ISTANBUL TECHNICAL UNIVERSITY ★ INFORMATICS INSTITUTE

CRYPTOCURRENCY PRICE PREDICTION BY USING SOCIAL MEDIA DATA

M.Sc. THESIS

Özlem Gül PAMUK

Department of Applied Informatics

Applied Informatics Programme

Thesis Advisor: Asst. Prof. Sefer BADAY

SEPTEMBER 2019

ISTANBUL TECHNICAL UNIVERSITY ★ INFORMATICS INSTITUTE

CRYPTOCURRENCY PRICE PREDICTION BY USING SOCIAL MEDIA DATA

M.Sc. THESIS

Özlem Gül PAMUK

(708151030)

Department of Applied Informatics

Applied Informatics Programme

Thesis Advisor: Asst. Prof. Sefer BADAY

SEPTEMBER 2019

<u>**İSTANBUL TEKNİK ÜNİVERSİTESİ ★ BİLİŞİM ENSTİTÜSÜ**</u>

MAKİNE ÖĞRENMESİ TEKNİKLERİ KULLANILARAK SOSYAL MEDYA VERİLERİ İLE KRİPTO PARA FİYAT TAHMİNİ

YÜKSEK LİSANS TEZİ

Özlem Gül PAMUK

(708151030)

Bilişim Uygulamaları Anabilim Dalı

Bilişim Uygulamaları Programı

Tez Danışmanı: Dr. Öğr. Üyesi Sefer BADAY

EYLÜL 2019

Özlem Gül Pamuk, a M. Sc student of ITU Informatics Institute student ID 708151030 successfully defended the thesis/dissertation entitled Cryptocurrency Price Prediction by Using Social Meida Date, which she prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Asst. Prof. Dr. Sefer BADAY** İstanbul Technical University	....
Jury Members :	**Assoc. Prof. Dr. Gökhan BİLGİN** Yıldız Technical University	....
	Prof. Dr. M. Oğuzhan KÜLEKCİ İstanbul Technical University	....

Date of Submission: 03 May 2019

Date of Defense: 13 June 2019

To my spouse, family and best friend,

FOREWORD

This thesis is written as completion to the master Applied Informatics at Istanbul Technical University. The subject of this thesis, cryptocurrency price prediction by using social media data and machine learning algorithms.

The subject is selected to work on machine learning methods. Machine learning is a grooving area and it is applicable to various and divergent areas like economy, digitalization, marketing, etc.

For data part bitcoin and social media selected. Bitcoin brings the new approach to economy and transactions of bitcoin data is available so collection bitcoin data for machine learning quite compassable.

I am working as a software test engineer and in that profession AI testing is a hot topic. I would like to improve my knowledge in similar areas and maybe continue to study AI testing in the future.

September 2019 Özlem Gül PAMUK

TABLE OF CONTENTS

Page

ABBREVIATIONS

API	: Application Programming Interface
SVM	: Support Vector Machine
SVC	: Support Vector Classifier
SVR	: Support Vector Regressor
MLP	: Multilayer Perceptron
App	: Appendix
JSON	: Java Script Object Notation
NLP	: Natural Language Processing
AI	: Artificial Intelligence
KNN	: K Nearest Neighbour
R^2	: Coefficient of Determination
TB	: Terabayt
DB	: Database
OS	: Operating System
NB	: Naïve Bayes

LIST OF TABLES

Page

LIST OF FIGURES

CRYPTOCURRENCY PRICE PREDICTION BY USING SOCIAL MEDIA DATA

SUMMARY

This study aimed to analyze and identify the relation between bitcoin cryptocurrency prices and social media mentions like related tweets or search percentages. Machine learning algorithms implemented to find out the relation between two topics.

The graduation thesis is composed of four chapters, each of them represents different aspects of the thesis.

Chapter one is the literature review. In that chapter, previous studies are searched, and successful and useful methods are combined followed and upgraded in this thesis. Google trend data has an influence on the prediction scores mentioned in other studies so google search results are decided to include in our analyze. Also most successful sentiment analyzes on tweet's text method influenced by previous studies. Time interval selection and shifting methods also inspired by previous studies. But this study surpassing the other studies by data volume and time duration. Other studies are mentioned that at least 9 months of data needed for better estimation results. Another improvement in this study is machine learning methods. Previous studies analyze the relationship between bitcoin price and social media methods by cross-correlation but in this study, relationship analyzed by machine learning methods.

Chapter two is an introduction to machine learning. What is machine learning and what is the logic behind the methods. Types of machine learning and differences between them introduced. This chapter gives better insights into the following chapters and approaches. Also, this chapter gives information about which kind of data collected from which platform.

In chapter 3, followed methods in the study are described in chronological order. Data collection is the beginning and most important part of the machine learning process. Data collected in three different sources. These are twitter, google, and cryptocurrency data. Twitter and google search percentage is representing the social media data. Twitter data streamed from Twitter API. Google search percentage gathered from Google Trend API. Cryptocurrency data captured from CoinarketCap API which is trusted and free cryptocurrency listing coins, tokens, exchanges platform. CoinMarketCap presents the top 100 cryptocurrency volume, trend, market cap, price, etc. information.

Data collected for 10 months, data collection codes written on python and open source python API of twitter and google trend used. Python also has own library for data manipulation and machine learning which is scikit-learn. Another reason to select python is the available community and platforms for support and problem-

solving. Collected data stored on ITU data laboratory Mac Os computer. Twitter data streamed daily bases in a .json file than import to the MongoDB database. Cryptocurrency and google search percentage data directly imported to MongoDB. MongoDB also an open-source and well documented tool and easy to use. It is recommended to use in big data projects, because it kept the data in file format and not allocate so much memory. It also allows regex search on the data. This feature is very helpful to search the desired keyword on the tweets.

Some of the collected data needs to preprocess before machine learning for more accurate and explicit results. Tweet’s text cleaning and then sentiment analysis performed on the data and new column which is sentiment analyze score column added to machine learning features. The critical step is merging all data to have one well-structured and suitable feature. Merging operation is critical because if there is an error in this phase it will directly affect the prediction scores. Lastly, data interpolation done on the features to overcome with the missing data.

Features are selected individually by human experience and observations. Merging made on time index column. All the features are numerical, expect tweet text column so while merging the data, text column’s count considered as feature. Other numerical columns merged and mean of the values considered as features. Data grouped by the time interval to have different data set and able to predict different time periods. Data grouped on 5 minutes, 15 minutes, 30 minutes, 45 minutes, 60 minutes and 1-day time intervals.

Features are ready for machine learning but there is another concern that overfitting. Overfitting is the problem that train data gives more accurate results than the test results. To avoid that grid search and cross-validation implemented to machine learning features. According to the output of the grid search and cross-validation machine learning model prepared with best hyperparameters.

Supervised machine learning methods selected for this study because our features have labeled data. Two machine learning methods analyzed first one is regression and the other is classification. The classification method gives us about increase or decrease in the bitcoin price. The regression method predicts the bitcoin price value.

If we check the results of 2 methods we can see that the regression methods give better estimation scores than the classification methods in sort time interval and when the time interval increases r2 score decrease rapidly. And best estimation scores belong to linear regression models which are; Linear Regression, Lasso and Rinde. Sort time intervals can assume like 5 minutes, 15 minutes and 30 minutes. For longer time intervals like 45 minutes, 60 minutes a 1-day classification model surpass the regression models. Other significant observations on classification models are given better accuracy scores on 2nd shift. As a result best regression model score is 0.997 belongs to Lasso model in 3 minutes time interval. And best classification model score is 0.979 belongs to Decision Tree model for the 1-day time interval.

MAKİNE ÖĞRENMESİ TEKNİKLERİ KULLANILARAK SOSYAL MEDYA VERİLERİ İLE CRYPTOCURRENCY FİYAT TAHMİNİ

ÖZET

Bu çalışma, bitcoin cryptocurrency fiyatı ile ilgili tweetler veya google arama yüzdesi gibi sosyal medya girişleri arasındaki ilişkiyi analiz etmek ve tanımlamak amacıyla yapılmıştır. İki konu arasındaki ilişkiyi bulmak için makine öğrenme algoritmaları uygulanmıştır.

Bu döküman, her biri tezin farklı yönlerini temsil eden dört bölümden oluşmaktadır.

Bölüm 1, literatür taramasıdır. Bu bölümde önceki araştırmalar gözden geçirilmiştir ve bu tezde de başarılı ve faydalı yöntemler kabul edilerek uygulanmıştır. Google arama yüzdesi verilerinin etkisi diğer çalışmalarda da belirtildiği için google arama sonuçları analizimize dahil edilmeye karar verilmiştir. Ayrıca, twitter metinlerinde daha önceki çalışmalarda, duygu analizinde en iyi sonucu veren method olan Text Blob bu çalışmada da kullanılmıştır. Önceki çalışmalara baktığımızda ortak sorunun verilerin yeterli hacimde ve zaman diliminde toplanamadığı önemle vurgulanmıştı. Bu durumu göze alarak bu çalışmada verilerin hacim ve süre olarak yeterli olgunlukta olmasına dikkat edildi. Önceki çalışmalarda ki diğer bir eksiklik ise makine öğrenmesi methodlarının kullanılmayıp yogunluklu olarak çapraz doğrulama tekniğinin kullanılmış oldugudur. Bu çalışmada, önceki çalışmalardan esinlenip daha iyi sonuçlar vermesi için diğer çalışmaların güçlü yönlerini harmanlayıp eksik kalan yönlerini de gidermeye yönelik çalışılmıştır.

Bölüm 2, makine öğrenmeye giriş niteliğindedir. Makine öğrenmesi nedir ve yöntemlerin ardındaki mantık nedir? Makine öğrenmesi türleri ve aralarındaki farklar. Bu bölüm, gelecek bölümler ve izlenen yaklaşımlar için açıklama niteliğindedir. Makine öğrenmesi çok daha fazla yöntem içeren bir konu olmasına rağmen bu çalışmada kullanılan regresyon ve sınıflandırma yöntemleri üzerinde durulmuştur. Bunun başlıca nedeni toplanmış olan verilerin etiketli oluşu. Etiketli veriler gözetimli öğrenme sınıfları için uygundur bu da bizi regresyon ve sınıflandırma öntemlerini kullanmamız için yönlendirmektedir. Regresyon yönteminde kullanılan methodlar: Linear Regresyon, Rasso, Ridge, SVR, Doğrusal SVR, K en Yakın Komşu, Karar Ağacı. Sınıflandırma yöntemleri seçilirken de mümkün oldugunca aynı metodlar kullanılmaya gayret edilmiştir. Sınıflandırma yönteminde kullanılar methodlar: Rassal Orman, K en Yakın Komşu, Karar Ağacı, SVM (Destek Vektör Makinesi), Doğrusal SVM, Lojistik Regresyon, Gaussian Naif Bayes, Bernoulli Naif Bayes ve MLP Sınıflandırıcı.

Bölüm üçte takip edilen yöntemler tezde takip edilen sırayla açıklanmaktadır. Veri toplama, makine öğrenmesi için başlangıç ve en önemli kısımdır. Üç farklı kaynaktan veriler toplanmıştır bunlar; Twitter, Google ve Cryptocurrency verileridir. Twitter ve google arama yüzdesi araştırmadaki sosyal medya verilerini temsil

etmektedir. Twitter verileri Twitter API üzerinden kripto para ile ilgili olanlar filtrelenerek gerçek zamanlı olarak çekilmiştir. Google arama yüzdesi verileri Google Trend API üzerinden elde edilmiştir. Google Trend arama yüzdesi google üzerinde yapılan aramaların tarihi ve coğrafi bazda gruplanarak tüm aramalara oranına bağlı olarak 0 ve 100 arasında ölçeklendirilmesiyle elde edilir. Bitcoin ve Ethereum ile arama sonuçları yüzdesi bu platform üzerinden elde edilmiştir. Kripto para verileri CoinMarketCap API üzerinden toplanmıştır. İnternet üzerinde kripto para değerlerine erişebileceğimiz birçok farklı site olmasına ragmen CoinMarketCap tercih edilmesinin nedeni güvenilir olması ve sağladığı verilerin çeşitliliği. En populer 100 kripto para verisine ait hacim, değer, pazar hacmi, değişim yüzdesi gibi birçok veri elde edilmiştir. On ay boyunca sürekli toplanan veriler, ITU veri labaratuvarında bulunan 1 terabaytlik Mac os bilgisayarında depolanmıştır. Bu bilgisayar ortak kullanımda olduğu için elektrik kesintisi, bilgisayar açılıp kapanması yada veri toplama işinin yanlışlıkla sonlandırılması durumunlarıyla başa çıkmak için otomatik görevler yaratılmıştır. Bu görevin ana olarak veri toplama işini gerçekleştiren görevin çalışıp çalışmadıgını her 15 dakika aralıklarla kontrol edip çalışmadığı durumda otomatik başlatmaya yaramaktadır. Bu sayede veri toplamada maksimum verim sağlanmıştır.

Toplanan veriler yine aynı bilgisayarda bulunan ve sadece bu çalışma için kurulan Mongo veritabanında saklanmıştır. Mongo veritabanı özellikleriyle bu çalışma için diğer veritabanlarının önüne geçmiştir. Örnek olarak mongo veritabanı içerisinde regex sorgu atma özelliğine sahiptir. Bu özellik tweet metinleri içinde belirtilen kripto para birimine göre arama kolaylıgı sağlamıştır. Diğer bir özelliği ise açık kaynak olması ve üyelik veya ödeme gerektirmeden kullanılabilmesi.

Python yazılım dili araştırmamızda tercih ediliştir. Başlıca sebepleri arasında twitter ve google'ın sağladığı açık kaynak API lerinin bulunması. Makine öğrenmesi ve veri toplama ve manipule etmede çok basarılı ve kullanım kolaylığı sağlayan kütüphanelerinin olması son olarakta problem çözme ve bilgi edinme konusunda faydalanılabilecek birçok platform ve mevcut topluluklarının olmasıdır. Ayrıca scikit-learn makine öğrenmesi kütüphanesinin python dilini destekliyor oluşuda python dilini tercih etmemizin başlıca nedenleri arasında gelmektedir.

Toplanan verilerin bir kısmının daha doğru ve kesin sonuç için makine öğrenmesi sürecinden önce ön işleme tabi tutulması gerekmektedir. Bu ön işleme fazında tweetler içerisindeki özel karakterler, URL, etiketlerden arındırılmıştır. Özel karakterler duyarlılık analizi yaparken yanlış sonuçlara yol açtıgı için temizlenmesi ve salt metnin analizinin yapılması elde edilen sonuçların doğruluğu için önemlidir. Daha sonraki adımda ise elimizdeki üç ayrı verinin zamana göre birleştirilip makine öğrenmesi için gereken tek bir veri setinin oluşturulması işlemi vardır. En fazla dikkat gerektiren ve hata yapılması durumunda çalışmanın sonuçlarını direk olarak etkileyebilecek bir adımdır. Üç ayrı veri tarih sütunu üzerinden birleştirildi ve eksik olan zaman dilimine ait veriler veri enterpolasyonu yöntemiyle dolduruldu.

