CUKUROVA UNIVERSITY

INSTITUTE OF NATURAL AND APPLIED SCIENCES

PhD THESIS

Bitcoin Price Prediction with Machine Learning

İlkay Sibel KERVANCI

Computer Engineering Department

March, 2023

CUKUROVA UNIVERSITY

INSTITUTE OF NATURAL AND APPLIED SCIENCES

PhD THESIS APPROVAL

Bitcoin Price Prediction with Machine Learning

İlkay Sibel KERVANCI

Computer Engineering Department

This Doctorate Thesis was evaluated by the following Jury Members on 29/03/2023 and was approved by unanimity of votes.

Jury : Prof. Dr. Mehmet Fatih AKAY(Advisor) ……

: Prof. Dr. İbrahim Taner OKUMUŞ ……

: Dr.Öğr. Üyesi Serkan KARTAL ……

: Doç. Dr. M. Uğraş CUMA ……

: Doç. Dr. Fatih KILIÇ ……

This Thesis was written in the Department of Computer Engineering, Institute of Natural and Applied Sciences.

Thesis Number:

Prof. Dr. Sadık DİNÇER
Director
Institute of Natural and Applied Sciences

This work was supported by the Çukurova University BAP
Project ID: FDK-2019-12360

Note: The usage of the presented specific declarations, tables, figures, and photographs either in this thesis or in any other reference without citation is subject to "The law of Arts and Intellectual Products" number of 5846 of Turkish Republic

CONTENTS

ABSTRACT I
ÖZ II
GENİŞLETİLMİŞ ÖZET III
ACKNOWLEDGEMENTS VIII
LIST OF TABLES IX
LIST OF FIGURES X
SYMBOLS AND ABBREVIATIONS XI
1. INTRODUCTION 1
2. PRELIMINARY WORK 5
2.1. Causality 5
2.2. Time Interval 7
2.3. Hyper-parameter Optimization 7
2.4. Methods 8
3. MATERIAL AND METHOD 11
3.1. Material 11
3.1.1. Datasets 11
3.1.2. Performance Metrics 13
3.1.3. Min-Max Normalization 13
3.1.4. The Structure of Bitcoin 14
3.2. Methods 14
3.2.1. Multilayer Perceptron (MLP) 15
3.2.2. Support Vector Machine (SVM) 16
3.2.3. Generalised Regression Neural Network (GRNN) 17
3.2.4. Recurrent Neural Networks (RNN) 18
3.2.5. Long Short-Term Memory (LSTM) 20
3.2.6. Gated Recurrent Unit (GRU) 21
3.2.7. Convolutional Neural Network (CNN) 22
3.2.8. Hyper-parameter Optimization Methods 24
4. RESULTS AND DISCUSSIONS 29
4.1. MLP 30
4.2. SVM 32
4.3. GRNN 34
4.4. RNN 35
4.5. LSTM 38
4.6. GRU 41

4.7. CNN 45
4.8. The Effect of Parameters on Methods 47
4.8.1 Parameter of RNN-LSTM-GRU 47
4.9. Optimization Results 54
4.9.1. Hparam Parameters 54
4.9.2. Grid Search 56
4.9.3. Random Search 57
4.9.4. Bayesian Optimization 59
4.10. DISCUSSIONS 61
5. CONCLUSIONS 65
REFERENCES 67
CURRICULUM VITAE 77

CUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES

PhD THESIS

Bitcoin Price Prediction with Machine Learning

İlkay Sibel KERVANCI

Advisor: Prof. Dr. Mehmet Fatih AKAY

Department of Computer Engineering

ABSTRACT

The most well-known cryptocurrency is Bitcoin, as it was the first cryptocurrency. The approximate value of the cryptocurrency market capitalization, of which 39% is Bitcoin, is 807 billion dollars (December 2022). It has become an important research topic due to the difficulty in predicting the price of Bitcoin due to the extreme volatility of the price, and there are many studies in the literature. For investors, this high volatility means high profits and risks. This thesis aims estimation results close to the actual price and reduces the risks for the investors, using machine learning methods, different data sets, and optimization techniques. The methods used: Multilayer Perceptron (MLP), Support Vector Machines (SVM), Generalized Regression Neural Network (GRNN), Recurrent Neural Network (RNN), Long-Short-Term Memory (LSTM), Gated Repetitive Unit (GRU), and Convolutional Neural Network (CNN). The thesis includes experiments with each machine learning model with the combinations of Bitcoin, gold, crude oil, natural gas, Ethereum, and dollar-euro parity. Hyper-parameter optimization methods such as Bayesian optimization (BO), random search, grid search, and Hparam parameters are examined. Models with BO achieved better results than others. This thesis proposes a new model for Bitcoin price prediction that effectively reduces prediction error. This new BO model with Gradient Incremental Regression Trees (GBRT), Gaussian Process (GP), Random Forest (RF), and Extra Trees (ET) was applied to optimizers and corresponding surrogate functions. In addition, to increase the comparability of the results with the other paper, it was evaluated with four different performance metrics: root square mean error (RMSE), mean square error (MSE), mean absolute error (MAE), and mean absolute percentage error (MAPE). In general, among the seven algorithms, predictions using only the closing price of Bitcoin yielded better results. In addition, we obtained very close results in the estimates made by adding Ethereum, crude oil, and natural gas to the data set. Better results were obtained with LSTM, CNN, and GRU, respectively, than with the other methods. The experimental optimization results indicated that hparam, grid search, and random search achieved the worst results in all four error metrics. BO-GP with hybrid LSTM-GRU outperformed all methods in this thesis and the examined literature for the value of MAE=0.002302, MAPE=0.005497, MSE=0.000015, and RMSE=0.003269.

Keywords: Bitcoin, Price Prediction, Machine Learning, Deep Learning.

ÇUKUROVA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

DOKTORA TEZİ

Makine Öğrenmesi ile Bitcoin Fiyat Tahmini

İlkay Sibel KERVANCI

Danışman: Prof. Dr. Mehmet Fatih AKAY

Bilgisayar Mühendisliği Anabilim Dalı

ÖZ

Bitcoin, ilk kripto para olması sebebi ile dünya çapında en çok bilinen kripto para birimidir. %39'nu Bitcoin'nin oluşturduğu kripto para piyasasının yaklaşık değeri 807 milyar dolardır (Aralık 2022). Bitcoin fiyatının aşırı oynaklığından dolayı fiyatını tahmin etmenin zorluğunun sonucu olarak, önemli bir araştırma konusu haline gelmiş ve literatür de bu konuda birçok çalışma bulunmaktadır. Yatırımcılar için ise bu yüksek oynaklık yüksek karlar ve riskler demektir. Bu tez de, makine öğrenmesi yöntemleri, farklı veri setleri, ve optimizasyon teknikleri kullanılarak gerçek fiyata yakın tahmin sonuçları elde etme ve yatırımcılar için riskleri azaltmak hedeflenmiştir. Kullanılan yöntemler; Çok Katmanlı Algılayıcı (MLP), Destek Vektör Makineleri (SVM), Genelleştirilmiş Regresyon Sinir Ağı (GRNN), Tekrarlayan Sinir Ağı (RNN), Uzun-Kısa Süreli Bellek (LSTM), Geçitli Tekrarlayan Birim (GRU) ve Konvolüsyonel Sinir Ağı (CNN). Altın, ham petrol, doğal gaz, Ethereum ve dolar-euro parite veri setlerinin Bitcoin fiyatı ile kombinasyonlarının makine öğrenmesi modellerinin her biri ile yapılan deneyleri içermektedir. Bayes optimizasyonu (BO), rastgele arama, ızgara arama ve Hparam parametresi gibi hiper parametre optimizasyon yöntemleri incelenmiştir. BO diğer optimizasyon yöntemlerinden daha iyi sonuçlar elde etmiştir. Bu tezde, Bitcoin fiyat tahmini hatasını etkili bir şekilde azaltan yeni bir model önerilmektedir. BO'lu bu yeni model; Optimize edicilere ve karşılık gelen vekil işlevlere Gradyan Artımlı Regresyon Ağaçları (GBRT), Gauss Süreci (GP), Rastgele Orman (RF) ve Ekstra Ağaçlar (ET) uygulanması ile elde edilir. Ayrıca sonuçların karşılaştırıla bilirliğini arttırmak için; karesel ortalamalarının karekökü hatası (RMSE), ortalama karesel hata (MSE), ortalama mutlak hata (MAE) ve ortalama mutlak yüzde hatası (MAPE) olmak üzere dört farklı performans metriği ile değerlendirilmiştir. Genel olarak, makine öğrenmesi algoritmalarının Bitcoin'in kapanış fiyatını kullanan tahminleri daha iyi sonuçlar vermiştir. Ayrıca veri setine Ethereum, ham petrol ve doğal gazı eklenerek yapılan tahminler sırasıyla daha iyi sonuçlar elde etmiştir. Sırasıyla LSTM, CNN ve GRU ile diğer yöntemlere göre daha iyi sonuçlar elde edilmiştir. Optimizasyonun deneysel sonuçları; hparam, ızgara arama ve rastgele arama dört hata metriği içinde kötü sonuçlar elde etmiştir. BO-GP hibrit LSTM-GRU ile MAE=0.002302, MAPE=0.005497, MSE=0.000015 ve RMSE=0.003269 ile incelediğimiz kadarı ile literatürdeki en iyi sonuçları elde etmiştir.

Anahtar Kelimeler: Bitcoin, Fiyat tahmini, Makine öğrenmesi, Derin Öğrenme.

GENİŞLETİLMİŞ ÖZET

Kripto para borsasının yaklaşık %39'unu (Aralık 2022) oluşturan Bitcoin, temeli 2008 yılında 'uçtan uça elektronik nakit sistemi' başlıklı makale ile duyuruldu. Bu sistem elektronik bir paranın anlık olarak uçlar arasında herhangi bir finansal kurum olmadan değiş tokuşuna dayanmaktadır. Kriptografik kanıta dayalı sistem sayesinde değiş tokuş yapmaya istekli iki uçun birbirleriyle doğrudan işlem yapmasına izin veren bir elektronik ödeme sistemidir. Bu şekilde arabulucu maliyeti minimuma düşecektir. Bu sistem gücünü yapılan işlemlerin sırasının hesaplamalı kanıtının bir zaman damgası ile damgalanması ve yapılan alım-satım işlemleri için yüksek hesaplamalar sebebi ile tekrar hesaplamanın pratik olmadığı, dürüst düğümlerin CPU gücünün işbirliği yapmış dolandırıcı düğümlerin gücünden fazla olduğu sürece sistem güvenli bir şekilde çalışacaktır. Her bir elektronik para dijital bir imza zincirindeki bir kayıtta tutulmaktadır. Sistemin başında merkezi bir otorite bulunmadığından yapılan her işlemin tüm ağa duyurulması ve ağın çifte harcamayı önlemek için her bir işlemin tek bir geçmişi üzerinde fikir birliği olması gerekir. Sistem bir zaman damgası sunucusu, zaman damgası eklenecek bir öğe bloğunun karmasını ve karmaya girebilmesi için verilerin o sırada var olması gerektiğini kanıtlar. Bir önceki zaman damgasının karmasını içeren blok ile bir sonraki blok güçlendirilmiş olur. Böylece eklenen her blok ile bloklar zinciri güçlendirilmiş olur. Kripto paraların temelini oluşturan art arda bağlanan sistemdeki uçlar arasında paylaşılan ve üzerinde fikir birliğine varılan dağıtılmış bir veri tabanıdır. Bu dağıtılmış veri tabanı Blockchain adı ile adlandırılmaktadır. Bitcoin' in açık kaynak kodlu yapısı sayesinde Bitcoin den türemiş birçok kripto paranın ortaya çıkmasına da olanak sağlamıştır.

2009 yılında ilk Bitcoin madenciliği yapılmıştır ve 2013 yılında artan bir ilgi ile bir Bitcoin 22 $ seviyelerine yükselmiştir. 2017 yılı sonunda 19.000 $ yükselen değeri 2020 sonu ve 2021 başında 60.000 $ a kadar yükselmiş, aynı yıl Haziran ayında 23.000 $ seviyelerine gerilemişken, 2021 Kasım ayında 68.000 dolar a tekrar yükselmiştir. 2021 sonundan 2022 sonlarına kadar düşüş eğilimindeki dalgalanmalar devam etmiştir. Bitcoin piyasa değerindeki bu dalgalanmalar sebebiyle yüksek karlar ve kayıplara sebep olduğundan fiyatının tahmin edilmesi önemli bir araştırma konusu olmuştur. Bitcoin fiyat tahmini için istatistiksel yöntemler, makine öğrenimi (ML) ve derin öğrenme (DL) algoritmalarının kullanan birçok çalışma vardır. Derin öğrenme birçok alanda umut verici gelişmeler sağlamıştır. Örneğin ses tanıma, nesne tanıma, sağlık, eğitim, finansal tahminler vb. Ayrıca makina öğrenmesinin uygulandığı alanlarda çok büyük miktarda veri seti, ML ve DL algoritmaları, algoritmaların parametreleri ve parametrelerin optimizasyonunun tahminlerin doğruluğunu önemli ölçüde artırabildiği gözlemlenmiştir. Ayrıca incelenen veri seti büyüklüğü ve değerlerin gürültü seviyesi, boşluklar, veri setinden yapılan çıkarımları etkiler. Bu sebepten veri ön işleme önemli ve gerekli bir makina öğrenmesi adımıdır. İncelenen veri setinden sonuç çıkarmak için, makine öğrenmesi modellerinin parametrelerinin girdi değerlerine göre ayarlanması gerekir. Makine öğrenimi algoritmalarının eğitim sürecini ve topolojisini etkileyen bazı faktörler vardır.

Model parametreleri ve algoritma hiper-parametreleri olarak gruplandırılırlar. Bu tezde bahsi geçen parametreler algoritma hiper-parametreleridir. Makine öğrenimi algoritmaları ve özellikle derin öğrenme gibi karmaşık algoritmalar, hiper-parametrelerinin farklı örnekleriyle çok farklı sonuçlar üretir. Bunlar çalıştıkları problemlere özgü değildirler ve bu nedenle yöntemlerinin adaptasyona ihtiyacı vardır. Ayrıca kullanılan optimizasyon teknikleri de önemlidir. Çünkü yerel optimum değerler yerine global optimum değerlerin bulunması amaçlanmaktadır. Basitce hiper-parametre optimizasyon teknikleri manuel ve otomatik olarak ikiye ayrılır.

Manuel yaklaşımlar: Bu yöntemde hiper-parametrelerin her biri için en uygun hiper-parametre değerlerini elde etmek farklı değer kombinasyonlarıyla tekrar tekrar denemeler sonucunda elde edilir. Bunun için bir uzmanlığa ihtiyaç vardır. Ayrıca çok zaman alıcı zahmetli bir süreçtir.

Otomatik yaklaşımlar: Birçok hiper-parametre optimizasyon algoritması vardır. Grid Search, Random Search, Particle Swarm Optimization (PSO), ve Bayesian Optimization (BO) otomatik hiper-parametre Optimizasyon yöntemleridir. Yüksek hesaplama maliyetleri nedeniyle otomatik yaklaşımların uygulanması zor olsa da, ancak artık, bilgisayar kümeleri veya grafik işlemci birimi (GPU) gibi donanımlar sayesinde geçmişten daha ucuza kullanmak mümkün olmuştur.

Izgara Arama, Hparam parameters ve Rastgele Arama, hiper-parametre ayarları için manuel olarak girilen bilgilere ihtiyaç duyarken, BO ilk yapılandırma dışında herhangi bir manuel girişe ihtiyaç duymaz. BO, geçmişte başarılı olan parametre değerlerini dikkate alarak bir sonraki parametreleri ayarlar ve bu, istenen sonuçlar elde edilene kadar tekrarlanır. Izgara Arama ve Hparam, uygulama açısından basittir, ancak hangi parametre değerlerinin seçileceği konusunda deneyim gereklidir. Ayrıca parametre sayısı arttıkça etkinliğin oldukça düştüğü ve işlem süresinin oldukça arttığı gözlemlenmiştir. Tensorboard, makine öğrenimi algoritmalarının sonuçlarının ve operasyonların iç işleyişindeki parametrelerdeki değişikliklerin görselleştirilmesini ve ölçülmesini sağlayan bir araçtır. Bu tezde Random Search, Hparam parameter ve Grid Search optimizasyon yaklaşımlarında hiper-parametrelerin görselleştirilmesi için Tensorboard kullanılırken BO (Tensorboard BO desteği yok) için kullanılamamaktadır. Modelinizle ilgili sorunları hızlı bir şekilde belirlemenin en etkin yolu, modelinizde neler olup bittiğini gerçek zamanlı olarak grafiksel olarak görselleştirmektir. İşte bu sebepten Tensorboard'un katkısı yadsınamaz. Birçok tahmin değişkeni olduğunda BO gibi "kara kutu" tahmin yöntemiyle üretilen modelleri yorumlarken duyarlılık analizini kullanmak oldukça faydalıdır. En iyi sonucu elde ettiğimiz BO-GP Hibrit LSTM-GRU için duyarlılık analizi yapılarak görselleştirilmiş ve parametreler arası ilişki anlamlandırılmıştır. Hiper-parametrelerin alaka düzeyini anlamak için derin öğrenme yöntemleri RNN, LSTM ve GRU hiper-parametre optimizasyon yöntemlerinden Grid search, ve Random search ile uygulanmıştır. Bayesian Optimization; LSTM, GRU, ve hibrid LSTM-GRU ile uygulanmıştır. Modellerin hiper parametrelerine ek olarak, modeller üzerindeki BO iç

parametrelerinin optimize edici ve vekil işlevi araştırıldı. BO'nun optimize edicileri ve karşılık gelen vekil işlevler GBRT, GP, RF ve ET uygulandı. Boru hattı ile farklı öğrenme yöntemlerinden LSTM, GRU, ve hibrit LSTM-GRU için BO vekil işlevlerinin verimliliği araştırıldı. Hibrit LSTM-GRU BO-GP ile en iyi sonucu elde etti.

Bu yöntemlerinin kullandığı veri setileri aynı şekilde veri ön işlemeye ve normalizasyona tabi tutulmuştur. Ardından zaman serisi problemine çevrilerek her bir yöntem RNN, LSTM ve GRU için; Grid search ve Random search optimizasyon yöntemleri uygulanmıştır.

Bu tezin konusu olan MLP, SVM, GRNN, RNN, LSTM, GRU ve CNN algoritmalarının öncelikle yapıları 3.2. Methods başlığında açıklanmış ve ardından Bitcoin fiyat tahmini problemi için kullanılmıştır. Bitcoin kapanış fiyatı, altın, Ethereum, ham petrol, doğal gaz, ve dolar-euro paritesi fiyatlarının Bitcoin fiyat tahmine katkılarının olumlu ya da olumsuz etkilerini anlamak için her bir algoritmaya bu veri setleri uygulanmıştır. Her bir algoritmaya için ilk olarak Bitcoin kapanış fiyatı algoritmaların farklı parametreleri ile denenmiş daha sonra aynı parametrelere Bitcoin kapanış fiyatına sırası ile altın, Ethereum, ham petrol, doğal gaz, ve dolar-euro parity fiyatlarının her biri eklenerek algoritmalar çalıştırılmıştır. Elde edilen sonuçlar MSE, ve MAPE için tablolara yerleştirilmiş ve en iyi iki sonuç için veri seti üçlü olarak kullanılmıştır. Elde edilen sonuçlar ve veri setlerinin tamamı birleştirilerek elde edilen sonuçlarla birlikte her bir algoritma için toplam sekiz adet deney yapılmış ve deneylerin sonuçları tablolara yerleştirilmiştir. Ayrıca her bir algoritma için farklı sayıda parametreleri için yapılan deneylerin sonuçları da tablolarda yer almaktadır. MLP ; activation function ve solver, SVM; kernel, LSTM; neuron, activation function, optimizer, GRU; neuron, activation function, optimizer, ve CNN; filter size, activation function, optimizer gibi algoritma parametreleri için elde edilen parametrik sonuçlar veri setleri sonuçları ile birlikte tablolaştırılmıştır.

Bir tahmin yönteminin performansı değerlendirilirken, modelin iç parametrelerinin öğrendiği veri seti ile değil de yöntemin daha önce görmediği verilerle test edilmesi gerekir. Aksi halde çok yüksek doğruluklar elde edilirken buna aşırı uyum denir ve yeni bir veri ile karşılaştığında model düşük performans gösterir. Bu sebepten makina öğrenmesi algoritmaları kullanılırken öğrenme ve test veri setine ayırma yöntemi kullanılır. Bu tezdeki bütün deneylerde veri seti öğrenme ve test olmak üzere ikiye ayrılmıştır. Veri setini ayırma işlemleri için sklearn kütüphanesinden model_selection özelliğinin train_test_split kullanılmıştır. Veri seti ayrılma oranları 0.8 eğitim verisi ve 0.2 test verisi olarak ayrılırken zaman serisinde verilerin sıralaması önemli olduğundan shuffle=False olmasına dikkat edilmiştir.

Ayrıca hiper-parametreler ayarlanırken parametrelerin farklı olasılıkları için deneyler yapılırken sızmalar olabilir. Bu sızmaların etkisinden kurtulmak için veri setini eğitim, test, doğrula diye üç gruba ayırmak bir yöntem olabileceği gibi onun yerine çapraz doğrulama da kullanılabilir.

Bu tezde veri seti eğitim, test olarak ayrılmış ve çapraz doğrula özelliğine sahip tüm deneylerde çapraz doğrulama kullanılmış olup cv=3 olarak kullanılmıştır.

Bölüm 1 de ‘Introduction’ başlığı altında Bitcoin ve makine öğrenme yöntemlerinden kısaca bahsedilmiş ve tezin konusuna giriş yapılmıştır.

Bölüm 2 de ‘Preliminary Work’ başlığı altında Bitcoin fiyat tahmini hakkında olan makaleler 4 ana başlık altında incelenmiştir. ‘Causality’ başlığında Bitcoin fiyatı ile nedensellik ilişkilerinin incelendiği çalışmalar yer almakta ve bu makaleler Tablo 1 özetlenmiştir. ‘Time interval’ başlığında Bitcoin fiyat tahmini ile ilgili çalışmalar kullandıkları veri setileri zaman aralığı açısından incelenmiştir. ’Hyper-parameter optimization’ başlığında Bitcoin fiyat tahmininde optimizasyon yöntemleri kullanmış olan çalışmalar incelenmiş ve Tablo 2 de özetlenmiştir. ’Methods’ başlığı altında daha önceki çalışmalar kullanılan yöntemler açısından incelenmiş ve Tablo 3 de özetlenmiştir.

‘MATERIAL AND METHODS’ başlığı ‘Material’ ve ‘Methods’ olarak iki başlıkta incelenmiştir. ’Material’ başlığının altında ‘Datasets’ tezde kullanılmış olan Bitcoin, Ethereum, altın, Dolar Euro değişim oranı, ham petrol ve doğal gaz için 7 Ağustos 2015 den 13 Aralık 2022 tarihleri aralığındaki fiyatları içeren verilerle hazırlanmıştır. Tezde kullanılmış olan yöntemlerin performanslarını değerlendirmek için kullanılan metrikler MSE, RMSE, MAE, ve MAPE olup ‘Performance metrics’ başlığı altında matematiksel formülasyonları yer almaktadır. ‘Min-Max normalization’ başlığı altında normalizasyon tekniğinin formülü yer almaktadır. Bu tezde kullanılan her bir veri seti ML yöntemleri ile kullanılmadan önce normalize edilmiştir. Normalizasyon yöntemlerinden ‘Min-Max normalization’ tekniğinin kullanılmasının sebebi Bitcoin fiyatının çok büyük değerler almasıdır. Ayrıca veri setleri arasındaki sayısal farklardan oluşabilecek hataları en aza indirerek aralarındaki artış azalış trenlerini yakalamaya olanak sağlamaktadır. Böylece tüm veri setlerinin değerleri [0,1] aralığına indirgenmiştir.

’Methods’ başlığı altında bu tezde kullanılan 7 adet ML algoritması MLP, SVM, GRNN, RNN, LSTM, GRU ve CNN ayrı ayrı incelenmiş ve yapıları açıklanmıştır. Ayrıca bu tezde, hiper-parametrelerin öğrenme üzerindeki etkisini anlamak için geri gitme, dönem, parti boyutu, bırakma, aktivasyon fonksiyonu, optimize edici, öğrenme hızı ve katman sayısı parametreleri manuel olarak 'Parameter of RNN-LSTM-GRU' başlığı altında her biri için alt başlıklarda değerlendirilmiştir.

Ayrıca, ML uygulandığı alanlarda veri seti, model parametresi sayısı ve parametrelerin optimizasyonu tahminlerin doğruluğunu önemli ölçüde arttırdığı için ‘Hyper-parameter optimization’ başlığı altında 4 yöntem için ‘Hparam parameters’, ‘Grid Search’, ‘Random Search’ ve ‘Bayesian optimization’ yöntemleri; LSTM ve GRU ve hibrit LSTM-GRU algoritmalarına uygulanmış ve sonuçlar her bir başlık altında önce yöntem kısaca açıklanmış ve ardından sonuçlar eklenmiştir. BO için LSTM ve GRU ve hibrit LSTM-GRU modelleri için optimizer ve vekil işlevler optimize edilmiş ve sonuçlar tablolaştırılmıştır.

Veri setleri ile deneyler yapılırken Bitcoin fiyatı ilk tekbaşına tahmin için kullanılmış ve daha sonrasında her bir veri seti ile ‘csv’ formatında birleştirilerek zaman serisine çevrilmek sureti ile ertesi gün fiyatları tahmin edilmiştir. Tahminler yapılırken her yöntem için veri setleri tek tek

incelenmiş ardından çıkan en iyi iki sonuç ile birlikte yapılan tahmin sonucu ve tüm veri setlerinin yer aldığı tahminler tablolaştırılmıştır. Böylece her bir yöntem için toplam sekiz deney yapılmış ve sonuçlar 'RESULTS AND DISCUSSIONS' başlığı altında incelenmiştir. Sonuçlar 'Results' altında yer alırken 'Discussion' da result başlığındaki sonuçların literatür ile karşılaştırılmasına yer verilmiştir. Son olarak 'CONCLUSIONS' başlığı altında tezdeki her bir başlık altında yer alan sonuçların değerlendirilmesinden ulaşılan genel sonuç ve önerilere yer verilmiştir.

