerical values may coexist in the same data set, machine learning models to be built with these variables may not always perform well. At this point, the necessary transformations can be done by normalization and standardization, which are the most popular scaling methods.

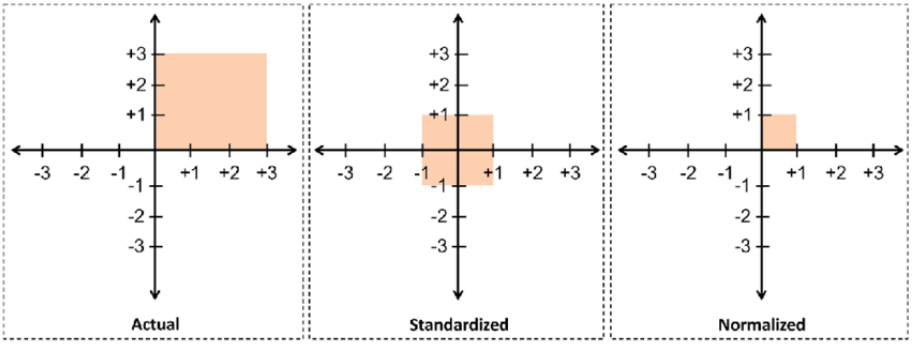


**Figure 2.3.** Effect of standardization and normalization on data (Kocak, 2022)

#### *2.2.2.3. Reduction*

Dimensionality reduction is a very important method for data science. Real-life data has too many dimensions (attributes), and the larger the dimension, the more time, and resources we need to spend on all processes, from data cleaning to model building. In addition, due to the high correlation between attributes, it may encounter overfitting problems. The easiest way to reduce the size is to find the attributes that best describe our data, that is, the attributes that hold the maximum variance information of the distribution in the data and discard the others.

### 2.2.3. Test and train set

The procedure of dividing a dataset into separate training and testing sets is an essential step in the creation of machine learning models. At this stage, the data set is divided into two separate parts so that most of the data is in the training set to train the model and evaluate its performance.

Rácz et al. (2021) highlighted the variations seen across machine learning methods

and the influence of dataset sizes and split ratios on model performance. Usually, 70% or 80% of the data is reserved for the training set.

### 2.2.4. Model selection

Model selection is the act of selecting the best appropriate algorithm or approach for a certain issue. It is a critical factor in deciding the capacity of supervised learning to generalize (Sugiyama and Ogawa, 2001).

In machine learning, there are many models that can be applied to different purposes and datasets. The approaches, prediction capabilities, costs and prediction types of these models may differ. The models used in machine learning and classified according to learning approaches are shown in Figure 2.4

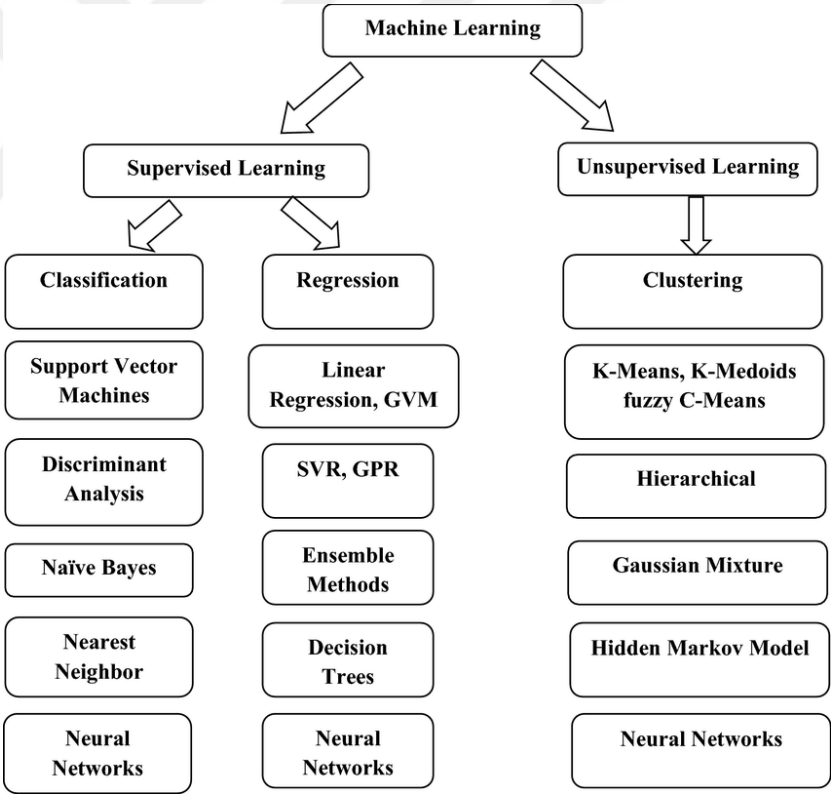


**Figure 2.4.** Machine learning algorithms (Abirami and Kumar, 2021)

This study aims to find a solution to a classification problem, and these algorithms are described in the next section.

### 2.2.5. Model evaluation

The evaluation of classification algorithms is crucial for assessing their predictive ability and choosing the most suitable technique for a specific problem (Santafé et al., 2015). Common metrics for assessing classification predictions include accuracy, kappa, sensitivity, specificity, and positive and negative predicted values (Kühn and Johnson, 2013). These metrics help in understanding the model's performance and making informed decisions.

#### *2.2.5.1. Confusion matrix*

A confusion matrix is a table that outlines the actual and predicted classifications performed by a classification system, which is commonly used for performance evaluation (Ting, 2017).

It works especially well for binary classification problems because it shows how much a model mixes up different classes (Amin, 2022).

| | | **Actual Class** | |
|---|---|---|---|
| | | Positive (**P**) | Negative (**N**) |
| **Predicted Class** | Positive (**P**) | True Positive (**TP**) | False Positive (**FP**) |
| | Negative (**N**) | False Negative (**FN**) | True Negative (**TN**) |

**Figure 2.5.** Confusion matrix

False Positive (FP) and False Negative (FN) values symbolize inaccuracies in the model's prognostications. On the other hand, True Positive (TP) and True Negative (TN) values signify occurrences in which the model made accurate predictions regarding the outcomes.

It is possible to derive four secondary metrics from the Confusion Matrix: F1-score, accuracy, precision, and recall.

#### *2.2.5.2. Precision*

Precision is a metric that quantifies the correctness of positive predictions provided

by a model (Ruuska et al., 2018). The precision calculation formula given in the 2.1.

$$precision = \frac{TP}{TP + FP} \tag{2.1}$$

#### *2.2.5.3. Recall*

Accuracy refers to the ratio of correctly classified instances in the test dataset by an algorithm (Ruuska et al., 2018). The accuracy calculation formula is shown in 2.2.

$$recall = \frac{TP}{TP + FN} \tag{2.2}$$

#### *2.2.5.4. Accuracy*

Accuracy refers to the ratio of correctly classified instances in the test dataset by an algorithm (Ruuska et al., 2018). The accuracy calculation formula is shown in 2.3.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.3}$$

#### *2.2.5.5. F-1 score*

The F1-Score measures the balanced accuracy between precision and recall in a confusion matrix based on the harmonic mean of these metrics (Goutte & Gaussier, 2005). The formula is given in the 2.3.

$$F1score = 2\ x\ \frac{precision\ x\ recall}{precision + recall} \tag{2.3}$$

## 3. CLASSIFICATION ALGORITHMS

This study is based on a classification problem and the algorithms used are described in this section.

### 3.1. Decision Trees

Decision Trees (DT) are hierarchical constructs that depict the underlying rules of data by recursively dividing it into groups. These may be used for data exploration using three methods: description, categorization, and generalization. Decision trees have been used in several domains such as statistics, engineering, and decision theory. A multitude of current data mining tools rely on the process of generating decision trees from data.

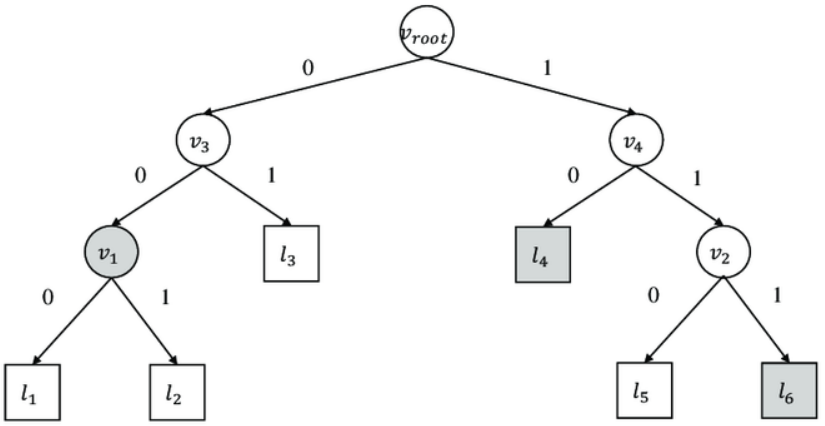


**Figure 3.1.** Decision tree demonstration (Fu et al., 2023)

Decision trees are built using a training set of objects, where each item is characterized by characteristics and class labels. The fundamental idea behind data collection is the accurate correlation between qualities and classes. A training set that is devoid of any noise develops all objects by using the fundamental principle. Decision trees consist of internal nodes and leaf nodes. Internal nodes might have two or more child nodes, whereas leaf nodes have no child nodes. Every internal node in the structure has splits that evaluate the value of an attribute expression. Every terminal node is assigned a class designation (Murthy, 1998).

### 3.2. Random Forests

Random forests are an ensemble of decision trees, where each tree's structure is determined by a random vector that is separately sampled and follows the same distribution for all trees in the forest.

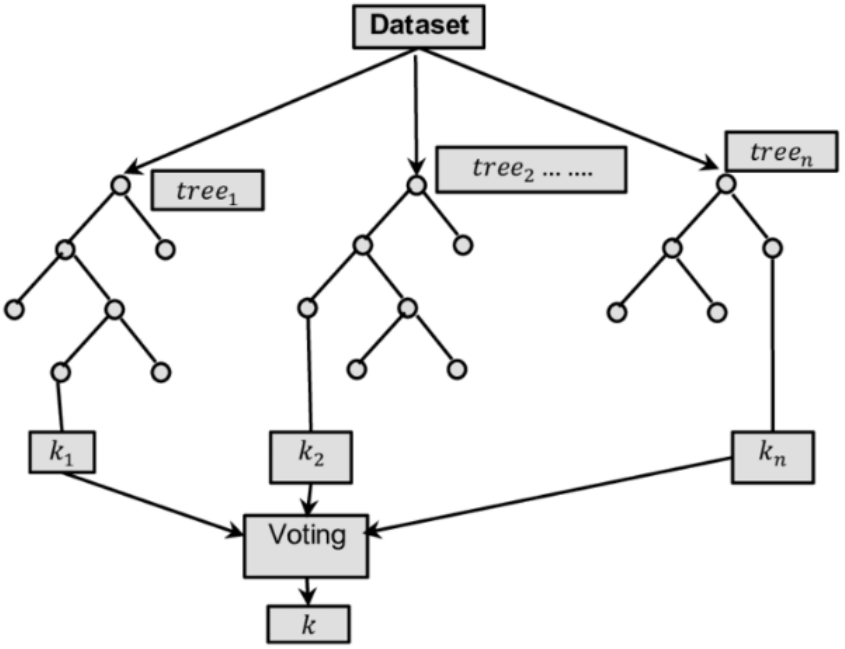


**Figure 3.2.** Random forest demonstration (Ahmed et al., 2020)

As seen in Figure 3.2, the random forest model picks n distinct subsets from both the dataset and the feature set and trains them. Thus, many decision trees are generated, with each tree making predictions independently. We chose the tree with the most accurate forecast for our model. The generalization error for forests asymptotically converges to a limit as the number of trees in the forest approaches infinity. The generalization error of a forest of tree classifiers is contingent upon the potency of the individual trees within the forest and the level of correlation between them (Breiman, 2001).

### 3.3. Extra Tree Classifier

The algorithm known as Extremely Randomized Trees, or Extra-Trees (ET), constructs a collection of unpruned decision trees in the traditional top-down method. This tree-based ensemble technique is different in two main ways: it uses

random cut-points to separate nodes, and it builds the trees from the whole learning sample instead of a bootstrap replica.

The predictions of the trees are combined to get the ultimate forecast, using majority voting for classification problems and arithmetic averaging for regression issues. The Extra-Trees approach provides a high level of randomization, efficient computing, and resilience in both classification and regression problems. It is noteworthy that although bootstrapping is not initially included in ET's structure, it may be included in some implementations (Geurts et al., 2006).

### 3.4. Naïve Bayes

The Naive Bayes (NB) classifier is a straightforward and widely used probabilistic approach for supervised learning that relies on Bayes' theorem. The Naive Bayes Classifier computes the likelihood of each state for an element and categorizes it based on the state with the greatest probability. It can get very effective results with little training data. It has been empirically shown to be successfully used in tasks such as text categorization, medical diagnosis, and system performance management (Al-Aidaroos et al., 2010; Lewis, 1998).