Regresyon modeli nümerik bir tahmin çıktısı verirken sıflandırmayöntemi ise bir gruplandıma sonucu tahminide bulunmaktadır. Bu nedenle 2 yöntem için de kullanılacak veri seti çıktısı sütunu farklı değerlendiriliştir. Regresyon yöntemince bitcoin fiyat değeri çıktı tahmini sonuç verisi olarak değerlendirilirken, sınıflandırmada ise artış azalış sonucu tahmin verisi değerlendirilmiştir. Tahmin verileri farklı zaman dilimleri için tahmin yapması için düzenlenmiştir. Tahmin

verileri 5 dakika, 15 dakika, 30 dakika, 45 dakika, 1 saat ve 1 gün tahmin verileri olarak ele alınmıştır. En kısa süreli tahminde 5 dakika sonraki durum tahmin edilirken. 1 günlük zaman diliminde ise 1 gün sonrasının değer veya artış azalış tahmini hesaplanmıştır. Ayrıca farklı zaman dilimlerini gözlemleyebilmek için örneğin 10 dakika sonraki tahmin verisi hesaplamak için kaydırma yöntemi uygulanmıştır. Kaydırma yöntemiyle 5 dakikalık datayı 2 kere kaydırarak 5*2 dakika soraki yani 10 dakika sonraki tahmin değerlendirimiş olmaktadır. Kaydırma yöntemi herbir zaman dilimi için 4 kere uygulanmıştır.

Önceki adımlarla hazır hale gelen veri setimiz makine öğrenmesine geçmeden önce makine öğrenmesi yöntemlerinin olumsuz sonucu olan aşırı öğrenme durumunu engellemek için çapraz gerçelleme yöntemiyle en iyi hiper parametre değerleri herbir model için ayrıca hesaplandı. Hesaplanan en iyi hiper parametreler makine öğrenmesinde kullanıldı.

Makine öğrenmesinin sonucu olarak, regresyon methodlarından lineer regrasyon yöntemleri; Linear Regresyon, Lasso ve Ridge modelleri en iyi sonuçları vermiştir. K en Yakın Komşu ve Karar ağacı yöntemleri ise kısa süreli zaman tahminlerinde iyi sonuç verirken tahmin süresi arttığında hızla düşüş sağlamıştır. Yine de linear regrasyon yöntemlerini kısa süreli tahminlerde de geçememiştir. Sınıflandırma yönteminde ise regrasyon yönteminden farklı olarak uzan zaman dilimlerdeki tahmin değerlerinin iyileştiği gözlemlenmiştir. Yani sınıflandırma yöntemiyle daha uzun süreli daha doğru tahmin yapılabilmektedir. Sınıflandırma yönteminde farkedilen diğer bir durum ise 2 kere kaydırma sonuçları gözle görünür şekilde artış göstermiştir.

Özetle, regresyon ve sınıflandırma tahmin sonuçları, bitcoin fiyatının sosyal medyadan etkilendiğini göstermektedir. Daha kesin sonuçlar için daha iyi makine öğrenme modellemesi ve daha fazla veri gereklidir.

1. INTRODUCTION

In the past few years, data mining and generating new insights based on the processed data has become a trending topic. This should not come as a surprise since being able to find connections between events comes with the benefits of saving lives or making a great profit. Researchers have used data mining to find connections between the chemicals, the proteins they act upon and their adverse effects to discover the adverse effects of some drugs that were not known before [1]. Similarly, data mining and classification methods were used to discover correlations between Bitcoin and gold prices and even exchange rates [2].

There are a lot of companies that are based on data analytics. For example; marketing companies use data from customer behavior and, as a result, searching a product or a keyword on the web leaves you are facing with related advertisements. It can be seen that this has a really big potential and there are opportunities waiting to be used for other purposes.

Another key point is social media. There are a lot of studies about social media's effects on society's perception; it is obvious that social media gives their users the power to corrupt information and for various reasons companies and people are able to cause disinformation. Everybody has at least one social media account such as Twitter, Facebook, Instagram or LinkedIn. Twitter was selected as a data source since it has an API that allows data to be collected easily, and it boasts around 139 million daily active users [3]. This guarantees a huge amount of data flow per day which we can exploit by collecting and processing for our research. Some people may not have social media accounts, but they are most definitely using a search engine for internet search which returns results based on the majority of other people's interests. It means that whenever you are interacting with the web, you are sharing information, ideas, and interest with others.

Nowadays cryptocurrency is a trending topic. And it should be because it is a novel technology where every transaction, volume, and list of every action is accessible and transparent. There is a really great opportunity in analysis of the data on cryptocurrencies as they are being traded for-profit and there is a lot of data on the internet that circulates about them.

Cryptocurrencies are a side product of another invention. Satoshi Nakamoto never intended to invent new currencies, but a new method of transaction which was more secure, and which was not controlled by a single entity [4].

"Announcing the first release of Bitcoin, a new electronic cash system that uses a peer-to-peer network to prevent double-spending. It's completely decentralized with no server or central authority." – Satoshi Nakamoto, 09 January 2009, announcing Bitcoin on SourceForge.

In light of this information, we decided to analyze the social media data and observe its effects on cryptocurrency price and user behavior

1.1 Purpose of the Thesis

The aim of this thesis to prove if there is any relation between social media and cryptocurrency price fluctuation. To analyze the relation of machine learning techniques used.

1.2 Literature Review

Martina Matta, Ilaria Lunesu, Michele Marchesi "Bitcoin Spread Prediction Using Social and Web Search Media" Study Social media (Twitter and Google Trends) data cross-correlation with Bitcoin price. They compare Tweets volume to Bitcoin price, Positive Tweets to Bitcoin price and Google Trends to Bitcoin price. And as a result, Google Trends has a kind of predictor for the Bitcoin price [5].

Ali Hasan, Sana Moin, Ahmad Karim and Shahaboddin Shamshirband "Machine Learning-Based Sentiment Analysis for Twitter Accounts" Analyze the different sentiment analyzers and their efficiency. As a result, TextBlob has the best score among others [6]. Ifigeneia Georgoula and others "Using Time-Series and Sentiment Analysis to Detect the Determinants of Bitcoin Prices" 2015, use SVM for sentiment analyzer on twitter's text then apply regression machine learning model and found out twitter has short term influence on bitcoin price [7].

Jermain C. Kaminski, MIT Media Lab "Nowcasting the Bitcoin Market with Twitter

Signals "2016, in that paper he studied cross-correlation between emotions on tweets and bitcoin close price. As a result, negative tweets have a moderate correlation with bitcoin [8].

Evita Stevqvist, Jacob Lönnö "Predicting Bitcoin price fluctuation with Twitter sentiment analysis" 2017, Twitter text sentimental effect on bitcoin price prediction analyzed with one-month data. Shifting operation done to predict different time intervals. As a result, 1 to 3 hours shifting has the best accuracy score. Shifting the data for preparing the prediction dataset is a good approach [9]. Most of the study's weak point is a data volume. Studies emphasis that with the larger and long term data better scores may achieve. Sean MacNally, School of Computing, National College of Ireland, "Predicting the Price of Bitcoin Using Machine Learning", used 3 years' bitcoin data for classification the bitcoin price increase and decrease. As a result, accuracy between different models lies around %50 to %52 [10].

In this study, unlike other studies, machine learning methods are employed instead of cross-validation methods. Data is collected for 9 months with a much higher volume compared to the other studies. Also; this research yields a 97 percent accuracy score which is much higher than the previous studies.

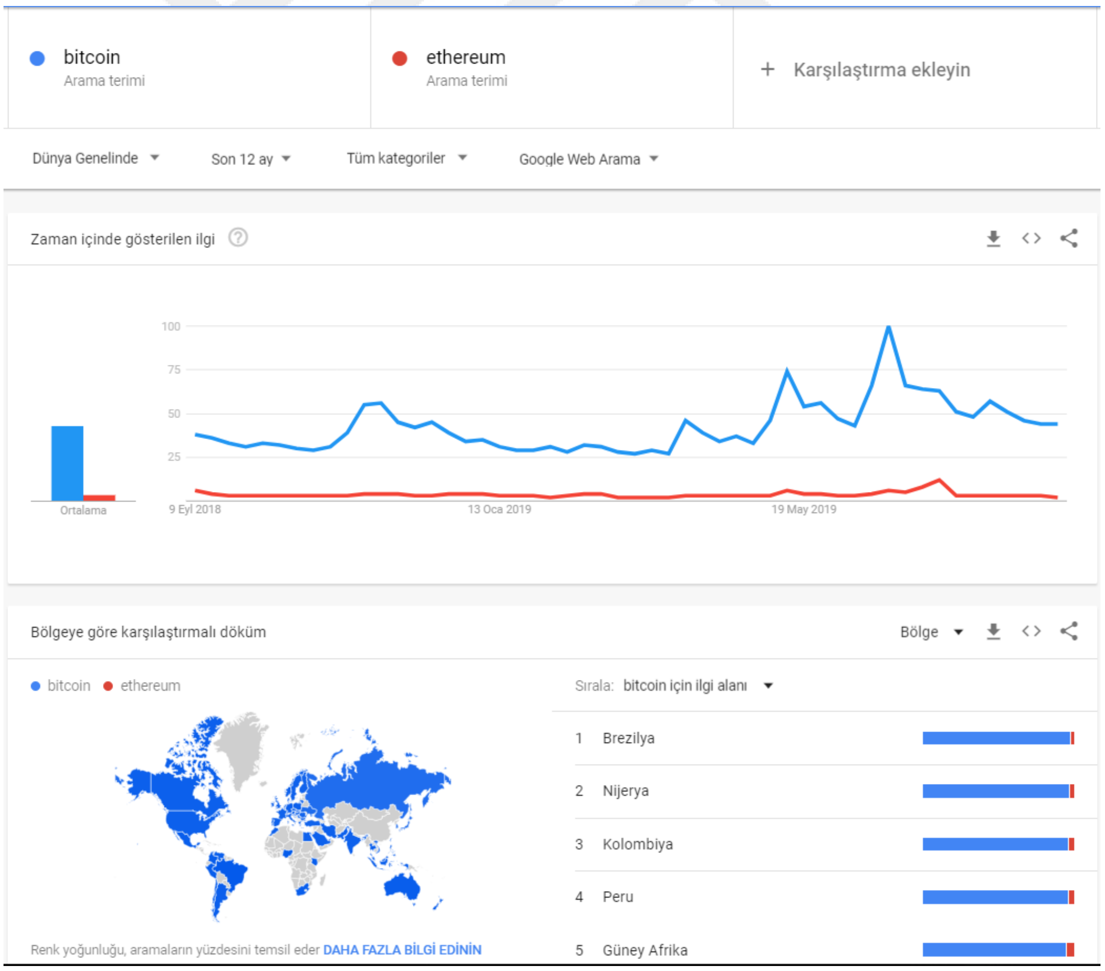
2. BACKGROUND

2.1 Google Trends

Google Trends is a service which provides google search statistics according to search parameter, location, language and time interval. Marketing companies or individual people using this kind of information.

Google Trends offer user interface and API solutions to reach the google trend service [11].

Google trend search percentage adjusted by total google searches and grouped according to geography and time range. This helps the fair adjustment, if not geography which has the highest search volume always has the highest search percentage. After that search results scaled between 0 to 100 according to topic's proportion to all searches.


● bitcoin
Arama terimi

● ethereum
Arama terimi

+ Karşılaştırma ekleyin

Dünya Genelinde · Son 12 ay · Tüm kategoriler · Google Web Arama

Zaman içinde gösterilen ilgi

Bölgeye göre karşılaştırmalı döküm

● bitcoin ● ethereum

Sırala: bitcoin için ilgi alanı

1	Brezilya
2	Nijerya
3	Kolombiya
4	Peru
5	Güney Afrika

Renk yoğunluğu, aramaların yüzdesini temsil eder DAHA FAZLA BİLGİ EDİNİN

Figure 2.1 Google trend website.

2.2 Cryptocurrency

Bitcoin is the most known and most important cryptocurrency. Satoshi Nakamoto who is cryptology and computer science experts began to develop Bitcoin in 2007. As a part of the bitcoin implementation blockchain database structure designed and used.

What is so special about Bitcoin?

Decentralized control system or distributed ledger. It means that not controlled by a group or central authority. Member of blockchain network are called Nodes. And each transaction data sends to Nodes (users) by using Peer to Peer Network synchronization.

This decentralization managed by the blockchain system. It named blockchain because of the structure of the database like adding blocks to the previous block by using cryptographic hash. Blocks can be created by blockchain miners or valid transactions. Cryptographic hash guarantee that data is not modified, blockchain structure not allow the changes in the data. And each transaction assures more secure the blockchain system [24].

This structure eliminates the centralized authority.

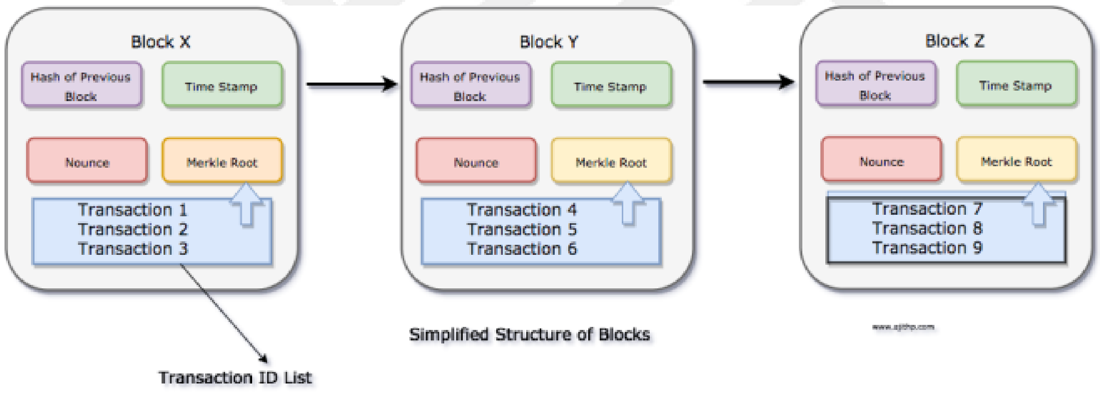


Figure 2.2 Blockchain structure.

2.3 Twitter

Twitter is an online news and social networking service that users able to post and interact with messages known as tweets. Twitter, based on the idea of status sharing and limited to 140 characters. Users share their ideas with their connections via twitter platform. Nowadays twitter considered as more information network than social network platform.

Twitter's monthly active users were 330 million and 2019 first-quarter revenue is 787 million dollars. This number recently continues to increase [3].

Twitter also shares its API (Application Programming Interface) by the name of the Twitter developer. User could subscribe to twitter developer platform and use twitter API for their analysis. Twitter API provides location, tweet text, friends number, time, retweet et cetera features. That is why twitter is a very important data provider platform for data analytics [12].