ACKNOWLEDGEMENTS

Thank my supervisor Prof. Dr. M. Fatih AKAY, Prof. Dr. I. Taner OKUMUŞ, and Dr. öğretim üyesi Serkan KARTAL, for providing guidance and feedback throughout this project. My mother, husband, and children for their patience and encouragement.

I want to dedicate this thesis to my beloved father's soul with all my heart.

LIST OF TABLES

Table 2.1. Causality relation with the Bitcoin price prediction papers is as follows 6
Table 2.2. Hyper-parameter optimization for the Bitcoin price prediction papers is as follows 8
Table 2.3. ML and conventional statistical methods papers.....10
Table 3.1. Daily datasets to use for Bitcoin price prediction 11
Table 3.2. Combination of datasets and models for Bitcoin price prediction 15
Table 4.1. Results of MLP experiments 30
Table 4.2. Results of SVR experiments 32
Table 4.3. Results of GRNN experiments 34
Table 4.4. Results of RNN experiments 36
Table 4.5. Results of LSTM experiments.....39
Table 4.6. Results of GRU experiments 42
Table 4.7. Results of CNN experiments 45
Table 4.8. Comparing the activation function with Models 48
Table 4.9. The effects of the go-backward on RNN, LSTM, and GRU (neuron=128, batch size=128, epoch=100, and Lr=0.01).....49
Table 4.10. Optimizer and activation function results 49
Table 4.11. The effect of the learning rate (Activation=tanh, batch size=128, neuron=128) 51
Table 4.12. The effect of the batch size on performance 52
Table 4.13. The effect of the dropout on the performance test set 53
Table 4.14. Hyper-parameters and their values.....55
Table 4.15. The results of the hourly dataset with hparam parameters (epoch=100).....55
Table 4.16. The results of the daily dataset with hparam parameters (epoch=100) 56
Table 4.17. Grid search best parameter for each model 56
Table 4.18. Random search best parameter for each model with univariate time series.....57
Table 4.19. The results of different models with Random search 58
Table 4.20. The results of different models with their best parameters for a random search.....58
Table 4.21. BO with LSTM and GRU layers 59
Table 4.22. The results of MSE with their best parameters for BO. 60
Table 4.23. BO with LSTM, GRU, and hybrids 60
Table 4.24. In terms of ML algorithms, and data sets 61
Table 4.25. The results of the literature review comparison of the articles MSE, RMSE, MAE, and MAPE values with our best results.....62

LIST OF FIGURES

Figure 3.1. Historical price 12
Figure 3.2. Multilayer Perceptron 15
Figure 3.3. SVR 17
Figure 3.4. GRNN architecture 18
Figure 3.5. RNN architecture 19
Figure 3.6. LSTM 20
Figure 3.7. GRU (Gated Recurrent Unit) 22
Figure 3.8. Multiple-channel CNN 23
Figure 4.1. Visualized from the results of Table 4.1 (MSE) 31
Figure 4.2. Visualized from the results of Table 4.2 (MSE) 33
Figure 4.3. All feature space, the relatedness matrix 34
Figure 4.4. GRNN predicted-actual price, prediction error, residuals 35
Figure 4.5. RNN best result of (MSE) each dataset in Table 4.4 38
Figure 4.6. LSTM best result of (MSE) each dataset in Table 4.5 41
Figure 4.7. GRU best result of (MSE) each dataset in Table 4.6 44
Figure 4.8. CNN best result of (MSE) each dataset in Table 4.7 47
Figure 4.9. Loss of different activation functions 48
Figure 4.10. Tensorboard parallel coordinate view LSTM in Table 4.10 50
Figure 4.11. Conventional neural network and Neural network with dropout 53
Figure 4.12. Dropout effect on RNN, LSTM, and GRU 54
Figure 4.13. Grid search Table 4.16 result in Tensorboard MSE, MAE, MAPE for the test set (blue: GRU, dark blue: LSTM, green: RNN) 57
Figure 4.14. Random search MSE values 58
Figure 4.15. Random search, MSE results with layers (8000 and 100 time step) 59
Figure 4.16. Bayesian optimization LSTM and GRU MSE results with layers (8000 and 100 time step) 60
Figure 4.17. Bayesian optimization LSTM, GRU, and hybrid MSE results (8000 and 100 time step) 61
Figure 4.18. Prediction of the next 30 days with hybrid LSTM-GRU loss 63
Figure 4.19. Sensitivity analysis of the hyperparameters, LSTM-GRU optimized with BO-GP 63

SYMBOLS AND ABBREVIATIONS

Act	: Activation Function
ANN	: Artificial Neural Network
API	: Application Programming Interface
ARDL	: Autoregressive-Distributed Lag
ARMA	: Autoregressive Moving Average
CNN	: Convolutional Neural Network
CV	: Cross Validation
DL	: Deep Learning
ET	: Extra Trees
FFNN	: Feed Forward Neural Network
GARCH	: Generalized AutoRegressive Conditional Heteroskedasticity
GBDT	: Gradient Boosted Trees
GBRT	: Gradient Incremental Regression Trees
GP	: Gaussian Process
GRNN	: Generalized Regression Neural Networks
GRU	: Gated Recurrent Unit
HR	: Huber Regression
KNN	: K-Nearest Neighbors
LSTM	: Long-Short-Term Memory
Lr	: Learning rate
LR	: Linear Regression
MAE	: Mean Absolute Error
MAPE	: Mean Absolute Percentage Error
ML	: Machine Learning
MLP	: Multilayer Perceptron
MLP-based NARX	: MLP Based Nonlinear Auto-Regressive Network
MSE	: Mean Squared Error
NARDL	: Nonlinear Autoregressive-Distributed Lag
Opt	: Optimizer
PSO	: The Particle Swarm Optimization
R2	: R-Square
RBF	: Radial Basis Function
RF	: Random Forest
RMSE	: Root Mean Squared Error
RNN	: Recurrent Neural Network

SGD : Stochastic Gradient Descent

SMA : Simple Moving Average

SVM : Support Vector Machine

SVR : Support Vector Regression

XGBOOST : Extreme Gradient Boosting

1. INTRODUCTION

Cryptocurrencies are a new investment vehicle in global finance. Its identity, structure, and function constantly evolve thanks to its great potential and applicability to different fields. Cryptocurrencies are growing rapidly and are increasingly seen as a recognized investment vehicle (Abboushi, 2017). Bitcoin is a decentralized digital currency system (Rahouti et al., 2018). There is no central authority in the Bitcoin system to issue new money or verify transfers, as in fiat money. Cryptocurrency has been developed to avoid using reliable third parties such as banks, cards, and governments and to avoid time delays and money redirection costs (Juarez and Manzanilla, 2019). The virtual currency system was announced by (Satoshi, 2008), and in 2009 Bitcoin was applied as the core component of Bitcoin. The first cryptocurrency is distinguished from the others because it is well-known, and many cryptocurrencies are clones of it (Rauchs and Garrick, 2017). In the Bitcoin network, which includes miners, all transactions are carried out through miners in the network. Bitcoin is an open-source online system that conducts on different operating systems and devices. It can be downloaded from https://bitcoin.org or any other website to join the Bitcoin network. The entire history of transactions, is called a Blockchain ledger and downloaded for free. Users have anonymity; while transactions can be viewed, no information about whom they belong to can be seen. Unlike traditional methods, banks, and official currencies, the Bitcoin system is designed so that there is no central authority. In this structure, which consists of miners without a central authority, transaction data are kept in Blockchain. It also provides anonymity by allowing users to transact financial transactions without revealing personal information (Jianfu, 2014). These systems provide users with data security, privacy, and solutions for double spending problems using an access control mechanism via Blockchain and decentralized network. Blockchain is a ledger that records and verifies transactions on the Bitcoin network. The Blockchain ledger is open-source and not owned by any other protocol or governing authority. All computing devices participating in the protocol can access this shared public ledger and store transaction logs. Cryptographic blocks are used to verify the transaction. All blocks can be monitored, added, and verified. Blockchain ensures that the integrity of the Bitcoin system is maintained. Since each block consisting of many transaction records is associated with the hash value and the previous block information, the user can see the transaction details, ensuring that the written transaction has not been changed. Only valid transactions are stored in memory and are adjacent to the Blockchain (Prachi and Sudhir, 2019). Since the Bitcoin network is 24/7 regarding working hours and days, transactions and price changes can be seen instantly. The state of this price change timeframe allows for using short or long timeframes as desired, making it possible to obtain many different datasets in varying timeframes. In addition to the time variability, Bitcoin is an environment where everyone can see the transfers. Due to the extreme volatility of the Bitcoin price compared to other investment instruments, investors have increased their interest in price prediction with the

possibility of making high returns. For this reason, it has attracted many researchers' attention and articles about price prediction with machine learning. Machine learning looks for patterns in the dataset and tries to conclude. Many features such as daily opening, closing, highest, lowest price, and volume in the Blockchain structure can be added to the data set, and causality can be searched for price prediction. For most of the Bitcoin price prediction problems, was used, the closing price (Tandon et al., 2019; Hamayel and Owda, 2021; Drahokoupil, 2022; Aras, 2021) and some multivariate datasets (Cavalli and Amoretti, 2021; Rajabia et al., 2022). In general, hourly (Gradojevic et al., 2023) and daily (Gong and Huser, 2022; Felizardo et al., 2019) datasets are used in forecasting problems, while more sensitive minute (Thearasak and Thanisa, 2018; Aggarwal et al., 2019), and second datasets are used in real-time forecasting.

While examining the studies that make Bitcoin price estimation, it was necessary to look at the differences in methods, the datasets, optimization, and the effect of causality relationships on the results. From the comprehensive review article on Bitcoin price prediction, they concluded that machine learning (ML) and deep learning (DL) algorithms play an essential role and that the accuracy of the prediction largely depends on the input characteristics and the ML/DL technique used. Better predictions can be made through price prediction model selection, optimization, and incorporation of different data sources into the dataset to provide a reference for investors to avoid risks.

In this thesis, two data sets were used in terms of time intervals, daily and hourly.

Daily datasets are Bitcoin close, dollar-euro parity, gold price, natural gas, crude oil, and Ethereum.

These data sets were placed in the tables, first as a single one, then by combining the two data sets that gave the best results for each model. Finally, we experimented with each model for eight different data sets with the data set altogether. These operations were performed for each ML algorithm, like MLP, SVM, GRNN, RNN, LSTM, GRU, and CNN. Seven algorithms and a few parameters determined by these algorithms experimented with each data set separately. In the review articles (Kervancı and Akay, 2020; Khedr et al., 2021), LSTM and GRU are popular estimation approaches, so they were chosen for hyper-parameters optimization. There are many methods and libraries for hyper-parameter optimization. This thesis applied the hparam parameter, grid search, random search, and BO methods to LSTM and GRU models for the Bitcoin price prediction problem. Hparam parameter, grid and random search require manually entered information for hyper-parameter settings, while BO does not need manual input other than the first configuration. BO adjusts the following parameters, taking into account parameter values that were successful in the past, and these repeat until the desired results are obtained. Thanks to this structure, using BO will save time and provide opportunities for real-time applications. The performance is increased by the hybrid models (Aras, 2021). So we added hybrid models of GRU and LSTM to BO. Hparam parameter with LSTM; grid and random search with RNN, LSTM, and

GRU; BO with LSTM, GRU, and Hybrid LSTM-GRU experiments performed. The data set was subjected to data preprocessing and normalization. After the normalization process, the data set is turned into a time series problem by shifting one day (or hour), and it is ready for use in experiments.

Hyperparameter optimization methods (BO), Hparam parameter, random search, and grid search were examined. Models with BO achieved better results than others. To improve LSTM, GRU, and Hybrid LSTM-GRU results with BO; GBRT, GP, RF, and ET were applied to optimizers and corresponding surrogate functions. We explore the efficiency of BO surrogate functions on different learning methods with the pipelines. Using BO will save time and provide opportunities for real-time application. To our knowledge, for the Bitcoin price prediction problem, no such study has been published in the literature using an optimal model framework based on the optimal hyper-parameter set provided by the BO optimizer and corresponding surrogate functions RF, GP, ET, and GBRT.

The contribution of the thesis, Bitcoin price prediction problem literature, is below;

1. Using machine learning algorithms like MLP, SVM, GRNN, RNN, LSTM, GRU, CNN, and hybrid LSTM-GRU (GRU-LSTM)
2. Includes experiments with each machine learning model with the combinations of Bitcoin, gold, crude oil, natural gas, Ethereum, and dollar-euro parity gold, crude oil, natural gas, Ethereum, and dollar-euro parity datasets.
3. A few articles used more than one Optimization technique in the literature, and this thesis aims to provide a broader perspective by using Grid Search, Random Search, BO, and optimized BO.
4. For the Bitcoin price prediction problem, no such study has been published in the literature using an optimal model framework based on the optimal hyper-parameter set provided by the BO optimizer and corresponding surrogate functions RF, GP, ET, and GBRT.
5. It allows comparing previous studies by listing the best results with four different score values (MSE, RMSE, MAE, MAPE).
6. Hyper-parameters were included and evaluated, which yielded the best results for each technique.
7. It contains the best results among the reviewed articles for the three parameters: MAE=0.002302, MSE=0.000015, and RMSE=0.003269.
8. While it provides an hourly prediction opportunity for investors, it will also give an overview of the future price with its prediction in a time interval such as 30 days.

2. PRELIMINARY WORK

This section analyzed previous articles in terms of causality, time interval, hyper-parameter optimization, and methods. In reference (Kervancı and Akay, 2020), Bitcoin review paper, statistical methods, machine learning -statistical methods, machine learning-machine learning, frequency effect of the selected time, the impact of social media, and web search engine, causality, optimization of hyperparameters are examined under seven headings. However, since certain ML methods will be used in this thesis, method groups have been discussed under four main headings.

2.1. Causality

The Generalized Auto Regressive Conditional Heteroskedasticity (GARCH) model was used for Bitcoin price prediction (Baur et al., 2018). They found that the dollar and gold have a causality relation to Bitcoin. According to reference (Smith, 2016), gold is the most convenient way to think about Bitcoin. While market exchange rates are not affected by changes in Bitcoin prices, Bitcoin exchange rate volatility is one of the sources of other cryptocurrency's price volatility. According to (Baur et al., 2018), Bitcoin differs from fiat currencies and gold. Autoregressive-Distributed Lag (ARDL) and Nonlinear Autoregressive-Distributed Lag (NARDL) models were used to examine the relationship between Bitcoin and gold (Bouri et al., 2018). So they found no link between Bitcoin and gold prices in the short and long terms. (Bouoiyour and Selmi, 2015) stated that used ARDL and found that the gold price has no considerable effect on Bitcoin. The effect of gold on the bitcoin prediction performance, CNN, LSTM, and GRU models were used (Aggarwal et al., 2019). LSTM obtained the best RMSE result among them. When the gold price was added as a parameter to the Bitcoin dataset, and the tests were repeated for the LSTM, CNN, and GRU models, no positive effect of gold on the Bitcoin price was observed. According to (Gkillas et al., 2020), they detected closer relationships between gold and Bitcoin compared to crude oil and Bitcoin as well as crude oil and Bitcoin.

They are (Okorie and Lin, 2020) examined the link between crude oil and cryptocurrency markets. Their study, which they took as examples of Bitcoin, the top four, and the bottom five cryptocurrencies by market value, showed that both bidirectional and unidirectional volatility spread from the crude oil market to the cryptocurrency markets.

According to (Ghorbel and Jeribi, 2021), there are interdependencies between cryptocurrencies, and secondly, there is a bidirectional volatility spillover between cryptocurrencies and stock indices, oil, and gold markets.

They stated (Moussa et al., 2021) that natural gas does not influence Bitcoin, while Gold and Oil Brent Crude logarithmic prices potentially and significantly affect Bitcoin. As stated (Jiang et al., 2022), Bitcoin, gold, foreign exchange, and natural gas markets serve as volatility communicators.

They are (Beneki et al., 2019) found that while showing a volatility transfer from Bitcoin to Ethereum that peaked quickly and disappeared over a long period, the reverse effect was significantly weaker.

As stated by (Balcilar et al., 2017), volume positively affects price prediction when there are no fluctuations in the Bitcoin market. However, when the market is volatile, the estimation from the historical Bitcoin price produces the best results, and the volume is insignificant. According to (Aalborg et al., 2019), Bitcoin price changes can be predicted reasonably from its historical values. In this respect, Bitcoin is like any other financial asset. They also found that Bitcoin's trading volume improves predictions of Bitcoin volatility.

Table 2.1. Causality relation with the Bitcoin price prediction papers is as follows

	correlation	**no correlation**
Gold	(Baur et al., 2018), (Smith, 2016), (Gkillas et al., 2020), (Moussa et al., 2021), (Ghorbel and Jeribi, 2021)	(Bouoiyour and Selmi, 2015), (Bouri et al., 2018), (Aggarwal et al., 2019)
Dollar-euro	(Baur et al.,2018), (Smith, 2016), Garch Model (Szetela et al., 2016)	(Bouri et al., 2018), Arma Model (Szetela et al., 2016)
Crude-oil,	(Gkillas et al., 2020), (Okorie and Lin, 2020), (Ghorbel and Jeribi, 2021)	
Natural gas	(Jiang et al., 2022)	(Moussa et al., 2021)
Ethereum	(Beneki et al., 2019)	
Other cryptocurrencies	(Ghorbel and Jeribi, 2021)	
Social media	(Martina et al., 2015), (Kaminski and Lab, 2016), (Garcia et al., 2014), (Shen et al., 2019),	
Bitcoin volume	(Aalborg et al., 2019), (Balcilar et al., 2017)	
Bitcoin historical price	(Balcilar et al., 2017), (Aalborg et al., 2019)	

They showed linkages between markets in the short, medium, and long term, but cryptocurrencies are relatively insulated from these market shocks and stand out from popular financial assets (Corbet et al., 2018).

As stated in (Szetela et al., 2016), GARCH models have identified some reliance between Bitcoin and US Dollar, Euro, and Yuan, while Autoregressive Moving Average (ARMA) analysis has no relationship found. They examined tweet data and Web Search engine results for Bitcoin's price prediction. They found a significant correlation between Google Trends data and tweets. They confirmed that the number of tweets about Bitcoin prices in a positive mood could predict changes in Bitcoin prices. Also, in this study, positive and negative tweets about Bitcoin price are interpreted as a reason for Bitcoin's increasing popularity (Martina et al., 2015). Furthermore,

studies such as (Kaminski and Lab, 2016; Garcia et al., 2014; Shen et al., 2019) were conducted. They said it has little effect on Bitcoin price when examining the relationship between Bitcoin, Twitter, and Google Trends.

2.2. Time Interval

Applying DL is demanding and requires many learning samples (Devavrat and Kang, 2014; Madan et al., 2015). This thesis uses a daily data set while making Bitcoin price predictions. In addition to the daily data set, estimations were made with the hourly data set, and these two were compared. The hourly data set produced better results because it contained more data and was more sensitive than the daily data.

The following articles contain Bitcoin price predictions made with data sets in different time ranges, and the smallest time interval is not always recommended. In (Shintate and Pichl, 2019), they used time intervals of 1-minute and 30-minutes. They preferred the 30-minutes dataset due to the difficulties and losses of using the data in the 1-minute timeframe online. They used three different length datasets with Represent Bayesian regression algorithm and k-means. These datasets consist of 30-minutes, 1-hour, and 2-hours of data. K-means achieved a return of 89% with a 1-hour dataset in 50 days. They recommended the 1-hour dataset, as the 1-hour dataset provides the best returns (Devavrat and Kang, 2014). (Madan et al., 2015) compared the 10-minutes and 10-seconds datasets, they found that the 10-minutes dataset gave better results regarding sensitivity and specificity. Although the 10-seconds dataset predicts the price changes more accurately, they suggested the lower margin of 10-minutes because the algorithm could not send trades in the 10-seconds timeframe. The estimation made with the daily average value created by averaging the minute and second data set during the day gave the best results (Thearasak and Thanisa, 2018). They used Theil-Sen Regression, LSTM, Huber Regression, and GRU. As a result, GRU achieved the best results MSE of 0.00002. Based on (Karasu et al., 2018), they are used daily close price of price prediction.

In addition, considering the previous studies, using the hourly data set is quite reasonable, as minute and second data sets require a lot of disk space, and there are missing data.

2.3. Hyper-parameter Optimization

Since hyper-parameters are dataset and model specific, their setting is not global, and it does not have the same values and effects for the problem and model, and it must be repeated with each change. Some hyper-parameters in neural networks are learning rate, weight initialization, network layers, and the number of neurons (Yu et al., 2020). They are judged on their results in terms of learning rate. (Breuel, 2015) examined batch sizes, momentum, different non-linearities, and peephole parameters on the learning rate. This thesis uses BO, Hparam parameter, random search, and grid search methods for hyper-parameter optimization, while (Aggarwal et al., 2019)

and (Singh and Agarwal, 2018) only use grid search. As stated in (Buslim et al., 2021), ML models like LSTM, GRU, and RNN used grid and random search for hyper-parameter optimization. According to (Singh and Agarwal, 2018), after normalizing the Bitcoin and Ethereum datasets with the Min-Max normalization technique, they optimized the SVR, K-Nearest Neighbors (KNN), and Polynomial regression methods with grid search. In (Pour et al., 2022), they used LSTM with BO. In (Fadil et al., 2021), they used SVR with grid search.

Table 2.2. Hyper-parameter optimization for the Bitcoin price prediction papers is as follows

Year	Authors	Optimization	Methods
2018	(Singh and Agarwal, 2018)	Grid search	Polynomial regression, SVR, and KNN regression
2018	(McNally et al., 2018)	BO	RNN, LSTM
2019	(Felizardo et al., 2019)	Random search	ARIMA, RF, SVM, LSTM, and WaveNets
2021	(Buslim et al., 2021)	Grid search and Random search	GRU, RNN, and LSTM
2022	(Drahokoupil, 2022)	BO	XGBoost
2022	(Lahmiri et al., 2022)	BO	SVR
2022	(Pour et al., 2022)	BO	LSTM
2022	(Tripathi and Sharma, 2022)	BO	LSTM, BiLSTM, CNN- BiLSTM, DANN
2023	(Kervancı and Akay, 2023)	Hparam	LSTM

As shown in Table 2.2, while BO is the most used method, there is no study in which all three are combined for Bitcoin price prediction.

2.4. Methods

They used the hybrid process, ARIMA, and VAR parametric techniques for input selection, and after that, SVR and ANN nonparametric techniques for prediction (Ince and Trafalis, 2006). As a result, SVR exceeds the ANN, and hybrid methods perform better than naive. They used MLP-based NARX, MA, and PSO algorithms for the prediction model (Indera et al., 2017). This model has an acceptable correlation coefficient at the 95% confidence limit. They compared prediction performance between statistical models and machine learning methods. They used ARIMA, Bayesian optimized RNN, and LSTM (McNally et al., 2018). The ARIMA predictions were insufficient, while LSTM gained the maximal accuracy, and the RNN gained the minimal RMSE. LSTM and RNN were examined between 50 and 100 days, and the best prediction for LSTM was 100 days, while RNN was 50 days. They used the 1-minute dataset with Theil-Sen Regression,

Huber Regression, LSTM, and GRU for prediction (Thearasak and Thanisa, 2018). As a result, GRU has the best MSE at 0.00002. They used LR, L-SVM, and P-SVM for Bitcoin closing price prediction (Karasu et al., 2018). Bitcoin's closing price was predicted for the MA and WMA filters using the time series containing data for the daily closing price, highest price, and lowest price. When the estimates were evaluated regarding MSE, the P-SVM performed 0.00075, while the LR performed the worst. Based on (Shen and Wan, 2019), the daily interval trading dataset was used for the examination from April 30, 2013, to November 20, 2018. GRU, GARCH, and ARIMA methods were examined. GRU will outperform GARCH using historical datasets if it has enough datasets to learn. As a result, the GRU model outperformed the GARCH model. Based on (Ze et al., 2019), the GARCH model, SMA, and GRU are used, and the GRU model outperformed the other models.

Based on (Mangla et al., 2019), SVM, RNN, Logistic regression, and ARIMA methods were used, and their results, respectively, were 48%, 50%, 47%, and 53%. Regarding the prediction accuracy of these methods, ARIMA has an accuracy of 53% for the next day, and its performance gradually decreases as the number of days to predict increases. But RNN accuracy is 50% up to 6 days.

Based on (Dutta et al., 2020) used a number of exogenous and internal variables with RNN, LSTM, and GRU models to predict Bitcoin price. As a result, GRU found better results than LSTM and RNN in a shorter time. GRU and LSTM achieved the same results as the training dataset, while the test dataset with GRU had 0.005, better results than all others.

They used Bitcoin, Ethereum, and Ripple (Livieris et al., 2021). These cryptocurrency datasets are examined separately in CNN and LSTM hybrid models. After that, merge the three of them, then use layers. Their results for Bitcoin MAE=169.817 and RMSE=256.688.

Based on (Parekh et al., 2022), Dash, Litecoin, and Bitcoin datasets are generated and normalized to [1-10]. LSTM and GRU layers are applied to these datasets and combined with results from DL-GuesS, a hybrid model based on Twitter sentiment. The best results for Dash are MSE=0.0011, MAE=0.0196, and MAPE=4.4089.

Table 2.3. ML and conventional statistical methods papers

	ARIMA	**AR**	**MA**	**TR**	**HR**	**RF**	**MLP**	**SVM**	**RNN**	**LSTM**	**GRU**
(Madan et al., 2015)						*		**			
(Thearasak and Thanisa, 2018)				**	*** *					*	** **
(Spilak, 2018)							**		*	*** *	
(McNally et al., 2018)	*								**	*** *	
(Roy et al., 2018)	*** *	**	*								
(Zheshi et al., 2019)						*		*** *		**	
(Shen et al., 2019)	*										**
(Dutta et al., 2020)									*	**	*** *

3. MATERIAL AND METHOD

3.1. Material

3.1.1. Datasets

Gold, crude oil, natural gas, Ethereum, the dollar-euro parity, and Bitcoin data sets were downloaded from https://www.investing.com/. These datasets consist of time series 7 August 2015 to 13 December 2022. Gold, crude oil, natural gas, Ethereum, and Bitcoin prices are in dollars.