$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)} \tag{3.1}$$

In the Bayes formula in Equal 3.1, A and B are separately labeled events. The notations P (A) and P (B) are the probabilities that events A and B occur independently of each other. According to this formula, P(A|B) denotes the probability of event A occurring when event B occurs, and P(B|A) denotes the probability of event B occurring when event A occurs.

### 3.5. Logistic Regression

Logistic regression is a classification algorithm that is mainly used in binary classification problems, despite the word "regression" in its name. The biggest difference between logistic regression and linear regression is how they apply the line to separate the two classes. While linear regression uses Least Squares to draw the optimum line, logistic regression uses Maximum Likelihood.

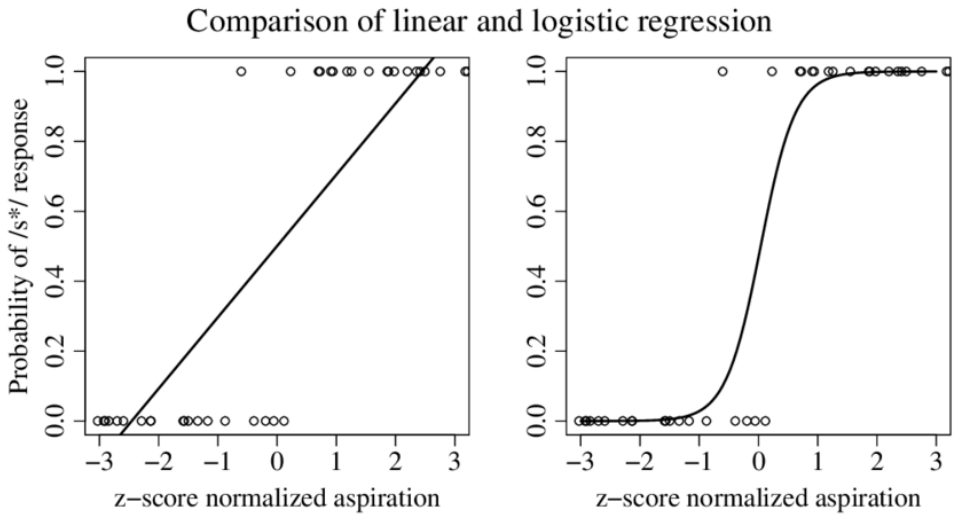


**Figure 3.3.** Comparison of linear and logistic regression (Kallay and Holliday, 2012)

Logistic regression uses the Sigmoid (Logistic) function, which is also frequently used under activation functions in Deep Learning, for classification. A sigmoid function is an "S" shaped curve that compresses the data between 0 and 1 and thus enables classification (Hosmer & Lemeshow, 2000).

$$S(x) = \frac{1}{1 + e^{-x}} \tag{3.2}$$

### 3.6. Support Vector Machine

The support-vector machine (SVM) is a machine learning algorithm built for solving two-group classification issues. The process involves the mapping of input vectors to a feature space with many dimensions, followed by the construction of a linear decision surface. The decision surface is created to optimize the learning machine's generalization capabilities. The support-vector network was first developed to handle training data that can be separated into distinct classes, but it has also been adapted to accommodate training data that cannot be easily separated. The support-vector network has shown its capacity for generalization by using polynomial input transformations (Cortes and Vapnik, 1995).

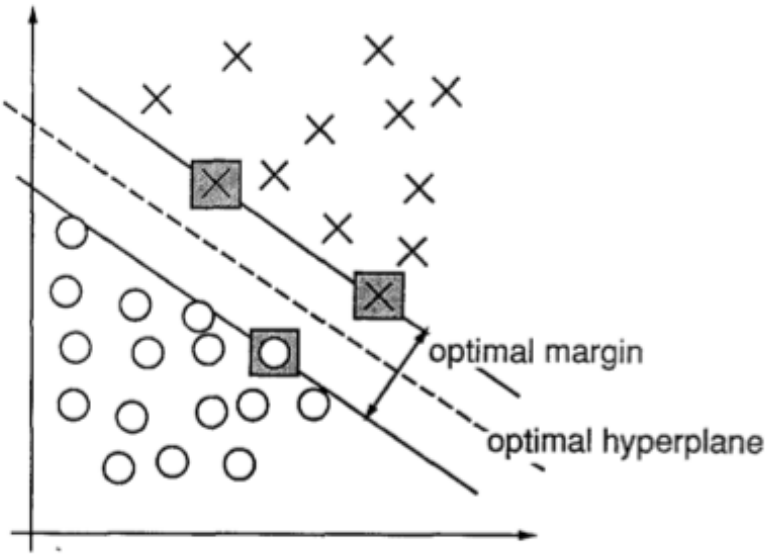


**Figure 3.4.** Support vector machine demonstration

The 3.4 figure gives an instance of a problem that may be divided into distinct parts in a two-dimensional space. The support vectors, shown by gray squares, establish the boundary with the greatest distance between the two classes.

### 3.7. K-Nearest Neighbors Classifier

The k-nearest neighbor algorithm (KNN) is a technique used to categorize objects by identifying the nearest training data points in the feature space. KNN is a kind of instance-based learning. The k-nearest neighbor technique is one of the most basic machine learning algorithms.

In the KNN algorithm, a new instance is categorized based on the class value of its closest neighbors. The class of a test sample is determined by selecting the k nearest samples from the training set. Every sample computes its proximity to the other nearest samples, and the samples with the closest proximity are combined. This procedure continues until the necessary quantity of clusters is achieved.

The distance between the test sample and the training samples may be computed using several distance metrics, such as Euclidean and Minkowski (Cunningham & Delany, 2021).

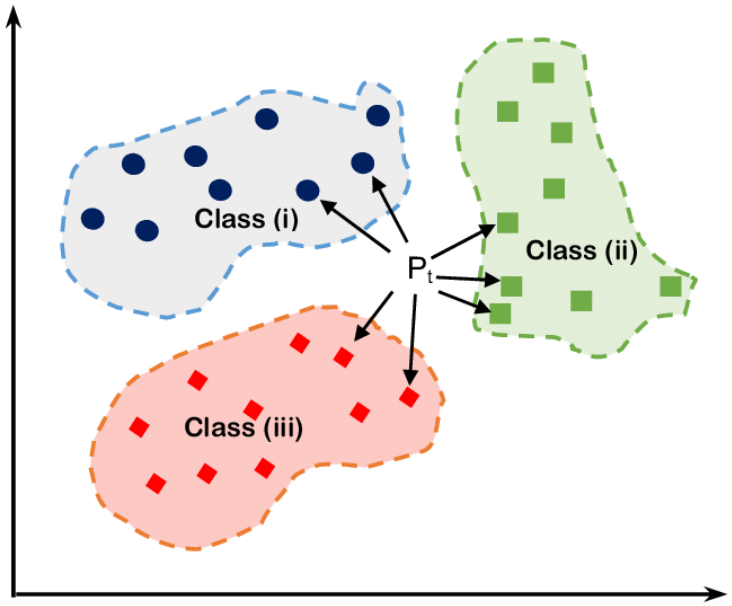


**Figure 3.5.** K-Nearest Neighbors demonstration (Atallah et al., 2019)

### 3.8. Dummy Classifier

The DummyClassifier is a classification model that is used with the Sklearn package. A classifier disregards the input attributes and acts as a basic reference point for comparison with other more intricate classifiers. The behavior of the baseline is determined by the strategy parameter. All techniques provide predictions that disregard the input feature values provided as the X parameter for the fit and predict functions. The predictions, however, often rely on the values seen in the y parameter that is provided to the fit function (http-1)

Sklearn does not share the theory and details on which the classifier is based. Information about the working logic of the model is limited.

### 3.9. Ada Boost

AdaBoost is a machine learning technique that uses boosting to create a prediction rule with high accuracy. It does this by mixing several weak and wrong rules. Decision trees having a maximum depth learned from the training set are excluded from usage. AdaBoost, which is considered the first boosting algorithm to be put into practice, uses decision trees with a depth of one consisting of one node and two leaves as a weak learner. Ranking has significance due to the cumulative nature of the structure. The mistake made

by the first decision tree impacts the weights of the subsequent tree. Unlike the bagging technique, it does not do calculations in parallel; instead, it performs calculations sequentially (Schapire, 2013).

Figure 3.6 exemplifies the application of the AdaBoost algorithm to classification problems.

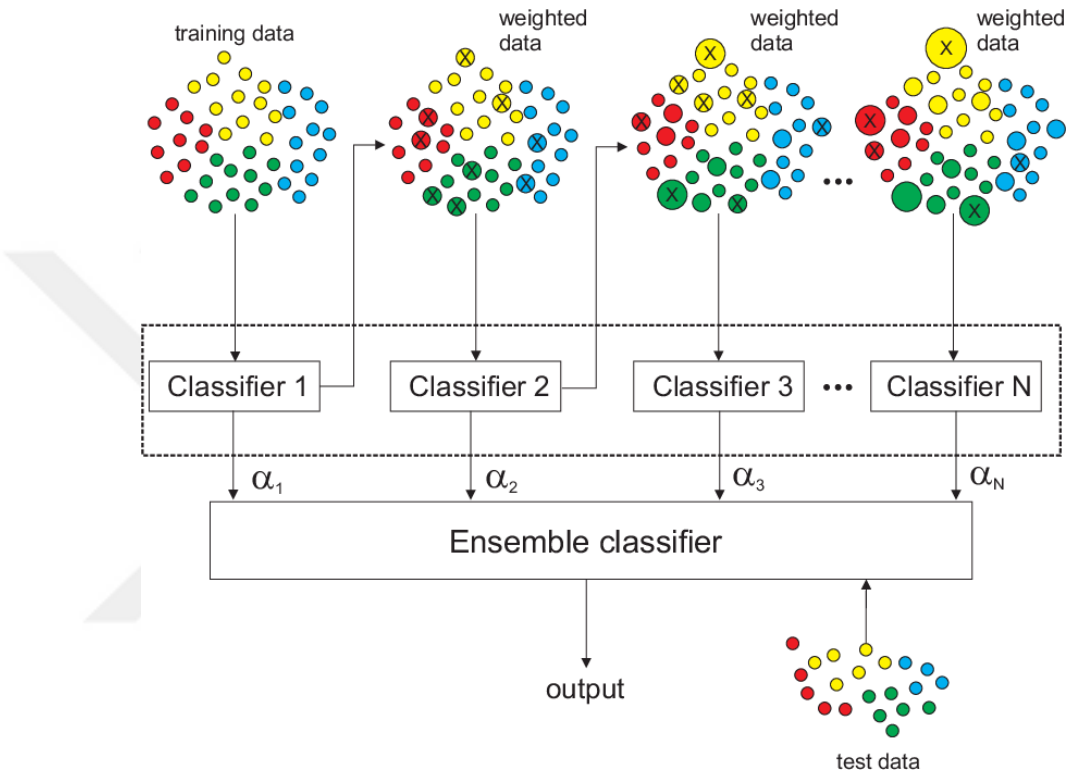


**Figure 3.6.** Example of AdaBoost (Salcedo-Sanz et al., 2023)

### 3.10. Gradient Boosting

The gradient boosting method is a classification and regression model that uses the gradian descent algorithm to improve the results of decision trees. It generates many consecutive and dependent trees by looking at the decision trees and their errors. By finding the sum of the decisions made by all these trees, it makes the final decision (Friedman, 2001).

The Gradient Boosting technique has facilitated the development of many machine learning algorithms that exhibit enhanced efficiency and dependability. XGBoost and LightGBM, which are also used in this investigation, are included in these models.

### 3.11. XGBoost

XGBoost is a Machine Learning (ML) method that was created at the University of Washington. It is renowned for its gradient boosting architecture. The primary advantages of this system lie in its ability to optimize performance via techniques such as parallelization and tree pruning, as well as its capacity to enhance hardware efficiency by using cache awareness approaches. XGBoost utilizes sparsity awareness to automatically learn the ideal missing value, leverages the Weighted Quantile Sketch (WQS) technique to identify the most effective splitting points, and incorporates a built-in cross-validation mechanism for each iteration, removing the need for explicit programming. The algorithm employs L1 and L2 regularization techniques to mitigate overfitting. The enhancements have played a significant role in the achievement of XGBoost in the domain (Chen & Guestrin, 2016).

### 3.12. Light Gradient Boosting Machine

Light Gradient Boosting Machine also known as LightGBM is a robust gradient boosting Machine Learning framework that is used especially in classification. It is a distributed, high-performance gradient boosting framework based on decision trees, also named Light because it is faster than other models. Despite other boosting algorithms that divide the tree level-wise, LightGBM splits the tree leaf-wise. It exhibits superior loss reduction compared to level-wise algorithms, leading to enhanced accuracy. Light GBM employs a vertical growth strategy for decision trees, in contrast to other methods that use a horizontal growth approach (Ke et al, 2021). The algorithm attempted to be a competitor to XGBoost in terms of both accuracy and time.


**Figure 3.7.** Differences between level-wise and leaf-wise (Bui et al., 2021)

# 4. METHODOLOGY

This section describes the methodology of the study. The steps from the data collection process to the comparison of the performance of the algorithms used to detect player loss are explained in detail in the following chapters. The general methodology of this work is presented in Figure 4.1, which outlines the step-by-step process followed to achieve the research objectives. The figure provides a visual representation of the different stages involved in the study, highlighting the key components and their interconnections. It serves as a useful guide for understanding the overall approach adopted in this research.