2.4 Machine Learning

Machine Learning is a mathematical algorithm that a computer system performs designed task without instructions. Machine learning based on computational statistics, which is inherited from statistic and probability theory [13].

Machine learning based on predictive modeling, with training data machine learning model learn the pattern by using process and techniques. Machine learning model gets experienced after performing training task so now have the ability to perform predictions with new and unseen cases.

2.4.1 Types of machine learning algorithms

2.4.1.1 Supervised learning

In supervised learning model data should be labeled, it means that input and output fields taught to the computer by human experts. So the model finds the best-fitted relationships and dependencies between input and output fields.

Supervised learning includes 2 main usage categories. One is Regression and the other is Classification problems. Regression technique based on estimating the best proper values and relationships between variables. Classification on the other hand separates the values to the group according to their features.

Regression Algorithms: Linear Regression, SVR (Support Vector Regression), Decision Tree, ect.

Classification Algorithms: SVM (Support Vector Machines), Neural Networks, Nearest Neighbor, Naive Bayes, etc.

2.4.1.2 Unsupervised learning

Inputs and outputs of the machine learning model is unknown, data is not labeled. There is no human interaction it means no teacher factor. Unsupervised machine learning method used to detect patterns and cluster the data.

Unsupervised Learning Algorithms: K_means Clustering, Hidden Markov Model, Gaussian Mixture, Hierarchical.

2.4.1.3 Semisupervised learning

Data contains labeled and unlabeled features. Unlabeled data is being labeled by using assumption techniques like continuity, cluster or manifold. Board games and self-driving cars are a good example of showing studies on that learning model.

Semisupervised Learning Algorithms: Deep Adversarial Networks, Temporal Difference(TD), Q Learning.

2.4.2 Machine learning models

I would like to introduce some machine learning models and logical approach behind the model.

2.4.2.4 Linear regression

Linear Regression is a type of regression analysis based on independent X variables and dependent Y variables linear relationship. Based on data points best fit straight line is plotted.

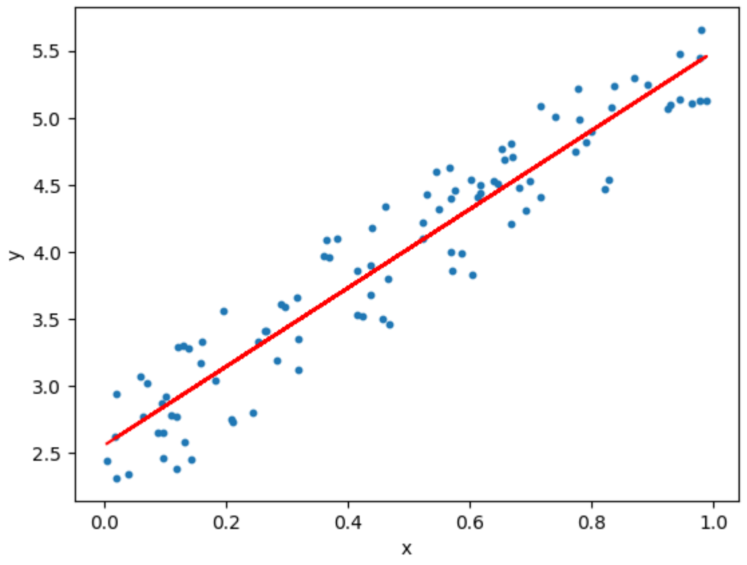


Figure 2.3 Linear regression logic.

Retrieved August 28, 2019, https://towardsdatascience.com/linear-regression-using-python-b136c91bf0a2

2.4.2.5 Lasso regression

Lasso: least absolute shrinkage and selection operator. Lasso also a linear regression method but with shrinkage and variable selection features lasso minimize the prediction error.

Shrinkage: shrinks coefficients toward zero. The shrinkage process provides a better interpretation of the model.

Selection: In this process, the most important X variables discovered according to association with the response Y variable.

2.4.2.6 Ridge regression

Ridge regression also linear regression model and very similar to the Lasso model. In lasso model shrinks coefficients toward zero but Ridge model does not set the coefficients to zero but force them to be lower. This behavior decreases the impact of the irrelevant features on the model.

2.4.2.7 SVM (Support Vector Machine)

SVM classification model of the SVR. The idea between two models is the same.

SVR model two boundary lines defined according to the least error rate, to minimize the error maximize the margin. Regression made according to closest to the original hyperplane or within the boundary line.

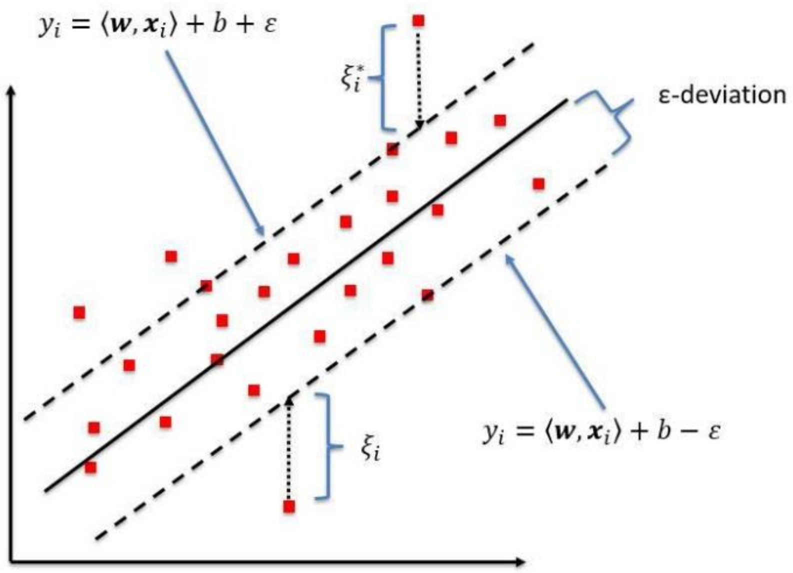

$y_i = \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b + \varepsilon$
ξ_i^*
ε-deviation
ξ_i
$y_i = \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b - \varepsilon$

Figure 2.4 SVR model regression logic.

Retrieved April 20, 2019, from https://medium.com/coinmonks/support-vector-regression-or-svr-8eb3acf6d0ff

For the classification problems goal is to design hyperplane that classifies all data. The idea is the maximize the margin between two different classes for designing better hyperplane,

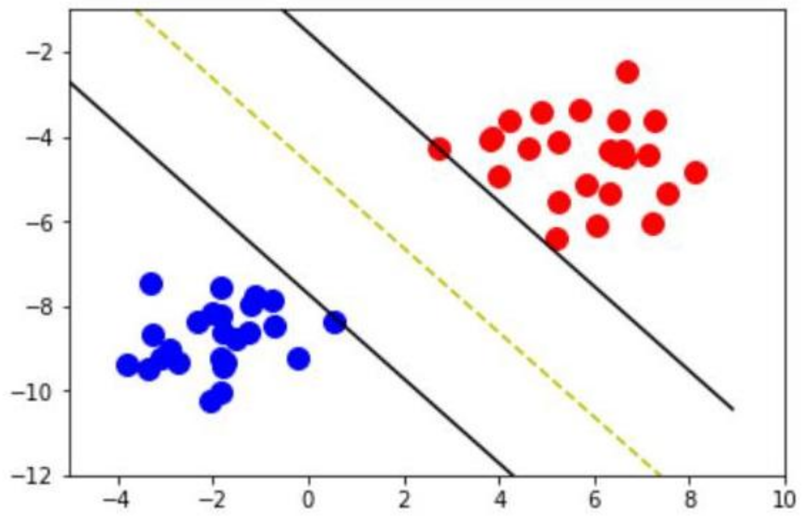


Figure 2.5 SVM model classification logic.

Retrieved April 20, 2019, from https://medium.com/coinmonks/support-vector-regression-or-svr-8eb3acf6d0ff

2.4.2.8 KNN (k-nearest neighbors)

Knn algorithm based on the distance of the selected data point to other data points. Knn can be used as a classification or regression model. Data classified by the majority of the neighbors. For the regression average of the k-neighbors values used [23].

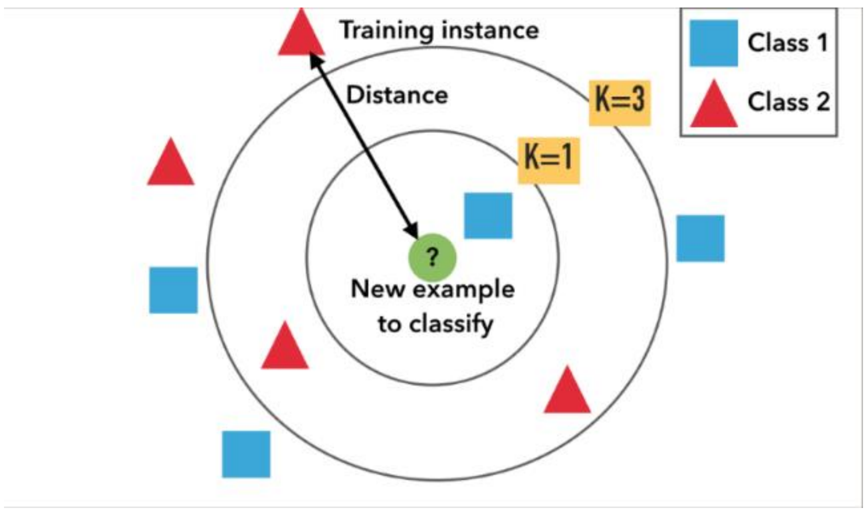


Figure 2.6 K_nearest neighbors logic.

Retrieved July 07, 2019, from https://towardsdatascience.com/

2.4.2.9 Decision tree

Decision Tree can be considered as conditional control statements. Each feature represents nodes, each branch represents decision rule and leaf represent outcome. One of the important part is how to select root node. Root node must be the best classifies attribute the training data.

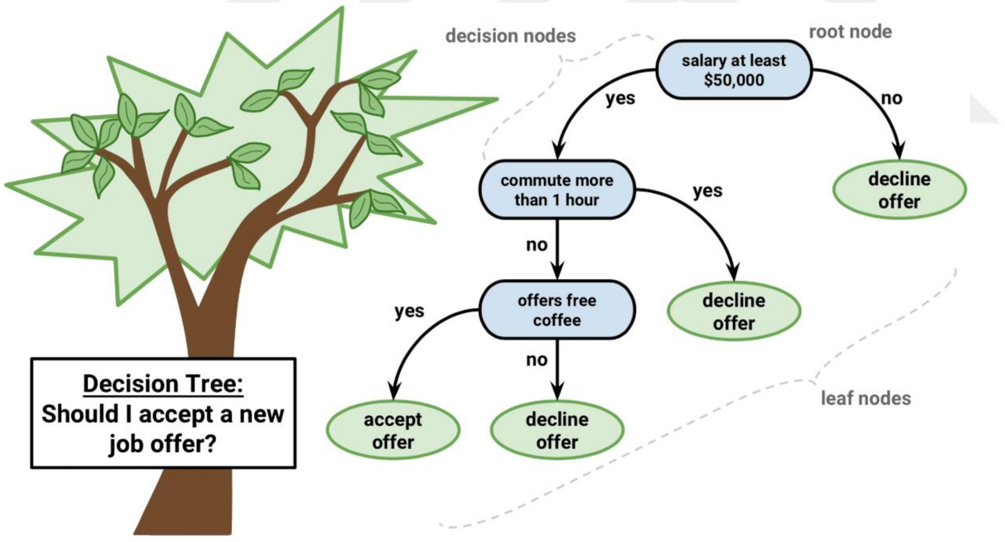


Figure 2.7 Decision tree logic.

Retrieved July 07, 2019, from http://dataaspirant.com/2017/01/30/how-decision-tree-algorithm-works/

2.4.2.10 Random forest

Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction

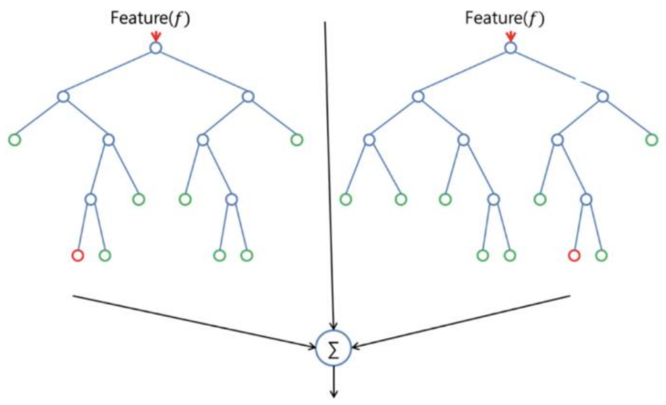


Figure 2.8 Random forest logic.

2.4.2.11 Naïve bayes

There are some classification models based on Naïve Bayes theorem which are Gaussian and Bernoulli models. This theorem assumes a probabilistic model. Naïve assumes independence among predictors. Method basically calculates the probability of a point or data belonging to a certain class based on its attributes.

There is an example to clarify the Bayesian theorem;
X: 35 years old customer with an income of $40.000 and a fair credit rating.
H: Hypothesis that the customer will buy a computer?
P(H|X): The probability of the customer will buy a computer based on customer information like age credit record and income (Posterior probability of H).
P(H): The probability of the customer will buy a computer regardless of the customer's other features (Prior probability of H).
P(X|H): Probability that customer X with fair earning bought the computer (Posterior probability of X).
P(X): Probability that a person in the dataset met the same requirement with customer X (Prior probability of X).

$$P(H|X) = \frac{\mathrm{P(X|H) * P(H)}}{\mathrm{P(X)}}$$

2.4.3 Machine learning model scorer

Machine learning model scores used to know if the machine learning model has good performance on the data set. Scores give numerical values so it helps compare the model's efficiency with each other. Also by checking the score value, it is seen that if the data set suitable for the selected machine model or not.

There are a lot of scoring for each model like F1 score, R^2, Accuracy, Recall, Precision, roc_auc, etc. In our study, we evaluate R^2 score for regression and accuracy score for classification models.

2.4.3.12 R^2 score

R^2 score or coefficient of determination. The best possible score is 1.0 for this scorer.

R^2 score may be a negative value, it means that chosen model fits worse than horizontal line through the mean of the data. As a result, if the R^2 score is negative chosen model not fit for the dataset.

$$\boldsymbol{SS_{Total}} = \sum (yi - \bar{y})^2$$

$$\boldsymbol{SS_{Regression=}} \sum (yi - yregression)^2$$

$$\boldsymbol{R^2 = 1 - \frac{SS_{Regression}}{SS_{Total}}}$$

SS Regression Error is the distance between the original data points and the regression line. Sum Squared Total Error is the distance between the original data points and the mean of the original values.

2.4.3.13 Accuracy score

The accuracy score shows how accurate the prediction is. Accuracy score metric used for classification models.