Table 3.1. Daily datasets to use for Bitcoin price prediction

Datasets	Contents
X1	Bitcoin
X2	Bitcoin, gold
X3	Bitcoin, Ethereum
X4	Bitcoin, crude-oil
X5	Bitcoin, dollar-euro parity
X6	Bitcoin, natural gas
X7	Bitcoin, gold, Ethereum
X8	Bitcoin, gold, crude-oil
X9	Bitcoin, gold, dollar-euro parity
X10	Bitcoin, gold, natural gas
X11	Bitcoin, Ethereum, crude-oil
X12	Bitcoin, Ethereum, dollar-euro parity
X13	Bitcoin, Ethereum, natural gas
X14	Bitcoin, crude-oil, dollar-euro parity
X15	Bitcoin, crude-oil, natural gas
X16	Bitcoin, dollar-euro parity, natural gas
X17	Bitcoin, gold, Ethereum, natural gas, dollar-euro parity, crude-oil

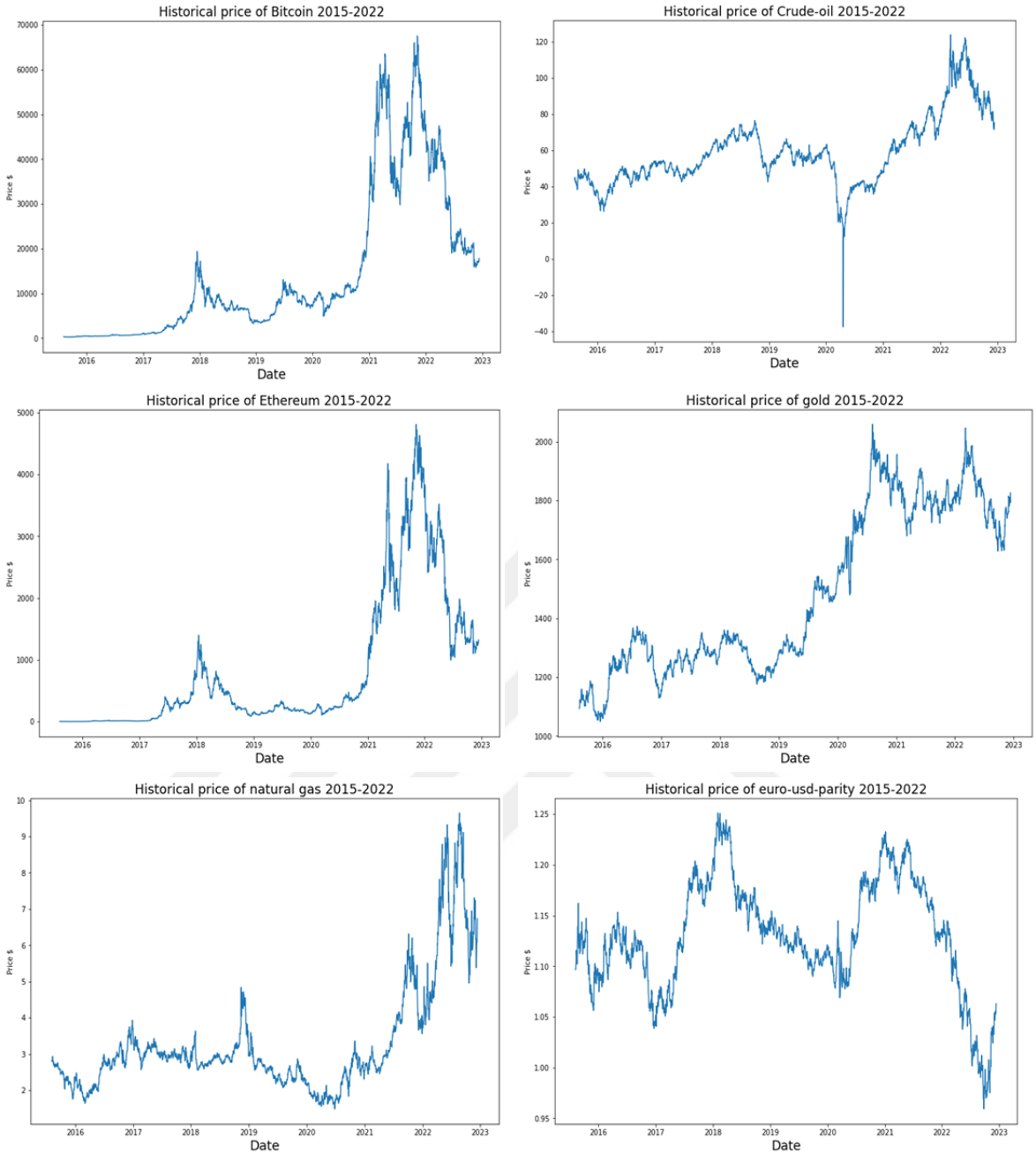


Figure 3.1. Historical price

X1: used daily closed prices and dates. The data set was subjected to data preprocessing, and the Min-max normalization technique was applied and then used the shift method. In this method, the closing price of Bitcoin was shifted one day, and each day was prepared to be estimated the next day, so it was transformed into a time series problem. All data sets are converted into time series like the X1 data set, then relevant fields are added, and normalization is done. The normalization technique is min-max normalization. And then, the train_test_split function of the 'sklearn.model_selection' library is used for dividing datasets.

$$X_train, x_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, shuffle = False)$$

Datasets were divided into training 80% and test %20 subsets.

3.1.2. Performance Metrics

There are many performance and error measurement methods for regression and classification problems. But while there is no definitive document or recommendation on which metric is appropriate for particular problems and datasets, there are studies on it (Naser and Alavi, 2021). Since a single error metric is not used in the articles that make Bitcoin price predictions, the error rate metrics MAE, MAPE, MSE, and RMSE are used to increase comparability with different articles. In this thesis, four metrics are used, and given below with their formulas.

X=predicted value, Y=real value, m=number of value;

$$MAE = \frac{\sum_{j=1}^{m} |y_j - x_j|}{m} \qquad \text{Equation 3.1}$$

$$MAPE = \frac{\sum_{j=1}^{m} (|y_j - x_j| / y_j)}{m} * 100 \qquad \text{Equation 3.2}$$

$$MSE = \frac{\sum_{j=1}^{m} |y_j - x_j|^2}{m} \qquad \text{Equation 3.3}$$

$$RMSE = \sqrt{\frac{\sum_{j=1}^{m} |y_j - x_j|^2}{m}} \qquad \text{Equation 3.4}$$

3.1.3. Min-Max Normalization

Normalization, which is the preprocessing step, is a scaling technique. When predicting, creating a new range for very large or small dataset features is helpful. It fits each field in the dataset into a uniform scale while preserving the relationship between the dataset and the normalized dataset (Saranya and Manikandan, 2013). Min-max normalization was applied to all data sets in this thesis. The Min-Max formulation is as follows Y_i values are in the range of $[0,1]$.

A is a original dataset,

Predefined range $[B_1, B_2]$, Equation 3.5

$$A_{i(\text{Min-Max Normalized})} = \frac{A_i - Min(A)}{(Max(A) - Min(A))} * (B_2 - B_1) + B_1$$

3.1.4. The Structure of Bitcoin

Bitcoin's "electronic payment system based on cryptographic proof" was announced for the first time in 2008 by Satoshi Nakamoto (Satoshi, 2008). Bitcoin stores digital money and transfers in a distributed digital ledger without a central authority or database. This digital ledger structure is called Blockchain. Blockchain is distributed data among nodes, and once the information processed there is immutable, anyone can view the information and question the validity of transactions (Yaga et al., 2018). Thanks to this unchangeable structure, Blockchain is a suitable solution for a wide variety of uses. Blockchain is cryptocurrencies, smart contracts, and distributed ledger systems (Sherman et al., 2019).

Blockchain application areas are quite wide, for example; ownership, royalty distribution, judiciary, cognitive computing, IoT, supply chain, and health applications (Ahram et al., 2017). Articles containing suggestions that each of these areas can be applied have been published (Griggs et al., 2018; Nugent et al., 2016; Kouhizadeh and Sarkis, 2018; Guo et al., 2020). There are three types, according to the digital book's access: Open Blockchain, private Blockchain, and consortium Blockchain (Wang et al., 2019). Participants can participate in open Blockchains without permission, while they can participate in private and consortium Blockchains with permission. Private Blockchains have a single jurisdiction or organization, but consortium blockchain has multiple organizations rather than just one (Sukhwani et al., 2017). In consensus-based Blockchain networks, it is possible to increase the speed and reduce the computational cost depending on the type of trust and needs among the participants (Yaga et al., 2018). Quorum, Tendermint, MultiChain, Hyperledger Fabric, Iroha, Symbiont, R3 Corda, IOTA, Kadena, Sawtooth Lake, Ripple, Chain, and Stellar are examples of Consortium Blockchain Networks (Cachin et al., 2017).

3.2. Methods

This thesis used MLP, SVM, GRNN, RNN, LSTM, GRU, and CNN methods of DL and ML. All experiments were coded with Python (version 3.9.7) in Spyder (version 5.1.5) with Keras library (version 2.7.0), Tensorflow (version 2.7.0) backend, and Tensorboard (version 2.7.0). At the end of each experiment, the kernel is reset so that no results affect each other. Table 3.2 contains all datasets and combinations of models. After these combinations are applied, the data sets that give the best two results for each model are combined and applied to the model. As the last step, the X17 data set, where all data sets are combined, is applied to all models. Thus, model performances are evaluated with different data sets.

Table 3.2. Combination of datasets and models for Bitcoin price prediction

Datasets							
X1	MLP	SVM	GRNN	RNN	LSTM	GRU	CNN
X2	MLP	SVM	GRNN	RNN	LSTM	GRU	CNN
X3	MLP	SVM	GRNN	RNN	LSTM	GRU	CNN
X4	MLP	SVM	GRNN	RNN	LSTM	GRU	CNN
X5	MLP	SVM	GRNN	RNN	LSTM	GRU	CNN
X6	MLP	SVM	GRNN	RNN	LSTM	GRU	CNN

These seven methods are explained below.

3.2.1. Multilayer Perceptron (MLP)

MLP is called feedforward neural networks or Deep feedforward networks. Usually, more than one neuron, even with many inputs, may be required. We might need more than one neuron operating parallel in the 'layer' (Nazzal et al., 2008). MLP is a fully connected feedforward artificial neural network (ANN); the input layer connects with neurons in the hidden layer by linking weights. It maps the inputs to the desired set of outputs by multiplying them by weights and applying them to the activation function in the neurons. An activation process takes place in each neuron. The structure of MLP is illustrated in Figure 3.2.

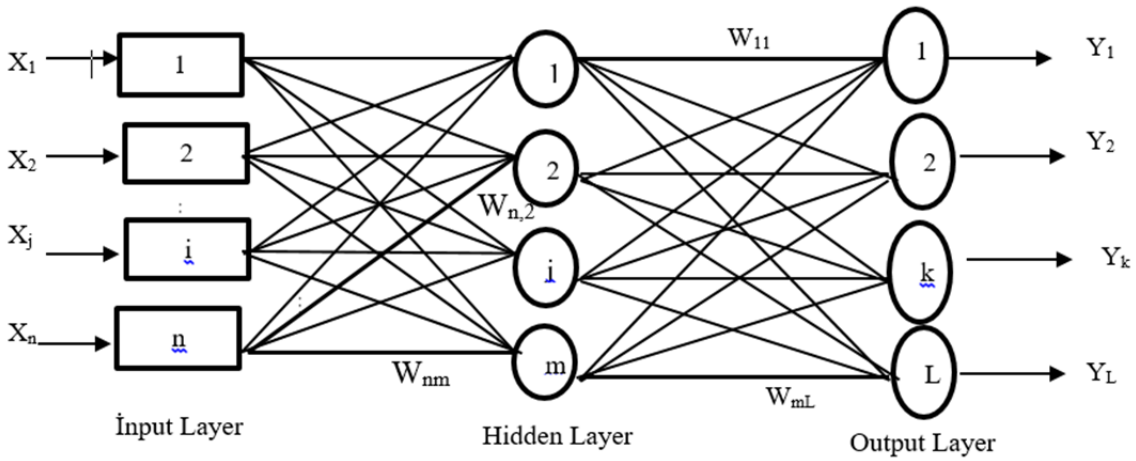


Figure 3.2. Multilayer Perceptron (Widiasari et al., 2017)

MLP is called feedforward because information flows from the input layer to the output y through the activation function, through the intermediate calculations used to define f. The model does not have feedback links through which outputs are fed back to itself (Goodfellow et al., 2016). MLP consists of three main parts: an input layer, a hidden layer, and an output layer.

The input and output layers can be accessed directly, but hidden layers cannot (Harun et al., 2010).

$$H_j = f\ (\sum_{i=1}^{n} W_{ji} X_i + b_i)$$ Equation 3.6

In Equation 3.6, X_i coming to each neuron in the hidden layer is multiplied by W_{ji} and bias is added, and H_j obtained through an activation function takes, its place in the output layer as in Equation 3.7.

$$y = f\,(\sum_{j=1}^{m} W_{kj} H_i + b_0)$$ Equation 3.7

3.2.2. Support Vector Machine (SVM)

The SV algorithm was first developed by Vapnik and co-workers in 1963-1964. After then, SVM was primarily developed at AT&T Bell Laboratories by Vapnik and co-workers in 1982-1995. SVM is a supervised learning method; first, SVM was developed to solve the classification problem. Later, a tool for regression was added and named Support Vector Regression (SVR) to solve regression problems (Smola and Scholkopf, 2004).

Training dataset

$$\{(X_1, Y_1) \ldots (X_t, Y_t)\} \subset X \times R,$$ Equation 3.8

X is input, and Y is output. Our goal is to find a function $f(x)$ for all training data input pairs with the least deviation ε for input X_i to Y_i. As long as the errors are smaller than ε, we ignore them, and when they are larger, we review and recalculate the W_i (weights) and b_i (bias).

$$f(x) = (w, x) + b \text{ with } w \in X\ , b \in R$$ Equation 3.9

$$\text{minimize} \frac{1}{2} \| w \|^2$$

$$\text{subject to} \begin{array}{l} y_i - (w, x_i) - b \le \varepsilon \\ (w, x_i) + b - y_i \le \varepsilon \end{array}$$ Equation 3.10

In Equation 3.9, the function f can be applied to optimization problems that approximate all pairs (X_i, Y_i) with precision ε, convex optimization problems. However, in (Cortes and Vapnik, 1995), this is sometimes not the case. When the loose variables $\varsigma_i, \varsigma^*_i$ apply Equation 3.9 because it is desired to allow some errors, the optimization problem Equation becomes like 3.10. When SVM (Equation 3.9) to nonlinear SVR, we get a hyper-parameter C that we can optimize (Equation 3.10). We allow for an extra $\varsigma_i, \varsigma^*_i$ deviation from the margin of our data.

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2} w^T w + C \sum_{i=1}^{n} (\varsigma_i + \zeta^*_i) \\ & y_i - (w, x_i) - b \le \varepsilon + \varsigma_i \\ \text{subject to} \quad & (w, x_i) + b - y_i \le \varepsilon + \zeta^*_i \\ & \varsigma_i, \zeta^*_i \ge 0 \end{aligned}$$

Equation 3.11

$$|\zeta|_\varepsilon := \begin{matrix} 0 & \text{if } |\zeta| \le \varepsilon \\ |\zeta| - \varepsilon & \text{otherwise} \end{matrix}$$

Equation 3.12

SVM algorithms use many mathematical functions defined as different types of kernel functions. For example Linear, Nonlinear, Polynomial, Gaussian, Radial Basis Function (RBF), the Gaussian Radial Basis Function, Hyperbolic Tangent, and Sigmoid.

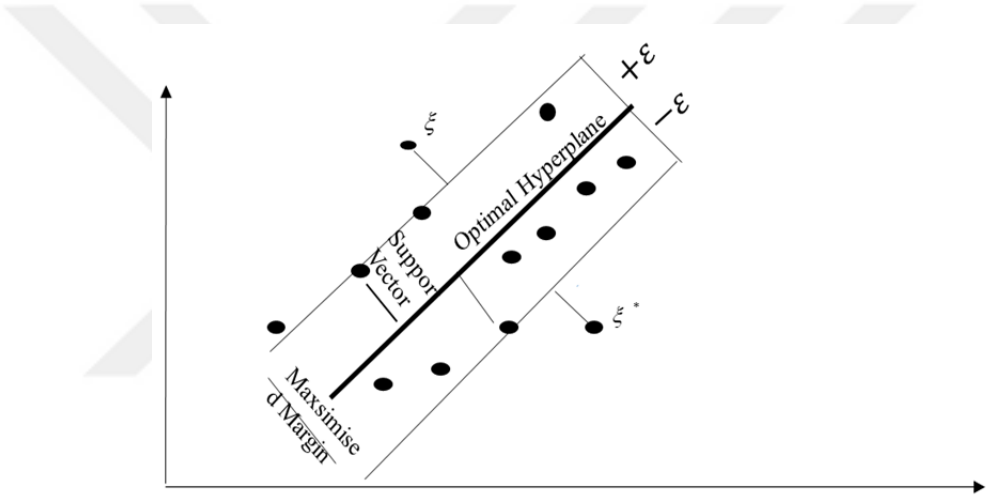


Figure 3.3. SVR

We used the SVR of the sklearn.svm library in the Bitcoin price prediction problem.

3.2.3. Generalised Regression Neural Network (GRNN)

GRNN is a radial basis function first introduced by (Specht, 1991). It is similar to the common FFNN but the process differs from the nonlinear regression idea for functional assessment. GRNN matches any arbitrary function between input and output vectors, and draws the function assessment directly from the training dataset (Feng et al., 2017). GRNN can effectively solve nonlinear problems thanks to its high fault tolerance, strong nonlinear mapping capability, robustness and simplicity of network structure (Li et al., 2012). The GRNN has layers like the input, pattern, summation, and output layers. The input layer is fed by vector X and has neurons the size of vector x.

The X's from the input layer and the σ parameter and nonlinear P_i's are calculated as in Equation 3.13.

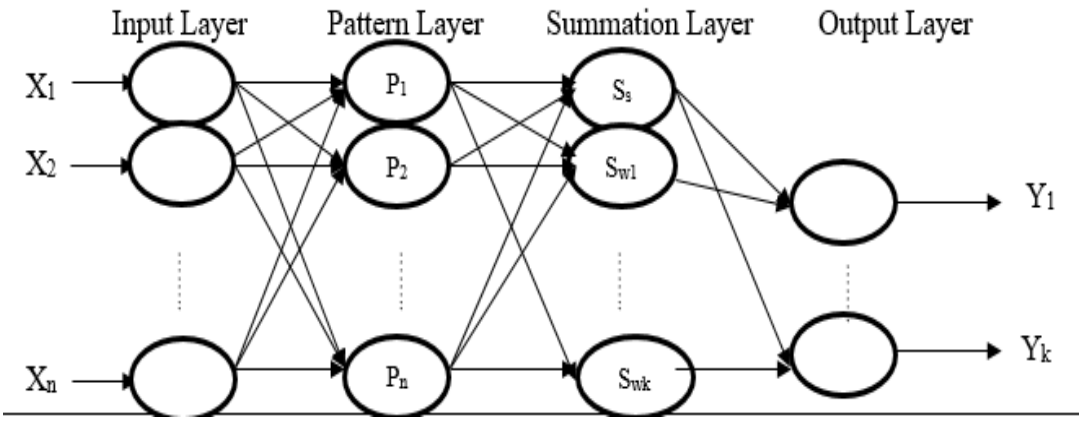


Figure 3.4. GRNN architecture (Li et al., 2012)

$$P_i = exp\frac{(X - X_i)^T (X - X_i)}{2\sigma^2}$$ Equation 3.13

In the summation layer, two operations occur, as in Equations 3.14 and 3.15.

$$S_s = \sum_{i=1} P_i$$ Equation 3.14

$$S_w = \sum_{i=1} P_i W_i$$ Equation 3.15

The simple sum is performed in Equation 3.14, and in Equation 3.15, the results of the multiplication of the connection weight W_i of the values from the pattern layer are summed.

$$Y = \frac{S_s}{S_w}$$ Equation 3.16

Assuming that there are k output layers, the result from equation 1.14 is divided by each of equation 3.15 to get k outputs Y from equation 3.16.

3.2.4. Recurrent Neural Networks (RNN)

RNNs form loops through connections between neurons, which is why it is called a recurrent neural network. These loops in the RNN store the data from the previous step and transmit it as feedback to the neuron in the next step. This mechanism creates an internal memory and explicitly enables sequential data learning (Madan and Mangipudi, 2018). An RNN allows information to be carried because it has loops. It makes them different from the other methods; their memory helps them find correlations between inputs. These circles can be used bidirectional

and theoretically anywhere in the network (Haykin, 1994). Among other features, the recurring connection provides a delay as it can store values from the previous time step that can be used in the current time step. Therefore, it can learn temporary and permanent patterns (Abdennour, 2006).

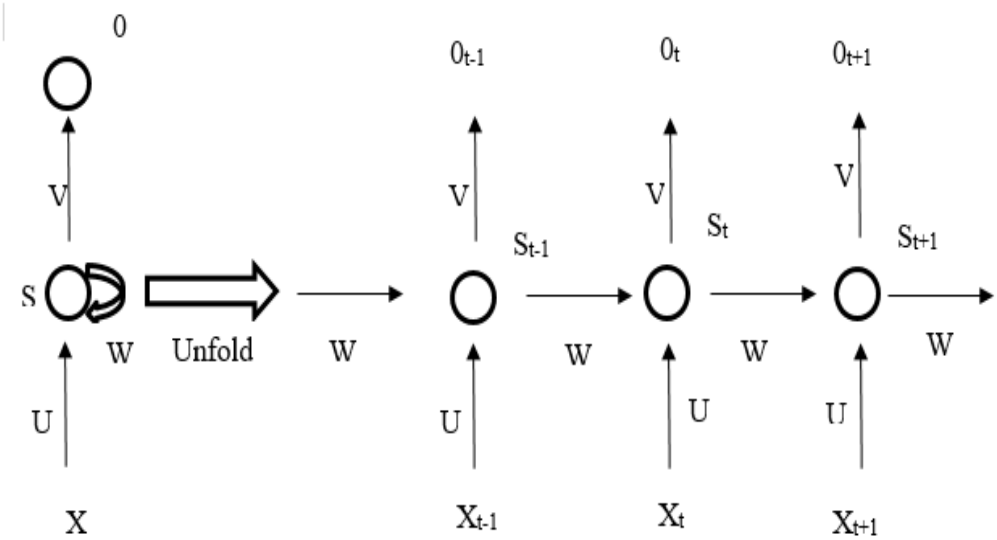


Figure 3.5. RNN architecture (LeCun et al., 2015)

RNNs process an input sequence one element at a time, holding a 'state vector' in their hidden units that implicitly contain information about the history of all the past input sequences. It becomes transparent how we can apply back-propagation to train RNNs when we think of the outputs of hidden units at different discrete time steps as if they were the outputs of different neurons in a deep multilayer network (Figure 3.5 above). RNNs are powerful dynamic systems, but training them has proved challenging because the back-propagation of gradients either grows or shrinks at each step. Over many steps, they can explode or vanish (LeCun et al., 2015).

$y \rightarrow$ output of neuron, $f \rightarrow$ nonlinear function,
$\alpha \rightarrow$ activation of neuron Equation 3.17
$y = f(\alpha)$

$m \rightarrow$ input variables, $w_i \rightarrow$ weights,
$\tau \rightarrow$ threshold, $u \rightarrow$ input vector, Equation 3.18

$$\alpha = \sum_{i=1}^{m} w_i u_i + \tau,$$

3.2.5. Long Short-Term Memory (LSTM)

LSTM is a special type of RNN developed to solve the exploding or vanishing gradient problems in traditional RNNs. It is possible, thanks to the gated structure of the LSTM cell, that decides whether the data in the cells in the current and previous time step can pass or not, thus keeping its flow under control. As an RNN, the LSTM network has feedback links to maintain information order, making it a powerful tool for processing sequential data. LSTM can solve unsolvable time series tasks by feed-forward networks utilizing different time intervals. Time series data often have cyclic patterns, where the observations rise and fall over long periods. The LSTM networks can be constructed so that LSTM can remember long-term relationships in the data. The LSTM networks have been shown to model temporal sequences and their long-range dependencies more accurately than the original RNN model (Kazybek et al., 2018). We might need five or ten, operating parallel in what we will call a "layer". This concept of a layer is discussed below. Each LSTM cell has three gates. The first gate is the input gate, which processes the input information at a given time. The second gate is the forget gate that allows removing the information from the previous cell. The last gate merges the information from the input and output gates to feed the next cell of the LSTM network with selected information (Brunel et al., 2019).

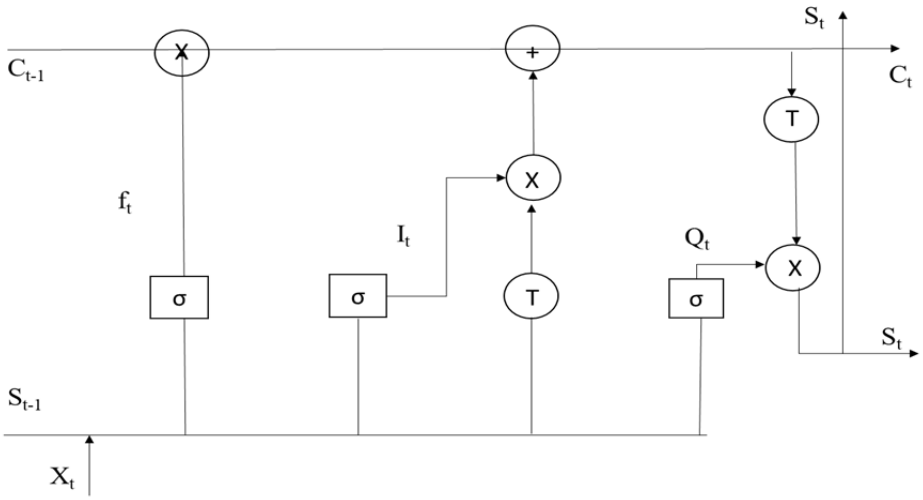


Figure 3.6. LSTM (Le et al., 2019)

The first step of is f_t forget gate in LSTM is to decide which of the incoming data will be forgotten thanks to the existence of the forget gate.

$$f(t) = \sigma\left(W\left[S_{t-1}, X_t\right] + b\right)$$

Equation 3.19

1- The second step is to decide which of the new incoming information should be stored in the cell or not.