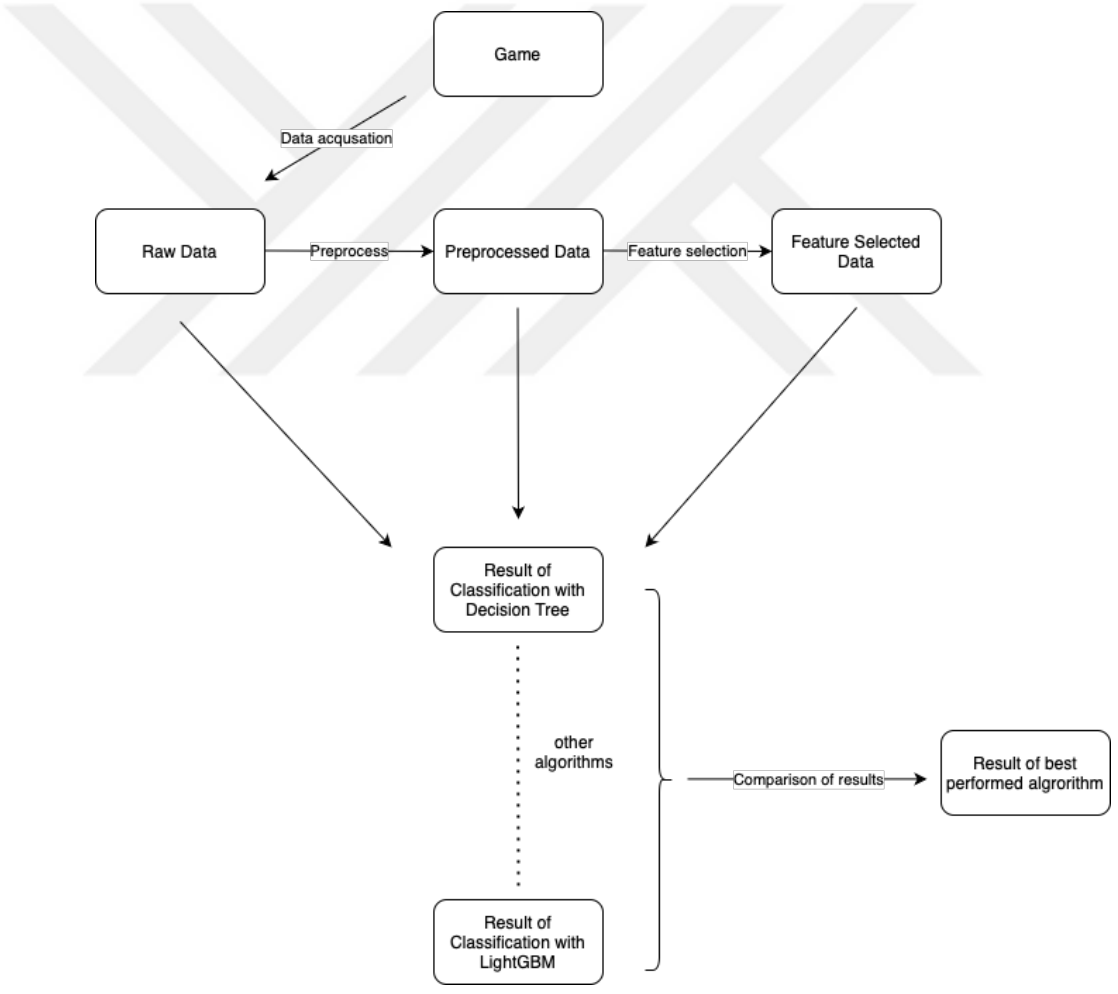


**Figure 4.1.** Explanation of general methodology of this work

## 4.1. Data Collection

This thesis uses data from a mobile puzzle game developed by me for the Apple ecosystem. The data was provided directly by us, and the dataset contains the data of 99 levels in the game with 35 attributes. The data used is level-based and does not include data such as player demographics and behaviors. In different versions of the game, there are differences in the features and level design used for the game. Therefore, a stable version was selected and used in the study in order not to affect the analysis performance of the data set. The tests of prototype trials in the mobile game industry are usually tested with users in the United States of America (USA). This can be explained by the quality of the player base. 95% of the game users are in the USA and for this reason, users in other locations are not included in the dataset. Screenshots of the game whose data was used in this study are shown in Figure 4.2.

**Figure 4.2.** Screenshots from game

The data can be analyzed in two groups. The first one is the data related to the design of the game and the second one is analytical data based on level. The details of the data collected are as stated in the Table 4.1 and Table 4.2.

**Table 4.1.** Game design data before preprocessing.

| Attributes | Description |
|---|---|
| Level ID | Id of level |
| Main Level ID | Id of parent level |
| Stage | Id of child level |
| Width | Width of level design |
| Height | Height of level design |
| Total Object | Total object in level |
| Type Count | Number of different types |
| Locked Box | Locked boxes in level |
| Soft Locked Box | Is there any soft loft box in level? |
| Surprise Box | Is there any surprise box in level? |
| Vent Count | Number of vents in level |
| Vent Limit | Limit of every vent |
| Combo Box_L1 | Width of combo box at line 1 |
| Combo Box_L2 | Width of combo box at line 2 |
| Combo Box_L3 | Width of combo box at line 3 |
| Key Box_L1 | Required key at line 1 |
| Key Box_L2 | Required key at line 2 |
| Key Box_L3 | Required key at line 3 |
| Combo Box Tutorial | Is there combo box tutorial in level? |
| Bonus Level | Is level a bonus level? |
| Given Time | Time required to complete level |
| Feature Types | Number of feature types in level |

**Table 4.2.** Analytic data before preprocessing.

| Attributes | Description |
|---|---|
| Level ID | Id of level |
| User Start | The number of players start to level |
| User Complete | The number of players to complete level |
| Fail Count | The number of fails in level |
| Failed Players | The number of failed players |
| Avg Time | Average duration of players completing the level |
| Feature Count | The number of features in level |
| Shuffle Count | The number of shuffles using in level |
| Shuffled Players | The number of players using shuffle |
| Reverse Count | The number of reverses using in level |
| Reversed Players | The number of players using reverse |
| Continue Count | The number of continue using in level |
| Continued Player | The number of players using continue |

## 4.3. Data Preparation

After determining the features that we could use in the study, we created our data set. The pre-processing necessary for our model to give better results is described in the following section 4.3.1.

### 4.3.1. Cleaning

In data set, there are missing data in our Shuffle Count, Shuffled Players, Reverse Count, Reversed Players, Continue Count, and Continued Player features. This is because these features are available after certain levels. For this reason, instead of filling it with the average, we filled it with 0 because it is never used in these levels.

There are 3 bonus levels in the game. The mechanics of these levels are different from the general mechanics of the game and some of their features are excessive. In order not to decrease data quality, we deleted the records with bonus levels.

### 4.3.2. Feature extraction and deletion

In the shortest terms, the aim of this study is to detect player churn. Player churn means that a player leaves the game or stops being active. The churn value used in prediction is calculated in Formula 4.1 (Pahwa, 2018).

$$churn = \frac{(User\ Start - User\ Complete)}{User\ Start} x100 \qquad (4.1)$$

The churn values in the data set vary between 0% and 6.93%. When the game-specific details are evaluated, and according to our experience in the gaming industry, we considered values of 2.7% and above as high churn.

The data set consists of level-based data and the number of users starting the game at each level is different. It will be more effective to use some numerical data such as this by proportioning it according to the number of users in the level. No matter how numerically different it may seem, 1 and 100 can sometimes have the same effect. The important thing is the ratio rather than the number itself. For this, the changes specified in the table below have been made. After the transformation, the features that are no longer needed have been removed.

**Table 4.3.** Analytic data features after extraction.

| Attributes | Formula |
|---|---|
| Fail Percentage | $= Fail\ Count\ /\ User\ Start$ |
| Fails Per Failed | $= Fail\ Count\ /\ Failed\ User$ |
| Shuffle Percentage | $= Shuffle\ Count\ /\ User\ Start$ |
| Shuffles Per Shuffled | $= Shuffle\ Count\ /\ Shuffled\ User$ |
| Reverse Percentage | $= Reverse\ Count\ /\ User\ Start$ |
| Reverses Per Reversed | $= Reverse\ Count\ /\ Reversed\ User$ |
| Continue Percentage | $= Continue\ Count\ /\ User\ Start$ |
| Continues Per Continued | $= Continue\ Count\ /\ Continued\ User$ |
| Time Percentage | $= Avg\ Time\ /\ Given\ Time$ |

There are different types of objects and features in the game. The player may be unfamiliar with or have difficulty in the first level where these are seen. For this reason, we have created a new feature that shows the levels in these situations. The newly added features are shown in the Table 4.4.

**Table 4.4.** Design data features after extraction.

| Attributes | Description |
|---|---|
| New Feature Seen | Is there a new feature in level? |
| New Object Seen | Is there a new object in level? |

When we analyzed the dataset, we observed that the 'Width' feature is 4 for all levels. For this reason, we remove this feature, as it will not have any effect on the classification model. We also remove the LevelID and Main Level ID so that it does not cause data memorization. Since the game data is meaningful according to the level, we grouped the levels every 9 chapters. After the changes made, the properties in the data set are as stated in the Table 4.5.

**Table 4.5.** Final version of the features in the data set

| Attributes | Attribute Type |
|---|---|
| Stage | Int |
| Height | Int |
| Total Object | Int |
| Type Count | Int |
| Locked Box | Int |

| Attributes | Attribute Type |
|---|---|
| Soft Locked Box | Bool |
| Surprise Box | Bool |
| Vent Count | Int |
| Vent Limit | Int |
| Combo Box_L1 | Int |
| Combo Box_L2 | Int |
| Combo Box_L3 | Int |
| Key Box_L1 | Int |
| Key Box_L2 | Int |
| Key Box_L3 | Int |
| Feature Type Count | Int |
| New Feature Seen | Bool |
| New Object Seen | Bool |
| Avg Time | Float |
| Fail Percentage | Float |
| Fails Per Failed | Float |
| Shuffle Percentage | Float |
| Shuffles Per Shuffled | Float |
| Reverse Percentage | Float |
| Reverses Per Reversed | Float |
| Continue Percentage | Float |
| Continues Per Continued | Float |
| Time Percentage | Float |

### 4.3.3. Normalization

The data set contains a wide range of different values between 0 and 600. For the model to better understand the values, a normalization transformation was performed. All data were transformed between 0 and 1. The Min Max Scaler method was used to perform this normalization.

### 4.3.4. Test and train splitting

In this section, our data set is divided to train and test the model. As mentioned in the previous sections, 70% or 80% is usually used for the training data set. In this study, different data set variations were created to measure the contribution of the preprocessing we performed on the data set to the success of the classification models. In this study, we

set the train size ratio as 85% and evaluated the results with k-fold as 5 in the cross-validation method to evaluate the model performance more accurately. Version of sets are given in the Table 4.6.

**Table 4.6.** Description of data sets

| Data Set | Description |
|---|---|
| Data_v0 | Data set before preprocess |
| Data_v0_Tuned | Tuned version of Data_v0 |
| Data_v1 | Preprocessed data set |
| Data_v1_Tuned | Tuned version of Data_v1 |
| Data_v2 | Applied feature selection version |
| Data_v2_Tuned | Tuned version of Data_v2 |

### 4.3.5. Feature selection sets by algorithms

In this study, we created different feature sets to measure the effect of feature selection on model performance. A model was created on the Data_v1 dataset with the algorithms used in the study. For each model, the 10 most effective features were determined, and separate sets were created. These sets are explained in the Table 4.7.

**Table 4.7.** Description of feature selection data sets

| Data Set | According to |
|---|---|
| lr | Logistic Regression |
| ada | AdaBoost |
| dt | Decision Tree |
| et | Extra Tree Classifier |
| gbc | Gradient Boosting Classifier |
| lgbm | Light Gradient Boosting Machine |
| rf | Random Forest |
| svm | Support Vector Machines |
| xgb | XGBoost |

## 4.4. Model Evaluation

In this step, the prepared data set was tested with the classification algorithms. The performance of each classification algorithm with sub-datasets is evaluated separately.

The F1 score was chosen to evaluate the model performance. Result of all classification models explained with details in Chapter 5.

# 5. RESULTS AND DISCUSSION

## 5.1. Results of Models

In this section, the results of the models used in the classification problem are evaluated and discussed separately.

### 5.1.1. Performance of decision tree

In this section, the classification problem is evaluated with the Decision Tree algorithm. Firstly, the performances of the raw and preprocessed versions of the dataset are compared.

**Table 5.1.** Performance comparison of DT with preprocessed and non-preprocessed data

| Data | Accuracy | Recall | Precision | F1 |
|---|---|---|---|---|
| Data_v0 | 62.50% | 62.50% | 63.55% | 62.33% |
| Data_v1 | 69.14% | 69.14% | 64.40% | **66.41%** |

The data preprocessing improved the performance of the Decision Tree algorithm by approximately 6.7% for the accuracy metric and 4.1% for the F1 score. The most important features for this model have been selected to further improve this model. These features are shown in Figure 5.1.