The confusion matrix is used to present the prediction results. We can understand the accuracy score by analyzing the confusion matrix.

Table 2.1 Confusion matrix.

		Predicted Value	
		Yes	No
Actual Value	Yes	True Positive	False Negatives
	No	False Positives	True Negatives

True Positive (TP): Correctly predicted positive values. Predicted yes and it is true.

True Negatives(TN): Correctly predicted negative values. Predicted no and it is true.

False Positives(FP): The actual value is no and predicted is yes.

False Negatives(FN): Actual class is yes but predicted class is no.

The accuracy score measures the ratio of the correctly predicted observation of the total observations.

$$Accuracy = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$$

There are other scorers like Precision, Recall and F1 scores.

Precision score called positive predictive value.

$$Precision = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

Recall score called sensitivity. Positives events that predicted correctly.

$$Recall = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

F1 score is the harmonic mean of the precision and the recall. Higher F1 score means better model.

$$F1 = \frac{2}{\frac{1}{\mathrm{precision}} + \frac{1}{\mathrm{recall}}}$$

$$F1 = \frac{2 * (\mathrm{precision} * \mathrm{recall})}{\mathrm{precision} + \mathrm{recall}}$$

3. METHOD

3.1 Purpose

This study began with the literature review. Literature review help to understand the other studies on prediction methods based on social media data. In this chapter followed methods and technologies are described in chronological order.

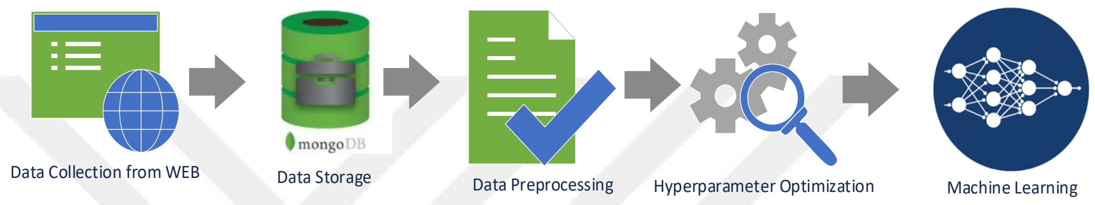


Figure 3.1 Machine learning steps in order.

3.1.1 Data collection

Data is the most important element of the machine learning process. Good structured and continuous data may lead to success in machine learning. Data collection is the most challenging and time resuming part of this study. Maintenance, storage, and continuity of the data needs to be well design and implement. For continuous data collection, 7/24 available and online computer needed. For that purpose, Mac Os 1 TB computer on the data analysis laboratory at the ITU used.

In case of environmental incapability like power off, computer restart or killing data collection job, bash script task is written. Bash script task's main purpose is checking if the data collection job is running or not in every 15 min. If the data collection job is not running starting the task automatically. For automatic starting cron job software utility used.

The software utility cron is a time-based job scheduler in Unix-like computer operating systems. People who set up and maintain software environments use cron to schedule jobs (commands or shell scripts) to run periodically at fixed times, dates, or intervals [13].

3.1.1.1 Cryptocurrency data

Top 100 cryptocurrency live data collected and stored to the database by using CoinMarketCap API [14].

Cryptocurrency data collected every 5 sec interval. Data include these laballed fields; id, name, symbol, rank, price_usd, price_btc, 24h_volume_usd, market_cap_usd, available_supply, total_supply, max_supply, percentage_change1h, percentage_change_24h, percentage_change_7d, last_updated.

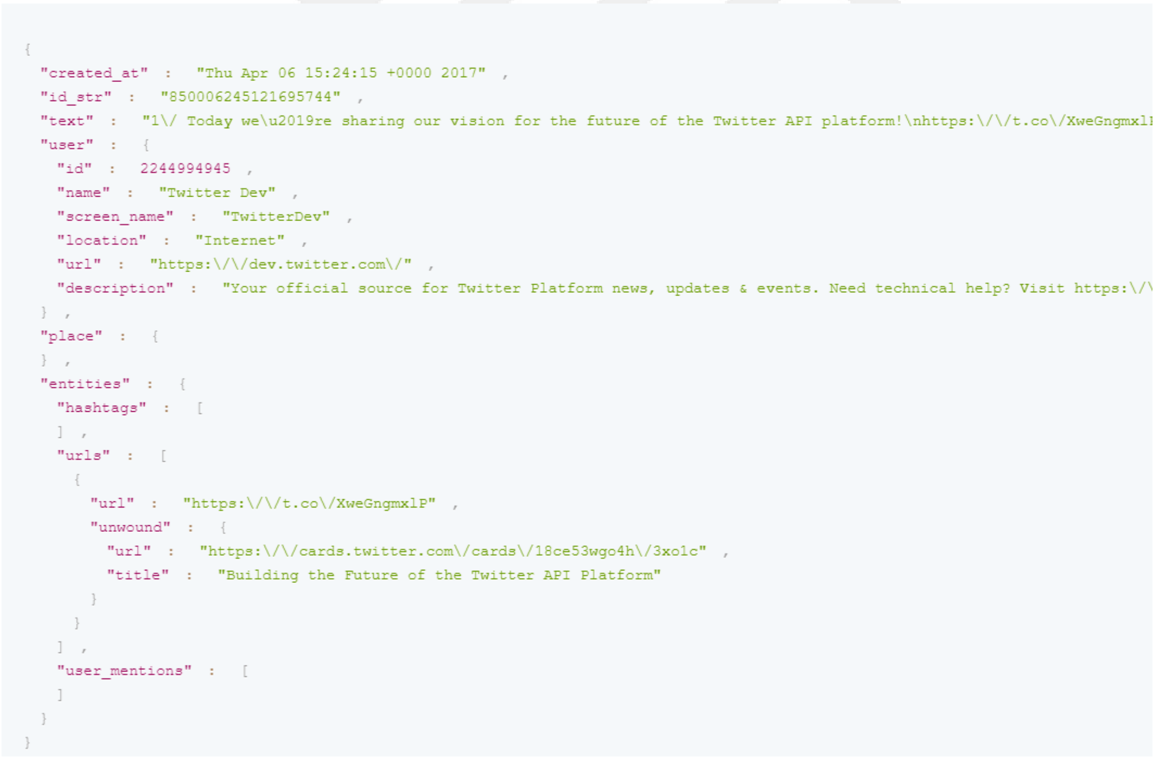
Table 3.1 Captured cryptocurrency data columns.

id	name	symbol	rank	price_usd
price_btc	24h_volume_usd	market_cap_usd	available_supply	total_supply
max_supply	percentage_change1h	percentage_change_24h	percentage_change_7d	last_updated

3.1.1.2 Twitter real time data

Twitter data stream by using the twitter developer API. API provides quite useful endpoints to the developers and it is very important for social media analyzing. For more info, visit the twitter developer API [15].

Data is streamed by twitter using keywords which are the most popular cryptocurrencies like Bitcoin, Ethereum, Dash, IOTA, Ripple, Litecoin, EOS, XRP, Tether, etc. Streamed data stored as JSON file with the tag of current date timestamp in the dedicated computer. Data collected from August 2018 to April 2019 so to maintain the computer storage .json file zipped automatically the next day.



```
{
  "created_at"  :  "Thu Apr 06 15:24:15 +0000 2017"  ,
  "id_str"  :  "850006245121695744"  ,
  "text"  :  "1\/ Today we\u2019re sharing our vision for the future of the Twitter API platform!\nhttps:\/\/t.co\/XweGngmxlP
  "user"  :  {
    "id"  :  2244994945  ,
    "name"  :  "Twitter Dev"  ,
    "screen_name"  :  "TwitterDev"  ,
    "location"  :  "Internet"  ,
    "url"  :  "https:\/\/dev.twitter.com\/"  ,
    "description"  :  "Your official source for Twitter Platform news, updates & events. Need technical help? Visit https:\/\
  }  ,
  "place"  :  {
  }  ,
  "entities"  :  {
    "hashtags"  :  [
    ]  ,
    "urls"  :  [
      {
        "url"  :  "https:\/\/t.co\/XweGngmxlP"  ,
        "unwound"  :  {
          "url"  :  "https:\/\/cards.twitter.com\/cards\/18ce53wgo4h\/3xo1c"  ,
          "title"  :  "Building the Future of the Twitter API Platform"
        }
      }
    ]  ,
    "user_mentions"  :  [
    ]
  }
}
```

Figure 3.2 Tweet stream JSON objects.

3.1.1.3 Google trend data

According to previous research google trend’s data has an influence on the Bitcoin price (Martina Matta, 2015). So, google trends data included in this research.

Google trend is a website that analyzes the google search queries according to location, language or available criteria. Also, provide API for the developers to do their analysis. Bitcoin and Ethereum search percentage and time interval data gathered from google trend. The minimum available time interval was the 8min at that time (when the data collection is began for this thesis) but now less time interval available on the API [11].

3.1.2 Data storage

Data collection continue for 9 months started from Agust 2018 to April 2019. Data collected from three different sources; Twitter, Google Trend, and CoinMarketCap API. To work with different types of data sources efficiently, data stored in the database.

MongoDB database used to store the data, there is some reason to select MongoDB in our research. First of all, open-source library and very easy to install. Have a developer community to get answers to possible problems. MongoDB supports search by regex, this feature very useful to search keyword in the tweets [16].

For each data source, three different databases created and desired data inserted with understandable column naming.

Google Trend and Cryptocurrency data neat and genuine enough so no need to do preprocessing before insert to database.

For twitter data, human interaction is needed. First of all, relevant twitter objects selected according to user experience. Twitter APIs returns many objects in JSON format, so no need to store unnecessary field to the database for the performance and storage maintenance. Selected tweet objects are; tweet_id, screen_name, user_id, followers, friends, location, following, statuses_count, timezone, text, hashtags, timestamp, quote_count, reply_count, retweet_count, favorite_count.

Tweet text object needs data preprocessing to better analysis outcome.

3.1.3 Data preprocessing

Data preprocessing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors. Data preprocessing is a proven method of resolving such issues [17].

Twitter text includes a lot of meaningless elements. To use text data in machine learning techniques some preprocessing should be done. Which kind of preprocessing necessary or not related to usage of the text data on the machine learning techniques.

3.1.3.4 Twitter text cleaning

Tweeter text includes meaningless emoji, character, URL, pictographs, flags, etc. These elements need to be cleaned for more successful sentiment analysis.

Clean_text function implemented on python to clean each tweet's text fields. Function basically removing emotions, symbols, flags, &amp, @, URL, hashtag, and retweets (RT) from the text.

3.1.3.5 Twitter text sentiment analysis

Sentiment analysis mainly about understanding the emotion inside the text. Text could be referred to negative, positive, happy, angry, bad feelings. As a human we have an understanding of feelings but how we teach the computers to understand the idea of a text. At that point neural language processing studies take place.

Sentiment analysis based on Natural Language Processing. Natural language processing (NLP) is a subfield of computer science, information engineering, and artificial intelligence concerned with the interactions between computers and human languages, in particular how to program computers to process and analyze large amounts of natural language data [13].

Twitter text sentiment analysis studies and best-fitted method defined according to study TextBlob technique has the highest percentage (Ali Hasan and others 2018).

TextBlob sentiment analysis technique used for scoring text polarity. The sentiment polarity feature enriches the dataset for the machine learning model.

Only the polarity score of TextBlob used in the data model. Polarity score gives an idea about text is negative or positive. Polarity scores between -1 to 1. -1 represent negative and 1 represent positive feelings.

Figure 3.3 TextBlob sentiment analysis on tweets.

3.1.3.6 Merging data

To create a successful data model, input variables which are X and output variable Y should be defined and arrange according to machine learning model. The aim of this thesis prediction bitcoin price data, so the output variable is clear. But input variables are collected at different time intervals. Twitter data is streaming, and time interval is millisecond. Cryptocurrency data captured every 5 seconds. Google Trend’s interval is 8 seconds.

First of all, the desired twitter columns selected from TwitterStream database. These are; text, favorite_count, followers, friends, listed_count, statuses_count, quote_count, retweet_count, reply_count, sentiment_polarity. For detailed information about columns and what it is representing please check the Twitter Developer API. Then data resampled to 5 min time interval. While doing that text column converted to count of text column. Other columns converted as mean of the values in selected time series.

From the cryptocurrency database, Bitcoin and Ethereum selected for machine learning analysis. Data resampled as 5 min interval and mean of the values considered as the new feature. Bitcoin data include these fields; date, price_btc_usd, percent_change_1h, percent_change_24h, percent_change_7d, h24_volume_usd, total_supply, market_cap_usd.

The last part merging the resampled data. Data merge on date column. Each of them is a time-series data so merging on the date is the most logical and efficient way. This way, a single and valid data frame is created by merging three other data frames.

All data operations performed by using Python. Python includes Pandas library. Pandas is an open-source library providing high-performance, easy-to-use on the data structures, and data analysis tools for the Python programming language [18].

Table 3.2 Data points after merge operation.

Time interval	DataFrame row-column count
5 minutes	59764
15 minutes	17991
30 minutes	9181
45 minutes	6207
60 minutes	4707
1 day	228

3.1.3.7 Data interpolation

Interpolation is an estimation of a value between two known values. [19]

After merging the data there are some missing time interval values observed. For missing values, the data interpolation method implemented to have continuous data.

3.1.4 Machine learning model hyperparameter optimization

Before the machine learning process, to get the best results from the model, grid-search should be applied to dataset. Grid search is an exhaustive search over specified parameter values for an estimator [20]. Grid search finds out optimal hyperparameters for accurate prediction. Some examples to the hyperparameters; k in k-nearest neighbors, C in SVM or kernel in SVR. The grid-search method mainly tries each parameter on your data model and brings the best score. Grid search also has cross-validation. Cross-validation necessary to avoiding overfitting. Overfitting means that your train data fit your machine learning model than test data.

To perform machine learning our data split into train and test. There is always a risk that losing some crucial data from the training set. To avoid that cross-validation method used on the data. Training data used for teaching the pattern of our data to the machine learning model. After that machine learning prediction done with test data.

3.1.5 Machine learning model

After finding the best hyperparameters, the machine learning method will be applied to the dataset. Dataset has labeled data so supervised learning like regression and classification model are suitable. There is not such a defined model or previous research for best fitted model so most popular and suitable machine learning models performed in this study. Another main criteria for selecting machine learning model is the scikit-learn library.

Scikilearn machine learning library in python. It is an open-source library and simple to use. Also, have rich documentation very useful for understanding the modules [20].

Selected regression models; SVR, Lasso, Ridge, Kneighbors, MLPRegressor, Decision Tree.

Selected classification models: SVM, Kneighbors, DesicionTree, GaussianNB, MLPClasifier, RandomForest.

Some method has both regression and classification model. So, it is a good opportunity to observe which method is more suitable and better estimation for the dataset.