The second step has two layers,

i) It layer decides which values to be updated or not

$$I(t) = \sigma(W[S_{t-1}, X_t] + b) \quad \text{Equation 3.20}$$

ii) Tanh layer that creates a vector of new candidate values C_t

$$\tanh(t) = \tanh(W[S_{t-1}, X_t] + b) \quad \text{Equation 3.21}$$

Then, update the old cell state, C_{t-1} into the new cell state C_t, which can be given

$$C_t = C_{t-1} f_t + I_t \tanh_t \quad \text{Equation 3.22}$$

2- Finally, decide what is going to be produced as output. This output will be based on the cell state, but will be a filtered version. In this step, the output gate O_t decides what parts of the cell state are going to be produced as output.

$$O_t = \sigma(W[S_{t-1}, X_t] + b) \quad \text{Equation 3.23}$$

3- Then, the cell state C_t goes through the tanh layer and multiply it by the output gate O_t as follows.

$$S_t = O_t \tanh(C_t) \quad \text{Equation 3.24}$$

3.2.6. Gated Recurrent Unit (GRU)

Based on (Cho et al., 2014), GRU was suggested in 2014. GRU solves the disappearing gradient problem in RNNs. Thanks to its gating units, it adaptively controls the flow of information in each time step. For example, although the gated structure of the GRU is similar to the LSTM unit, thanks to GRU transition units, it controls the flow of information in the same cell unit without needing different memories (Chung et al., 2014). The gate mechanisms at the GRU are a simple copy of the structure of RNNs. Although LSTM and GRU are different from each other, their gated structures are similar to each other. Deciding which one will perform better, in general,

is problematic because it depends on the problem. (Chung et al., 2014; Fu, Zhang, and Li, 2016) reported that GRU progresses faster. GRU and LSTM get better results than RNNs thanks to the structure provided by the gating units. RNNs replace the unit's content with calculated from the new input and the information from the previous state in the hidden layer. But both new information and old content are preserved in LSTM and GRU.

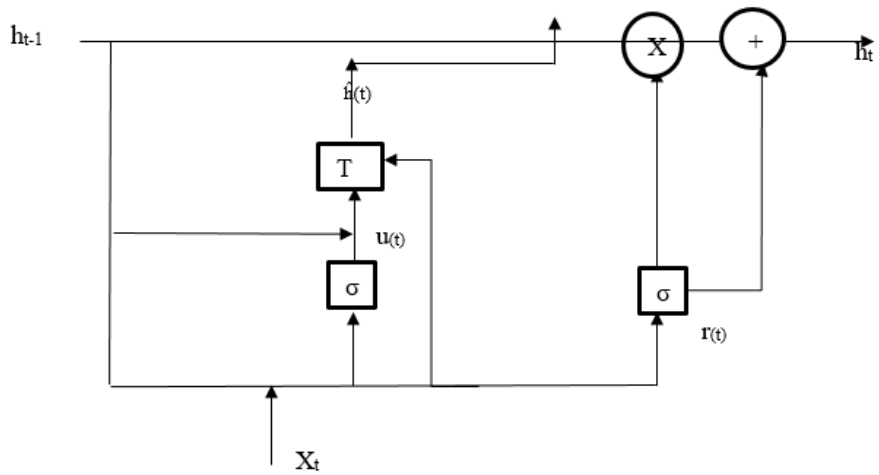


Figure 3.7. GRU (Gated Recurrent Unit) (Zhao et al., 2017)

$$\mathbf{r}_{(t)} = \sigma\left(W_r X_t + W_r h_{t-1} + b_r\right) \qquad \text{Equation 3.25}$$

$$u_{(t)} = \sigma\left(W_u X_t + W_u h_{t-1} + b_u\right) \qquad \text{Equation 3.26}$$

$$h_{(t)} = tanh\left(W.\left[r_{(t)} h_{t-1}\right] + W X_t\right) \qquad \text{Equation 3.27}$$

$$h_{(t)} = \left(1 - u_{(t)}\right)\ (h_{t-1} + u_{(t)}) h_{(t)} \qquad \text{Equation 3.28}$$

3.2.7. Convolutional Neural Network (CNN)

CNN is known as Convolutional Network. CNN is a data-specific structure with a grid-type structure, such as images and time series. The convolution layer, one of the essential components of CNN, is the layer that makes feature extraction from the combination of linear and nonlinear operations. The structure of CNN has one or more convolutional layers followed by just like a multilayer neural network with one or more fully connected layers. A convolutional neural network consists of pooling, fully connected layers, and convolutional. Thanks to a back-propagation algorithm, it is designed to automatically and adaptively learn spatial hierarchies of features extracted by the convolution layer (Yamashita et al., 2018). A convolutional layer consists of three stages. In these stages, it first applies a kernel to its input, performing a series of

convolutions in parallel to produce a series of linear activations. Each linear activation is executed via a nonlinear activation function in the second step. The layer's output is modified through the pooling layer in the third step. (Mai et al., 2018). CNN has achieved a remarkable triumph in pattern recognition. Instead of features extracted by data analysts, CNNs automatically extract in-depth features from images. In addition, CNN algorithms used for time series classification problems are preferred due to weight sharing and rotation advantages (Zhang et al., 2018). Unlike other feature extraction-based approaches for time series, CNNs can automatically extract and generate in-depth features of time series thanks to the convolution and pooling layer (Zhao et al., 2019).

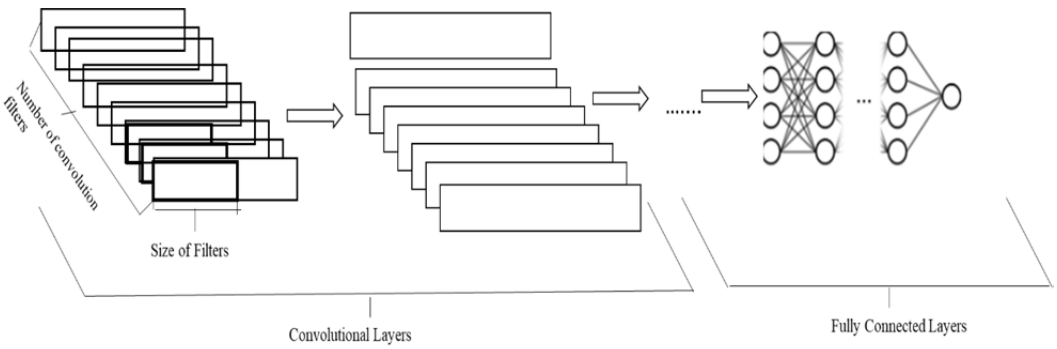


Figure 3.8. Multiple-channel CNN (Mozo et al., 2018)

X_t is a time series dataset

Y_t denotes the target output

$((X_1, Y_1), (X_2, Y_2), ...(X_{N\ Sample}, Y_{N\ Sample}))$ Equation 3.29

$X_t \in R^{Nxk}, Y_t \in R^n$ for $1 \le t \le N_{sample}$

$s \rightarrow$ convolution stride, w $\rightarrow$ weights

$C_r() \rightarrow$ feature map, b $\rightarrow$ bias

$$C_r(t) = f(\sum_{i=1}^{l}\sum_{j=1}^{k} x(i + s(t-1), j) w_r(i, j) + b(r))$$ Equation 3.30

$P_r() \rightarrow$ pooling layer

g $\rightarrow$ pooling strategy (max pooling) Equation 3.31

$P_r(t) = g(C_r((t-1)l + 2), ..., C_r(tl))$

$O() \rightarrow$ output layer $j = 1, 2, ...n$

$$O(j) = f(\sum_{i=1}^{M} z(i) w_f(i, j) + b_f(j))$$ Equation 3.32

3.2.8. Hyper-parameter Optimization Methods

ML is widely used in many application areas, and one of these areas is prediction. In this thesis, in which machine learning models MLP, SVM, GRNN, RNN, LSTM, GRU, and CNN are examined, hyper-parameter optimization techniques are essential to determining the values or value ranges of the parameters that perform well in the models. Using hyper-parameter optimization methods, we save time, human resources, and processor power required to optimize ML algorithms containing big data and many parameters.

The methods used in this thesis for optimization are as follows. 'Hparam' was imported from 'tensorboard.plugins.hparams' API, 'GridSearchCV' was imported from model_selection of the sklearn library for Grid search, 'RandomisedSearchCV' was imported from model_selection of the sklearn library for Random search, and 'BayesSearchCV' imported from the skopt library was used for Bayes optimization.

It is aimed to ensure fairness in comparison by using KerasRegressor as wrappers in the experiments among the three methods.

In addition, we provide a fair comparison environment using the same set of parameters and values when comparing ML algorithms (Yang and Shami, 2020). The thesis does not evaluate all possible values for BO, grid search, and random search. The parameters in this thesis and their values were decided as a result of the evaluations made on the models. Because it is recommended to use the model by examining the changing behavior of the model by repeatedly trying to solve the problem with new values from the set of possible values (Asselman et al., 2021).

The parameters used in hyper-parameter optimization methods in this thesis are as follows.

While defining Keras layers for optimization methods, plain and same layers are used as much as possible. Below can be seen the sequential model of GRU, LSTM, and RNN with univariate time series.

Hparam Parameter

Many parameters can be used LSTM with Keras. Keras has two main models: sequential () and the model class used with the functional API. This thesis was used as a model, sequential ().To understand hyper-parameters' effect on learning, this section thoroughly assessed the batch size, dropout, activation function, optimizer, and learning rate parameters. The results of all combinations of parameters and their values are obtained and visualized using the hparam parameter feature of the Tensorboard. The hyper-parameters manual approach shows success that cannot be ignored. This section used the relatively small time series Bitcoin price as the manual approach.

Grid Search

Grid search is a traditional optimization technique. A given model performs a complete search of the subsets in the hyper-parameter space for the specified set of parameters and their specified values. Grid search is simple in execution, but determining which parameters and the

values to be given to these parameters from the hyper-parameter space requires expertise. Once the hyper-parameters are determined, it takes much time as it applies all the combinations for each value of the parameters (Yu and Zhu, 2020).

Random Search

Unlike grid search, which searches for all combinations one by one, Random search performs random searches in this combination space, catching or exceeding the grid search results. With the number of parameters increasing, time and resource savings are achieved using random search instead of the time and processor power expectation needed by grid search. Based on (Liashchynskyi and Liashchynskyi, 2019), they are used random search, grid search, and genetic algorithm.

Their comparative results are as follows;

- Grid search with 83% accuracy in approximately 4.3 hours,
- Random search with 86% accuracy in approximately 2.7 hours, and
- Genetic algorithm with 86% accuracy in approximately 4.13 hours.

As can be seen from this result, random search has reached a higher accuracy rate in a shorter time. If the search space were larger, the search time would increase too much for grid search.

Bayesian Optimization

BO is an optimization method used for problems that take a long time to evaluate. A surrogate function was created to solve the problem, and then the Gaussian process was applied to this surrogate function. It uses the acquisition function it defines from the surrogate function to decide which combinations to choose. Among all hyper-parameter combinations, the set of hyper-parameters evaluated with f(x) and whose error values take optimum values is X *.

$$x^* = \underset{x \varepsilon X}{\arg\min}\, f(x) \qquad \text{Equation 3.33}$$

$$\text{Prior point of } x_{t-1} \left(f\ (x_t) < \left(\text{best value of } f(x) \text{ until now}\right) - \varepsilon \right) \qquad \text{Equation 3.34}$$

GP is a Gauss process regression.

$$\mathrm{f}(x) \sim GP\left(mean(x), var\left(x, x^{'}\right)\right) \qquad \text{Equation 3.35}$$

Based on (Frazier, 2018), Bayes optimization is the most suitable method for optimizing areas with more than 20 dimensions. The all-experiment surrogate function is GP with BayesSearchCV class of skopt, and the parameter is the base estimator. The different estimators available for BO are; ET, RF, and GBRT. GP uses a probabilistic regression model, RF and GBRT models use an ensemble learning strategy. Although they are similar, there are minor differences between RF and GBRT. The difference between GBRT and RF is that each tree in RF uses a parallel estimator called bagging. In contrast, GBRT grows trees sequentially, and each tree receives information from ancestral trees (Yang et al., 2022).

B is a set of regression trees in the GBRT,

s are the regression values from B tree,

$|B|$ is number of regression trees,

$$mean(x) = \frac{1}{|\mathcal{B}|}\sum_{s\in\mathcal{B}} s(\mathbf{x}) \qquad \text{Equation 3.36}$$

$$var(x)^2 = \frac{1}{|\mathcal{B}|-1}\sum_{s\in\mathcal{B}} (s(\mathbf{x}) - mean(x))^2 \qquad \text{Equation 3.37}$$

When we use RF as the surrogate function for BO, the estimation result produces a single result that falls within the training sample boundaries, rather than a probability distribution. To combine the estimates from the RF, we consider the estimated variances from B different trees as var_b and the mean as $mean_b$ (Jeroen. and Vanschoren, 2021).

$$mean(x) = \frac{1}{B}\sum_{B=1}^{B} mean(x)_b \qquad \text{Equation 3.38}$$

$$var(x)^2 = \frac{1}{B}\left(\sum_{B=1}^{B} var(x)_b{}^2 + mean(x)_b{}^2\right) - mean(x)^2 \qquad \text{Equation 3.39}$$

ET can be used instead of RF to reduce variance, although this can cause bias. This helps to choose the best one from the set of thresholds for the most distinctive point, providing more randomness with ET modelling for the splitting rule. With ET, we train a set of different decision trees with randomly selected features, while RF trains trees with bootstrap samples for each candidate partition based on a randomly selected subset. While RF uses the bagging procedure to iteratively generate sub-training sets, ET uses all the training examples to construct each tree with a varying number of parameters. Result of this, the number of ET leaves is larger than RF leaves (Geurts et al.).

$$mean(x) = \frac{1}{B}\sum_{B=1}^{B} mean(x)_b \qquad \text{Equation 3.40}$$

The different estimators for BO are GP, ET, RF, and GBRT were used in the experiments and their effects on the results are section 4.9.4. Bayesian Optimization.

4. RESULTS AND DISCUSSIONS

MSE, RMSE, MAE, and MAPE values of the test results obtained for each model, model parameters, and data sets are in the table under the model headings below. Each model used Bitcoin, Gold, crude oil, natural gas, Ethereum, and the dollar-euro parity datasets to predict Bitcoin price. Then, the two data sets that gave the best results which were combined and the prediction was repeated. Finally, the prediction of the combination of all data sets was added. These operations are done in seven ML models. All graphs for all models are plotted with MSE values. Obtained test results were evaluated in terms of each model, and the results obtained from all models will be evaluated in the conclusion section.

4.1. MLP

Table 4.1. Results of MLP experiments

Dataset	Activation	Solver	MSE	RMSE	MAE	MAPE
X1	tanh	sgd	0.100643	0.317242	0.285917	0.519764
		adam	**0.000512**	**0.022622**	**0.015445**	**0.028022**
	relu	sgd	0.284869	0.533732	0.470908	0.832931
		adam	0.015512	0.124549	0.098928	0.155063
	logistic	sgd	0.249535	0.499535	0.450493	0.818814
		adam	0.070077	0.264720	0.238350	0.433069
X2	tanh	sgd	0.245655	0.495636	0.436224	0.764451
		adam	**0.000583**	**0.024151**	**0.016878**	**0.030502**
	relu	sgd	0.304586	0.551893	0.495346	0.892092
		adam	0.005424	0.073649	0.058978	0.097078
	logistic	sgd	0.253071	0.503062	0.454821	0.829894
		adam	0.082314	0.286905	0.248561	0.426701
X3	tanh	sgd	0.214543	0.463188	0.417889	0.759472
		adam	**0.000564**	**0.023755**	**0.016335**	**0.029291**
	relu	sgd	0.121833	0.349045	0.308657	0.553677
		adam	0.006862	0.082840	0.062976	0.098929
	logistic	sgd	0.266969	0.516691	0.466246	0.847967
		adam	0.008208	0.090598	0.064052	0.117229
X4	tanh	sgd	0.188982	0.434721	0.394251	0.720935
		adam	**0.000526**	**0.022940**	**0.015792**	**0.028570**
	relu	sgd	0.260243	0.510140	0.463640	0.849250
		adam	0.007276	0.085298	0.068289	0.111846
	logistic	sgd	0.266375	0.516115	0.464617	0.843024
		adam	0.095756	0.309444	0.278179	0.504642
X5	tanh	sgd	0.141467	0.376120	0.343016	0.634622
		adam	**0.000616**	**0.024828**	**0.017505**	**0.031548**
	relu	sgd	0.196693	0.443501	0.397142	0.715863
		adam	0.050651	0.225058	0.189396	0.316940
	logistic	sgd	0.260118	0.510017	0.459380	0.833800
		adam	0.059776	0.244492	0.221902	0.408045
X6	tanh	sgd	0.328746	0.573364	0.523490	0.967865
		adam	**0.000547**	**0.023383**	**0.016557**	**0.030673**
	relu	sgd	0.205461	0.453279	0.404745	0.727188
		adam	0.011310	0.106351	0.083517	0.131513
	logistic	sgd	0.248750	0.498748	0.449051	0.814646
		adam	0.004545	0.067419	0.060015	0.114894
X11	tanh	sgd	0.213996	0.462597	0.425336	0.792651
		adam	**0.000515**	**0.022685**	**0.015600**	**0.028287**
	relu	sgd	0.229136	0.478682	0.434735	0.796312
		adam	0.031945	0.178733	0.143620	0.227356
	logistic	sgd	0.269890	0.519510	0.469591	0.855313
		adam	0.004196	0.064780	0.045136	0.089055
X17	tanh	sgd	0.111968	0.334617	0.296843	0.532265
		adam	**0.002034**	**0.045104**	**0.031724**	**0.056184**
	relu	sgd	0.222701	0.471912	0.414769	0.727953
		adam	0.033072	0.181858	0.140651	0.218439
	logistic	sgd	0.279088	0.528288	0.477754	0.871807
		adam	0.020598	0.143520	0.114779	0.193803

When Table 4.1 is interpreted as a result of the tests made with MLP, the best estimates are X1, X4, and X3, respectively. After X1 and X11, X3 and X4 have the best results, except for X17, all results are close and have acceptable values. When the Crude-oil and Ethereum datasets are together (X11), the results are better than separately. The activation function and solver from MLP parameters were evaluated in these tests. All best results without exception were found with the tanh and adam couple. There are enormous differences in the results achieved by this couple compared to the others.

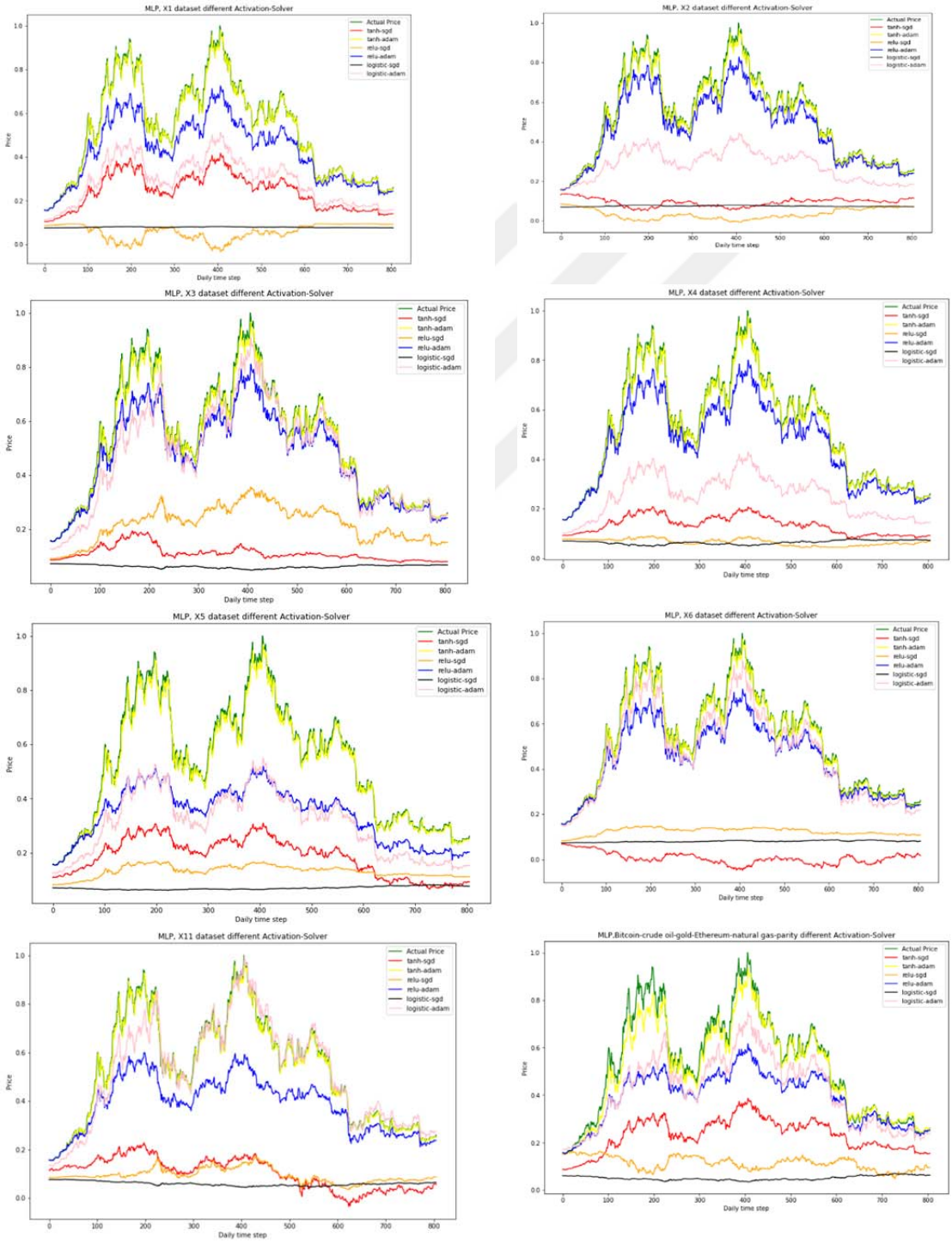


Figure 4.1. Visualized from the results of Table 4.1 (MSE)

4.2. SVM

Which kernel we use in SVR is essential because it allows us to find the hyperplane that best fits our data in higher dimensional space with the least cost. For example, linear, RBF, sigmoid, and poly are used below in Table 4.2 with datasets.