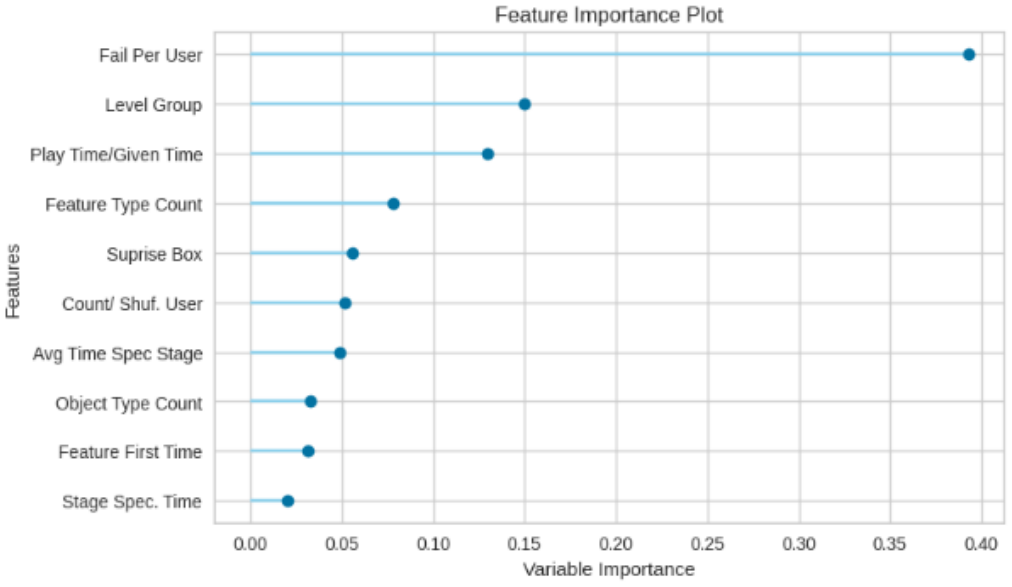


**Figure 5.1.** Most important features according to DT

It is seen that the fail ratio of the players has the highest effect on the outcome with this model. The model was reconstructed with the features shown in Figure 5.1. Performance results after feature selection are as follows in Table 5.2.

**Table 5.2.** Performance comparison of DT after feature selection

| Data | Accuracy | Recall | Precision | F1 |
|---|---|---|---|---|
| Data_v1 | 69.14% | 62.50% | 64.40% | 66.41% |
| Data_v2 | 70.48% | 70.48% | 69.41% | **69.61%** |

After the feature selection, it is seen that there is a 1.9% improvement in the accuracy metric and a 1.3% improvement in the F1 score. The changes in the data set were finalized according to the Decision Tree algorithm. The results so far were the results produced by the model with default settings. To achieve the best results, we adjusted the hyperparameters of our model according to the best version. The results of the improvement are given in Table 5.3.

**Table 5.3.** Performance comparison of DT after model tuning

| Data | Accuracy | Recall | Precision | F1 |
|---|---|---|---|---|
| Data_v2 | 70.48% | 70.48% | 69.41% | 69.61% |
| Data_v2_Tuned | 79.90% | 79.90% | 78.95% | **78.09%** |

When the results are analyzed, it is seen that the tuning process increased the performance by 9.4% in the accuracy metric and 8.5% in the F1 score. The results of all the improvements made to the DT algorithm are given in the Table 5.4.

**Table 5.4.** Performance comparison of the improvement steps with DT

| Data | Accuracy | Change | F1 | Change |
|---|---|---|---|---|
| Data_v0 | 62.50% | - | 62.23% | - |
| Data_v1 | 69.14% | 6.6% | 66.41% | 4.18% |
| Data_v2 | 70.48% | 7.98% | 69.61% | 7.38% |
| Data_v2_Tuned | 79.90% | 17.4% | **78.09%** | **15.86%** |

At the beginning of the classification problem, the accuracy and F1 metrics of our model were around 62%. Improvements in the prediction ability of the model are clearly seen after the operations are performed. An increase of 17.4% for the accuracy value and

15.9% for the F1 score was achieved.

### 5.1.2. Performance of random forest

In this section, the classification problem is evaluated with the Random Forest algorithm. Firstly, the performances of the raw and preprocessed versions of the dataset are compared.

**Table 5.5.** Performance comparison of RF with preprocessed and non-preprocessed data

| Data | Accuracy | Recall | Precision | F1 |
|---|---|---|---|---|
| Data_v0 | 70.33% | 70.33% | 63.44% | 66.24% |
| Data_v1 | 74.48% | 74.48% | 69.85% | **70.76%** |

The data preprocessing improved the performance of the Random Forest algorithm by approximately 4.15% for the accuracy metric and 4.5% for the F1 score. The most important features for this model have been selected to further improve this model. These features are shown in Figure 5.2.

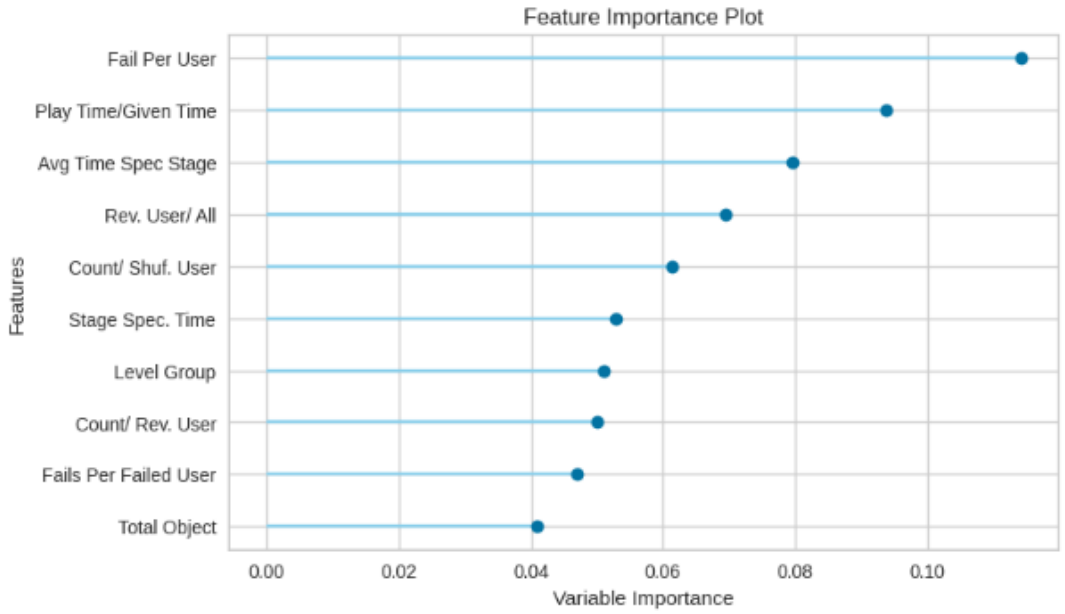


**Figure 5.2.** Most important features according to RF

It is seen that the fail ratio of the players has the highest effect on the outcome with this model. The model was reconstructed with the features shown in Figure 5.2.

Performance results after feature selection are as follows in Table 5.6.

**Table 5.6.** Performance comparison of RF after feature selection

| Data | Accuracy | Recall | Precision | F1 |
|---|---|---|---|---|
| Data_v1 | 74.48% | 74.48% | 69.85% | **70.76%** |
| Data_v2 | 74.48% | 74.48% | 69.85% | **70.76%** |

After the feature selection, no change was observed in the performance values. We think that such an output is obtained due to the working logic of the RF algorithm. The results so far were the results produced by the model with default settings. To achieve the best results, we adjusted the hyperparameters of our model according to the best version. The results of the improvement are given in Table 5.7.

**Table 5.7.** Performance comparison of RF after model tuning

| Data | Accuracy | Recall | Precision | F1 |
|---|---|---|---|---|
| Data_v2 | 74.48% | 74.48% | 69.85% | 70.76% |
| Data_v2_Tuned | 77.14% | 77.14% | 73.04% | **72.48%** |

When the results are analyzed, it is seen that the tuning process increased the performance by 2.7% in the accuracy metric and 1.7% in the f1 score. The results of all the improvements made to the RF algorithm are given in the Table 5.8.

**Table 5.8.** Performance comparison of the improvement steps with RF

| Data | Accuracy | Change | F1 | Change |
|---|---|---|---|---|
| Data_v0 | 70.33% | - | 66.24% | - |
| Data_v1 | 74.48% | 4.15% | 70.76% | 4,52% |
| Data_v2 | 74.48% | 4.15% | 70.76% | 4.52% |
| Data_v2_Tuned | 77.14% | 6.81% | **72.48%** | **6.24%** |

At the beginning of the classification problem, the accuracy was 70.33% and the F1 metrics of the model were around 66.24%. An increase of 6.81% for the accuracy value and 6.24% for the F1 score was achieved.

### 5.1.3. Performance of extra tree classifier

In this section, the classification problem is evaluated with the Extra Tree Classifier. Firstly, the performances of the raw and preprocessed versions of the dataset are compared.

**Table 5.9.** Performance comparison of ET with preprocessed and non-preprocessed data

| Data | Accuracy | Recall | Precision | F1 |
|---|---|---|---|---|
| Data_v0 | 67.75% | 67.75% | 62.89% | 64.96% |
| Data_v1 | 70.38% | 70.38% | 64.24% | **66.71%** |

The data preprocessing improved the performance of the Extra Tree Classifier 2.63% for the accuracy metric and 1.75% for the F1 score. The most important features for this model have been selected to further improve this model. These features are shown in Figure 5.3.

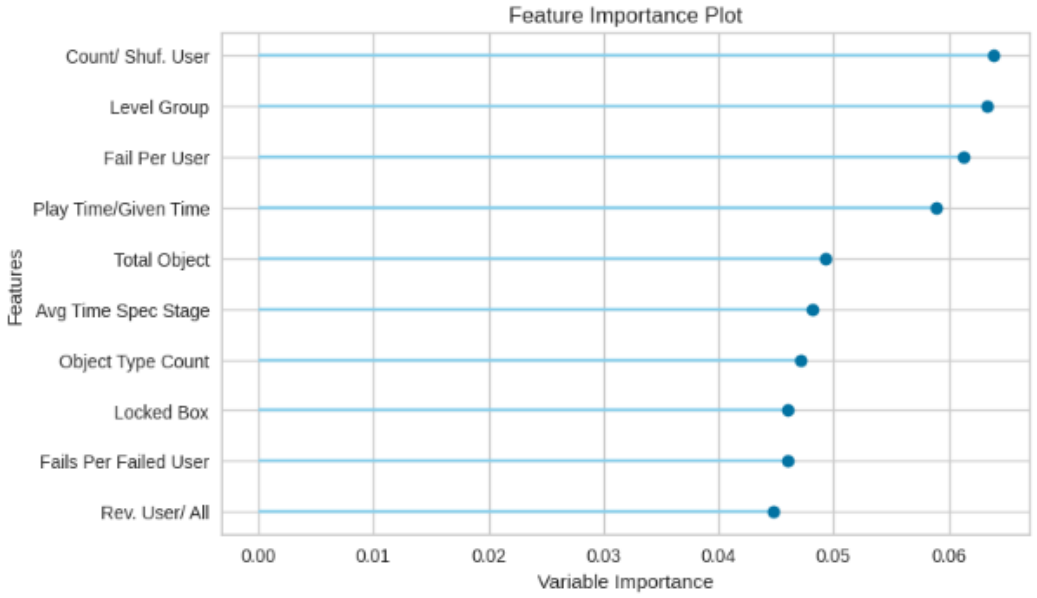


**Figure 5.3.** Most important features according to ET

It is seen that the level group and using shuffle per shuffled players has the highest effect on the outcome with this model. The model was reconstructed with the features shown in Figure 4.3. Performance results after feature selection are as follows in Table 5.10.

**Table 5.10.** Performance comparison of ET after feature selection

| Data | Accuracy | Recall | Precision | F1 |
|---|---|---|---|---|
| Data_v1 | 70.38% | 70.38% | 64.28% | 66.71% |
| Data_v2 | 74.38% | 74.38% | 63.99% | **68.46%** |

After the feature selection, it is seen that there is a 1.9% improvement in the accuracy metric and a 1.3% improvement in the F1 score. The changes in the data set were finalized according to the Decision Tree algorithm. The results so far were the results produced by the model with default settings. To achieve the best results, we adjusted the hyperparameters of our model according to the best version. The results of the improvement are given in Table 5.11.

**Table 5.11.** Performance comparison of ET after model tuning

| Data | Accuracy | Recall | Precision | F1 |
|---|---|---|---|---|
| Data_v2 | 74.38% | 74.38% | 63.99% | 68.46% |
| Data_v2_Tuned | 73.14% | 72.14% | 72.42% | **72.48%** |

When the results are analyzed, it is seen that the tuning process decreased the performance by 1.24% in the accuracy metric and increased 3.96% in the f1 score. The results of all the improvements made to the ET algorithm are given in the Table 5.12.