3.1.6 Time intervals and shifts

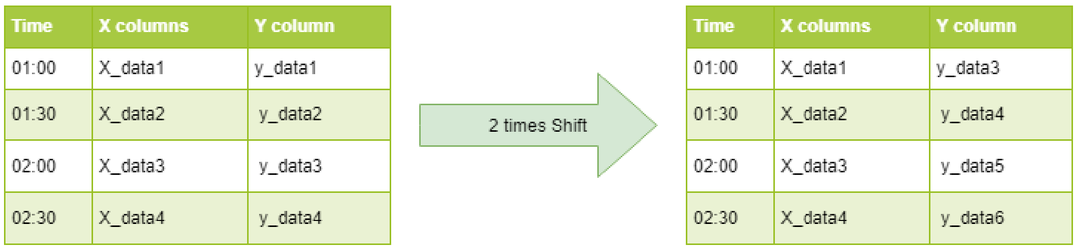
To substance the possible identification of correlation between cryptocurrency price change and social media data, two temporal aspects are considered; time interval (frequency length) and shift.

Table 3.3 Time intervals.

Intervals
5 minutes
15 minutes
30 minutes
45 minutes
60 minutes
1 day

Each time series is evaluated by four different shifts forward: 1, 2, 3, and 4.

For example, if data frame time interval (data frequency) is 30 min and shifted 4 times prediction made for 30 min*4 and it is 120 min which is 2 hours later.



Time	X columns	Y column
01:00	X_data1	y_data1
01:30	X_data2	y_data2
02:00	X_data3	y_data3
02:30	X_data4	y_data4

Time	X columns	Y column
01:00	X_data1	y_data3
01:30	X_data2	y_data4
02:00	X_data3	y_data5
02:30	X_data4	y_data6

Figure 3.4 Shifting logic.

4. RESULTS

4.1 Data Collection

Data collection began in August 2018 until April 2019. Nine months' data available for the machine learning process. Some data missing because of environmental problems like internet connection problems or power cut off.

4.1.1 Twitter data

During the data collection total 66430752 cryptocurrency related data collected via twitter API. While streaming twitter data keywords used to collect related data.

Keywords: 'Bitcoin', 'Ethereum', 'Ripple', 'Litecoin', 'Cardano', 'Stellar', 'Monero', 'IOTA', 'Dash', 'TRON', 'Tether', 'Qtum','OmiseGO', 'Zcash','Bytecoin', 'altcoin','BTC', 'ETH', 'XRP', 'BCH', 'LTC', 'XLM', 'XMR', 'DASH', 'XEM', 'TRX', 'USDT'

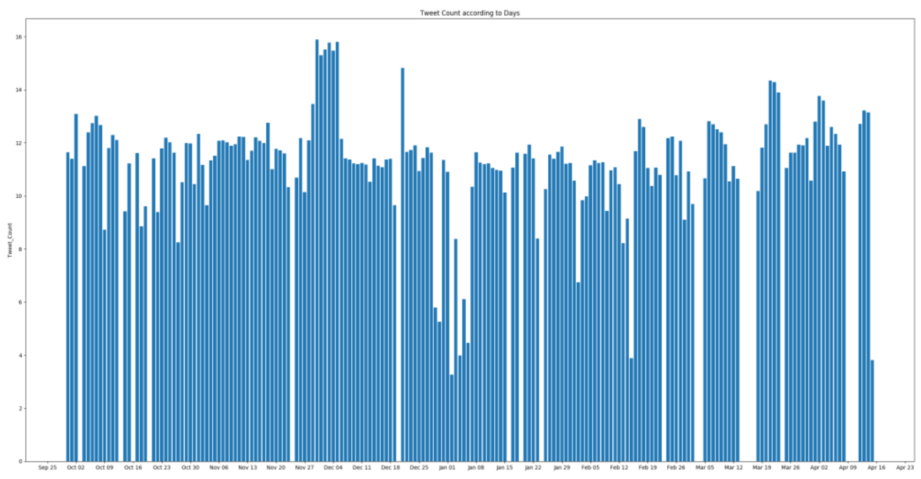


Figure 4.1 Twitter data daily distribution plot.

4.1.2 Google trend data

From google trend API total 1652202 data collected. As keyword Bitcoin and Ethereum used. Google trend returns search percentage of the keywords on google.

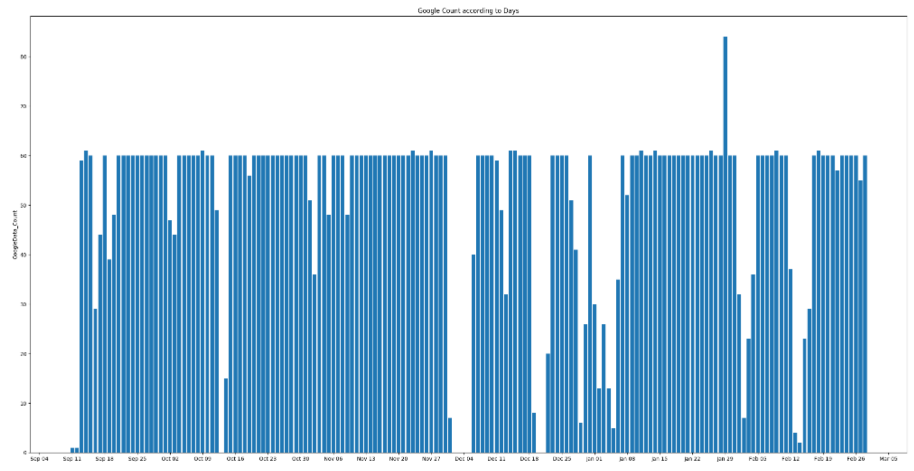


Figure 4.2 Google trend data daily distribution plot.

4.1.3 Cryptocurrency data

From the CoinMarketCap API total 7457337 data collected. There is not any keyword option for that but CoinMarketCap returns the top 100 cryptocurrencies price, volume and transaction information. And top 2 cryptocurrencies are Bitcoin and Ethereum. In dataset only Bitcoin and Ethereum related data are filtered and used.

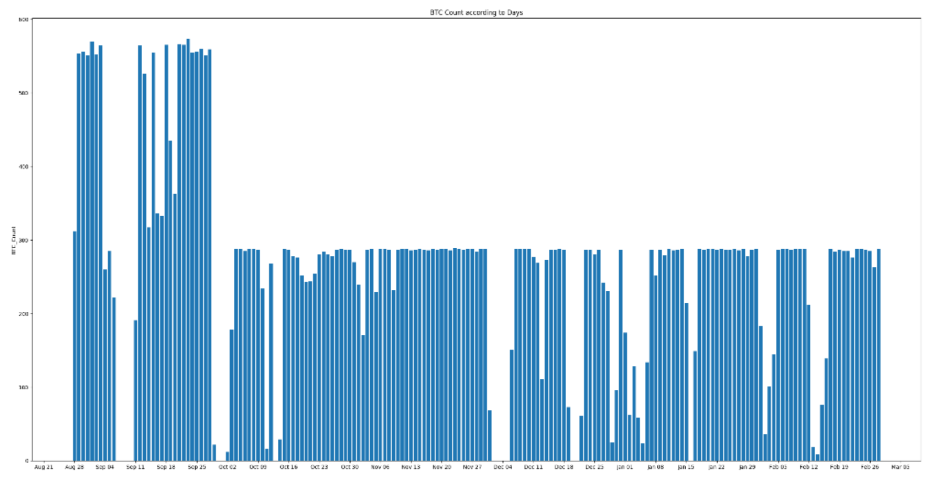


Figure 4.3 Cryptocurrency daily distribution plot.

4.2 Grid Search Optimum Hyperparameters

Scikit-learn is a python machine learning API. Grid search is one of the model selection and evaluation module. Grid search working logic is mainly trying each parameter as a matrix and return the best accuracy.

Grid search applied to each data set for regression and classification models. Data sets created according to different time intervals. Time intervals are 5 min, 15 min, 30 min, 45 min, 1 hour and 1 day.

Table 4.1 Grid search optimum parameters for 5 min interval regression model.

Parameters	Lasso	Ridge	K-Neighbors	MLP	Decision Tree	Linear SVR	SVR
alpha	0.1	0.01	-	-	-	-	-
epsilon	-	-	-	1e-07	-	0.1	-
kernel	-	-	-	-	-	-	linear
C	-	-	-	-	-	100	100
gamma	-	-	-	-	-	-	auto
tol	-	-	-	-	-	0.001	-
max_dept	-	-	-	-	16	-	-
min_sample	-	-	-	-	4	-	-
n_neighbours	-	-	2	-	-	-	-
shuffle	-	-	-	False	-	-	-

Grid search returns best possible parameters for each model. Then related hyperparameters used on the machine learning models.

Table 4.2 Grid search optimum parameters for 5 min interval classification model.

Parameters	Random Forest	K-Neighbors	Decision Tree	MLP	Gaussian NB	Linear SVC	SVR
max_depth	10	-	4	-	-	-	-
min_sample_split	100	-	2	-	-	-	-
n_neighbors	-	25	-	-	-	-	-
alpha	-	-	-	1e-05	-	-	-
epsilon	-	-	-	1e-08	-	-	-
learning_rate_init	-	-	-	0.0001	-	-	-
shuffle	-	-	-	True	-	-	-
var_smoothing	-	-	-	-	1e-10	-	-
C	-	-	-	-	-	1	1
multi_class	-	-	-	-	-	ovr	ovo
tol	-	-	-	-	-	0.0001	0.1
kernel	-	-	-	-	-	-	rbf
gamma	-	-	-	-	-	-	auto

4.3 Machine Learning Process

To detect the relation between bitcoin price and social media data machine learning method used. Dataset contains labeled data so the supervised learning model selected. Supervised learning has 2 categories one is regression and the other is classification. To use both model dataset needs to be arranged accordingly.

4.3.1 Regression model and dataset

Regression is the task of predicting a continuous quantity [21]. Also, a predictive model function according to the relation between input and output variables. The first step is to check if the dataset will be fitted with the regression model. Dataset contains the date column and ordered by date, so it is time-series data. While collecting the data aim was having continuous data, so dataset is also continuous. The third feature of the dataset is labeled columns. And all columns already converted to numerical values. As a result, we can assume that our dataset consistent with regression models.

The most important factor of machine learning is data and data is ready for the regression method. In this study time interval of data and prediction of best accuracy relation is observed. To achieve that data merged and shifted in defined time interval. For example, to predict the bitcoin 5 min later price data merged 5 min time interval, then the bitcoin column shifted 1 raw. The machine learning model applied and gives the relation function between input variables X to next 5 min output y variable.

The grid-search result is set to the machine learning model to get better estimation sores. The most popular and different approach machine learning model selected. While selecting the machine learning model compatibility with the data is considered. If the regression score between 0 to 1 selected model suitable for the data set. If the score is minus, the selected machine learning model is not suitable for the data set.

For regression model's metric r² scores observed for nine regression models which are Linear regression, Lasso, Ridge, K_Neighbours, Decision Tree, SVR, Linear SVR, MLP (Multi-Layer Perceptron regressor) and Gaussian Processes regressor. MLP and Gaussian Process regressor model does not fit for the dataset. So other 7 model analyzed for the regression. 4 times shifting done for each regression model. Shifting time intervals are 5 min, 5*2 min, 5*3 min, 5*4 min, 15 min, 15*2 min …. Until 1*4 Day shift. 5*2 minutes shift means time interval is 5 minutes and X and y column calculated accordingly and y column shifted 2 times so predicted value belongs to 10 min later's data.

Linear Regression, Ridge and Lasso models give very good estimation scores until 1Day time interval. 1Day time interval data count may not be enough for these regression models. But data model fits very good with Linear Regression, Ridge, and Lasso.

K-neighbors, Decision Tree, and SVR models give acceptable estimation scores only for 5 min time interval data sampling. After 5 min r² score decrease prominently.

On the other hand, Linear SVR model fit to 5 min and 15 min resampled dataset very well but after 15 min r² score decrease. It means that not a good match for the data set and regression model for 30 minutes, 45 minutes, 60 minutes and 1 day resampled data.

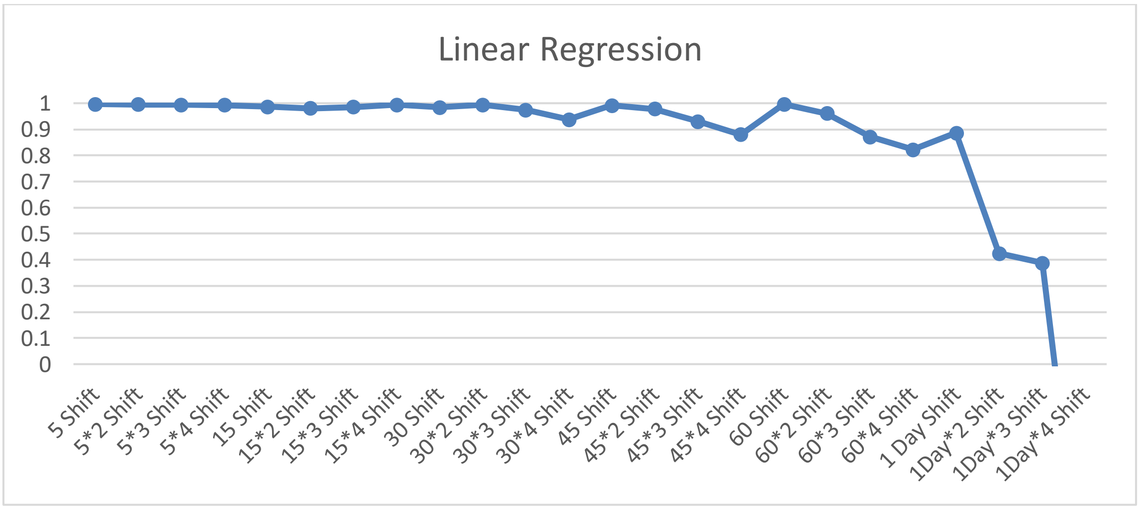


Figure 4.4 Linear Regression r² scores for each shifting.

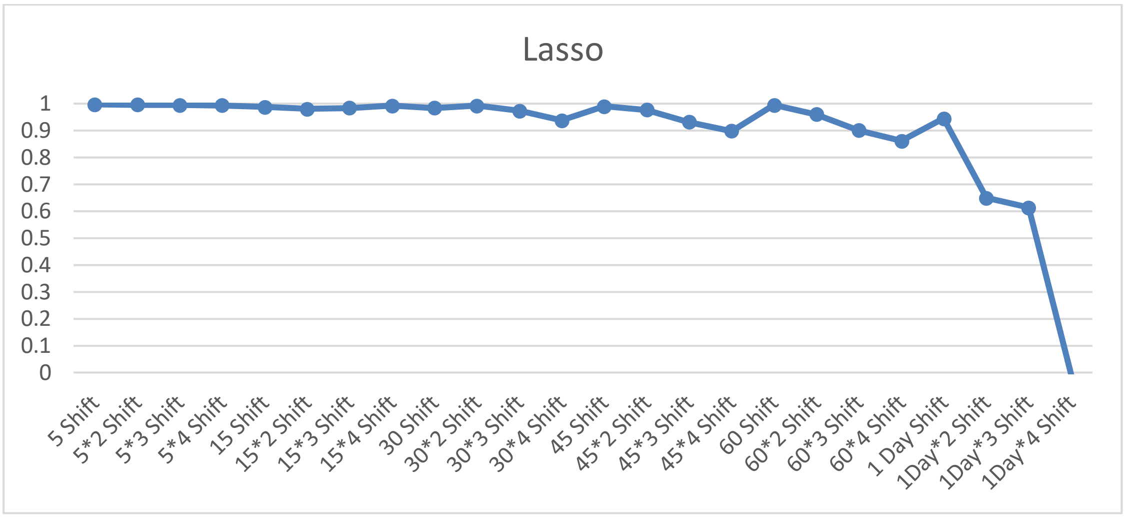


Figure 4.5 Lasso r^2 scores for each shifting.