Table 4.2. Results of SVR experiments

Dataset	Kernel	MSE	RMSE	MAE	MAPE
X1	Linear	**0.000976**	**0.031237**	**0.026301**	**0.066896**
	RBF	0.003472	0.058921	0.048211	0.089939
	Sigmoid	0.006749	0.082150	0.064493	0.105727
	Poly	0.033106	0.181950	0.172398	0.376170
X2	Linear	**0.085592**	**0.292561**	**0.252063**	**0.428608**
	RBF	0.176499	0.420118	0.370212	0.649947
	Sigmoid	0.191976	0.438150	0.387366	0.683343
	Poly	1.893713	1.376123	1.085130	1.715022
X3	Linear	**0.074066**	**0.272150**	**0.233810**	**0.399076**
	RBF	0.167847	0.409692	0.362596	0.641485
	Sigmoid	0.188755	0.434460	0.385653	0.684897
	Poly	5.974187	2.444215	1.607239	2.260458
X4	Linear	0.086222	0.293635	0.252464	0.429853
	RBF	0.181679	0.426239	0.377193	0.667109
	Sigmoid	0.195465	0.442114	0.392126	0.695645
	Poly	**0.030858**	**0.175665**	**0.140184**	**0.246120**
X5	Linear	0.092487	0.304117	**0.268350**	**0.471804**
	RBF	0.185969	0.431242	0.385108	0.689003
	Sigmoid	0.201374	0.448747	0.400661	0.716725
	Poly	**0.091909**	**0.303165**	0.272288	0.512899
X6	Linear	**0.088571**	**0.297609**	**0.259118**	**0.446926**
	RBF	0.180479	0.424829	0.376478	0.666621
	Sigmoid	0.197920	0.444882	0.395132	0.701953
	Poly	1.003797	1.001897	0.729414	1.222657
X11	Linear	0.073807	0.271674	0.231545	0.392484
	RBF	0.167528	0.409302	0.361509	0.638263
	Sigmoid	0.186458	0.431807	0.382650	0.678348
	Poly	**0.020601**	**0.143529**	**0.112038**	**0.216219**
X17	Linear	0.075022	0.273901	0.232779	0.392555
	RBF	0.164314	0.405357	0.357072	0.627432
	Sigmoid	0.182601	0.427318	0.377808	0.667055
	Poly	**0.029910**	**0.172946**	**0.149401**	**0.271501**

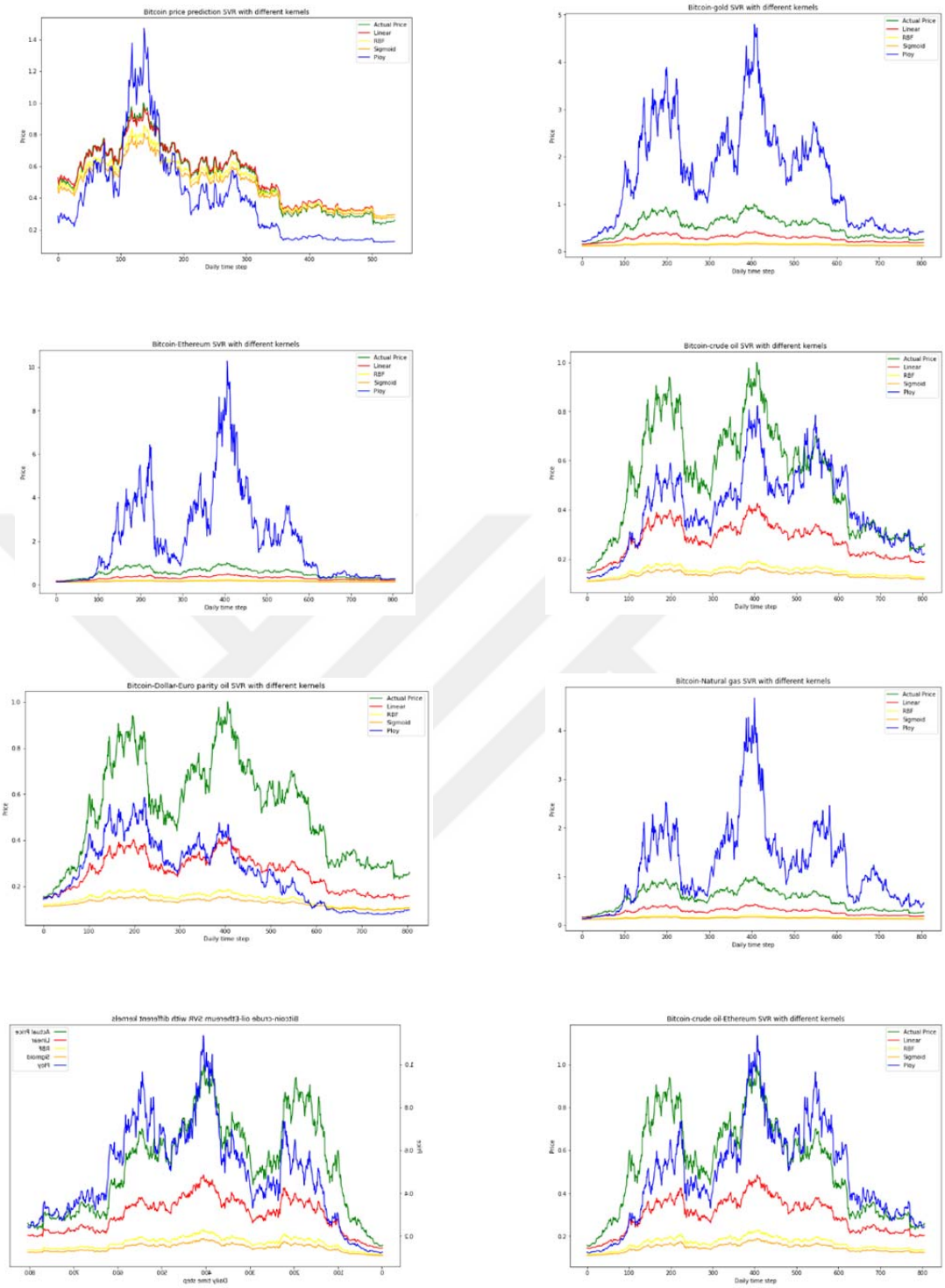


Figure 4.2. Visualized from the results of Table 4.2 (MSE)

When Table 4.2 is interpreted as a result of the tests made with SVM, the best estimates are X1, X4, and X3, respectively. The Crude-oil (X4) estimation was second but about six times worse than X1 regarding MAPE values. Estimates made with Ethereum (X3) are about 1.6 times worse than crude oil (X4). As a result of the tests performed with the X11 dataset consisting of X4 and X3, an improvement of approximately 0.9 was observed. It is seen that X17 dataset results, where

all the data sets are included, deteriorate compared to X1 and X11 and are better than X2, X3, X4, X5, and X6. Linear and poly kernel results are reasonable compared to other kernels. However, only the result of the linear kernel with X1 is acceptable.

4.3. GRNN

PyGRNN library was used for GRNN. ANNs are based on parametric regression, while GRNN is based on non-parametric regression. The nonlinear relationships of the features in the X17 dataset were found with GRNN for feature selection in Figure 4.3. All experiments in GRNN are optimized with a 'warm start'.

Figure 4.3. All feature space, the relatedness matrix

Table 4.3. Results of GRNN experiments

Dataset	MSE	RMSE	MAE	MAPE
X1	0.000358	0.018919	0.013902	0.038798
X2	0.000384	0.019585	0.015156	0.042689
X3	0.000358	0.018920	0.013903	0.038800
X4	0.000373	0.019310	0.014680	0.041060
X5	0.000537	0.023183	0.017434	0.048663
X6	**0.000347**	**0.018638**	**0.013778**	**0.038527**
X13	0.000355	0.018838	0.013867	0.038729
X17	0.019954	0.141260	0.127335	0.401130

When Table 4.3 is interpreted as a result of the tests made with GRNN, it is seen best estimates X6, X1, and X3, respectively. After X6 and X13, X1 and X3 have the best results, except for X17, all results are close and have acceptable values. When the natural gas and Ethereum datasets are combined X13, the results deteriorate somewhat compared to the natural gas results.

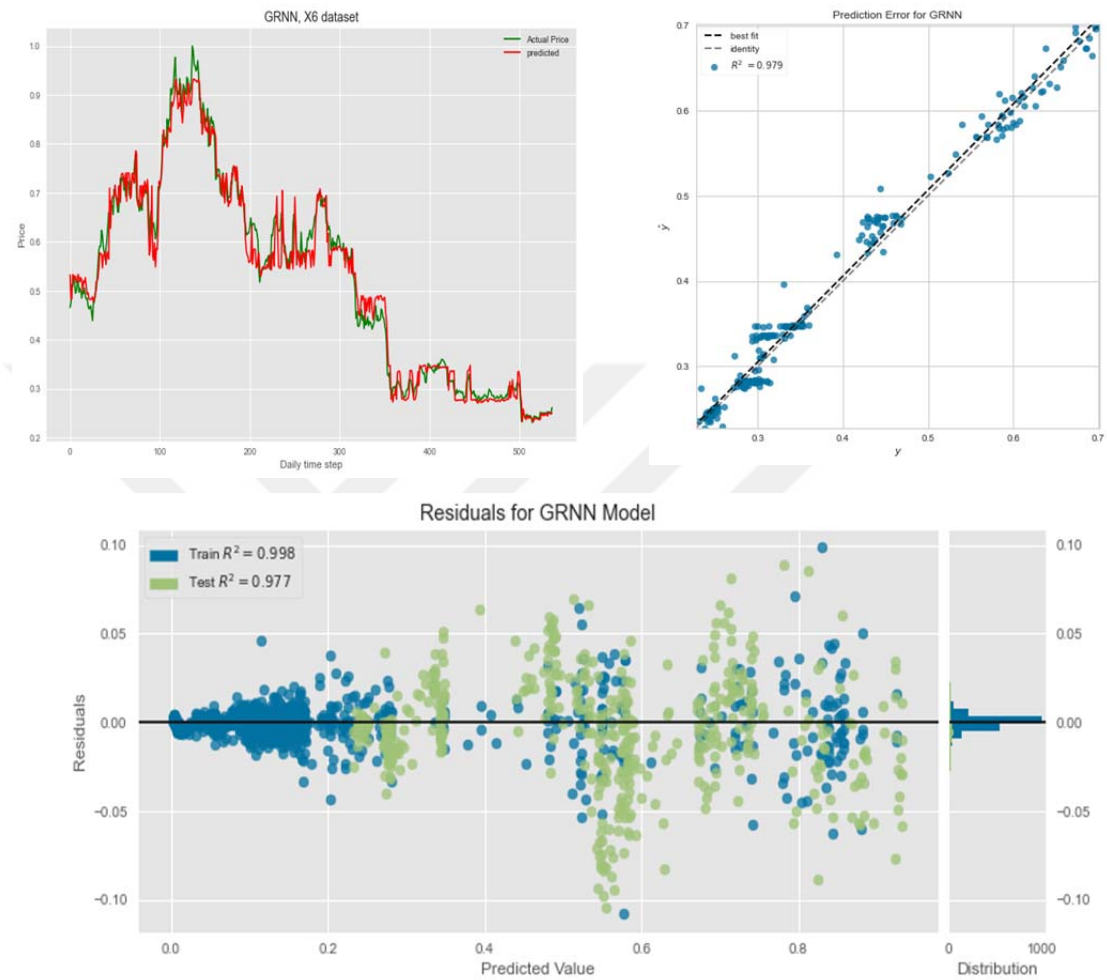


Figure 4.4. GRNN predicted-actual price, prediction error, residuals

The results for X6, which consists of natural gas and Bitcoin, where GRNN has the best results, are shown in Figure 4.4

4.4. RNN

Tensorflow's Keras library was used for RNN experiments. There are three options for recurrent neural networks in Keras: SimpleRNN, LSTM, and GRU. Only SimpleRNN experiments were performed under this title.

Table 4.4. Results of RNN experiments

Dataset	Neurons	Act	Opt	MSE	RMSE	MAE	MAPE
X1	64	tanh	sgd	0.073137	0.270438	0.240464	0.428805
			adam	0.000727	0.026956	0.018597	0.031758
		relu	sgd	0.085413	0.292256	0.257189	0.451642
			adam	0.000521	0.022824	0.015600	0.028176
	128	tanh	sgd	0.071908	0.268156	0.241393	0.438468
			adam	**0.000499**	**0.022343**	**0.015227**	**0.027981**
		relu	sgd	0.153377	0.391634	0.344768	0.605592
			adam	0.000506	0.022501	0.015316	0.028242
X2	64	tanh	sgd	0.016899	0.129996	0.109108	0.182453
			adam	0.000646	0.025417	0.018096	0.033529
		relu	sgd	0.144569	0.380223	0.322105	0.538527
			adam	0.000672	0.025924	0.018650	0.034221
	128	tanh	sgd	0.091422	0.302360	0.258682	0.438319
			adam	**0.000639**	**0.025269**	**0.017658**	**0.031094**
		relu	sgd	0.132754	0.364354	0.309520	0.519905
			adam	0.001042	0.032282	0.026845	0.056994
X3	64	tanh	sgd	0.003797	0.061619	0.041483	0.076141
			adam	0.001915	0.043759	0.030460	0.047465
		relu	sgd	0.075363	0.274524	0.237852	0.411959
			adam	0.002271	0.047659	0.038631	0.067914
	128	tanh	sgd	0.013470	0.116061	0.095347	0.171651
			adam	**0.000507**	**0.022523**	**0.015572**	**0.028369**
		relu	sgd	0.137600	0.370944	0.320071	0.547702
			adam	0.004225	0.065003	0.045567	0.069887
X4	64	tanh	sgd	0.006367	0.079793	0.070288	0.129099
			adam	0.001597	0.039965	0.031378	0.055012
		relu	sgd	0.174257	0.174257	0.357794	0.611904
			adam	0.021118	0.145320	0.108772	0.162363
	128	tanh	sgd	0.046061	0.214618	0.189538	0.338870
			adam	**0.000519**	**0.022774**	**0.015678**	**0.028499**
		relu	sgd	0.155886	0.394824	0.348303	0.615553
			adam	0.016268	0.127546	0.094659	0.141043

X5	64	tanh	sgd	0.030659	0.175098	0.158967	0.295783
			adam	**0.000579**	**0.024064**	**0.016599**	**0.029881**
		relu	sgd	0.198756	0.445821	0.399489	0.721031
			adam	0.047270	0.217417	0.188619	0.338570
	128	tanh	sgd	0.047317	0.217525	0.197965	0.366448
			adam	0.000875	0.029582	0.021570	0.038014
		relu	sgd	0.154859	0.393522	0.353527	0.641313
			adam	0.038533	0.196298	0.170093	0.307550
X6	64	tanh	sgd	0.002695	0.051916	0.042425	0.078905
			adam	0.003702	0.060848	0.049057	0.084329
		relu	sgd	0.154413	0.392954	0.356558	0.659173
			adam	0.001063	0.032597	0.023378	0.041030
	128	tanh	sgd	0.050836	0.225468	0.210435	0.409600
			adam	**0.000702**	**0.026492**	**0.018556**	**0.032145**
		relu	sgd	0.167354	0.409089	0.359867	0.633999
			adam	0.001066	0.032656	0.024651	0.047565
X11	64	tanh	sgd	0.001600	0.039995	0.030456	0.058524
			adam	0.001222	0.034962	0.024256	0.391475
		relu	sgd	0.170253	0.412617	0.362058	0.636651
			adam	0.082703	0.287581	0.225384	0.343367
	128	tanh	sgd	0.006266	0.079155	0.054832	0.100723
			adam	**0.000658**	**0.025660**	**0.017767**	**0.030736**
		relu	sgd	0.122604	0.350149	0.301364	0.520297
			adam	0.059925	0.244795	0.188433	0.283416
X17	64	tanh	sgd	0.008524	0.092323	0.074256	0.131506
			adam	0.009786	0.098922	0.079188	0.130007
		relu	sgd	0.122033	0.349333	0.299539	0.512484
			adam	0.211080	0.459435	0.402652	0.705851
	128	tanh	sgd	0.011882	0.109004	0.081660	0.137030
			adam	**0.005794**	**0.076118**	**0.058801**	**0.094987**
		relu	sgd	0.161697	0.402116	0.356825	0.641701
			adam	0.200821	0.448131	0.380268	0.633496

When Table 4.4 is interpreted as a result of the tests made with RNN, the best estimates are X1, X3, X4, X5, X2, and X11, respectively. Except for X17, all results are close to each other and have acceptable values. When the Crude-oil and Ethereum datasets are together (X11), there was a slight deterioration in the results than separately. Neurons (64 and 28), Activations (tanh, relu), and

Optimizers (Adam, Sgd) were evaluated in these tests. All best results without exception were found with the tanh and adam couple. All results except X5, neuron=128, have good results. All database results with Adam, tanh, and 128 are visualized below in Figure 4.5.

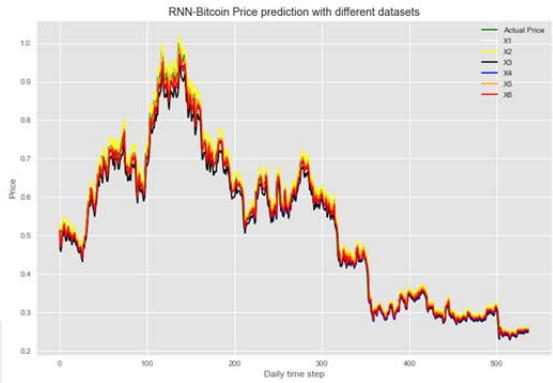


Figure 4.5. RNN best result of (MSE) each dataset in Table 4.4

4.5. LSTM

In this thesis, layers and models features of Tensorflow and Keras library were used for LSTM experiments.

Table 4.5. Results of LSTM experiments

Dataset	Neurons	Act	Opt	MSE	RMSE	MAE	MAPE
X1	64	tanh	sgd	0.088767	0.297937	0.264901	0.471651
			adam	0.000394	0.019857	0.013914	0.026678
		relu	sgd	0.144025	0.379507	0.337778	0.601877
			adam	**0.000380**	**0.019486**	**0.013338**	**0.024971**
	128	tanh	sgd	0.148086	0.384819	0.342476	0.610194
			adam	0.000413	0.020322	0.014150	0.025999
		relu	sgd	0.153936	0.392346	0.349159	0.622125
			adam	0.000391	0.019777	0.013929	0.026917
X2	64	tanh	sgd	0.048943	0.221231	0.172960	0.272824
			adam	0.000501	0.022385	0.015495	0.027589
		relu	sgd	0.058856	0.242602	0.193142	0.307830
			adam	0.000767	0.027692	0.022571	0.048143
	128	tanh	sgd	0.081686	0.285807	0.235768	0.386941
			adam	0.001102	0.033194	0.026498	0.048569
		relu	sgd	0.114572	0.338485	0.288854	0.491310
			adam	**0.000428**	**0.020693**	**0.014924**	**0.028655**
X3	64	tanh	sgd	0.014164	0.119013	0.099061	0.167296
			adam	**0.000444**	**0.021064**	**0.015070**	**0.028311**
		relu	sgd	0.053731	0.231799	0.200898	0.349024
			adam	0.000523	0.022875	0.016253	0.029597
	128	tanh	sgd	0.078793	0.280701	0.247764	0.437372
			adam	0.000476	0.021811	0.016589	0.034068
		relu	sgd	0.109103	0.330307	0.290770	0.512460
			adam	0.000683	0.026134	0.018359	0.031537
X4	64	tanh	sgd	0.086369	0.293886	0.241811	0.396340
			adam	0.000665	0.025790	0.019712	0.037244
		relu	sgd	0.130464	0.361199	0.311168	0.534777
			adam	0.000648	0.025462	0.019676	0.040110
	128	tanh	sgd	0.135266	0.367785	0.318062	0.548863
			adam	**0.000409**	**0.020221**	**0.014057**	**0.025872**
		relu	sgd	0.155578	0.394434	0.344891	0.602406
			adam	0.000505	0.022465	0.015630	0.028723

X5	64	tanh	sgd	0.127873	0.357593	0.343385	0.695345
			adam	0.000485	0.022028	0.015559	0.029390
		relu	sgd	0.154395	0.392932	0.367229	0.702767
			adam	0.154395	0.392932	0.367229	0.702767
	128	tanh	sgd	0.155568	0.394421	0.367944	0.701847
			adam	0.000650	0.025499	0.017912	0.030926
		relu	sgd	0.171230	0.413800	0.380194	0.708081
			adam	**0.000446**	**0.021128**	**0.015612**	**0.031531**
X6	64	tanh	sgd	0.089525	0.299207	0.235105	0.366289
			adam	0.002228	0.047198	0.039098	0.073651
		relu	sgd	0.153959	0.392377	0.337614	0.577631
			adam	0.000605	0.024590	0.018916	0.039747
	128	tanh	sgd	0.132003	0.363323	0.303886	0.503379
			adam	0.000607	0.024644	0.017336	0.030780
		relu	sgd	0.156704	0.395858	0.341434	0.585784
			adam	**0.000524**	**0.022887**	**0.016561**	**0.031513**
X11	64	tanh	sgd	0.017194	0.131126	0.099816	0.158864
			adam	0.001425	0.037751	0.027181	0.043922
		relu	sgd	0.070147	0.264852	0.222844	0.374146
			adam	0.001583	0.039783	0.031093	0.061755
	128	tanh	sgd	0.064000	0.252982	0.211715	0.352657
			adam	**0.000811**	**0.028486**	**0.020128**	**0.034273**
		relu	sgd	0.132869	0.364512	0.318846	0.557604
			adam	0.001082	0.032897	0.026046	0.053694
X17	64	tanh	sgd	0.013241	0.115069	0.094768	0.192318
			adam	**0.001747**	**0.041797**	**0.030125**	**0.054791**
		relu	sgd	0.038238	0.195544	0.154112	0.252719
			adam	0.008340	0.091326	0.064477	0.172802
	128	tanh	sgd	0.041485	0.203679	0.157777	0.251321
			adam	0.002108	0.045909	0.034984	0.066756
		relu	sgd	0.097731	0.312619	0.259589	0.429646
			adam	0.004292	0.065516	0.048528	0.116860

When Table 4.5 is interpreted as a result of the tests made with LSTM, the best estimates are X1, X4, X3, X2, X6, and X5, respectively. All results are close to each other and have acceptable values. Neurons (64 and 128), Activations (tanh, relu), and Optimizers (Adam, Sgd) were evaluated in these tests. Since the two best results after X1, X4 and X3, are evaluated together as X11 (the Crude-oil and Ethereum), there was a slight deterioration in the results than separately. All best results without exception were found with the adam. There are datasets where the activation functions tanh and relu give the best results for 64 and 128.

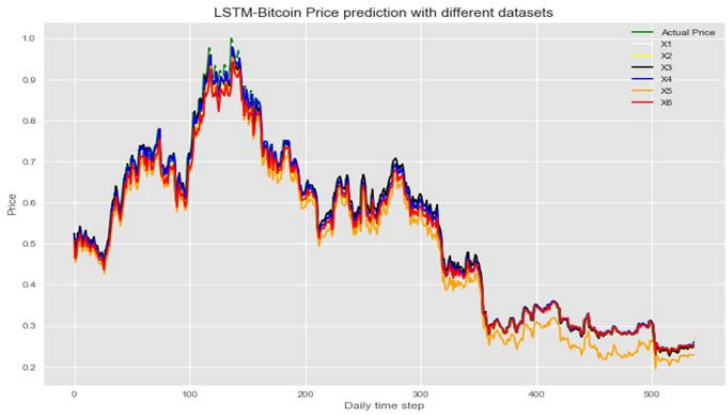


Figure 4.6. LSTM best result of (MSE) each dataset in Table 4.5

4.6. GRU

In this thesis, layers and model features of Tensorflow's Keras library were used for GRU experiments.

Table 4.6 .Results of GRU experiments

Dataset	Neurons	Act	Opt	MSE	RMSE	MAE	MAPE
X1	64	tanh	sgd	0.004316	0.065698	0.055206	0.095965
			adam	0.000545	0.023336	0.016988	0.031031
		relu	sgd	0.030605	0.174942	0.155426	0.277085
			adam	0.000450	0.021217	0.014832	**0.026762**
	128	tanh	sgd	0.045621	0.213592	0.189488	0.336888
			adam	**0.000411**	**0.020273**	**0.014396**	0.027120
		relu	sgd	0.106595	0.326489	0.290189	0.516398
			adam	0.000446	0.021120	0.015760	0.032529
X2	64	tanh	sgd	0.030863	0.175678	0.131923	0.212314
			adam	0.002509	0.050093	0.041090	0.072936
		relu	sgd	0.044778	0.211608	0.160713	0.252537
			adam	0.001150	0.033913	0.027526	0.051397
	128	tanh	sgd	0.038072	0.195120	0.147552	0.232703
			adam	0.000722	0.026871	0.020059	0.036148
		relu	sgd	0.061851	0.248698	0.194518	0.306402
			adam	**0.000452**	**0.021250**	**0.015926**	**0.032366**
X3	64	tanh	sgd	0.017825	0.133509	0.118518	0.228843
			adam	0.000504	0.022457	0.016648	0.032531
		relu	sgd	0.001899	0.043576	0.034784	0.074915
			adam	0.001983	0.044534	0.036062	0.064385
	128	tanh	sgd	0.001456	0.038159	0.030407	0.063557
			adam	**0.000464**	**0.021539**	**0.015923**	**0.031422**
		relu	sgd	0.013557	0.116433	0.092810	0.152372
			adam	0.000489	0.022113	0.015496	0.027668
X4	64	tanh	sgd	0.026015	0.161290	0.132557	0.243848
			adam	0.000440	0.020986	0.014591	**0.026492**
		relu	sgd	0.084145	0.290078	0.231666	0.368396
			adam	0.000674	0.025963	0.021140	0.043864
	128	tanh	sgd	0.072752	0.269725	0.212184	0.333019
			adam	0.000527	0.022962	0.016538	0.029796
		relu	sgd	0.126067	0.355059	0.302370	0.513226
			adam	**0.000407**	**0.020184**	**0.014576**	0.028745

X5	64	tanh	sgd	0.138327	0.371923	0.361601	0.791531
			adam	0.000663	0.025748	0.020055	0.040767
		relu	sgd	0.144822	0.380554	0.369279	0.776145
			adam	0.000654	0.025578	0.020083	0.045904
	128	tanh	sgd	0.141914	0.376715	0.364540	0.756189
			adam	**0.000398**	**0.019943**	**0.013771**	**0.025526**
		relu	sgd	0.157646	0.397047	0.378081	0.749928
			adam	0.000975	0.031218	0.025506	0.062744
X6	64	tanh	sgd	0.062340	0.249679	0.204149	0.545713
			adam	0.000829	0.028793	0.023539	0.050266
		relu	sgd	0.082921	0.287960	0.229694	0.371725
			adam	0.001508	0.038837	0.028098	0.047131
	128	tanh	sgd	0.073583	0.271262	0.222426	0.375985
			adam	0.000836	0.028912	0.022329	0.042342
		relu	sgd	0.113655	0.337128	0.269394	0.424576
			adam	**0.000631**	**0.025126**	**0.018061**	**0.032287**
X14	64	tanh	sgd	0.071305	0.267031	0.241294	0.458461
			adam	**0.000664**	**0.025759**	**0.018612**	**0.034258**
		relu	sgd	0.124539	0.352901	0.315251	0.571966
			adam	0.006816	0.082559	0.065897	0.176449
	128	tanh	sgd	0.125618	0.354426	0.316648	0.574078
			adam	0.001215	0.034863	0.026698	0.047756
		relu	sgd	0.155828	0.394751	0.350481	0.625995
			adam	0.000792	0.028139	0.023454	0.053247
X17	64	tanh	sgd	0.018670	0.136638	0.113502	0.224685
			adam	0.000753	0.027445	0.021288	0.041467
		relu	sgd	0.046502	0.215643	0.177314	0.308142
			adam	**0.000753**	**0.027442**	**0.020575**	**0.040999**
	128	tanh	sgd	0.050987	0.225804	0.181077	0.301715
			adam	0.002011	0.044845	0.036223	0.073675
		relu	sgd	0.119871	0.346224	0.287937	0.476544
			adam	0.001584	0.039805	0.032993	0.079020

When Table 4.6 is interpreted as a result of the tests made with GRU, the best estimates are X5, X4, X1, X3, X6, and X2, respectively. All results are close to each other and have acceptable values. Neurons (64 and 128), Activations (tanh, relu), and Optimizers (Adam, Sgd) were evaluated in these tests. Since the two best results, X5 and X4, are considered together as X14 (dollar-euro parity and the Crude-oil), there was a slight deterioration in the results than separately. All best results with neurons=128 except X14 and X17, and best results without exception were found with the adam.