**Table 5.12.** Performance comparison of the improvement steps with ET

| Data | Accuracy | Change | F1 | Change |
|---|---|---|---|---|
| Data_v0 | 67.75% | - | 64.96% | - |
| Data_v1 | 70.38% | 2.63% | 66.71% | 1.75% |
| Data_v2 | 74.38% | 6.63% | 68.46% | 3.50% |
| Data_v2_Tuned | 73.14% | 5.39% | **72.48%** | **7.52%** |

At the beginning of the classification problem, the accuracy was 67.75% and F1 metrics was 64.96% in our model. Improvements in the prediction ability of the model are clearly seen after the operations are performed. An increase of 5.39% for the accuracy value and 7.52% for the F1 score was achieved.

### 5.1.4. Performance of naïve bayes

In this section, the classification problem is evaluated with the Naïve Bayes. Firstly, the performances of the raw and preprocessed versions of the dataset are compared.

**Table 5.13.** Performance comparison of NB with preprocessed and non-preprocessed data

| Data | Accuracy | Recall | Precision | F1 |
|---|---|---|---|---|
| Data_v0 | 36.5% | 36.5% | 54.14% | 34.60% |
| Data_v1 | 36.48% | 36.48% | 64.24% | **35.86%** |

The most important features of the Naive Bayes algorithm were not reached. The model was reconstructed with the features with obtain DT. Performance results after feature selection are as follows in Table 5.14.

**Table 5.14.** Performance comparison of NB after feature selection

| Data | Accuracy | Recall | Precision | F1 |
|---|---|---|---|---|
| Data_v1 | 36.48% | 36.48% | 64.24% | 35.86% |
| Data_v2 | 67.71% | 67.71% | 70.33% | **68.79%** |

After the feature selection, it is seen that there is a 31.23% improvement in the accuracy metric and a 32.93% improvement in the F1 score. The changes in the data set were finalized according to the NB algorithm. Since the default settings of the model are the best settings, there is no effect on the model performance. The results of all the improvements made to the NB algorithm are given in the Table 5.15.

**Table 5.15.** Performance comparison of the improvement steps with NB

| Data | Accuracy | Change | F1 | Change |
|---|---|---|---|---|
| Data_v0 | 36.50% | - | 34.60% | - |
| Data_v1 | 36.48% | -0.02% | 35.86% | 1.26% |
| Data_v2 | 67.71% | 31.21% | 68.79% | 34.19% |
| Data_v2_Tuned | 67.71% | 31.21% | **68.79%** | **34.19%** |

NB algorithm performs very poorly without feature selection. With feature selection, the accuracy metric increased by 31.21% and the F1 score increased by 34.19%.

### 5.1.5. Performance of logistic regression

In this section, the classification problem is evaluated with the Logistic Regression algorithm. Firstly, the performances of the raw and preprocessed versions of the dataset are compared.

**Table 5.16.** Performance comparison of LR with preprocessed and non-preprocessed data

| Data | Accuracy | Recall | Precision | F1 |
|---|---|---|---|---|
| Data_v0 | 61.00% | 61.00% | 60.32% | 60.31% |
| Data_v1 | 75.71% | 75.71% | 59.18% | **66.42%** |

The data preprocessing improved the performance of the Logistic Regression algorithm by 14.71% for the accuracy metric and 6.1% for the F1 score. The most important features for this model have been selected to further improve this model. These features are shown in Figure 5.4.

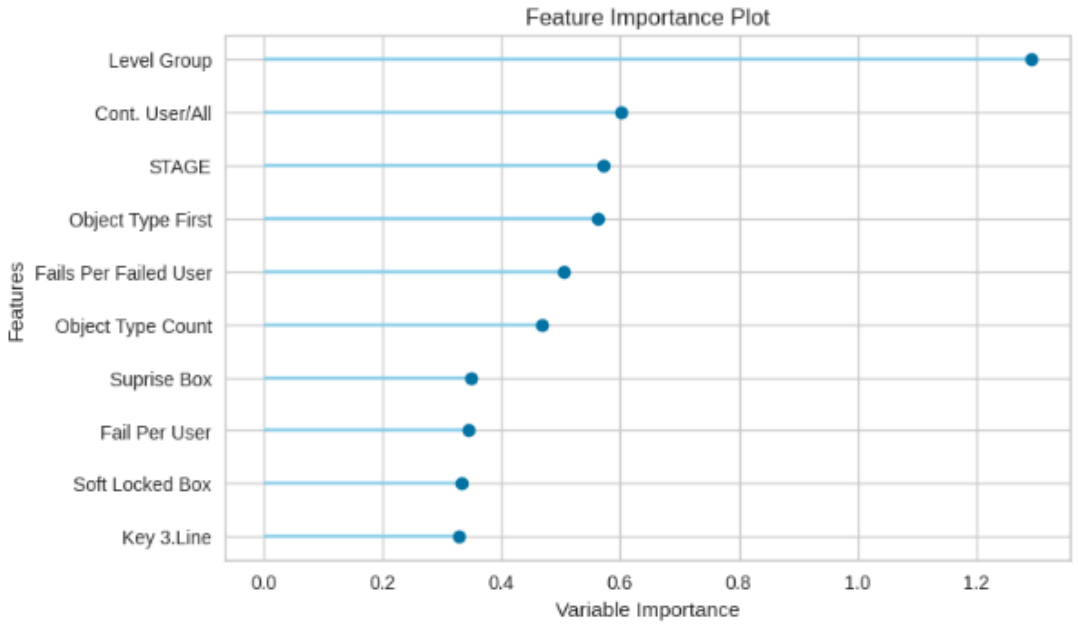


**Figure 5.4.** Most important features according to LR

It is seen that Level Group has the highest effect on the outcome with this model. The model was reconstructed with the features shown in Figure 5.4. Performance results after feature selection are as follows in Table 5.17.

**Table 5.17.** Performance comparison of LR after feature selection

| Data | Accuracy | Recall | Precision | F1 |
|---|---|---|---|---|
| Data_v1 | 75.71% | 75.71% | 59.18% | 66.42% |
| Data_v2 | 75.71% | 75.71% | 59.18% | **66.42%** |

According to the results, feature selection does not contribute to performance when evaluated in terms of data set and model. The results so far were the results produced by the model with default settings. To achieve the best results, we adjusted the hyperparameters of our model according to the best version. The results of the improvement are given in Table 5.18.

**Table 5.18.** Performance comparison of LR after model tuning

| Data | Accuracy | Recall | Precision | F1 |
|---|---|---|---|---|
| Data_v2 | 75.71% | 75.71% | 59.18% | 66.42% |
| Data_v2_Tuned | 66.38% | 66.38% | 73.82% | **68.56%** |

When the results are analyzed, it is seen that the tuning process decreased the performance by 9.33% in the accuracy metric and increased 2.14% in the F1 score. We made the model optimization process according to the F1 score, so there was a loss of performance in other values. The results of all the improvements made to the LR algorithm are given in the Table 5.19.

**Table 5.19.** Performance comparison of the improvement steps with LR

| Data | Accuracy | Change | F1 | Change |
|---|---|---|---|---|
| Data_v0 | 61.00% | - | 60.31% | - |
| Data_v1 | 75.71% | 14.71% | 66.42% | 6.11% |
| Data_v2 | 75.71% | 14.71% | 66.42% | 6.11% |
| Data_v2_Tuned | 66.38% | 5.38% | **68.56%** | **8.25%** |

At the beginning of the classification problem, the accuracy was 61% and F1 metrics was 60.31% in our model. Improvements in the prediction ability of the model are clearly seen after the operations are performed. An increase of 5.38% for the accuracy value and 8.25% for the F1 score was achieved.

### 5.1.6. Performance of support vector machines

In this section, the classification problem is evaluated with the Support Vector Machines. Firstly, the performances of the raw and preprocessed versions of the dataset are compared.

**Table 5.20.** Performance comparison of SVM with preprocessed and non-preprocessed data

| Data | Accuracy | Recall | Precision | F1 |
|---|---|---|---|---|
| Data_v0 | 68.50% | 68.58% | 68.44% | 64.22% |
| Data_v1 | 66.29% | 66.29% | 60.99% | 63.38% |

The data preprocessing negatively improved the performance of the Decision Tree algorithm by approximately 2.21% for the accuracy metric and 0.84% for the f1 score. The most important features for this model have been selected to further improve this model. These features are shown in Figure 5.5.

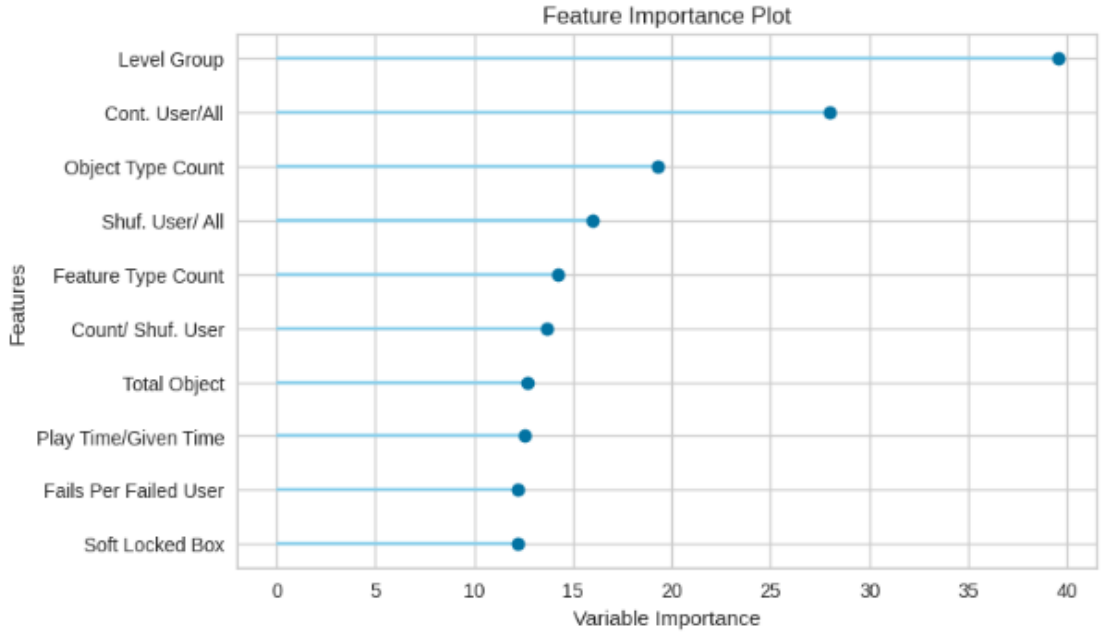


**Figure 5.5.** Most important features according to SVM

It is seen that Level Group has the highest effect on the outcome with this model. The model was reconstructed with the features shown in Figure 5.5. Performance results after feature selection are as follows in Table 5.21.

**Table 5.21.** Performance comparison of SVM after feature selection

| Data | Accuracy | Recall | Precision | F1 |
|---|---|---|---|---|
| Data_v1 | 66.29% | 66.29% | 60.99% | 63.38% |
| Data_v2 | 64.67% | 64.67% | 76.97% | **63.66%** |

After the feature selection, it is seen that there is a 1.62% negative improvement in the accuracy metric and a 0.28% improvement in the f1 score. After the preprocess and feature selection operations, the performance criteria were negatively affected. For this reason, we performed the model improvement process over Data_v0. To achieve the best results, we adjusted the hyperparameters of our model according to the best version. The results of the improvement are given in Table 5.22.

**Table 5.22.** Performance comparison of SVM after model tuning

| Data | Accuracy | Recall | Precision | F1 |
|---|---|---|---|---|
| Data_v0 | 68.50% | 68.58% | 68.44% | 64.22% |
| Data_v0_Tuned | 76.67% | 76.67% | 73.50% | **72.56%** |

When the results are analyzed, it is seen that the tuning process increased the performance by 8.17% in the accuracy metric and 8.34% in the f1 score. At the beginning of the classification problem, the accuracy was 68.58% and F1 metrics was 60.31% in our model. Improvements in the prediction ability of the model are clearly seen after the operations are performed. It can be said that the SVM algorithm performs better on less processed data under these conditions.

### 5.1.7. Performance of k-nearest neighbors

In this section, the classification problem is evaluated with the K-Nearest Neighbors algorithm. Firstly, the performances of the raw and preprocessed versions of the dataset are compared.

**Table 5.23.** Performance comparison of KNN with preprocessed and non-preprocessed data

| Data | Accuracy | Recall | Precision | F1 |
|---|---|---|---|---|
| Data_v0 | 70.33% | 70.33% | 60.29% | 64.71% |

| Data | Accuracy | Recall | Precision | F1 |
|---|---|---|---|---|
| Data_v1 | 74.38% | 74.38% | 63.59% | **67.60%** |

The data preprocessing improved the performance of the KNN algorithm by 4.05% for the accuracy metric and 2.89% for the f1 score. The most important features of the KNN algorithm were not reached. The model was reconstructed with the features with obtain DT. Performance results after feature selection are as follows in Table 5.24.

**Table 5.24.** Performance comparison of KNN after feature selection

| Data | Accuracy | Recall | Precision | F1 |
|---|---|---|---|---|
| Data_v1 | 74.38% | 74.38% | 63.59% | 67.60% |
| Data_v2 | 74.38% | 74.38% | 69.83% | **70.30%** |

Although there was no change in the accuracy metric after feature selection, there was a 2.7% increase in the F1 score. The changes in the data set were finalized according to the KNN algorithm and the results of all the improvements made are given in the Table 5.25.