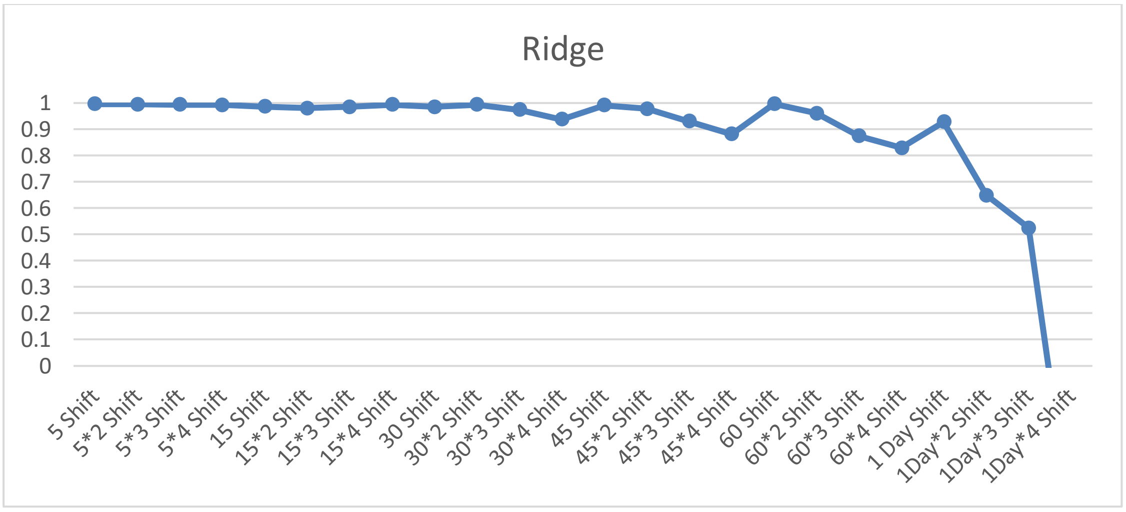


Figure 4.6 Ridge r^2 scores for each shifting.

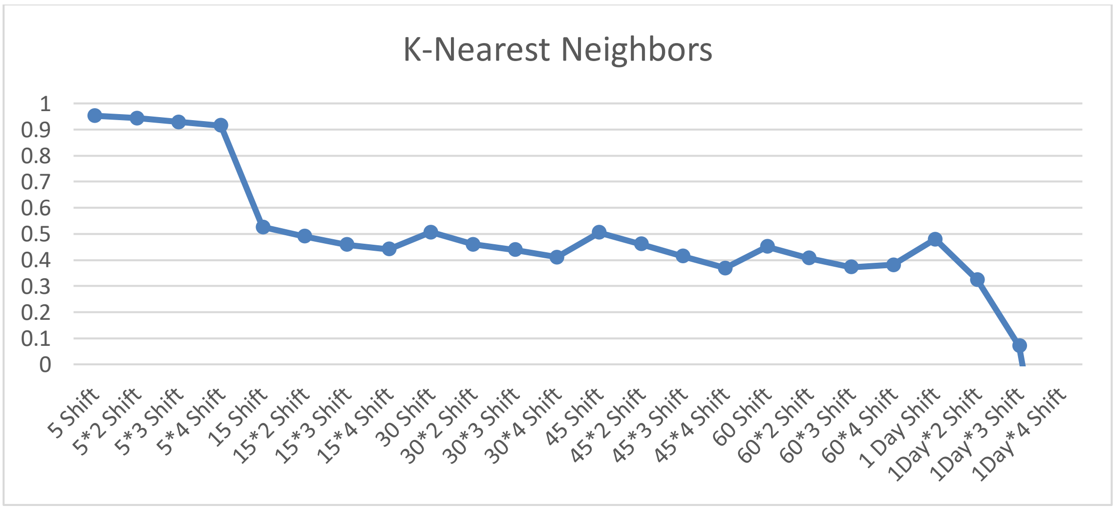


Figure 4.7 K-Nearest Neighbors r^2 scores for each shifting.

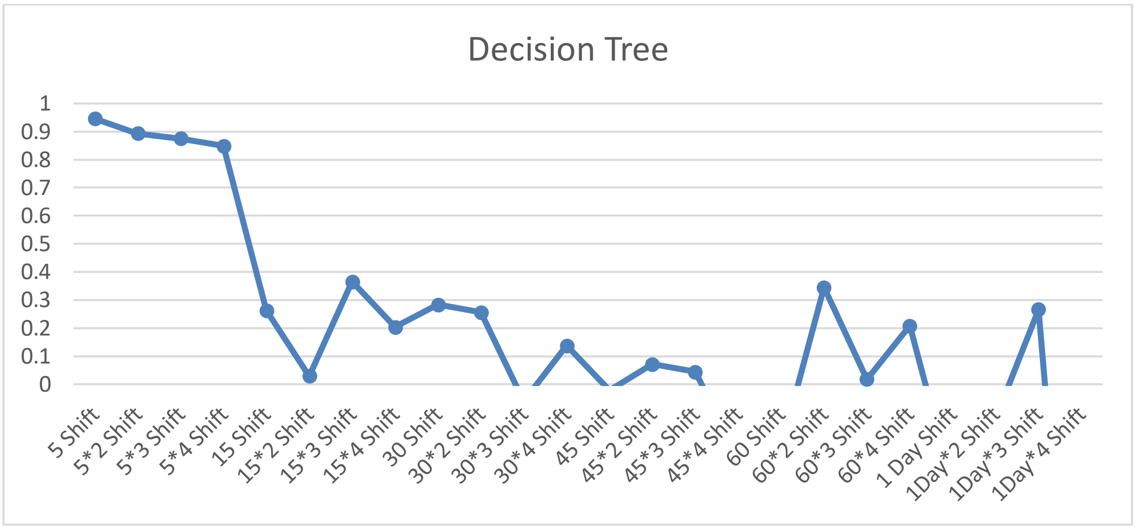


Figure 4.8 Decision Tree r² scores for each shifting.

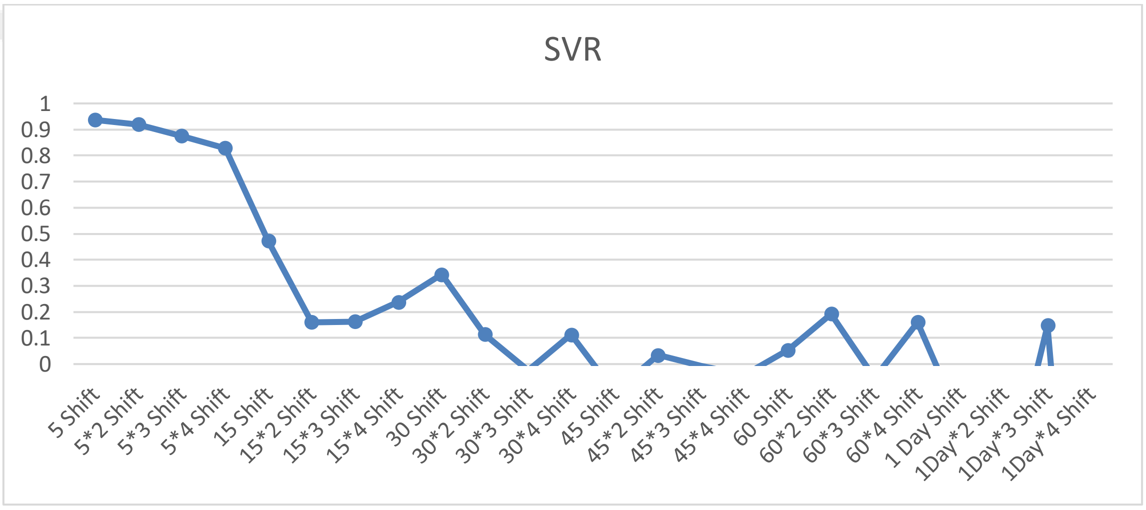


Figure 4.9 SVR r² scores for each shifting.

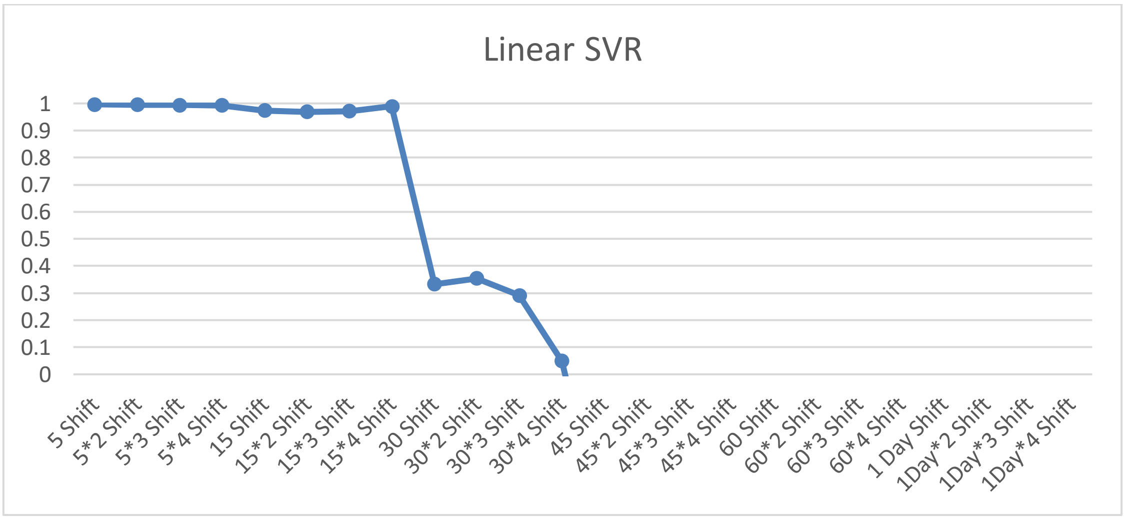


Figure 4.10 Linear SVR r² scores for each shifting.

For better comparison, all regression models R² scores plotted in one graph. Figure 4.11 easily observed that linear models are more powerful than other models.

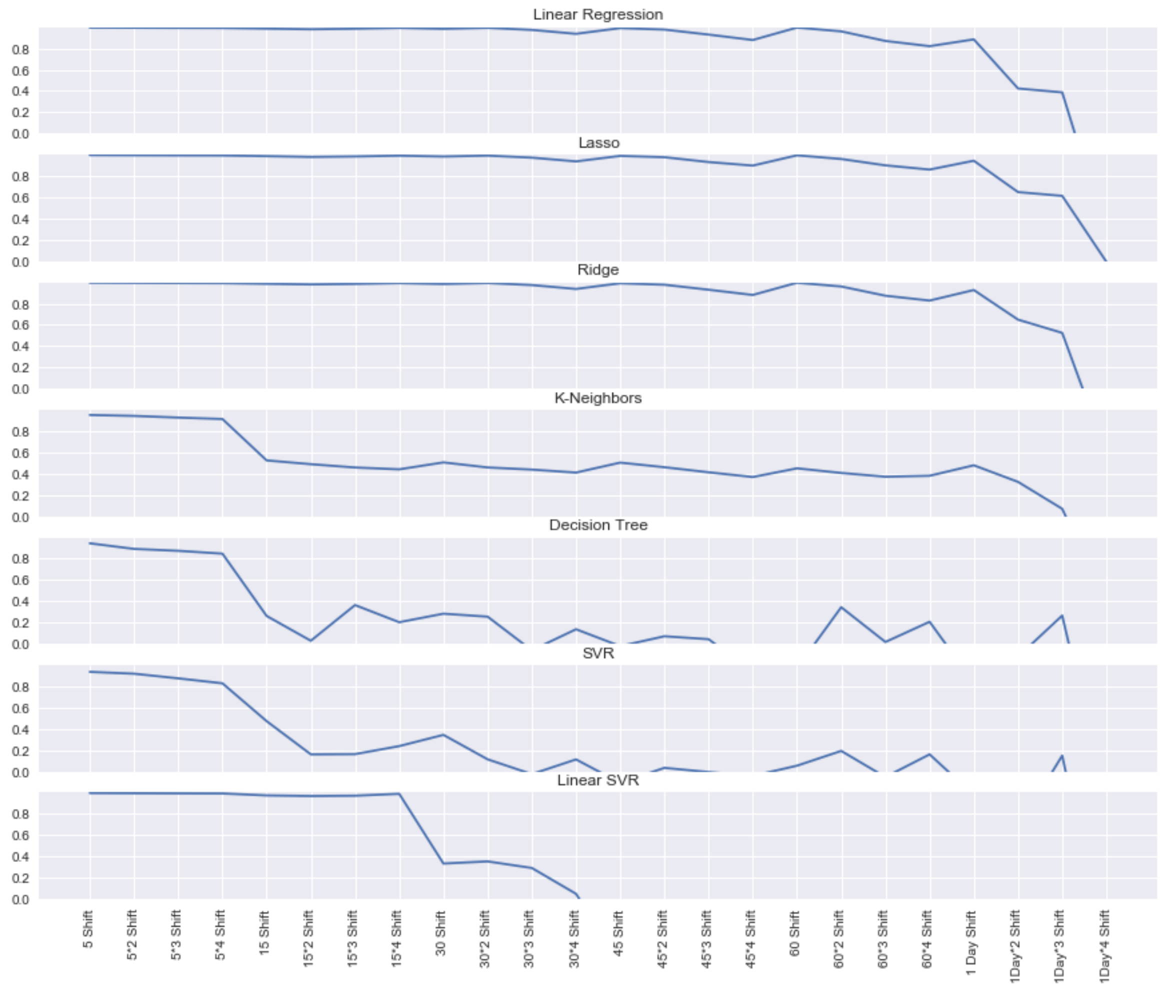


Figure 4.11 Regression r² scores for all models.

4.3.2 Classification model and dataset

Classification models and regression models have similar logic behind but when the regression model predicting the quantities, the classification model is predicting a discrete class label [22].