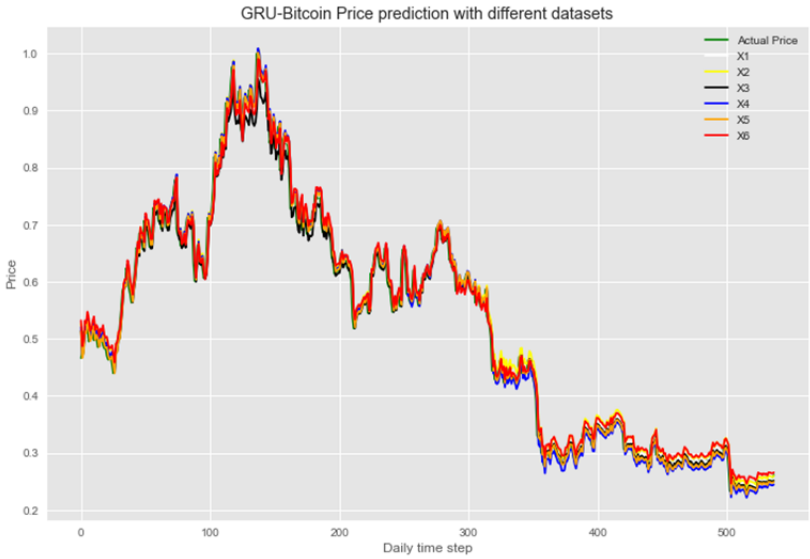


Figure 4.7. GRU best result of (MSE) each dataset in Table 4.6

4.7. CNN

Table 4.7. Results of CNN experiments

Dataset	Filter size	Act	Opt	MSE	RMSE	MAE	MAPE
X1	64	tanh	sgd	0.000425	0.020621	0.014705	0.028451
			adam	0.000912	0.030198	0.023546	0.045755
		relu	sgd	0.000688	0.026223	0.017162	0.030223
			adam	0.000392	0.019793	0.013973	0.026991
	128	tanh	sgd	0.000405	0.020128	0.014209	0.027279
			adam	0.000410	0.020257	0.014178	0.027137
		relu	sgd	0.000739	0.027186	0.017272	0.030143
			adam	**0.000388**	**0.019700**	**0.013603**	**0.025753**
X2	64	tanh	sgd	0.000465	0.021572	0.015655	0.030413
			adam	0.001206	0.034726	0.030570	0.071907
		relu	sgd	0.000809	0.028442	0.020246	0.037218
			adam	**0.000380**	**0.019501**	**0.013339**	**0.025092**
	128	tanh	sgd	0.000418	0.020438	0.014555	0.028048
			adam	0.000982	0.031342	0.027138	0.063079
		relu	sgd	0.000689	0.026244	0.019413	0.037053
			adam	0.000491	0.022156	0.016335	0.032865
X3	64	tanh	sgd	**0.000476**	**0.021815**	**0.016299**	**0.033062**
			adam	0.000952	0.030862	0.021722	0.037189
		relu	sgd	0.005138	0.071679	0.065805	0.141157
			adam	0.000853	0.029208	0.023751	0.050355
	128	tanh	sgd	0.003004	0.054806	0.048898	0.102320
			adam	0.002611	0.051096	0.043048	0.079152
		relu	sgd	0.001447	0.038036	0.033600	0.078235
			adam	0.000741	0.027229	0.022001	0.048062
X4	64	tanh	sgd	0.000549	0.023438	0.018098	0.037962
			adam	0.000405	0.020122	0.014071	0.026383
		relu	sgd	0.000790	0.028102	0.023164	0.052282
			adam	0.000880	0.029669	0.025081	0.058955
	128	tanh	sgd	0.000482	0.021954	0.016523	0.034022
			adam	0.005400	0.073483	0.067112	0.170086
		relu	sgd	0.000392	0.019798	0.013617	0.025710
			adam	**0.000382**	**0.019552**	**0.013564**	**0.025514**

X5	64	tanh	sgd	0.000430	0.020742	**0.014768**	**0.028611**
			adam	0.000540	0.023236	0.017969	0.036822
		relu	sgd	0.001583	0.039783	0.029133	0.054588
			adam	0.000538	0.023196	0.015782	0.028624
	128	tanh	sgd	**0.000427**	**0.020656**	0.014971	0.029869
			adam	0.000600	0.024485	0.019102	0.039230
		relu	sgd	0.002396	0.048948	0.040213	0.083878
			adam	0.000586	0.024201	0.017433	0.033029
X6	64	tanh	sgd	0.000560	0.023657	0.018023	0.035846
			adam	0.000835	0.028889	0.024092	0.054667
		relu	sgd	0.004531	0.067315	0.059872	0.136951
			adam	0.000651	0.025523	0.020022	0.041448
	128	tanh	sgd	0.000574	0.023965	0.019146	0.043170
			adam	0.000949	0.030806	0.022577	0.038751
		relu	sgd	**0.000526**	**0.022940**	**0.016743**	**0.032417**
			adam	0.002894	0.053792	0.045432	0.118705
X8	64	tanh	sgd	0.000637	0.025245	0.020421	0.045854
			adam	**0.000429**	**0.020705**	**0.014996**	0.029478
		relu	sgd	0.000640	0.025293	0.020259	0.044282
			adam	0.000589	0.024278	0.017940	0.033928
	128	tanh	sgd	0.000463	0.021516	0.016331	0.034391
			adam	0.000449	0.021188	0.015023	**0.028162**
		relu	sgd	0.000796	0.028215	0.019721	0.037548
			adam	0.003228	0.056816	0.049997	0.118344
X17	64	tanh	sgd	0.008488	0.092132	0.086013	0.205426
			adam	**0.000562**	**0.023706**	**0.017809**	**0.037489**
		relu	sgd	0.001569	0.039613	0.030369	0.061486
			adam	0.002210	0.047013	0.038687	0.072096
	128	tanh	sgd	0.003357	0.057937	0.051143	0.128609
			adam	0.000932	0.030524	0.023921	0.051292
		relu	sgd	0.000753	0.027444	0.021873	0.045441
			adam	0.001431	0.037828	0.027946	0.052601

When Table 4.7 is interpreted as a result of the tests made with CNN, the best estimates are X2, X4, X1, X5, X6, and X3, respectively. All results are close to each other and have acceptable values. Neurons (64 and 128), Activations (tanh, relu), and Optimizers (Adam, Sgd) were evaluated in these tests. Since the two best results, X2 and X4, are considered together as X8 (gold and the Crude-oil), there was a slight deterioration in the results than separately.

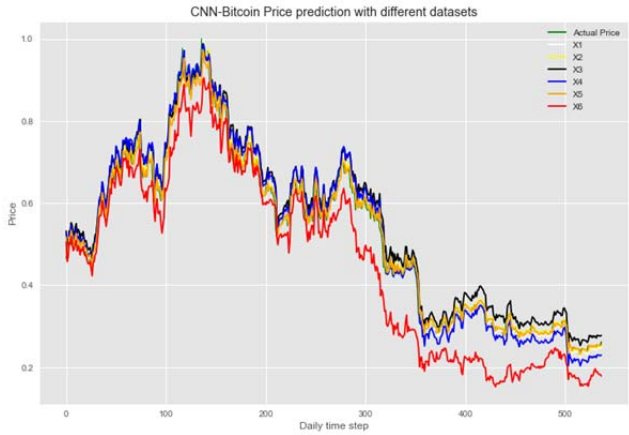


Figure 4.8. CNN best result of (MSE) each dataset in Table 4.7

4.8. The Effect of Parameters on Methods

Seven different ML and DL algorithms were examined in this thesis, and those with similar parameters were included in the same group and examined together. There are five different parameter types SVM, MLP, GRNN, RNN, and CNN. Since RNN, LSTM, and GRU parameters are the same in the RNN header, they were examined and interpreted together.

4.8.1 Parameter of RNN-LSTM-GRU

Recently, there have been many developments that will improve neural network performance. Increased GPU performance and adaptability to deep learning tasks have produced substantial productivity gains. In addition, efficiency improvements will continue to grow in importance as machine learning algorithms are optimized (Pomerat et al., 2019).

The Keras library includes RNN, LSTM, and GRU neural networks as recurrent layers. They are working with similar parameters, so they were examined under the same heading in terms of parameters. The parameter of RNN, LSTM, and GRU are activation function, go-backward, optimizer, batch size, learning rate, and epochs are evaluated separately below.

Activation Function: The activation function is used in artificial neural networks and converts the input signal to the output signal and feeds the following layers as input. As a result of applying activation to the weighted sums of the inputs in each neuron, feed the next layer. This section examined four different activation functions: tanh, relu, sigmoid, and softmax (Sharma et al., 2017).

tanh $$f(x)=\frac{2}{1+e^{-2x}}-1$$ Equation 4.1

relu $$f(x)=\max(0,x)$$ Equation 4.2

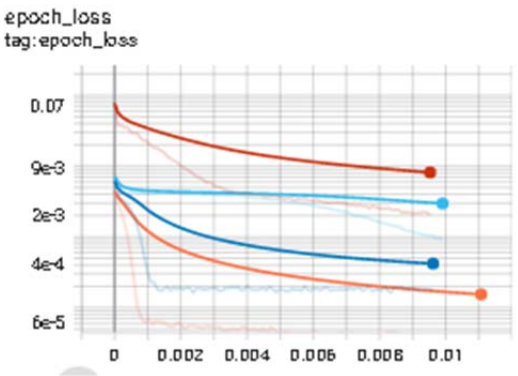
Sigmoid $$f(x)=\frac{1}{1+e^{-x}}$$ Equation 4.3

Softmax $$f_i(x) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$ Equation 4.4

Table 4.8. Comparing the activation function with Models

Models	Activation function	MSE	RMSE	MAE	MAPE
RNN	tanh	0.000488	0.022089	0.016044	0.031039
	relu	**0.000417**	**0.020410**	**0.014218**	**0.026976**
	sigmoid	0.000693	0.026325	0.021891	0.046657
	softmax	0.000569	0.023858	0.017166	0.031868
LSTM	tanh	**0.000424**	**0.020601**	**0.014364**	**0.026210**
	relu	0.000439	0.020963	0.015291	0.030562
	sigmoid	0.000616	0.024817	0.019836	0.042401
	softmax	0.000734	0.027101	0.019761	0.035890
GRU	tanh	0.000466	0.021588	0.015905	0.031748
	relu	0.000470	0.021690	0.016175	0.032815
	sigmoid	**0.000388**	**0.019710**	**0.013615**	**0.025793**
	softmax	0.000532	0.023075	0.016578	0.030814

The tests were performed with batch size=128, neuron=128, optimizer=Adam, and Lr=0.01.



Brown=sigmoid

Blue= softmax

Orange= tanh

Darkblue=relu

Figure 4.9. Loss of different activation functions

When Table 4.8 is interpreted, each RNN, LSTM, and GRU has reached the best results with a different activation function. All three achieved acceptable and close results. It achieved the best results with GRU sigmoid. According to reference (Pomerat et al., 2019), sigmoid performed better than relu and tanh.

In Figure 4.10, losses of different activation functions are shown in the experiment for LSTM. Since the less the loss, the better the result, it is seen that the activation function that best fits the data set is 'tanh'.

"go-backward" Parameter: As a result of the tests performed to evaluate the go-backwards parameter for RNN, LSTM, and GRU algorithms, the 'False' value was better than the 'True' value, while all other parameters were constant, as can be seen in Table 4.9. Based on reference (Singh et al., 2022), they found effective results with an efficient parallelized calculation of backward gains in NLP applications, which is the state corresponding to go-backward parameters in neural networks. As a result, if the go-backward parameter is suitable for the problem, it can improve performance.

Table 4.9. The effects of the go-backward on RNN, LSTM, and GRU (neuron=128, batch size=128, epoch=100, and Lr=0.01)

Methods	Go-backward	Act	Opt	MSE	RMSE	MAE	MAPE
simple RNN	False	tanh	Adam	0.000388	0.019706	0.013590	0.025246
	True	tanh	Adam	0.000438	0.020939	0.014657	0.026584
LSTM	False	tanh	Adam	0.000418	0.020435	0.014317	0.026334
	True	tanh	Adam	0.000448	0.021174	0.015454	0.030691
GRU	False	tanh	Adam	0.000383	0.019572	0.013455	0.025105
	True	tanh	Adam	0.000423	0.020565	0.014435	0.027569

Optimizers: It is an important parameter affecting the optimizer model's compiling process in neural networks. Adagrad, SGD, Adam, Ftrl, Adamax, RMSprop, Nadam, and Adadelta are optimizers. Stochastic gradient descent (SGD), where it is difficult to adjust the learning rate in deep neural networks, is among the most popular training methods. To overcome this difficulty were developed different variants of the optimizer, such as Adadelta (Zeiler, 2012), Adam (Kingma and Ba, 2015), Adagrad (Duchi et al., 2011), and RMSprop (Hinton et al., 2012) derived from SGD. Those uses for different types of problems.

Table 4.10 .Optimizer and activation function results

Optimizers	Activation function	MSE
Adamax	relu	0.000381
Adagrad	tanh	0.000385
adam	relu	0.000391
Adamax	tanh	0.000393
rmsprop	relu	0.000396
Adagrad	relu	0.000399
adam	tanh	0.000401
rmsprop	tanh	0.000406
rmsprop	tanh	0.000417

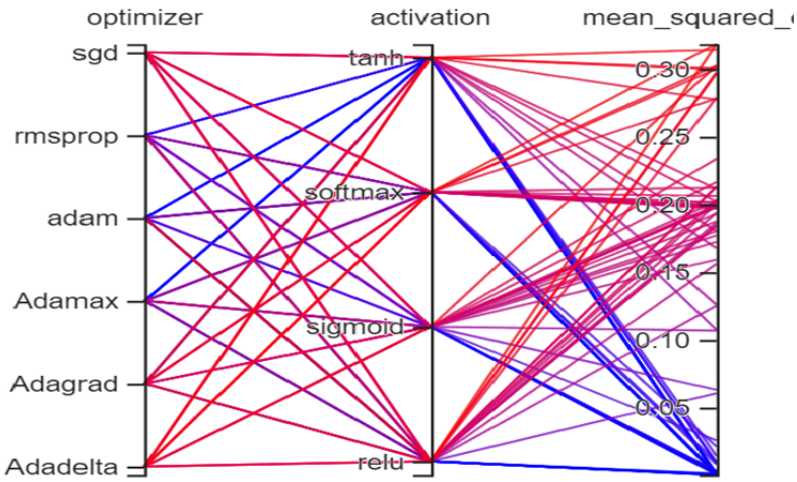


Figure 4.10. Tensorboard parallel coordinate view LSTM in Table 4.10

Looking at Figure 4.10 and Table 4.10, the best optimizers corresponding to the activation functions relu and tanh, giving the best results, are Rmsprop, Adam, Adagrad, and Adamax with blue-ranked arrows.

Learning Rate (Lr): Choosing the correct learning rate can significantly affect generalization accuracy and training speed. Neural networks determine how much the weights can change in response to an error observed during the training period in the training dataset by the value of the learning rate parameter (Wilson and Martinez, 2001).

Table 4.11. The effect of the learning rate (Activation=tanh, batch size=128, neuron=128)

Model	Optimizer	Lr	MSE	RMSE	MAE	MAPE
RNN	Rmsprop	0.1	0.016778	0.129529	0.115756	0.209804
		0.05	0.007685	0.087663	0.080915	0.204880
		0.001	0.000513	0.022645	0.017139	0.034864
		0.0001	**0.000396**	**0.019893**	**0.013921**	**0.026779**
	Adamax	0.1	0.000397	0.019930	0.013784	0.025435
		0.05	0.000394	0.019841	0.013819	0.026438
		0.001	0.000380	0.019504	0.013389	0.025144
		0.0001	**0.000381**	**0.019523**	**0.013403**	**0.025112**
	Adam	0.1	0.000612	0.024736	0.019010	0.037969
		0.05	0.001323	0.036375	0.030577	0.061105
		0.001	0.000380	0.019506	0.013402	0.025241
		0.0001	**0.000380**	**0.019491**	**0.013369**	**0.025117**
LSTM	Rmsprop	0.1	0.006897	0.083048	0.073517	0.179949
		0.05	0.003852	0.062063	0.054564	0.106817
		0.001	**0.001155**	**0.033986**	**0.029518**	**0.065547**
		0.0001	0.065999	0.256903	0.228539	0.407339
	Adamax	0.1	0.000394	0.019848	0.013975	0.027109
		0.05	0.000384	0.019603	0.013453	0.025251
		0.001	**0.000380**	**0.019497**	**0.013369**	**0.025055**
		0.0001	0.110814	0.332887	0.296252	0.527911
	Adam	0.1	0.000404	0.020110	0.013914	0.025627
		0.05	0.000387	0.019674	0.013674	0.026172
		0.001	**0.000380**	**0.019495**	**0.013371**	**0.025083**
		0.0001	0.100432	0.316910	0.281850	0.501889
GRU	**Rmsprop**	0.1	0.005210	0.072178	0.068253	0.144570
		0.05	0.001104	0.033221	0.028279	0.069826
		0.001	0.000394	0.019840	0.013877	0.026167
		0.0001	**0.000386**	**0.019638**	**0.013439**	**0.025070**
	Adamax	0.1	0.000394	0.019851	0.013874	0.026708
		0.05	0.000451	0.021237	0.014859	0.026841
		0.001	**0.000389**	**0.019719**	**0.013622**	**0.025534**
		0.0001	0.001396	0.037364	0.030711	0.056126
	Adam	0.1	0.000898	0.029970	0.022989	0.043170
		0.05	0.000423	0.020567	0.014485	0.027717
		0.001	**0.000382**	**0.019543**	**0.013419**	**0.025193**
		0.0001	0.000390	0.019742	0.013599	0.025439

RNN obtained approximately the same values with all three optimizers. When analyzed regarding the RNN learning rate, it reached the best results with Lr=0.0001. Adam and Adamax, when the very small and large values of Lr are evaluated in terms of results, very large deviations are not observed. On the contrary, results close to each other are obtained. Rmsprop had poor results at large Lr with RNN, LSTM, and GRU. However, as Lr gets smaller, it is seen that it closes the values obtained by other optimizers.

When the neural networks RNN, LSTM and GRU are compared between them, LSTM outperformed the other two. For LSTM, Adamax and Adam were close to each other and achieved the best result. However, for Rmsprop, the results other than Lr=0.001 are bad. For all activation functions for GRU and LSTM, the best result was obtained with Lr=0.001. RNN obtained the best prediction value in all three activation functions with an Lr=0.0001 value.

Batch Size: Batch is the slice size at which the dataset is sliced for processing during the learning process. Each batch of data is a collection of independently processed data in parallel. Increasing the batch size learns from more data at each step while the time and memory requirement for the process grows. Since the data set used in this thesis is time series, shuffle=False feature is used. They concluded that the larger the batch size, the better the performance (Golmant et al., 2018).

Table 4.12. The effect of the batch size on performance

Model	Batch size	MSE	RMSE	MAE	MAPE
RNN	32	0.003201	0.056581	0.050357	0.096757
	64	0.000676	0.025991	0.019227	0.034605
	128	0.000558	0.023630	0.017665	0.034671
	256	**0.000395**	**0.019875**	**0.013930**	**0.026264**
LSTM	32	0.000393	0.019817	0.013715	0.025819
	64	0.000455	0.021326	0.014925	0.026969
	128	0.000437	0.020912	0.014962	0.029100
	256	**0.000388**	**0.019702**	**0.013597**	**0.025280**
GRU	32	0.000480	0.021907	0.016358	0.032902
	64	0.000459	0.021430	0.015581	0.030572
	128	0.000450	0.021217	0.015159	0.029247
	256	**0.000389**	**0.019722**	**0.013622**	**0.025825**

In Table 4.12, the values of neuron=128, activation= tanh, optimizer=adam, and Lr=0.001 are fixed, and the variable tested value is batch size. As a result of the tests performed for the 32, 64, 128, and 256 values of the variable batch size, the performance values improved as the batch size increased for all three methods. When the performance is evaluated in terms of models, the results of all three methods are very close to each other, and GRU and RNN follow the best LSTM.

Dropout: In standard neural networks, all units change and adjust their parameters individually to reduce loss, and as a result, the final loss is reduced. But the problem is that there is no common adaptation, leading to overfitting. Dropout is random neurons are removed during the training of a neural network to drop incoming and outgoing edges to neurons (Srivastava, 2013). These networks, thinned during training, are combined to form an approximate model mean when applied to the test dataset. Thanks to this approximate model, the problem of overfitting is essentially eliminated. Dropout prevents neural networks from adjusting weights and biases to overfit the dataset, improving the neural network's generalization feature, helping it to perform better when new data is added.

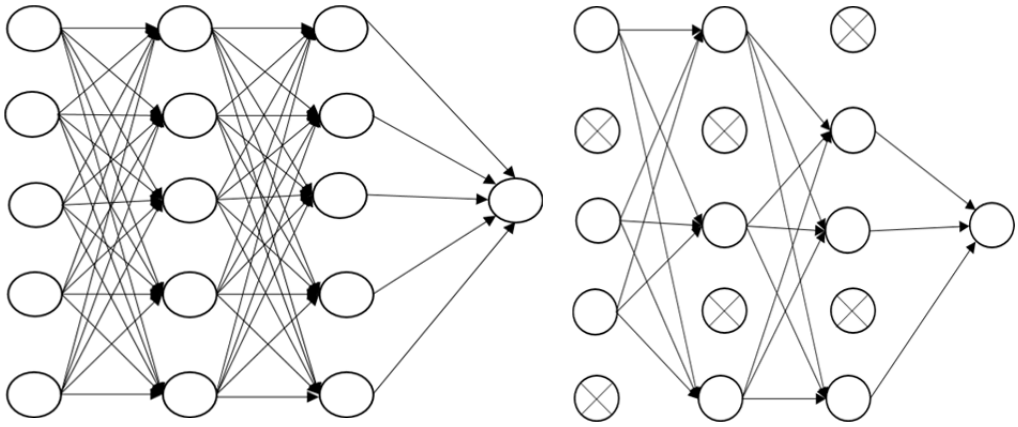


Figure 4.11. Conventional neural network and Neural network with dropout (Srivastava et al., 2014)

In Table 4.13, the values of neuron=128, two layer, activation= tanh, optimizer=adam, batch-size=128 and Lr=0.001 are fixed, and the variable tested value is a dropout.

Table 4.13. The effect of the dropout on the performance test set

Model	Dropout	MSE	RMSE	MAE	MAPE
RNN	**-**	**0.000386**	**0.019652**	**0.013498**	**0.025396**
	0.1	0.000591	0.024301	0.018129	0.033524
	0.2	0.000492	0.022190	0.016007	0.030653
	0.3	0.000396	0.019911	0.013831	0.026344
LSTM	**-**	**0.000382**	**0.019549**	**0.013411**	**0.025006**
	0.1	0.000518	0.022755	0.016006	0.028333
	0.2	0.000391	0.019770	0.013879	0.026692
	0.3	0.000423	0.020564	0.014310	0.026221
GRU	**-**	**0.000395**	**0.019875**	**0.013885**	**0.026119**
	0.1	0.000467	0.021612	0.015770	0.031027
	0.2	0.000421	0.020526	0.014331	0.026266
	0.3	0.000433	0.020797	0.014495	0.026302

When Table 4.13 is interpreted, the effect of dropout on performance is not very big, but the results are very close to each other. In a data set of 2686 rows, not applying drop gave better results in all three models.

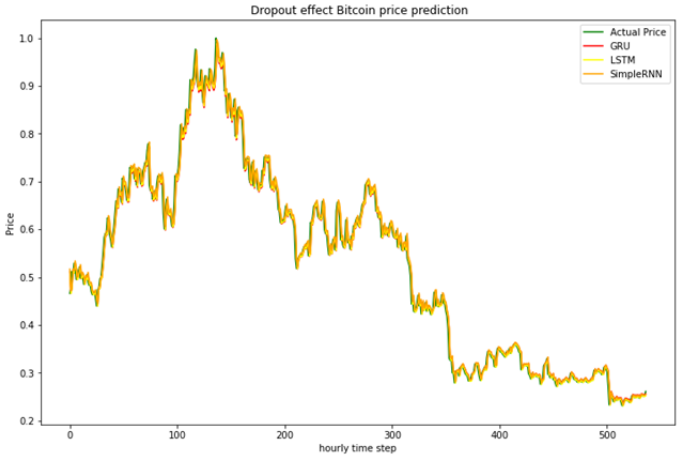


Figure 4.12. Dropout effect on RNN, LSTM, and GRU

Epoch: Epoch shows how many times a dataset is given to the network to train the network. In deep learning algorithms, we need to update the weights and thus pass the whole dataset multiple times to obtain a better and more accurate prediction model to optimize gradient descent. However, it is not clear how many epochs are needed to train a model with the same dataset to achieve optimum weights. For the best train to the network, different datasets proceed differently. Therefore, epoch numbers are different (Siami-Namini et al.,2018).

Adadelta, Adagrad, and SGD require many epochs. As stated by (Reimers and Gurevych, 2017), they measured the number of epochs for an optimizer, and convergence observed considerable variations in the number of epochs until Nadam converged the fastest and only required a few training epochs to achieve good performance. The increased number of epochs improved the RMSE and MSE values. However, after a certain increase, the number of an epoch does not improve the performance, which varies according to the training dataset.

4.9. Optimization Results

Better predictions can be made through price prediction model selection and optimization by combining different data sources. This thesis uses data sets consisting of Bitcoin closing, Gold, crude oil, natural gas, Ethereum, and dollar-euro parity prices. ML and DL algorithms results are examined from sections 4.1 to 4.7 to provide a reference for investors to avoid risks. In section 4.9, in previous articles (Buslim et al., 2021; Pour et al., 2022), the effect of parameters on performance is examined, showing us the importance of hyper-parameter optimization.

4.9.1. Hparam Parameters

This section emphasizes hyper-parameter optimization, which is concluded to affect performance (Kervancı and Akay, 2023). The hparam parameters of the Tensorboard were tested by giving the values in Table 4.14. We observed that the test MSE values of the hourly dataset gave better results than the daily dataset. The dropout technique prevents overfitting and leaves unrelated

information from the network to improve performance (Rehman et al., 2019). Education of deep neural networks is complicated as the parameters of the previous layer change. The distribution of each layer's inputs changes during training (Ioffe and Szegedy, 2015), so the effect of the parameters (activation function, optimizer, batch size, learning rate, and dropout) with a single layer for LSTM has been observed.

Table 4.14. Hyper-parameters and their values

Hyper-parameters	Values
HP_NUM_UNITS	32,64,128
HP_DROPOUT	[0.1,0.2]
HP_LEARNING_RATE	0.1, 0.05, 0.001, 0.2
HP_OPTIMIZER	Adam, sgd, adadelta,adagrad, rmsprop, adamax
HP_ACTIVATION	Tanh, softmax, relu,sigmoid
METRIC	Mean squered error

These tests were carried out to see the contribution of daily and hourly datasets to Bitcoin price prediction.