**Table 5.25.** Performance comparison of the improvement steps with KNN

| Data | Accuracy | Change | F1 | Change |
|---|---|---|---|---|
| Data_v0 | 70.33% | - | 64.71% | - |
| Data_v1 | 74.38% | 4.05% | 67.60% | 2.89% |
| Data_v2 | 74.38% | 4.05% | 70.30% | 5.59% |
| Data_v2_Tuned | 71.71% | 1.38% | 71.81% | **7.10%** |

At the beginning of the classification problem, the accuracy was 70.33% and F1 metrics was 64.71% in our model. Improvements in the prediction ability of the model are clearly seen after the operations are performed. An increase of 1.38% for the accuracy value and 7.10% for the F1 score was achieved.

### 5.1.8. Performance of adaboost

In this section, the classification problem is evaluated with the AdaBoost algorithm. Firstly, the performances of the raw and preprocessed versions of the dataset are compared.

**Table 5.26.** Performance comparison of ADA with preprocessed and non-preprocessed data

| Data | Accuracy | Recall | Precision | F1 |
|---|---|---|---|---|
| Data_v0 | 69.00% | 69.00% | 62.32% | 65.30% |
| Data_v1 | 69.14% | 69.14% | 66.74% | **67.81%** |

The data preprocessing improved the performance of the AdaBoost algorithm by 0.14% for the accuracy metric and 2.51% for the f1 score. The most important features for this model have been selected to further improve this model. These features are shown in Figure 5.6.

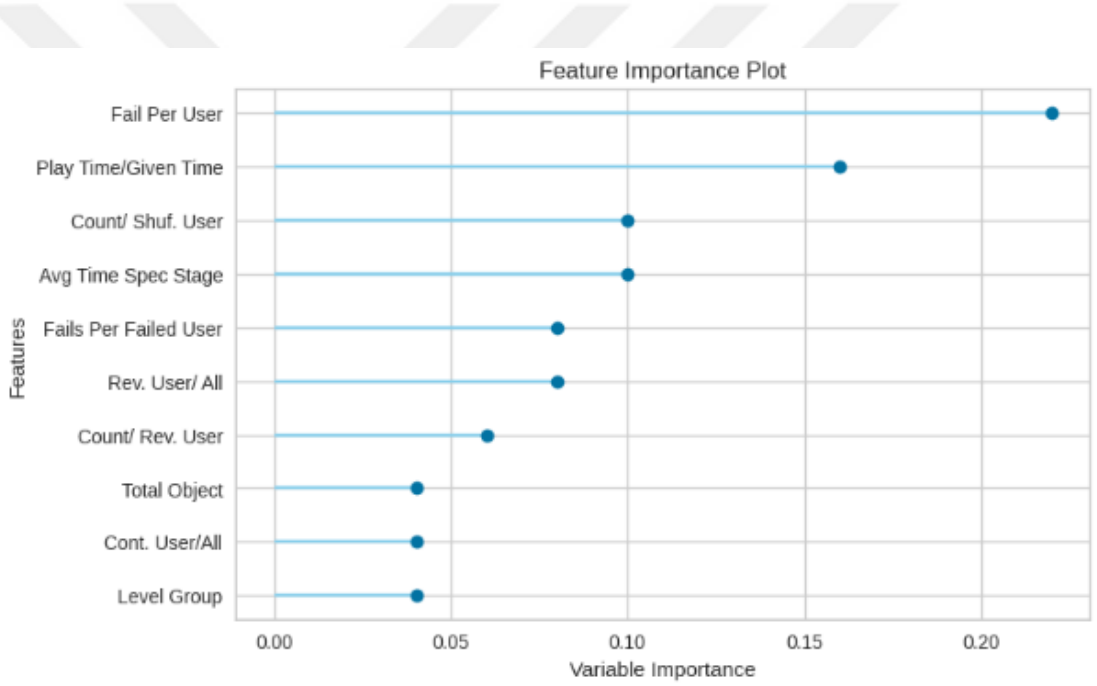


**Figure 5.6.** Most important features according to ADA

It is seen that the fail ratio of the players has the highest effect on the outcome with this model. The model was reconstructed with the features shown in Figure 5.6. Performance results after feature selection are as follows in Table 5.27.

**Table 5.27.** Performance comparison of ADA after feature selection

| Data | Accuracy | Recall | Precision | F1 |
|---|---|---|---|---|
| Data_v1 | 69.14% | 69.14% | 66.74% | 67.81% |
| Data_v2 | 73.05% | 73.05% | 71.95% | **71.94%** |

After the feature selection, it is seen that there is a 3.91% improvement in the accuracy metric and a 4.13% improvement in the F1 score. The changes in the data set were finalized according to the AdaBoost. The results so far were the results produced by the model with default settings. To achieve the best results, we adjusted the hyperparameters of our model according to the best version. The results of the improvement are given in Table 5.28.

**Table 5.28.** Performance comparison of ADA after model tuning

| Data | Accuracy | Recall | Precision | F1 |
|---|---|---|---|---|
| Data_v2 | 73.05% | 73.05% | 71.95% | 71.94% |
| Data_v2_Tuned | 73.05% | 73.05% | 71.95% | **71.94%** |

When the results are analyzed, it is seen that the tuning process not effect performance. Default parameters have the best values for these conditions. The changes in the data set were finalized according to the ADA and the results of all the improvements made are given in the Table 5.29.

**Table 5.29.** Performance comparison of the improvement steps with ADA

| Data | Accuracy | Change | F1 | Change |
|---|---|---|---|---|
| Data_v0 | 69.00% | - | 65.30% | - |
| Data_v1 | 69.14% | 0.14% | 67.81% | 2.51% |
| Data_v2 | 73.05% | 4.05% | 71.94% | 6.64% |
| Data_v2_Tuned | 73.05% | 4.05% | **71.94%** | **6.64%** |

At the beginning of the classification problem, the accuracy was 69% and the F1 metrics of the model were around 65.30%. An increase of 4.05% for the accuracy value and 6.64% for the F1 score was achieved.

### 5.1.9. Performance of Gradient Boosting Classifier

In this section, the classification problem is evaluated with the Gradient Boosting Classifier. Firstly, the performances of the raw and preprocessed versions of the dataset are compared.

**Table 5.30.** Performance comparison of GBC with preprocessed and non-preprocessed data

| Data | Accuracy | Recall | Precision | F1 |
|---|---|---|---|---|
| Data_v0 | 66.58% | 66.58% | 65.57% | 65.34% |
| Data_v1 | 74.38% | 74.38% | 71.00% | **71.79%** |

The data preprocessing improved the performance of the Gradient Boosting Classifier by 7.80% for the accuracy metric and 6.45% for the F1 score. The most important features for this model have been selected to further improve this model. These features are shown in Figure 5.7.

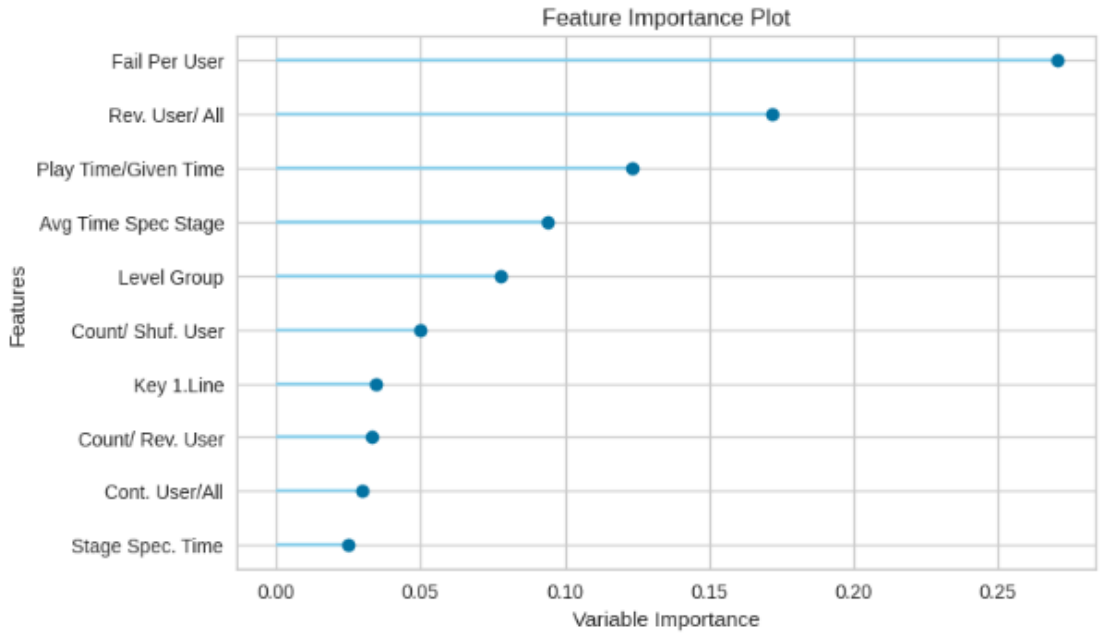


**Figure 5.7.** Most important features according to GBC

It is seen that the fail ratio of the players has the highest effect on the outcome with this model. The model was reconstructed with the features shown in Figure 5.7. Performance results after feature selection are as follows in Table 5.31.

**Table 5.31.** Performance comparison of GBC after feature selection

| Data | Accuracy | Recall | Precision | F1 |
|---|---|---|---|---|
| Data_v1 | 74.38% | 74.38% | 71.00% | **71.79%** |
| Data_v2 | 73.05% | 73.05% | 72.71% | 71.09% |

After the feature selection, it is seen that there is a 1.33% improvement negatively in the accuracy metric and a 0.7% negative improvement in the F1 score. The changes in the data set were finalized according to the GBC. The results so far were the results produced by the model with default settings. To achieve the best results, we adjusted the hyperparameters of our model according to the best version. The results of the improvement are given in Table 5.32.

**Table 5.32.** Performance comparison of GBC after model tuning

| Data | Accuracy | Recall | Precision | F1 |
|---|---|---|---|---|
| Data_v2 | 73.05% | 73.05% | 72.71% | 71.09% |
| Data_v2_Tuned | 76.86% | 76.86% | 73.84% | **72.28%** |

When the results are analyzed, it is seen that the tuning process increased the performance by 3.82% in the accuracy metric and 1.19% in the F1 score. The changes in the data set were finalized according to the GBC and the results of all the improvements made are given in the Table 5.33.

**Table 5.33.** Performance comparison of the improvement steps with GBC

| Data | Accuracy | Change | F1 | Change |
|---|---|---|---|---|
| Data_v0 | 66.58% | - | 65.34% | - |
| Data_v1 | 74.38% | 7.80% | 71.79% | 6.45% |
| Data_v2 | 73.05% | 6.47% | 71.09% | 5.75% |
| Data_v2_Tuned | 76.85% | 10.28% | **72.28%** | **6.94%** |

At the beginning of the classification problem, the accuracy was 66% and the F1 metrics of the model were around 65.34%. An increase of 10.28% for the accuracy value and 6.94% for the F1 score was achieved.

### 5.1.10. Performance of xgboost

In this section, the classification problem is evaluated with the XGBoost Classifier. Firstly, the performances of the raw and preprocessed versions of the dataset are compared.

**Table 5.34.** Performance comparison of XGB with preprocessed and non-preprocessed data

| Data | Accuracy | Recall | Precision | F1 |
|---|---|---|---|---|
| Data_v0 | 67.67% | 67.67% | 64.10% | 65.57% |
| Data_v1 | 71.71% | 71.71% | 67.12% | **68.60%** |

The data preprocessing improved the performance of the XGBoost by 4.04% for the accuracy metric and 3.03% for the F1 score. The most important features for this model have been selected to further improve this model. These features are shown in Figure 5.8.

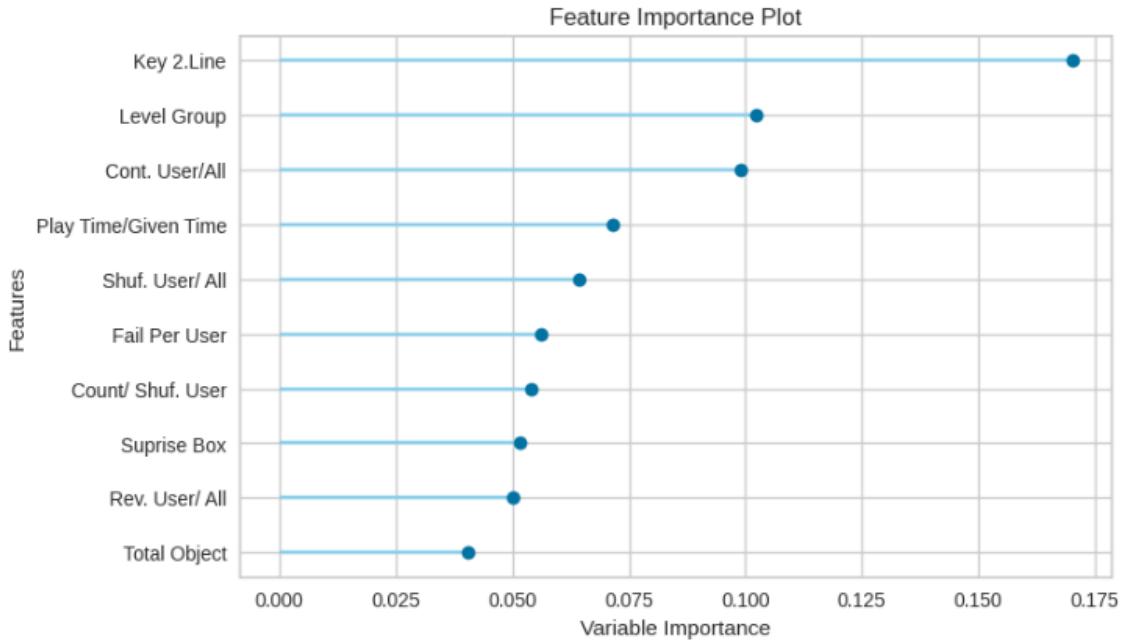


**Figure 5.8.** Most important features according to XGB

It is seen that key feature at line 2 of level has the highest effect on the outcome with this model. The model was reconstructed with the features shown in Figure 5.8. Performance results after feature selection are as follows in Table 5.35.