In classification methods same merging and interval approach used with the regression model. The only difference is the output y column. In Logistic Regression model, y output is a quantity but in classification y column defined as an increase or decrease of the bitcoin price. Increase or decrease calculated difference of the bitcoin price from next price value and results inserted to y column as negative, positive or zero.

For classification model's metric accuracy scores observed for ten classification models which are Random Forest, K_Neighbours, Decision Tree, Gaussian Naïve Bayes, Bernoulli Naïve Bayes, MLP, Gaussian Process, Linear SVC, SVC, Logistic Regression. The same shifting intervals are applied to the classification model. The same machine learning models tried to be selected for the classification and the regression. So which model has the best fit for the data can be evaluated.

Random forest, decision tree, Bernoulli Naïve Bayes, Linear SVC, SVC models give good accuracy scores after 45 min time interval for 2nd shifting.

Logistic Regression, Gaussian Process, MLP, Gaussian Naïve Bayes, K-Neighbors models fit with the data model but not give acceptable accuracy scores.

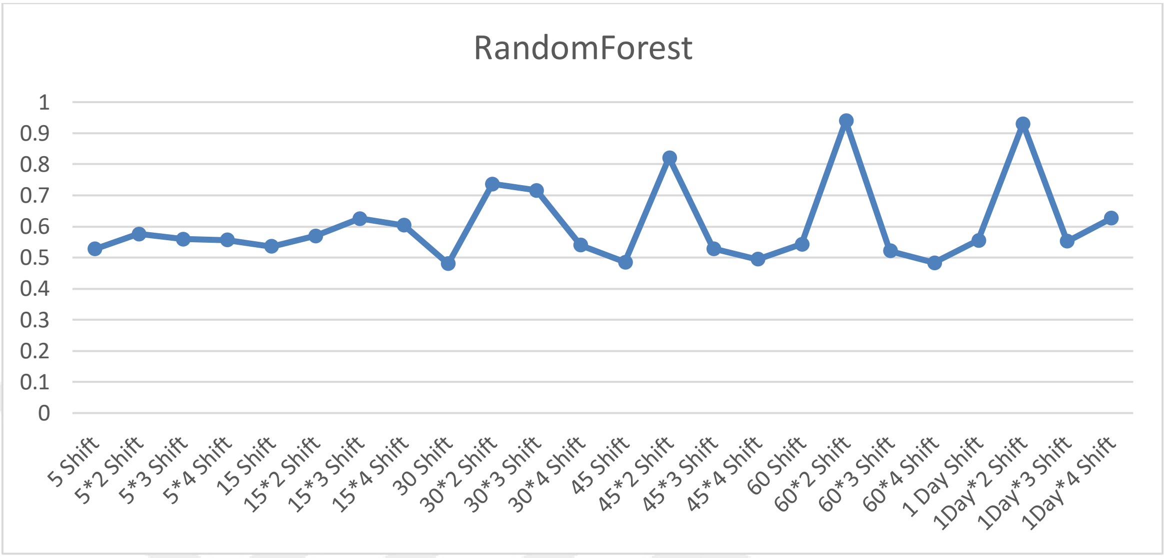


Figure 4.12 Random forest accuracy scores for each shifting.

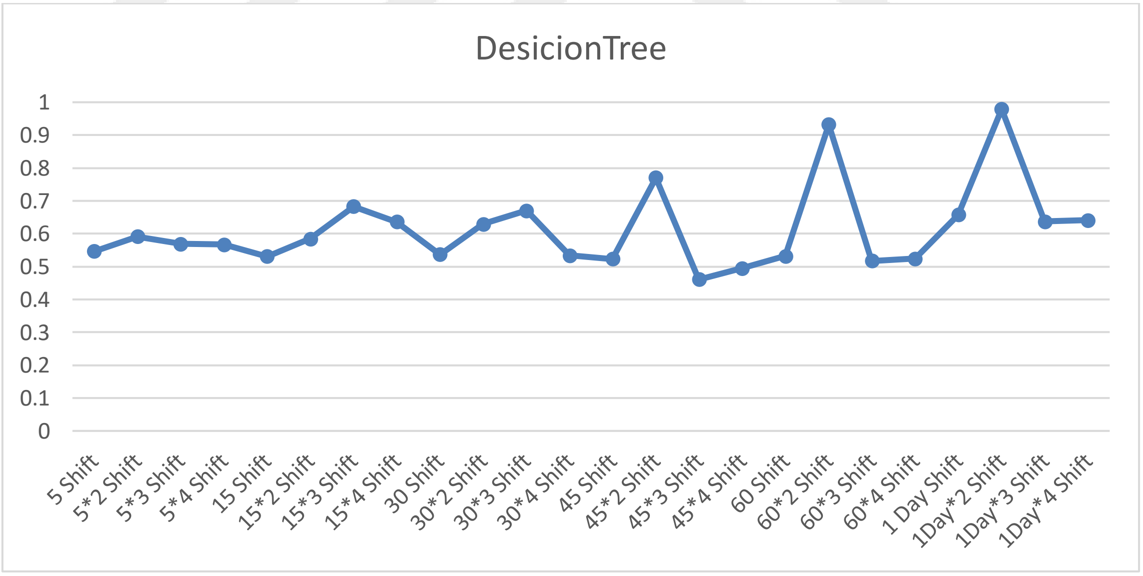


Figure 4.13 Decision tree accuracy scores for each shifting.

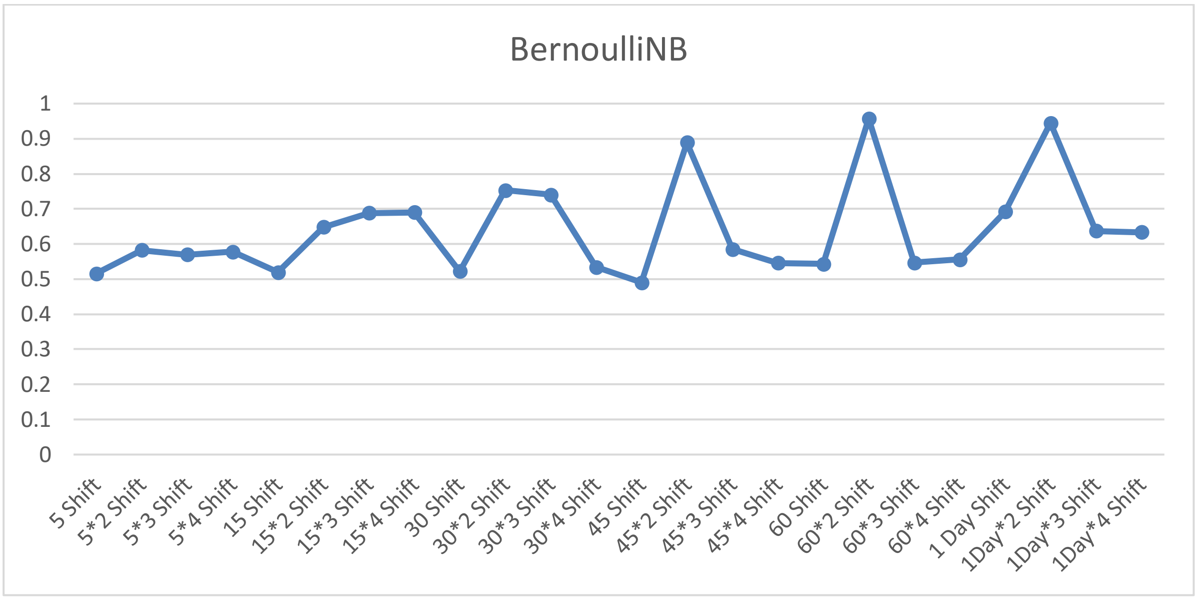


Figure 4.14 Bernoulli naïve bayes accuracy scores for each shifting.

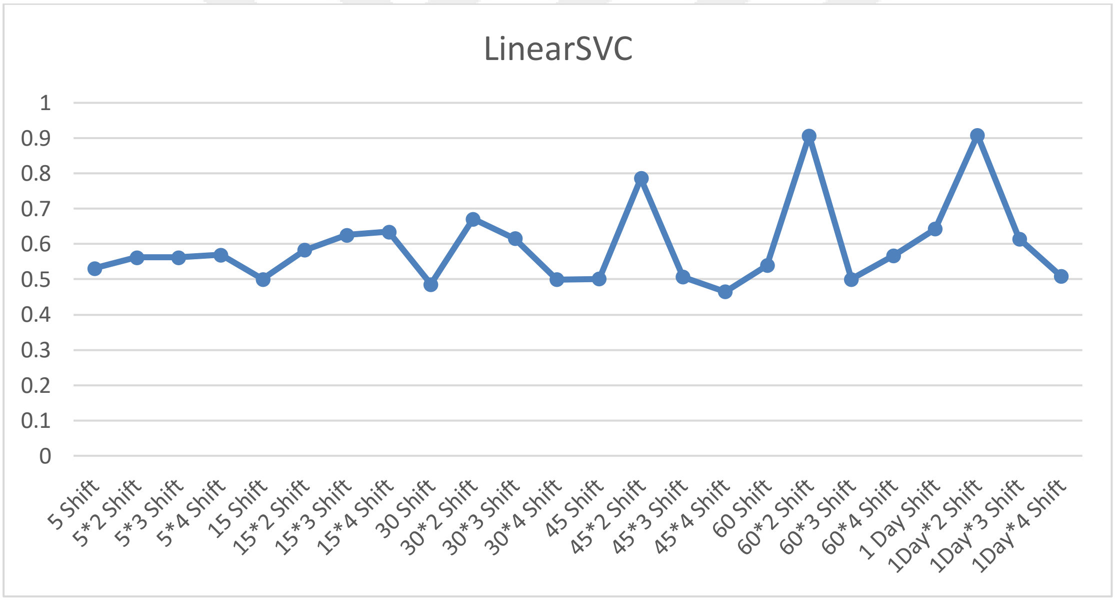


Figure 4.15 Linear SVC accuracy scores for each shifting.

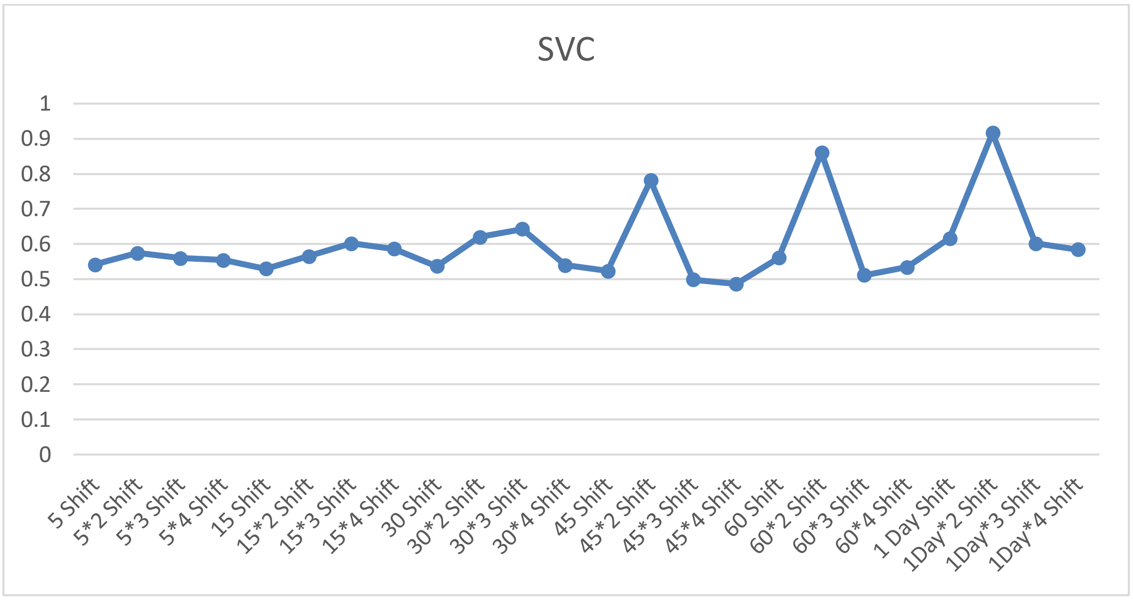


Figure 4.16 SVC accuracy scores for each shifting.

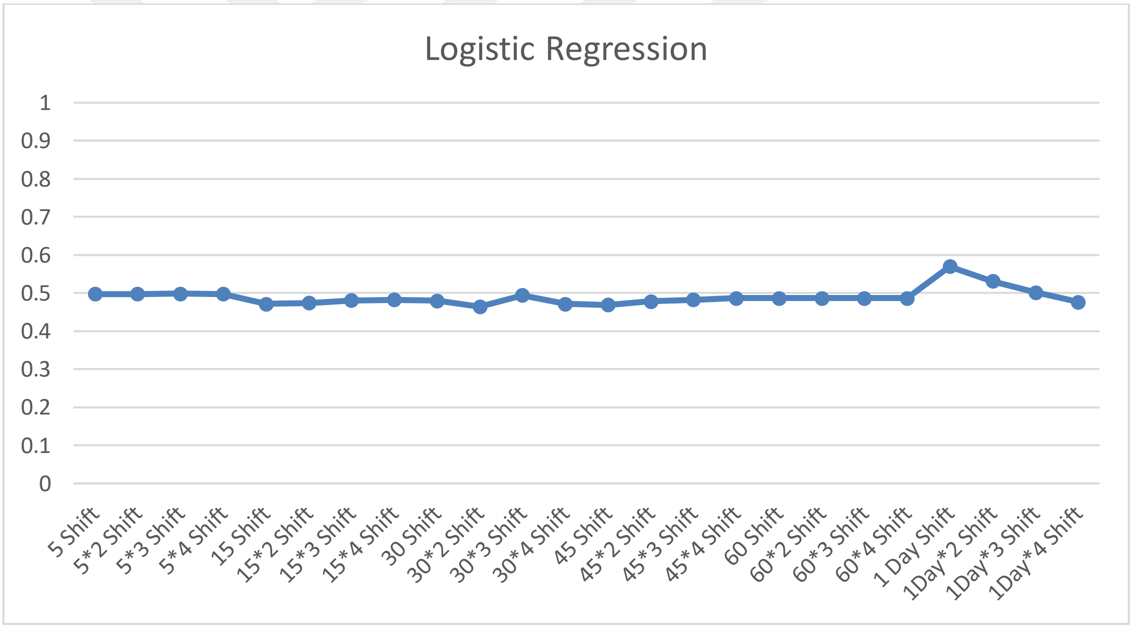


Figure 4.17 Logistic regression accuracy scores for each shifting.