Table 4.15. The results of the hourly dataset with hparam parameters (epoch=100)

Batch size	**Optimizer**	**Activation**	**Dropout**	**Lr**	**Test MSE**
64	Adamax	tanh	0.2	0.001	0.000043633
64	Adadelta	tanh	0.1	0.05	0.000045415
64	Adadelta	tanh	0.1	0.1	0.000047583
64	Adadelta	tanh	0.2	0.05	0.000047906
32	Adagrad	tanh	0.1	0.1	0.000048397
64	Adamax	tanh	0.1	0.001	0.000048572
32	Adamax	tanh	0.1	0.001	0.000050041
64	Adagrad	tanh	0.2	0.1	0.000051697
64	Adadelta	tanh	0.1	0.2	0.000052008
32	Adagrad	tanh	0.2	0.2	0.000052436

Table 4.14 shows the best 10 values and parameters out of 576 combinations (Table 4.14) for the hourly dataset.

Table 4.16. The results of the daily dataset with hparam parameters (epoch=100)

Batch size	**Optimizer**	**Activation**	**Dropout**	**Lr**	**Test MSE**
128	rmsprop	relu	0.1	0.001	0.00073843
128	adam	relu	0.1	0.001	0.00079225
128	Adamax	tanh	0.2	0.001	0.00079877
64	Adamax	tanh	0.2	0.001	0.00081403
128	adam	tanh	0.2	0.001	0.00081657
128	adam	tanh	0.1	0.001	0.00084659
128	Adamax	tanh	0.1	0.001	0.00086999
64	Adamax	tanh	0.1	0.001	0.00088882
64	rmsprop	tanh	0.1	0.001	0.00089062
64	Adamax	relu	0.1	0.001	0.00092647

Table 4.16 shows the best ten values and parameters out of 576 combinations (Table 4.14) for the daily dataset. The daily dataset produces better results with a small learning rate and small dropout values, whereas the hourly dataset produces better results with a large learning rate and large dropout values. The results of Tables 4.15 and 4.16 show better MSE values were obtained when Lr decreased and batch size was constant, or Lr was constant, and batch size increased, or dropout increased, and the other parameters were fixed. When the test MSE values of the daily and hourly datasets are examined, the hourly dataset is recommended since the hourly dataset produces better test MSE values.

4.9.2. Grid Search

The results do not affect each other as each parameter combination is run independently of the other. GRU, LSTM, and RNN processes take minutes 24, 23, and 16, respectively.

Table 4.17. Grid search best parameter for each model

Model	Optimizer	Activation Function	Dropout	Lr	Neuron
RNN	Adamax	tanh	0.2	0.05	64
LSTM	Adam	relu	0.1	0.001	128
GRU	Adamax	sigmoid	0.1	0.0001	128

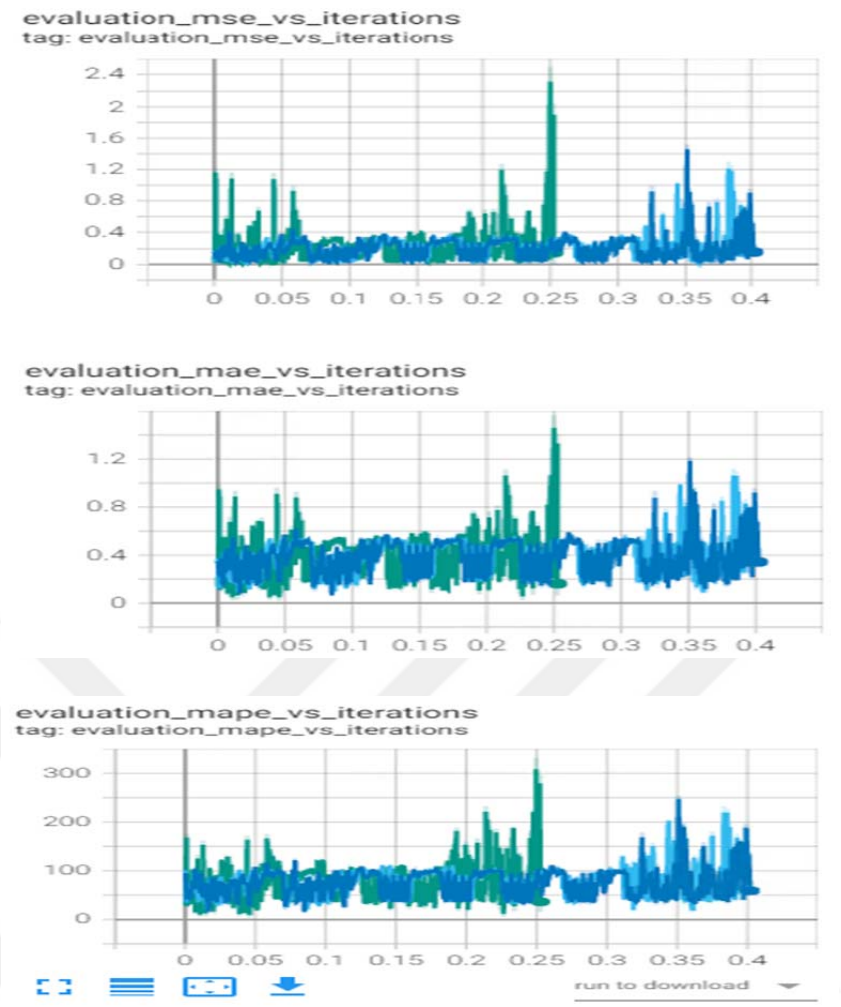


Figure 4.13. Grid search Table 4.16 result in Tensorboard MSE, MAE, MAPE for the test set (blue: GRU, dark blue: LSTM, green: RNN)

In Figure 4.13, the results of the tests for GRU, LSTM, and RNN in the Grid search are visualized on the Tensorboard for MSE, MAE, and MAPE. When these results visualized in Tensorboard are evaluated, LSTM and RNN come best behind GRU among all three error metrics.

4.9.3. Random Search

Table 4.18. Random search best parameter for each model with univariate time series

Model	Optimizer	Func	Dropout	Lr	Neuron	MSE	MAE	MAPE
RNN	Adam	tanh	0.2	0.001	64	0.000426	0.014335	0.026198
LSTM	rmsprop	tanh	0.1	0.05	128	0.019617	0.130829	0.262235
GRU	rmsprop	tanh	0.1	0.001	128	0.001277	0.029441	0.058585

GRU, LSTM, and RNN processes take minutes 18.5, 16.8, and 9.6 respectively.

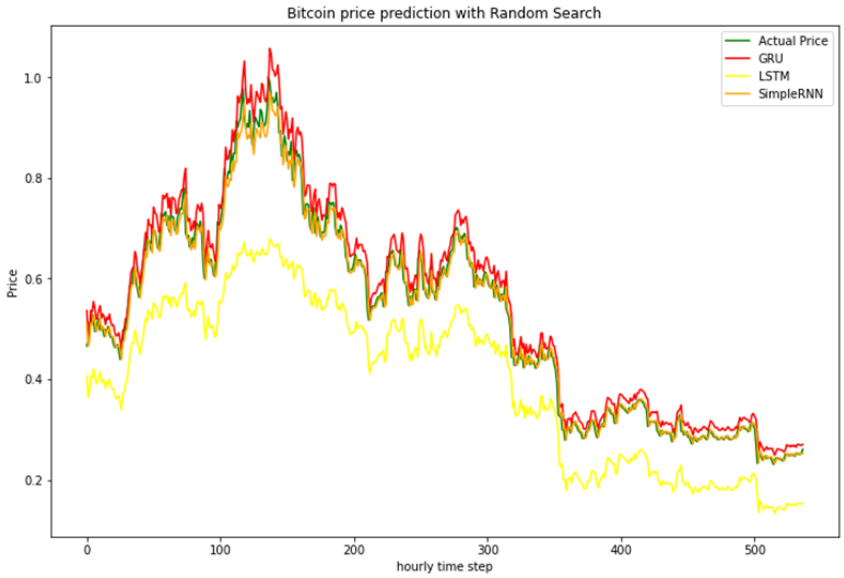


Figure 4.14. Random search MSE values

Table 4.19. The results of different models with Random search

Error Metrics	1-Layer GRU	1-Layer LSTM	2-Layers GRU	2-Layers LSTM	3-Layers GRU	3-Layers LSTM
MAPE	**0.012479**	0.016798	0.060828	0.043883	0.017961	0.088956
MAE	**0.005431**	0.007045	0.027493	0.017537	0.006493	0.039999
MSE	**0.000040**	0.000061	0.000809	0.000345	0.000081	0.001686
RMSE	**0.006327**	0.007791	0.028445	0.018577	0.009020	0.041065

Table 4.20. The results of different models with their best parameters for a random search

Models	Neurons	Activation	Dropout	Optimizer	Lr	MSE
1-Layer GRU	128	tanh	0.2	adamax	0.05	**0.000040**
2-Layers GRU	32	tanh	0.2	rmsprop	0.001	0.000809
3-Layers GRU	128	tanh	0.2	adam	0.0001	0.000081
1-Layer LSTM	128	tanh	0.1	adam	0.05	0.000061
2-Layers LSTM	64	tanh	0.2	adam	0.001	0.000809
3-Layers LSTM	128	tanh	0.2	rmsprop	0.001	0.001686

As we can see from Table 4.19, the best results for GRU and LSTM with 1, 2, and 3 layers worked better in 1 layer, and GRU outperformed LSTM. Table 4.19 shows a graphically represented in Figure 4.15 for test MSE values. For the test results in Table 4.19, the best parameters for each layer are shown in Table 4.20. The 'tanh' activation function produced the best result for each situation, and the number of neurons 128 is the best result in both LSTM and GRU. Considering Table 4.19 and 4.20 together, optimizer = adam generally fits the dataset. As a result of the papers, learning rates and nonlinearity 'tanh' activation function significantly affect performance (Breuel, 2015; Tandon et al. , 2019).

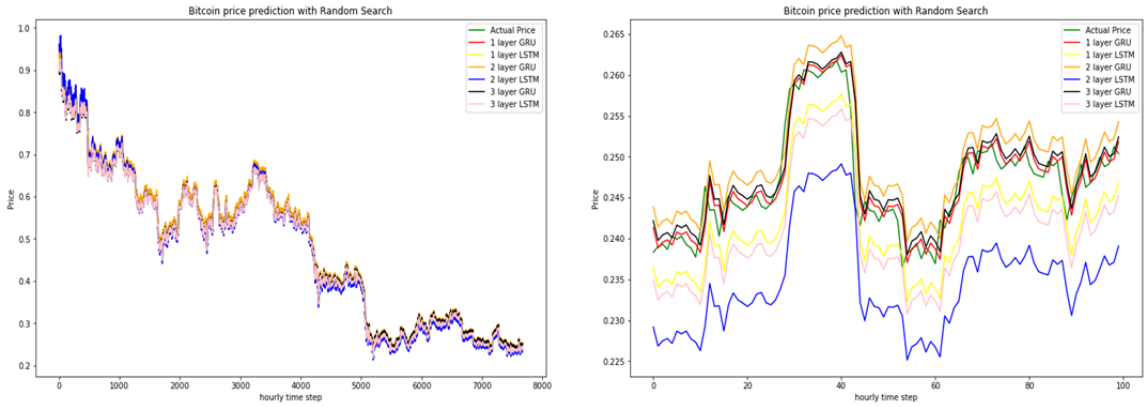


Figure 4.15. Random search, MSE results with layers (8000 and 100 time step)

4.9.4. Bayesian Optimization

According to (Wu et al., 2019), BO can find the optimal value even with a small number of samples. It is also quite efficient when compared to grid search and random search in terms of uptime. While minimizing the f(x) function, a random restart should be used in the code so that it can be tested on parameters other than the hyper-parameters around the local minimum. As a result of the tests, as we can see in Tables 4.22 and 4.23, in terms of the number of layers. Both LSTM and GRU give the best results in a layer, and GRU outperformed LSTM.

Table 4.21. BO with LSTM and GRU layers

Error Metrics	1-Layer GRU	1-Layer LSTM	2-Layers GRU	2-Layers LSTM	3-Layers GRU	3-Layers LSTM
MAPE	**0.005807**	0.010949	0.006991	0.044789	0.050767	0.022976
MAE	**0.002725**	0.004762	0.003319	0.017208	0.020365	0.011783
MSE	**0.000017**	0.000033	0.000024	0.000348	0.000455	0.000202
RMSE	**0.004088**	0.005763	0.004849	0.018662	0.021341	0.014216

In Tables 4.21 and 4.22, as a result of the experiments performed with BO, for the different numbers of layers of GRU and LSTM, the parameters that give the best results among the parameter combinations are activation function=tanh, optimizer=adam, dropout=0.1, neuron=128 and learning rate=0.001' (epoch=100). Activation functions, generally, the most suitable activation function is tanh for the data set. Although not like the activation function for optimizers, it generally gives better results with the adamax, adam and rmsprop with the dataset. But when all experiments are examined, the adam optimizer is recommended.

Table 4.22. The results of MSE with their best parameters for BO.

Models	Neurons	Activation	Dropout	Optimizer	Lr	MSE
1-Layer GRU	128	tanh	0.1	adamax	0.05	**0.000017**
2-Layers GRU	128	tanh	0.1	rmsprop	0.0001	0.000024
3-Layers GRU	128	tanh	0.2	adamax	0.001	0.000455
1-Layer LSTM	32	tanh	0.1	adam	0.05	0.000033
2-Layers LSTM	128	tanh	0.2	adam	0.001	0.000348
3-Layers LSTM	128	tanh	0.2	adam	0.001	0.000202

Figure 4.16 is the visualized form of Table 4.22 according to MSE values.

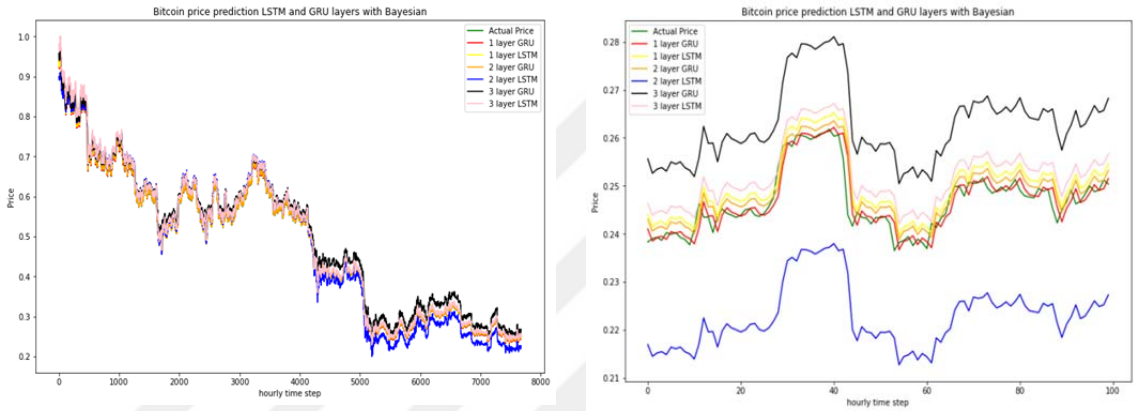


Figure 4.16. Bayesian optimization LSTM and GRU MSE results with layers (8000 and 100 time step)

Table 4.23. BO with LSTM, GRU, and hybrids

Error Metrics	1-Layer GRU	1-Layer LSTM	Hybrid GRU-LSTM	Hybrid LSTM-GRU
MAPE	0.005807	0.010949	0.006927	**0.005497**
MAE	0.002725	0.004762	0.003083	**0.002302**
MSE	0.000017	0.000033	0.000019	**0.000015**
RMSE	0.004088	0.005763	0.004306	**0.003269**

Hybrid GRU-LSTM and LSTM-GRU best performing parameters activation function is tanh, and optimizers are sgd and adamax, dropout=0.1, neuron=128 and learning rates are 0.1 and 0.05 (epoch=100). Both hybrid models, LSTM-GRU and GRU-LSTM, give reasonable results, and LSTM-GRU outperformed GRU-LSTM and all other models in the thesis.

Figure 4.17 is the visualized form of Table 4.23 according to MSE values.

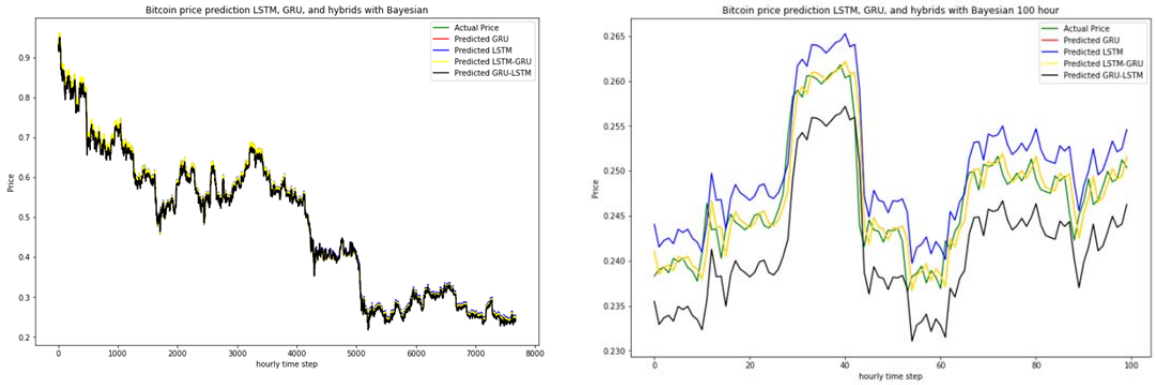


Figure 4.17. Bayesian optimization LSTM, GRU, and hybrid MSE results (8000 and 100 time step)

4.10. Discussions

Table 4.24. In terms of ML algorithms, and data sets

Models	Dataset	MSE	RMSE	MAE	MAPE
MLP	X1	0.000512	0.022622	0.015445	0.028022
SVM	X1	0.000976	0.031237	0.026301	0.066896
GRNN	X6	**0.000347**	**0.018638**	0.013778	0.038527
RNN	X1	0.000499	0.022343	0.015227	0.027981
LSTM	**X1**	0.000380	0.019486	**0.013338**	**0.024971**
GRU	X5	0.000398	0.019943	0.013771	0.025526
CNN	X2	0.000380	0.019501	0.013339	0.025092

Table 4.24 shows that models with the X1 dataset generally produced better results, while GRU with dollar-euro parity and GRNN with natural gas produced good results. Generally, models with the X1 data set produced good results, while GRU with X5 and GRNN with X6 produced good results. Except for SVM, the results are pretty close to each other. Acceptable results were obtained with X17, except GRNN and SVM. Except for SVM, there is at least one parameter set that generates MAPE values of around 0.03.

In terms of parameters, in ANN models, it is recommended to use the Lr parameter as it significantly improves performance. For numerical time series, setting the go-backward parameter to False is recommended. Tanh and relu activation functions are suitable for these datasets. It is quite clear that the SGD optimizer gives better results with the learning rate parameter. We recommend they be used together. In general, the impact of batch size on performance, the larger the batch size, the better the test MSE value. When all other parameters were fixed, it was seen that tanh and relu activation functions with SGD, Adam, RMSprop, and Adamax optimizers, and sigmoid activation function with Adadelta optimizer gave good results. We achieved the same results as the two-layer LSTM with the parameters we optimized the LSTM network, while we

obtained better results than the 3 and 4-layer LSTM, which shows us the importance of hyper-parameter optimization. Better MSE values were obtained when Lr decreased, batch size was constant, or Lr was constant and batch size increased, or dropout increases and the other parameters fixed. When the test MSE values of the daily and hourly datasets are examined, the hourly dataset is recommended since the hourly dataset produces better test MSE values.

In terms of Optimization methods, are used; the hybrid CNN-LSTM model with BO achieved the best results when compared to other models in the paper (Min et al., 2022). Hybrid model gave better results than convolutional neural networks.

Table 4.25. The results of the literature review comparison of the articles MSE, RMSE, MAE, and MAPE values with our best results

References	**Method**	**MSE**	**RMSE**	**MAE**	**MAPE**
(Aggarwal et al., 2019)	4 layer LSTM	-	32.98	-	-
(Tandon et al., 2019)	2 layer LSTM with 10 Fold Cross validation	-	-	0.0043	-
(Saad et al. , 2020)	Random Forest	-	0.0141	0.0072	-
(Rajabia et al., 2022)	MLP (learnable window size)	-	-	-	0.310048
(Drahokoupil, 2022)	XGboost with BO	3273.3	-	-	-
(Pour et al., 2022)	LSTM and BO	18155370	4260.9121	-	-
(Patra and Mohanty, 2022)	GRU with 3 layers	63307.312	251.6094	164.4882	0.0031
Our best result	BO hybrid LSTM-GRU	0.000015	0.003269	0.002302	0.005497

When Table 4.25 is investigated, it is seen that we obtained the best result in the examined literature, our results for MSE, RMSE, and MAE are better except for MAPE.

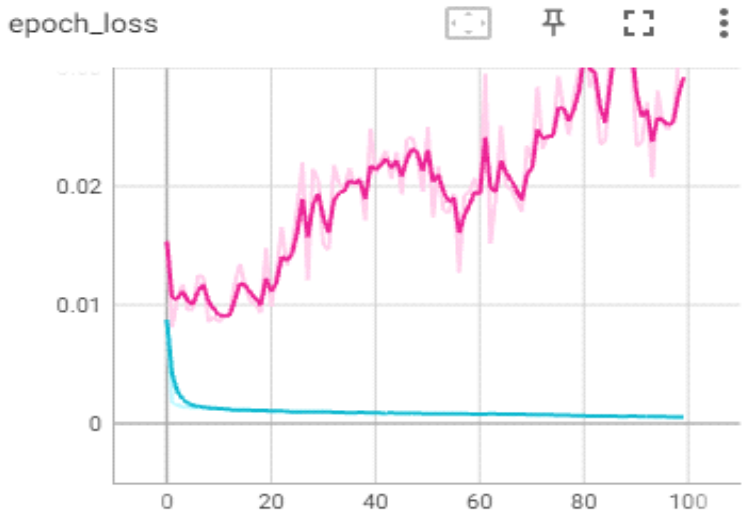


Figure 4.18. Prediction of the next 30 days with hybrid LSTM-GRU loss

As can be seen from this result and Figure 4.18, the loss in long-term prediction in Bitcoin is increasing gradually, affecting the prediction's performance.

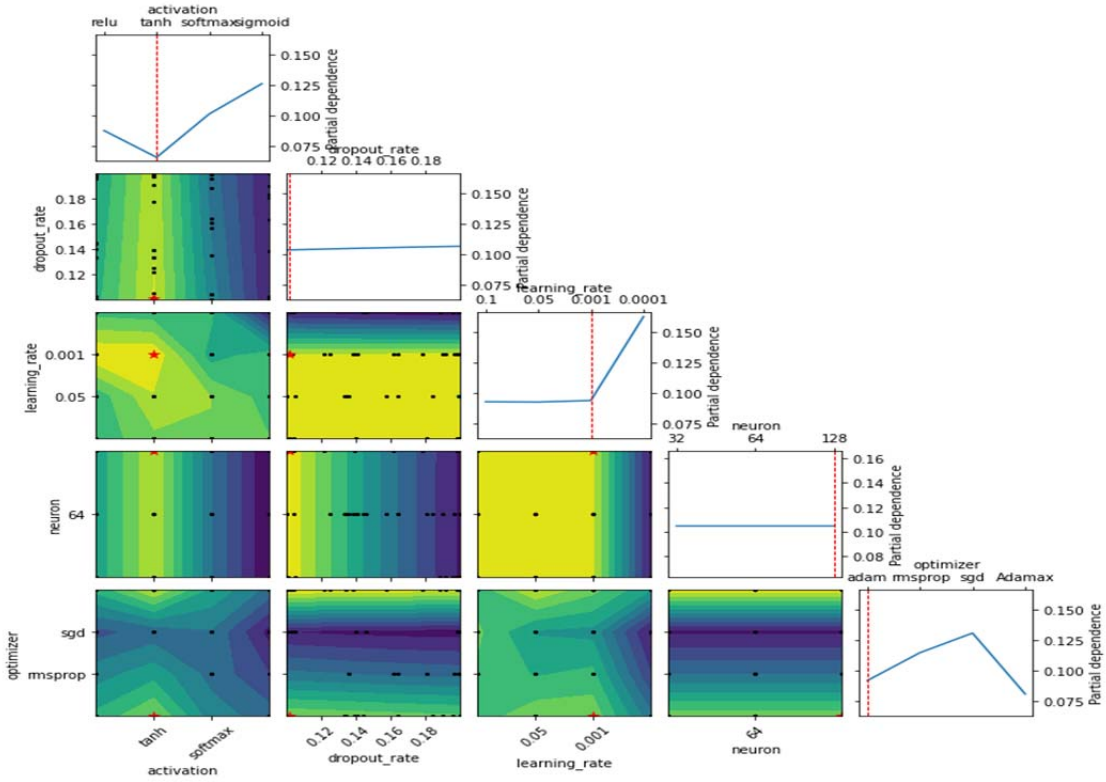


Figure 4.19. Sensitivity analysis of the hyperparameters, LSTM-GRU optimized with BO-GP

It is very useful to have sensitivity analysis when there are many predictive variables to help interpret the models produced by the "black box" estimation method of partial dependency functions. For many hyper-parameters and their values in Figure 4.19, it would be quite helpful to

have relevance to reduce combinations of variables to consider. Since the distribution for the number of neurons shifts towards 128, it is concluded that the number of neurons should be increased, while values less than 0.05 are preferred for Lr instead of values greater than 0.05. Thanks to these interpretations, gain computational cost and stability, it is ensured that the correct parameter values are preferred from unrelated regions to the relevant regions.

5. CONCLUSIONS

The following conclusions were reached considering all the experiments made in the 'Results and Discussions' section. In general, dataset size increases when the larger date range is used, and the accuracy increases accordingly. However, due to Bitcoin's extreme volatility, the dataset's variability may also increase. For this reason, it cannot be said that a large dataset for Bitcoin always improves performance. For this reason, it cannot be said that a large dataset for Bitcoin always improves performance. The most appropriate activation function for the proposed time series dataset is 'tanh'.