**Table 5.35.** Performance comparison of XGB after feature selection

| Data | Accuracy | Recall | Precision | F1 |
|---|---|---|---|---|
| Data_v1 | 71.71% | 71.71% | 67.12% | 68.60% |
| Data_v2 | 73.14% | 73.14% | 68.67% | **69.48%** |

After the feature selection, it is seen that there is a 1.40% improvement in the

accuracy metric and a 0.88% improvement in the F1 score. The changes in the data set were finalized according to the XGB. The results so far were the results produced by the model with default settings. To achieve the best results, we adjusted the hyperparameters of our model according to the best version. The results of the improvement are given in Table 5.36.

**Table 5.36.** Performance comparison of XGB after model tuning

| Data | Accuracy | Recall | Precision | F1 |
|---|---|---|---|---|
| Data_v2 | 73.14% | 73.14% | 68.67% | 69.48% |
| Data_v2_Tuned | 81.14% | 81.14% | 75.71% | **74.99%** |

When the results are analyzed, it is seen that the tuning process increased the performance by 8% in the accuracy metric and 5.51% in the F1 score. The changes in the data set were finalized according to the XGB and the results of all the improvements made are given in the Table 5.37.

**Table 5.37.** Performance comparison of the improvement steps with XGB

| Data | Accuracy | Change | F1 | Change |
|---|---|---|---|---|
| Data_v0 | 67.67% | - | 65.57% | - |
| Data_v1 | 71.71% | 4.04% | 68.60% | 3.03% |
| Data_v2 | 73.14% | 5.47% | 69.48% | 3.91% |
| Data_v2_Tuned | 81.14% | 13.47% | **74.99%** | **9.42%** |

At the beginning of the classification problem, the accuracy was 67.76% and the F1 metrics of the model were around 65.57%. An increase of 13.47% for the accuracy value and 9.42% for the F1 score was achieved.

### 5.1.11. Performance of light gradient boosting machine

In this section, the classification problem is evaluated with the Light Gradient Boosting Machine. Firstly, the performances of the raw and preprocessed versions of the dataset are compared.

**Table 5.38.** Performance comparison of LGBM with preprocessed and non-preprocessed data

| Data | Accuracy | Recall | Precision | F1 |
|---|---|---|---|---|
| Data_v0 | 74.25% | 74.25% | 70.26% | **70.63%** |
| Data_v1 | 73.14% | 73.14% | 68.16% | 68.59% |

The data preprocessing negatively improved the performance of the LGBM by 1.11% for the accuracy metric and 2.04% for the F1 score. The most important features for this model have been selected to further improve this model. These features are shown in Figure 5.9.

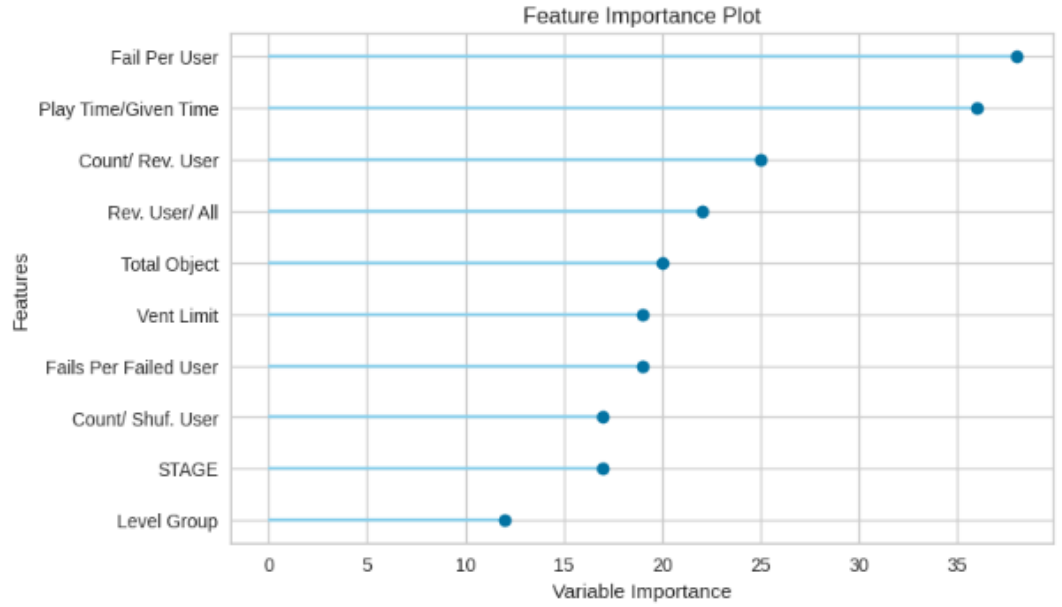


**Figure 5.9.** Most important features according to LGBM

It is seen that fail ratio of player has the highest effect on the outcome with this model. The model was reconstructed with the features shown in Figure 5.9. Performance results after feature selection are as follows in Table 5.39.

**Table 5.39.** Performance comparison of LGBM after feature selection

| Data | Accuracy | Recall | Precision | F1 |
|---|---|---|---|---|
| Data_v1 | 73.14% | 73.14% | 68.16% | 68.59% |
| Data_v2 | 73.14% | 73.14% | 68.67% | **69.48%** |

After the feature selection, it is seen that there is a 0.87% improvement in the F1 score. The changes in the data set were finalized according to the LGBM. The results so

far were the results produced by the model with default settings. To achieve the best results, we adjusted the hyperparameters of our model according to the best version. The results of the improvement are given in Table 5.40.

**Table 5.40.** Performance comparison of LGBM after model tuning

| Data | Accuracy | Recall | Precision | F1 |
|---|---|---|---|---|
| Data_v2 | 73.14% | 73.14% | 68.67% | 69.48% |
| Data_v2_Tuned | 83.14% | 81.14% | 75.71% | **74.99%** |

When the results are analyzed, it is seen that the tuning process increased the performance by 10% in the accuracy metric and 5.51% in the F1 score. The changes in the data set were finalized according to the LGBM and the results of all the improvements made are given in the Table 5.41.

**Table 5.41.** Performance comparison of the improvement steps with LGBM

| Data | Accuracy | Change | F1 | Change |
|---|---|---|---|---|
| Data_v0 | 74.25% | - | 70.63% | - |
| Data_v1 | 73.14% | -1.11% | 68.59% | -2.04% |
| Data_v2 | 73.14% | -1.11% | 69.48% | -1.15% |
| Data_v2_Tuned | 81.14% | 6.89% | 74.99% | **4.36%** |

At the beginning of the classification problem, the accuracy was 74.25% and the F1 metrics of the model were around 70.63%. An increase of 6.89% for the accuracy value and 4.36% for the F1 score was achieved.

## 5.2. Comparison of Models

In this section, the results obtained with different classification algorithms and different improvement processes are evaluated.

### 5.2.1. Performance impact of preprocessing

In this section, the performance of the preprocessed data set compared to the unprocessed data set is analyzed. The result of the comparison is given in the Table 5.42.

**Table 5.42.** Impact of preprocessing to model performance

| Algorithm | F1 without Preprocess | F1 with Preprocess | Change |
|---|---|---|---|
| Decision Tree | 62.23% | 66.41% | 4.08% |
| **Random Forest** | **66.24%** | **70.76%** | **4.52%** |
| Extra Tree | 64.96% | 66.71% | 1.75% |
| Naïve Bayes | 34.60% | 35.86% | 1.26% |
| Logistic Regression | 60.31% | 66.42% | 6.11% |
| Support Vector Machines | 68.44% | 60.99% | -7.45% |
| K-Nearest Neighbors | 64.71% | 67.60% | 2.89% |
| AdaBoost | 65.30% | 67.81% | 2.51% |
| **Gradient Boosting** | **65.34%** | **71.79%** | **6.45%** |
| XGBoost | 65.57% | 68.60% | 3.03% |
| LightGBM | 70.26% | 68.16% | -2.1% |

According to the comparison, SVM and LGBM algorithms are negatively affected by preprocessing. Other algorithms improved between 1.26% and 6.45%. GBC improved the most with the preprocessing process and obtained the highest f1 value in solving the classification problem at this stage. The worst result, both with and without preprocessing, was the NB classifier with 35.86%.

### 5.2.2. Performance impact of feature selection and tuning

In this section, the improvement of model performance after feature selection and tuning is analyzed. Feature selection was performed with a set of the 10 most important features after each model was tested with preprocessed data.

**Table 5.43.** Impact of tuned with feature selection to model performance

| Algorithm | F1 without Preprocess | F1 after feature selection and tuning | Change |
|---|---|---|---|
| **Decision Tree** | **62.23%** | **78.09%** | **15.86%** |
| Random Forest | 66.24% | 72.48% | 6.24% |
| Extra Tree | 64.96% | 72.48% | 7.52% |
| Naïve Bayes | 34.60% | 68.79% | 34.19% |
| Logistic Regression | 60.31% | 68.56% | 8.25% |
| Support Vector Machines | 68.44% | 73.50% | 8.34% |
| K-Nearest Neighbors | 64.71% | 71.81% | 7.10% |
| AdaBoost | 65.30% | 71.94% | 6.64% |
| Gradient Boosting | 65.34% | 72.28% | 6.94% |

| Algorithm | F1 without Preprocess | F1 after feature selection and tuning | Change |
|---|---|---|---|
| XGBoost | 65.57% | 74.99% | 9.42% |
| LightGBM | 70.63% | 74.99% | 4.36% |

According to the results, no model was found to be negatively affected by this step. Model performances improved between 4.36% and 34.19%. Although NB was the most improved model with 34.19%, it had the lowest classification ability. Excluding this case, the DT model has the highest classification skill, with a 15.86% increase in model performance to 78.09%.

### 5.2.3. Performance impact of different feature set

This section aims to test the important features generated by different classification algorithms with other algorithms to get better results. In the previous sections, feature sets obtained with different algorithms were explained. All sets and all models were tested against each other, and the sets that gave the best results were determined. The results are as shown in the Table 5.44.

**Table 5.44.** Best performed feature sets on algorithm

| Algorithm | First Set | F1 | Best Set | F1 | % |
|---|---|---|---|---|---|
| **Decision Tree** | **dt** | **78.09%** | **dt** | **78.09%** | **0.00%** |
| Random Forest | rf | 72.48% | ada | 74.39% | 1.91% |
| Extra Tree | et | 72.48% | ada | 74.97% | 2.49% |
| Naïve Bayes | dt | 68.79% | svm | 73.03% | 4.24% |
| Logistic Regression | lr | 68.56% | xgb | 70.85% | 2.29% |
| Support Vector Machines | svm | 73.50% | svm | 73.50% | 0.00% |
| K-Nearest Neighbors | dt | 71.81% | lr | 74.99% | 3.18% |
| **AdaBoost** | **ada** | **75.94%** | **xgb** | **79.90%** | **3.96%** |
| Gradient Boosting | gbc | 72.28% | dt | 75.81% | 3.53% |
| XGBoost | xgb | 74.99% | xgb | 74.99% | 0.00% |
| LightGBM | lgbm | 74.99% | lgbm | 74.99% | 0.00% |

According to the result, 7 out of 11 algorithms were positively affected by this step. The DT algorithm, with its own feature set, performed the best in the previous step. When the ADA algorithm was tested with the best features according to the XGB feature set, it was the most successful classification model among all variations with 79.90%. The first

two algorithms that gave the best results and the success they achieved because of the applied procedures are given in Figure 5.10.