The parameters affecting Bitcoin are generally unique, although some dependencies have been found in some periods. In general, it is seen that the parameters affecting Bitcoin have temporary effects. We believe in the necessity of utilizing deep learning and optimization methods when predicting the future price of Bitcoin. In this direction, we have obtained very low loss results in the thesis. Optimizing the surrogate function contributed to the improvement of reducing the loss. We have obtained the following results by adding the hybrid structure to our experiments for the two models that give the best general results within the thesis. Hybrid LSTM-GRU and GRU-LSTM experimented with BO-GP. The best performing parameters from the hourly dataset of the hybrid method LSTM-GRU is activation function=tanh, optimizers=sgd and adamax, dropout=0.1, neuron=128, and learning rates=0.1 and 0.05. As seen in Table 4.22 and Table 5.2, MAPE=0.005497, MAE=0.002302, MSE=0.000015, and RMSE=0.003269 values are quite good, except for the MAPE value of the (Patra and Mohanty, 2022) from the results obtained as a result of the literature review. Experiments with BO using different surrogate functions such as GP, ET, RF, and GBRT showed good interpolation ability in the operating range. But, applying the Gaussian process is also quite flexible to fit the data and update the posterior distribution. Therefore, GP is considered very suitable for BO. Sensitivity analysis is recommended to be used because it provides gain calculation cost and stability, and helps to choose the right parameter values from unrelated regions to relevant regions.

When we predict the next 30 days by applying these parameters (activation function=tanh, optimizers=sgd and adamax, dropout=0.1, neuron=128 and learning rates=0.1 and 0.05) to our daily data set (bitcoin close price) with hybrid LSTM-GRU, the results of loss increases as can be seen below Figure 5.1. The difference between the predicted price and the actual price is under $50 for five day, under $100 for eight days, and under $1000 for nineteen days. Looking at the real price differences between the following two days for the same date range, ten days with more than $100 and two days with around $1000. Although the effects of methods, datasets, optimization and causality relationships are examined, long-term predictions involve quite high risks.

REFERENCES

Aalborg, H. A., Molnár, P., and Vries, J. E. (2019). What can explain the price, volatility and trading volume of Bitcoin? *Finance Research Letters, 29*, 255-265.

Abboushi, S. (2017). Global virtual currency – Brief Overview. *The Journal of Applied Business and Economics, 19*(6).

Abdennour, A. (2006). Evaluation of neural network architectures for MPEG-4 video traffic prediction. *IEEE transactions on broadcasting, 52*(2), 184-192.

Aggarwal, A., Gupta, I., Garg, N., and Goel, A. (2019). Deep learning approach to determine the impact of socio economic factors on bitcoin price prediction. *Twelfth International Conference on Contemporary Computing (IC3).* Noida, India.

Ahram, T., Sargolzaei, A., Sargolzaei, S., Daniels, J., and Amaba, B. (2017). Blockchain technology innovations. *2017 IEEE technology & engineering management conference (TEMSCON).* Santa Clara, CA, USA.

Aras, S. (2021). Stacking hybrid GARCH models for forecasting Bitcoin volatility. *Expert Systems with Applications, 174*, 114747. doi:10.1016/j.eswa.2021.114747

Asselman, A., Khaldi, M., and Aammou, S. (2021). Enhancing the prediction of student performance based on the machine learning XGBoost algorithm. *Interactive Learning Environments*, 1-20.

Balcilar, M., Bouri, E., Gupta, R., and Roubaud, D. (2017). Can volume predict Bitcoin returns and volatility? A quantiles-based approach. *Economic Modelling, 64*, 74-81.

Baur, D. G., Dimpfl, T., and Kuck, K. (2018). Bitcoin, gold and the US dollar – A replication and extension. *Finance Research Letters, 25*, 103-110.

Beneki, C., Koulis, A., Kyriazis, N. A., and Papadamou, S. (2019). Investigating volatility transmission and hedging properties between Bitcoin and Ethereum. *Research in International Business and Finance, 48*, 219-227.

Bouoiyour, J., and Selmi, R. (2015). What does Bitcoin look like? *Annals of Economics and Finance, 16*(2), 449-492.

Bouri, E., Gupta, R., Lahiani, A., and Shahbaz, M. (2018). Testing for asymmetric nonlinear short- and long-run relationships between Bitcoin, aggregate commodity and gold prices. *Elsevier, 57*, 224–235.

Breuel, T. (2015). *Benchmarking of LSTM networks.* Google, Inc. Cornell University.

Brunel, A., and all, e. (2019). A CNN adapted to time series for the classification of Supernovae. *Electronic imaging, arXiv:1901.00461.*

Buslim, N., Rahmetullah, İ. L., Setyawan, B. A., and Alamsyah, A. (2021). Comparing bitcoin's prediction model using GRU, RNN, and LSTM by hyperparameter optimization grid search and random search. *9th International Conference on Cyber and IT Service Management (CITSM).* Bengkulu, Endonezya.

Cachin, C., Vukolic, M., and with many others at IBM. (2017). Blockchain consensus protocols in the wild. *arXiv preprint arXiv:1707.01873*.

Cavalli, S., and Amoretti, M. (2021). CNN-based multivariate data analysis for bitcoin trend prediction. *Applied Soft Computing, 101*, 107065. doi:10.1016/j.asoc.2020.107065

Cho, K., Merrienboer, B. v., Bahdanau, D., and Bengio, Y. (2014). On the properties of neural machine translation: encoder–decoder approaches. *arXiv preprint arXiv:1409.1259*.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Corbet, S., Meegan, A., Larkin, C., Lucey, B., and Yarovaya, L. (2018). Exploring the dynamic relationships between cryptocurrencies and other financial assets. *Economics Letters, 165*, 28-34.

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

Devavrat, S., and Kang, Z. (2014). Bayesian regression and Bitcoin. *Fifty-second Annual Allerton Conference.* Monticello, IL, USA.

Drahokoupil, J. (2022). Application of the XGBoost algorithm and Bayesian optimization for the Bitcoin price. *Prague University of Economics and Business, 4*(006).

Duchi , J., Hazan, E., and Singer , Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research, 12*, 2121-2159.

Dutta, A., Kumar, S., and Basu, M. (2020). A gated recurrent unit approach to bitcoin price prediction. *Journal of Risk and Financial Management, 13*(2), 23.

Fadil, I., Helmiawan, M. A., and Sofiyan, Y. (2021). Optimization parameters support vector regression using grid search method. *9th International Conference on Cyber and IT Service Management (CITSM).* Bengkulu, Indonesia.

Felizardo, L., Oliveira, R., Del-Moral-Hernandez, E., and Cozman, F. (2019). Comparative study of Bitcoin price prediction using WaveNets, Recurrent Neural Networks and other Machine Learning Methods. *6th International Conference on Behavioral, Economic and Socio-Cultural Computing.* Beijing, China.

Feng, Y., Peng, Y., Cui, N., Gong, D., and Zhang, K. (2017). Modeling reference evapotranspiration using extreme learning machine modeling and generalized regression neural network only with temperature data. *Computers and Electronics in Agriculture, 136*, 71-78.

Frazier, P. I. (2018). A Tutorial on Bayesian Optimization. *arXiv preprint arXiv:1807.02811*.

Fu, R., Zhang, Z., and Li, L. (2016). Using LSTM and GRU neural network methods for traffic flow prediction. *2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC).* Wuhan,China.

Garcia, D., Tessone, C. J., Mavrodiev, P., and Perony, N. (2014). Feedback cycles between socio-economic signals in the Bitcoin economy. *The Journal of The Royal Society*.

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

Ghorbel, A., and Jeribi, A. (2021). Investigating the relationship between volatilities of cryptocurrencies and other financial assets. *Decisions in Economics and Finance, 44*, 817–843.

Gkillas, K., Bouri, E., Gupta, R., and Roubaud, D. (2020). Spillovers in higher-order moments of crude oil, gold, and bitcoin. *The Quarterly Review of Economics and Finance, 84*, 398-406.

Golmant, N., Vemuri, N., Yao, Z., Feinberg, V., Gholami, A., Rothauge, K., . . . Gonzalez, J. (2018). On the computational inefficiency of large batch sizes for stochastic gradient descent. *arXiv preprint arXiv:1811.12941*.

Gong, Y., and Huser, R. (2022). Asymmetric tail dependence modeling, with application to cryptocurrency market data. *The Annals of Applied Statistics, 16*(3), 1822-1847. doi:10.1214/21-AOAS1568

Goodfellow, I., Bengio, Y., and Courville, A. (2016). Deep feedforward networks deep learning. *Deep learning, 1*(6).

Gradojevic, N., Kukolj, D., Adcock, R., and Djakovic, V. (2023). Forecasting Bitcoin with technical analysis: A not-so-random forest? *International Journal of Forecasting, 1*, 1-17.

Griggs, K. N., Ossipova, O., Kohlios, C. P., Baccarini, A. N., and Howson, E. A. (2018). Healthcare Blockchain system using smart contracts for secure automated remote patient monitoring. *Journal of Medical Systems, 42*, 130.

Grosse, R. (2017). *Bayesian hyperparameter optimization.* Toronto university. Toronto: Toronto university. Retrieved from https://www.cs.toronto.edu/~rgrosse/courses/csc321_2017/slides/lec21.pdf

Guo, J., Li, C., Zhang, G., Sun , Y., and Bie, R. (2020). Blockchain-enabled digital rights management for multimedia resources of online education. *Multimedia Tools and Applications, 79*, 9735–9755.

Hamayel, M. J., and Owda, A. Y. (2021). A novel cryptocurrency price prediction model using GRU, LSTM and bi-LSTM machine learning algorithms. *AI, 2*(4), 477-496. doi:https://doi.org/10.3390/ai2040030

Harun, N., Woo, W. L., and Dlay, S. S. (2010, May 11-12). Performance of keystroke biometrics authentication system using artificial neural network (ANN) and distance classifier method. *International Conference on Computer and Communication Engineering (ICCCE'10).* Kuala Lumpur, Malaysia.

Haykin, S. (1994). *Neural networks: A comprehensive foundation.* NJ, USA: Prentice Hall PTR Upper Saddle River,.

Hinton, G., Srivastava, N., and Swersky, K. (2012). *Lecture 6.5 Rmsprop- Divide the Gradient by a Running Average of its Recent Magnitude.* Toronto Uni.

Ince, H., and Trafalis, T. B. (2006). A hybrid model for exchange rate prediction. *Decision Support Systems, 42*, 1054–1062.

Indera, I. N., Yassin, I. M., Zabidi, A., and Rizman, Z. I. (2017). Non-linear autoregressive with exogeneous input (NARX) Bitcoin price prediction model using PSO-optimized parameters optimized parameters. *Journal of fundamental and applied sciences, 9*(3S), 791.

Ioffe, S., and Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training Reducing Internal Covariate Shift. *Proceedings of the 32 nd International Conference on Machine Learning. 37.* Lille, France: JMLR: W&CP .

Jianfu, L. (2014). The Tessera D&R computational environment designed experiments for R-HADOOP performance and Bitcoin analysis. *Doctor of Philosophy*. West Lafayette, Indiana.

Jiang, S., Li, Y., Lu, Q., Wang, S., and Wei, Y. (2022). Volatility communicator or receiver? Investigating volatility spillover mechanisms among Bitcoin and other financial markets. *Research in International Business and Finance, 59*, 101543.

Juarez, O. D., and Manzanilla, H. E. (2019). Forecasting Bitcoin pricing with hybrid models a review of the literature. *International Journal of Advanced Engineering Research and Science (IJAERS), 6*(9), 161-164.

Kaminski, J. C., and Lab, M. M. (2016). *Nowcasting the Bitcoin market with twitter signals.* Cornell University arXiv:1406.7577.

Karasu, S., Altan, A., Saraç, Z., and Hacioğlu, R. (2018). Prediction of Bitcoin prices with machine learning methods using time series data. *2018 26th Signal Processing and Communications Applications Conference (SIU).* Izmir, Turkey.

Kazybek , A., Kamilya , S., and Alex , P. J. (2018). Memristive LSTM network hardware architecture for time-series predictive modeling problems. *2018 IEEE Asia Pacific Conference on Circuits and Systems.* Chengdu, China.

Kervancı, S. I., and Akay, F. M. (2020). Review on Bitcoin Price Prediction Using Machine Learning and Statistical Methods. *Sakarya University Journal of Computer and Information Sciences (SAUCIS), 3*(3), 272-282.

Kervancı, S. I., and Akay, F. M. (2023). LSTM Hyperparameters optimization with Hparam parameters for Bitcoin Price Prediction. *Sakarya University Journal of Computer and Information Sciences(SAUCIS), in press*.

Khedr, A. M., Ifra , A., Raj, P. P., Magdi, E.-B., Alhashmi, S. M., and Sreedharan, M. (2021). Cryptocurrency price prediction using traditional statistical and machine-learning techniques: A survey. *Intelligent Systems in Accounting, Finance and Management, 28*(1), 3-34.

Kingma, D., and Ba, J. (2015). Adam: A Method for Stochastic Optimization. *3rd International Conference for Learning Representations.* San Diego.

Kouhizadeh, M., and Sarkis, J. (2018). Blockchain practices, potentials, and perspectives in greening supply chains. *Sustainability, 10*(10), 3652.

Lahmiri, S., Bekiros, S., and Bezzina, F. (2022). Complexity analysis and forecasting of variations in cryptocurrency trading volume with support vector regression tuned by Bayesian optimization under different kernels: An empirical comparison from a large dataset. *Expert Systems with Applications, 209*, 118349.

Le, X. H., Ho, H. V., and Lee, G. (2019). Application of long short-term memory (LSTM) neural network for flood forecasting. *Water, 11*(7), 1387.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature, 521*, 436–444.

Li, H.-z., Guo, S., Li, C.-j., and Sun, J.-q. (2012). A hybrid annual power load forecasting model based on generalized regression neural network with fruit fly optimization algorithm. *Knowledge-Based Systems, 37*, 378-387.

Liashchynskyi, P., and Liashchynskyi, P. (2019). Grid search, random search, genetic algorithm: A big comparison for NAS. *arXiv preprint arXiv:1912.06059*.

Livieris, I. E., Kiriakidou, N., Stavroyiannis, S., and Pintelas, P. (2021). An advanced CNN-LSTM model for cryptocurrency forecasting. *Electronics, 10*(3), 287.

Madan, I., Saluja, S., and Zhao, A. (2015). Automated bitcoin trading via machine learning algorithms. Google Scholar. Retrieved from Google Scholar: http://cs229. stanford. edu/proj2014/Isaac% 20Madan 20

Madan, R., and Mangipudi, P. S. (2018). Predicting computer network traffic: a time series. *Proceedings of 2018 Eleventh International Conference on Contemporary Computing (IC3).* Noida, India.

Mai, I., Marwan, T., and Elnainay, M. (2018). CNN based Indoor Localization using RSS time-series. *2018 IEEE Symposium on Computers and Communications (ISCC).* Natal, Brazil.

Mangla, N., Bhat, A., Avabratha, G., and Bhat, N. (2019). Bitcoin Price Prediction Using Machine Learning. *International Joutnal of Information And Computing Science, 6*(5), 318-320.

Martina, M., Ilaria, L., and Michele, M. (2015). Bitcoin Spread Prediction Using Social And Web Search Media. *UMAP Workshops, 2015.*

McNally, S., Roche, J., and Caton, S. (2018). Predicting the price of Bitcoin using machine learning. *26th Euromicro Conference on Parallel, Distributed and Network-Based Processing.* Cambridge, United Kingdom.

Min, W., Xikun, L., and Yi-di, Z. (2022). Gun life prediction model based on bayesian optimization CNN-LSTM. *Integrated Ferroelectrics, 228*(1), 107-116. doi:10.1080/10584587.2022.2072126

Moussa, W., Mgadmi, N., Béjaoui, A., and Regaieg, R. (2021). Exploring the dynamic relationship between Bitcoin and commodities: New insights through STECM model. *Resources Policy, 74*, 102416.

Mozo, A., Ordozgoiti, B., and Gómez-Canaval, S. (2018). Forecasting short-term data center network traffic load with convolutional neural networks. *PLOS one, 13*(2), e0191939.

Naser, M. Z., and Alavi, A. H. (2021). Error metrics and performance fitness indicators for artificial intelligence and machine learning in engineering and sciences. *Architecture, Structures and Construction*, 1-19.

Nazzal, J. M., El-Emary, I. M., and Najim, S. A. (2008). Multilayer perceptron neural network (MLPs) for analyzing the properties of Jordan Oil Shale. *World Applied Sciences Journal, 5*(5), 546-552.

Nugent, T., Upton, D., and Cimpoes, M. (2016). Improving data transparency in clinical trials using blockchain smart contracts. *F1000Res, 5*, 2541.

Okorie, D. I., and Lin, B. (2020). Crude oil price and cryptocurrencies: evidence of volatility connectedness and hedging strategy. *Energy economics, 87*, 104703.

Parekh, R., Patel, N. P., Thakkar, N., Gupta, R., Tanwar, S., and Sharma, G. (2022). DL-GuesS: Deep Learning and Sentiment Analysis-Based Cryptocurrency Price Prediction. *IEEE Access, 10*, 35398-35409.

Patra, G. R., and Mohanty, M. N. (2022). Price prediction of cryptocurrency using a multi-layer gated recurrent unit network with multi features. *Computational Economics*, 1-20.

Pomerat, J., Segev, A., and Datta, R. (2019). On neural network activation functions and optimizers in relation to polynomial regression. *IEEE International Conference on Big Data.* Los Angeles, CA, USA.

Pour, E. S., Jafari, H., Lashgari, A., Rabiee, E., and Ahmadisharaf, A. (2022). Cryptocurrency price prediction with neural networks of LSTM and Bayesian optimization. *EJBMR, 7*(2), 20-27.

Prachi, V. R., and Sudhir, N. D. (2019). Systematic erudition of Bitcoin price prediction. *2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS).* Coimbatore, India, India.

Rahouti, M., Xiong, K., and Ghani, N. (2018). Bitcoin concepts, threats, and machine-learning security solutions. *IEEE Access, 6*, 67189 - 67205.

Rajabia, S., Roozkhoshb, P., and Farima, N. M. (2022). MLP-based learnable window size for bitcoin price prediction. *Applied Soft Computing, Applied Soft Computing*, 109584. doi:10.1016/j.asoc.2022.109584

Rauchs, M., and Garrick, H. (2017). *Global cryptocurrency benchmarking study*. Cambridge Centre for Alternative Finance Reports.

Rehman, A. U., Malik, A. K., Raza, B., and Ali, W. (2019, June). A Hybrid CNN-LSTM Model for Improving Accuracy of Movie Reviews Sentiment Analysis. *Multimedia Tools and Applications*(78), 26597–26613.

Reimers, N., and Gurevych, I. (2017). Optimal Hyperparameters for Deep LSTM-Networks for Sequence Labeling Tasks. *EMNLP 2017*.

Roy, S., Nanjiba, S., and Chakrabarty, A. (2018). Bitcoin Price Forecasting Using Time Series Analysis. *ICCIT*, *21*. Dhaka, Bangladesh.

Saad, M., Choi, J., Nyang, D., Kim, J., and Mohaisen, A. (2020, March). Toward characterizing blockchain-based cryptocurrencies for highly accurate predictions. *IEEE Systems Journal, 14*(1), 321-332.

Saranya, C., and Manikandan, G. (2013). A study on normalization techniques for privacy preserving data mining. *International Journal of Engineering and Technology (IJET), 5*(3), 2701-2704.

Satoshi, N. (2008). A peer-to-peer electronic cash system. *Decentralized Business Review*, 21260.

Sharma, S., Sharma, S., and Anidhya, A. (2017). Activation functions in neural networks. *Towards Data Sci, 6*(12), 310-316.

Shen, D., Urquhart, A., and Wanga, P. (2019). Does twitter predict Bitcoin? *Economics Letters, 174*, 118–122.

Shen, Z., Wan, Q., and Leatham, D. J. (2019). Bitcoin return volatility forecasting: A comparative study of GARCH model and machine learning model. *Applied Economics Digital Library*.

Sherman, A. T., Javani, F., Zhang, H., and Golaszewski, E. (2019). On the origins and variations of blockchain technologies. *IEEE Security & Privacy, 17*(1), 72-77.

Shintate, T., and Pichl, L. (2019). Trend prediction classification for high frequency Bitcoin time series with deep learning. *Risk and Financial Management*.

Siami-Namini, S., Neda, T., and Siami Namin, A. (2018). Forecasting economic and financial time series : ARIMA VS. LSTM. *International Conference on Machine Learning and Applications (ICMLA)*, (s. 1394-1401). Orlando, FL, USA. arXiv:1803.06386 adresinden alındı

Singh, H., and Agarwal, P. (2018). Empirical analysis of Bitcoin market volatility using supervised learning approach. *2018 Eleventh International Conference on Contemporary Computing (IC3)*. Noida, India.

Singh, S., Slotine, J.-J., and Sindhwan, V. (2022). Optimizing Trajectories with Closed-Loop Dynamic SQP. *2022 IEEE International Conference on Robotics and Automation (ICRA).* Philadelphia, PA, USA.

Smith, J. (2016). An analysis of bitcoin exchange rates. *Available at SSRN 2493797*.

Smola, A. J., and Scholkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing, 14*, 199–222.

Specht, D. F. (1991). A general regression neural network. *IEEE Transactions on Neural Networks, 2*(6), 568 - 576.

Spilak, B. (2018). Deep Neural Networks For Cryptocurrencies Price Prediction. *Master's thesis, Humboldt-Universität zu Berlin*.

Srivastava, N. (2013). *Improving neural networks with dropout.* University of Toronto.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research, 15*, 1929-1958.

Sukhwani, H., Martínez, J. M., Chang, X., Trivedi, K. S., and Rindos, A. (2017). Performance modeling of PBFT consensus process for permissioned blockchain network (Hyperledger Fabric). *IEEE 36th Symposium on Reliable Distributed Systems* (pp. 253-255). IEEE.

Szetela, B., Mentel, G., and Gedek, S. (2016). Dependency analysis between bitcoin and selected global currencies. *Dynamic econometric models, 16*, 133-144.

Tandon, S., Tripathi, S., Saraswat, P., and Dabas, C. (2019). Bitcoin price forecasting using LSTM and 10-fold cross validation. *2019 International Conference on Signal Processing and Communication (ICSC)* (pp. 323-328). IEEE.

Thearasak, P., and Thanisa, N. (2018). Machine learning models comparison for Bitcoin price prediction. *2018 10th International Conference on Information Technology and Electrical Engineering (ICITEE).* Bali, Indonesia.

Tripathi, B., and Sharma, R. K. (2022). Modeling bitcoin prices using signal processing methods, Bayesian optimization, and deep neural networks. *Computational Economics*.

Wang, S., Ouyang, L., Yuan, Y., Ni, X., Han, X., and Wang, F.-Y. (2019). Blockchain-enabled smart contracts: architecture, applications, and future trends. *EEE Transactions on Systems, Man, and Cybernetics: Systems, 49*(11), 2266-2277.

Widiasari, I. R., Nugroho, L. E., and Widyawan. (2017, Nov 2-4). Deep learning multilayer perceptron (MLP) for flood prediction model using wireless sensor network based hydrology time series data mining. *International Conference on Innovative and Creative Information Technology (ICITech).* Salatiga, Indonesia.

Wilson, D. R., and Martinez, T. R. (2001). The need for small learning rates on large problems. *International Joint Conference on Neural Networks (IJCNN)*, *1*. Washington, DC, USA.

Wu , J., Chen, X. Y., Zhang, H., Xiong, L. D., Lei, H., and Deng, S.-H. (2019). Hyperparameter optimization for machine learning models based on Bayesian optimization. *Journal of Electronic Science and Technology*, 26-40.

Yaga, D., Mell, P., Roby, N., and Scarfone, K. (2018). *Blockchain technology overview.* U.S. Department of Commerce. National Institute of Standards and Technology Internal Report 8202.

Yamashita, R., Nishio, M., Gian Do, R. K., and Togashi, K. (2018). Convolutional neural networks: an overview and application in radiology. *Springer*.

Yang, L., and Shami, A. (2020). On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing, 465*, 295-316.

Yu, C., Qi, X., Ma, H., He, X., Wang, C., and Zhao, Y. (2020). LLR: Learning Learning Rates by LSTM for training neural Networks. *Neurocomputing*, 22-45.

Yu, T., and Zhu, H. (2020). Hyper-parameter optimization a review of algorithms. arXiv preprint arXiv:2003.05689.

Ze, S., Qing, W., and David, L. J. (2019). Model, Bitcoin return volatility forecasting: A comparative study of GARCH model and machine learning. *Agricultural & Applied Economics Association.* Atlanta, GA.

Zeiler, M. D. (2012). ADADELTA: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Zhang, A., Zhu, W., and Li, J. (2018). Spiking echo state convolutional neural network for robust time series classification. *2018 IEEE. Translations and Content Mining*, *7*.

Zhao, B., Lu, H., Chen, S., Liu, J., and Wu, D. (2019). Convolutional neural networks for time series classification. *College of Electronic Science and Engineering, National University of Defense Technology*, *28*, pp. 162– 169. Changsha 410073, China.

Zhao, R., and all, e. (2017). Machine health monitoring using local feature-based gated recurrent unit networks. *IEEE Transactions on Industrial Electronics, 65*(2), 1539-1548.

Zheshi, C., Chunhong, L., and Wenjun, S. (2019). Bitcoin price prediction using machine learning: An approach to sample dimension engineering. *Journal of Computational and Applied, 365*.

CURRICULUM VITAE

İlkay Sibel KERVANCI, Lisan eğitimini Kocaeli Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği bölümünde 2014'te tamamladı. Yüksek lisans eğitimini Kahramamaraş Sütçü İmam Üniversitesi Enformatik Ana Bilim Dalında tamamladı. 2017 yılında Çukurova Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Ana Bilim Dalında doktaraya başladı ve 2023 yılında doktora eğitimini tamamladı.Bu süreç içerisinde birçok kongre, ve dergi de çalışmaları bulunmaktadır.