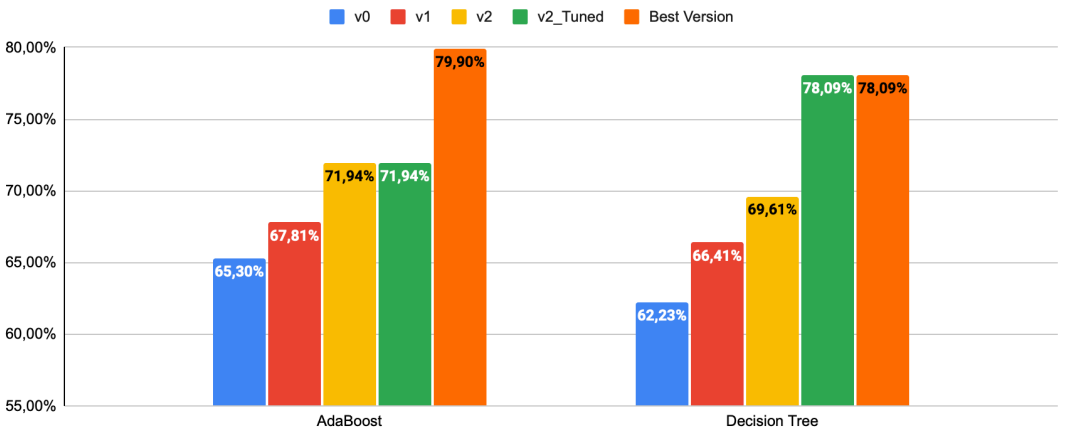


**Figure 5.10.** The best two algorithms and results

# 6. CONCLUSION AND FUTURE WORKS

This study attempts to detect level-based player churn with classification algorithms using in-game data and compares their performance. Eleven different classification algorithms, such as Decision Tree, Random Forest, KNN, and AdaBoost, were used to identify the levels of high player churn. Data preprocessing and feature selection steps were applied, and performances were also compared with nine different feature sets according to the results of different algorithms.

The results of the study show that the ADABoost model gives the highest F1-score of 79.90% when trained with the XGB feature set. The second-best model was Decision Tree, with an F1-score of 78.09%. When other classification models are compared, we see an average F1-score of 72%. The model with the lowest success rate was logistic regression, with an F1-score of 70.85%.

We also analyzed the impact of preprocessing, feature selection, and tuning steps on the models. When we look at the preprocessing process, we see that the performance of all models except the SVM and LGBM models increased by 1.26% to 6.45%. The Gradient Boosting algorithm showed the best performance improvement, achieving F1-score of 71.79% when only the preprocessing stage is considered.

When we look at the feature selection and tuning steps, we see improvements in all models compared to the previous step. The LIGHTGBM model had the lowest F1-score increase with 4.36%, and the NB model had the highest F1-score increase with 34.19%. Although NB is the model with the highest increase, it is one of the worst performing algorithms in this step.

When the models were tested with different feature sets, 7 out of 11 models showed performance improvement. Among the sets used, the xgb feature set produced with the XGBoost algorithm was found to be the best. The model with the highest performance improvement was NB. While AdaBoost achieved an F1-score of 75.94% when tested with its own set, it became the best performing model with an F1-score of 79.90% when tested with xgb feature set. According to these results, it is seen that algorithms can show different performances in detecting player loss. In addition, the data preprocessing, feature selection, and tuning steps played a very important role in improving the model's performance.

Many methods and approaches have been used to detect player churn. These studies

are reviewed in Section 1.1. All of the studies present a user-based approach. They have tried to find a solution to this problem by using user-based information. This information can include the user's personal information, such as demographic information, as well as in-game interaction. In addition, other studies have tried to find a solution to the question of which users we are losing rather than which levels are lost. Although this approach has advantages in detecting churn, obtaining user-based data may have some challenges and limitations under GDPR.

This paper presents a novel approach for targeted interventions using level-based data in player churn detection, which is a vital problem in the gaming industry. Rather than identifying which users are churned, we try to identify which levels are experiencing churn problems. Investigating the reasons behind player churn at specific levels can provide deeper insights and lead to improvements in game design. By analyzing level-based data, game developers can gain a better understanding of the specific challenges or frustrations players face at certain levels, allowing them to make targeted improvements and enhance the overall gameplay experience. Additionally, focusing on level-based churn detection enables developers to prioritize their efforts and allocate resources more effectively, as they can identify which levels require immediate attention.

### 6.1. Future Works

There are some limitations and challenges in this study. These limitations can be overcome with future studies and better prediction capabilities can be achieved in level-based player loss.

The size of the dataset is one of the challenges of this problem. Because thousands of levels are not very common in games. Therefore, a dataset with many records can be used to better understand this problem. This can improve the accuracy of the model. Also, a better analyzed feature set can help to focus on situations that have not been considered before. Further work can be done on feature extraction by considering the game in depth.

In addition, levels in games are interconnected, and the effects of the previous level can show up in subsequent levels. The result of a poorly designed level may appear later in an unpredictable way. Therefore, an approach that also examines the interconnectedness of levels can yield good results.

Machine learning algorithms are generally used in this study. These algorithms

belong to the group of supervised learning algorithms. In future studies, classification predictions can be made with unsupervised algorithms or deep learning models instead of these algorithms. Different algorithms may have the potential to provide better results.

As a result, a better dataset can be created using the expanded data and connections between levels. These new datasets can be tested with different models, such as Deep Learning to improve this study.

## REFERENCES


Abdel-Basset, M., Moustafa, N., Hawash, H., and Ding, W. (2021). Unsupervised deep learning for secure internet of things. Deep Learning Techniques for IoT Security and Privacy, 167-180.

Abirami, R. S. and Kumar, G. (2021). Comparative study based on analysis of coronavirus disease (covid-19) detection and prediction using machine learning models. SN Computer Science, 3(1).

Abram, K. J. and McCloskey, D. (2022). A comprehensive evaluation of metabolomics data preprocessing methods for deep learning. Metabolites, 12(3), 202.

Ahmed, M. U. and Begum, S. (2020). Artificial intelligence, machine learning and reasoning in health informatics—an overview. Intelligent Systems Reference Library, 171-192.

Al-Aidaroos, K. M., Bakar, A. A., and Othman, Z. (2010). Naïve bayes variants in classification learning. 2010 International Conference on Information Retrieval & Knowledge Management (CAMP).

Amin, M. F. (2022). Confusion matrix in binary classification problems: a step-by-step tutorial. Journal of Engineering Research, 6(5), 0-0.

Atallah, D. M., Badawy, M., & El-Sayed, A. (2019). Intelligent feature selection with modified k-nearest neighbor for kidney transplantation prediction. SN Applied Sciences, 1(10).

Breiman, L. (2001) Random Forests. Machine Learning (Vol. 45, Issue 1, pp. 5–32).

Bui, Q., Chou, T., Hoang, T., Fang, Y., Mu, C., Huang, P. H., et al. (2021). Gradient boosting machine and object-based cnn for land cover classification. Remote Sensing, 13(14), 2709.

Bzdok, D., Krzywinski, M., and Altman, N. (2018). Machine learning: supervised methods. Nature Methods, 15(1), 5-6.

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. Machine Learning, 20(3), 273-297.

Cunningham, P. and Delany, S. J. (2021). K-nearest neighbour classifiers - a tutorial. ACM Computing Surveys, 54(6), 1-25.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. The Annals of Statistics, 29(5), 1189–1232.

Fu, M., Zhang, C., Hu, C., Wu, T., Dong, J., & Zhu, L. (2023). Achieving verifiable decision tree prediction on hybrid blockchains. Entropy, 25(7), 1058.

Geurts, P., Ernst, D., and Wehenkel, L. (2006). Extremely randomized trees. Machine Learning, 63(1), 3-42.

Goutte, C. and Gaussier, É. (2005). A probabilistic interpretation of precision, recall and f-score, with implication for evaluation. Lecture Notes in Computer Science, 345-359.

Holliday, J. J. (2012). The emergence of L2 phonological contrast in perception: The case of Korean sibilant fricatives (Doctoral dissertation, The Ohio State University).

Hosmer, D.W. and Lemeshow, S. (2000) Introduction to the Logistic Regression Model, 2nd edn. New York: Wiley, pp. 1–30.

http-1: https://scikit-learn.org/stable/modules/generated/sklearn.dummy.DummyClassifier.html (Retrieved: 10.12.2023)

Jordan, M. I. and Mitchell, T. M. (2015). Machine learning: trends, perspectives, and prospects. Science, 349(6245), 255-260.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T. (2017). LightGBM: A highly efficient gradient boosting decision tree. Neural Information Processing Systems.

Kim, S., Choi, D., Lee, E., & Rhee, W. (2017). Churn prediction of mobile and online casual games using play log data. Plos One, 12(7), e0180735.

Kocak, B. (2022). key concepts, common pitfalls, and best practices in artificial intelligence and machine learning: focus on radiomics. Diagn Interv Radiol, 28(5), 450-462.

Kühn, M. and Johnson, K. (2013). Measuring performance in classification models. Applied Predictive Modeling, 247-273.

Lewis, D. (1998). Naive (bayes) at forty: the independence assumption in information retrieval. Machine Learning: ECML-98, 4-15.

Liu, M., Li, M., and Zhang, X. (2022). The application of the unsupervised migration method based on deep learning model in the marketing oriented allocation of high level accounting talents. Computational Intelligence and Neuroscience, 2022, 1-10.

Metuarau, T. (2017). A history of video games (2). Open Access Te Herenga Waka-Victoria University of Wellington.

Murthy, S. K. (1998). Data mining and knowledge discovery, 2(4), 345–389.

Oh, S., Park, J., Jeong, E., Kim, H., Bennis, M., and Kim, S. (2020). Mix2fld: downlink federated learning after uplink federated distillation with two-way mixup. IEEE Communications Letters, 24(10), 2211-2215.

Pahwa, A. (2018) Churn rate and its impact on telecom industry. International Journal of Research and Analytical Reviews, 5(3), 755-758.

Periáñez, Á., Alain, S., Guitart, A., & Magne, C. (2016). Churn prediction in mobile social games: towards a complete assessment using survival ensembles. 2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA).

Rácz, A., Bajusz, D., and Héberger, K. (2021). Effect of dataset size and train/test split ratios in qsar/qspr multiclass classification. Molecules, 26(4), 1111.

Río, A. F. d., Guitart, A., & Periáñez, Á. (2021). A time series approach to player churn and conversion in videogames. Intelligent Data Analysis, 25(1), 177-203.

Rothmeier, K., Pflanzl, N., Hüllmann, J. A., & Preuß, M. (2021). Prediction of player churn and disengagement based on user activity data of a freemium online strategy game. IEEE Transactions on Games, 13(1), 78-88.

Runge, J., Gao, P., Garcin, F., & Faltings, B. (2014). Churn prediction for high-value players in casual social games. 2014 IEEE Conference on Computational Intelligence and Games.

Ruuska, S., Hämäläinen, W., Kajava, S., Mughal, M., Matilainen, P., and Mononen, J. (2018). Evaluation of the confusion matrix method in the validation of an automated system for measuring feeding behaviour of cattle. Behavioural Processes, 148, 56-62.

Salcedo-Sanz, S., Pérez-Aracil, J., Ascenso, G., Del Ser, J., Casillas-Pérez, D., Kadow, etal. (2023). Analysis, characterization, prediction, and attribution of extreme atmospheric events with machine learning and deep learning techniques: a review. Theoretical and Applied Climatology.

Santafé, G., Inza, I., and Lozano, J. A. (2015). Dealing with the evaluation of supervised classification algorithms. Artificial Intelligence Review.

Schapire, R. E. (2013). Explaining adaboost. Empirical Inference, 37-52.

Seeger, M. (2017). Learning from labeled and unlabeled data. Encyclopedia of Machine Learning and Data Mining, 711-711.

Shah, S. M. A., Usman, S. M., Khalid, S., Rehman, I. U., Anwar, A., Hussain, S., et al. (2022). An ensemble model for consumer emotion prediction using eeg signals for neuromarketing applications. Sensors, 22(24), 9744.

Shi, M. and Zhang, B. (2011). Semi-supervised learning improves gene expression-based prediction of cancer recurrence. Bioinformatics, 27(21), 3017-3023.

Sifa, R. (2021). Predicting player churn with echo state networks. 2021 IEEE Conference on Games (CoG).

Su, J., Wen, Z., Jha, D. N., Li, Y., Guan, Y., Puthal, D., et al. (2019). Orchestrating the development lifecycle of machine learning-based iot applications: a taxonomy and survey.

Ting, K. M. (2017). Confusion matrix. Encyclopedia of Machine Learning and Data Mining, 260-260.

Video Game Market (2023). https://www.precedenceresearch.com/video-game-market

Vitola, J., Pozo, F., Burgos, D. A. T., and Anaya, M. (2017). Distributed piezoelectric sensor system for damage identification in structures subjected to temperature changes. Sensors, 17(6), 1252.

Winck, A. T., Machado, K., Souza, O. N. d., & Ruiz, D. D. (2013). Context-based preprocessing of molecular docking data. BMC Genomics, 14(S6).

Yang, W., Huang, T., Zeng, J., Chen, L., Mishra, S., & Liu, Y. (2022). Utilizing players' playtime records for churn prediction: mining playtime regularity.

IEEE Transactions on Games, 14(2), 153-160.

# ÖZGEÇMİŞ

| Kişisel Bilgiler | |
|---|---|
| Adı Soyadı | **Muhammed Selim TOKER** |
| E-Posta | |

| Eğitim | |
|---|---|
| Lisans | 2019, Yaşar Üniversitesi, Mühendislik Fakultesi, Yazılım Mühendisliği |
| Yüksek Lisans | 2021, Bakırçay Üniversitesi, Yüksek Lisans Enstitüsü, İş Zekası ve Veri Analitiği, Sosyal Bilimler A.B.D. |