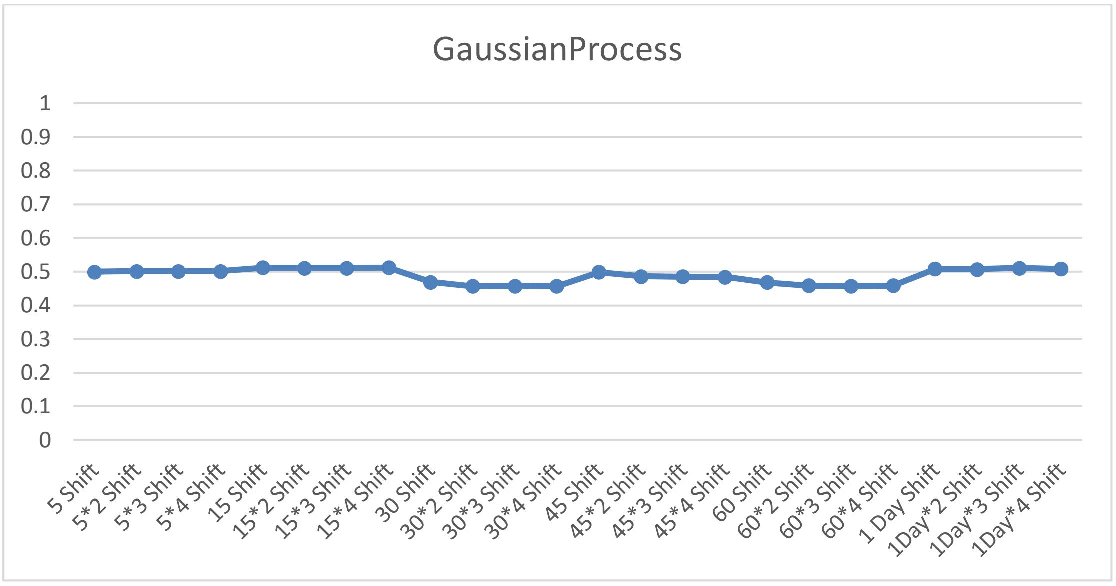


Figure 4.18 Gaussian Process accuracy scores for each shifting

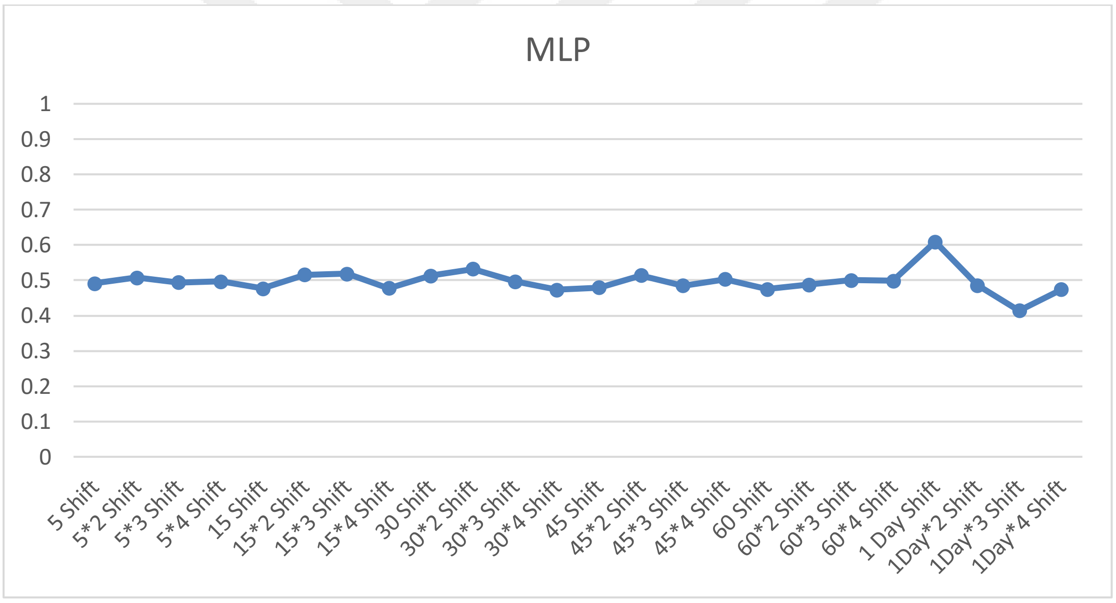


Figure 4.19 MLP accuracy scores for each shifting.

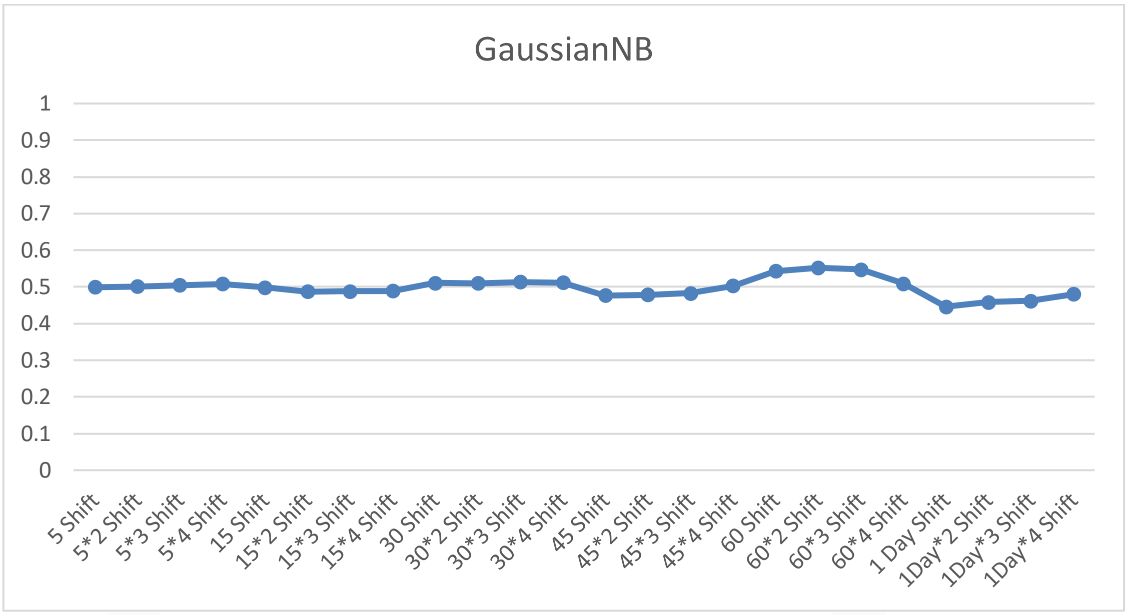


Figure 4.20 Gaussian naïve bayes accuracy scores for each shifting.

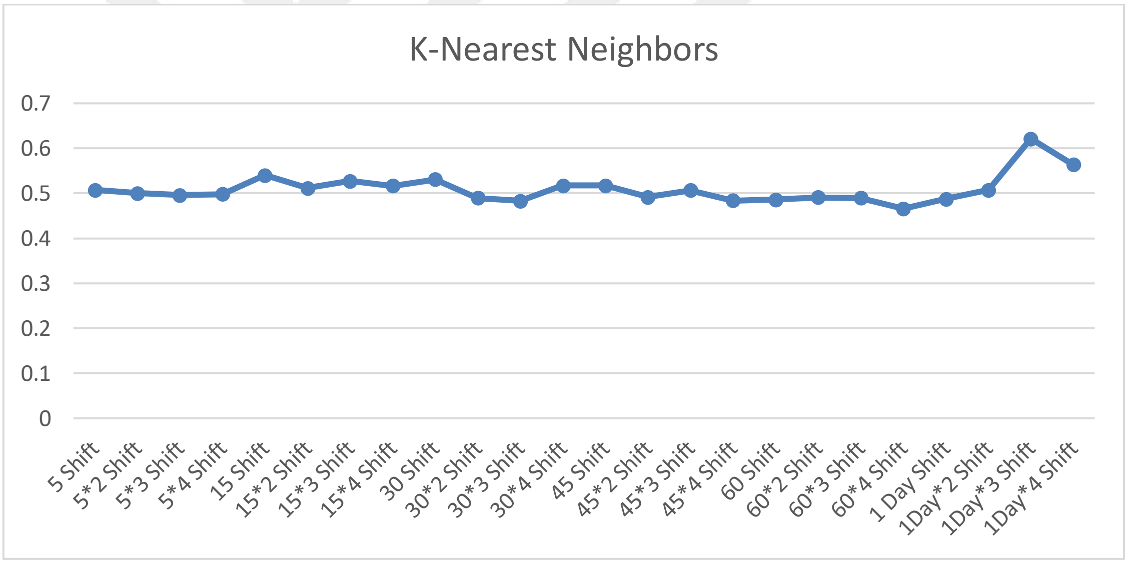


Figure 4.21 K-Nearest neighbors accuracy scores for each shifting.

For better comparison, all classification models' accuracy scores plotted in one graph. Figure 4.22 observed that SVC, Random Forest, Bernoulli Naive Bayes, and Decision Tree models are more powerful to others.

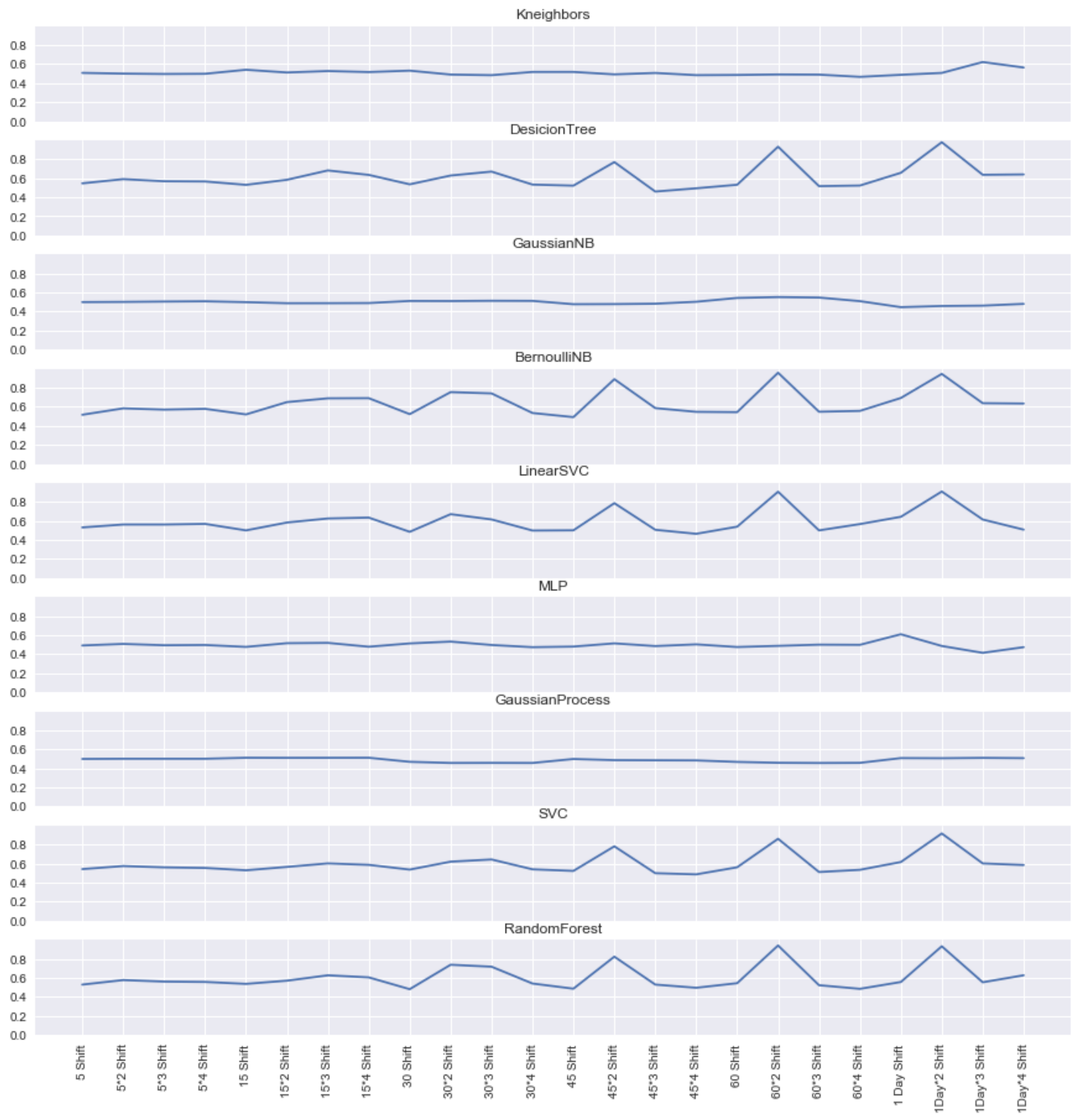


Figure 4.22 Classification accuracy scores for all models.

5. CONCLUSIONS AND RECOMMENDATIONS

Social media's effect on Bitcoin price is analyzed in two different approaches. Regression results show that the Twitter data has a correlation with Bitcoin prices in the short term; if the time interval increases, correlation percentage decreases slightly. The best prediction r^2 score of 0.997 belongs to Lasso model. Lasso model also produces a score of 0.944 until a 1-day time interval shift. All the linear models used in this research (Linear, Lasso and Ridge Regressions) fit well with the data and provide good r^2 scores. But after 1-day shift linear model's score goes below 0.5.

K-neighbors and decision tree models give acceptable r^2 scores only for 5-minute interval shifting. If we check 3 times 5-minute shifts and single 15-minute shift; we can observe that they predict the same time slot, but 3 times 5-minute shifts give 0.993 prediction r^2 score and one 15 minute shift gives 0.525 prediction r^2 score. We can interpret this difference to be the result of the data count. K-neighbors and decision tree models need more data points for a better estimation.

SVR model is also able to predict well for 5-minute intervals when shifted four times. Linear SVR model can make predictions with acceptable scores for 15-minute interval data.

Classification models' results are a totally different perspective than regression results. Best accuracy score of 0.979 is evaluated by the decision tree model for 2 shift in 1 day time interval (1 Day*2 Shift). Random forest and Bernoulli Naïve Bayes also calculated good accuracy scores in 2 shift of 1 day time intervals.

In classification models, when the increase and decrease of the Bitcoin price are analyzed, it is observed that after 45 minute time intervals accuracy scores got better. Also; another observation is that the second shifting always gave better accuracy scores than the other shifts. This observation is valid for all classification models which are used in this research.

As a result, regression models have better scores when they have more data and short time ranges such as 5-minute intervals. On the contrary, classification scores get better in longer time ranges such as 1-day intervals. Also; classification models have better fit with the second shift.

REFERENCES

[1] Marry K. La, Alexander Sedykh, Denis Fourches, Eugene Muratov, Alexander Tropsa, 06 June 2018, Predicting Adverse Drug Effects from Literature-and Database-Mined Assertion, Division of Practice Advancement and Clinical EducationUNC Eshelman School of Pharmacy, Chapel Hill, USA

[2] Obryan Poyser, 17 August 2018, Exploring the dynamics of Bitcoin's price: a Bayesian structural time series approach, Universitat Autonoma de Barcelona, Barcelona, Spain.

[3] Twitter, Inc., Jul 26, 2019, Twitter Announces Second Quarter 2019 Results, San Francisco.

[4] Satoshi Nakamoto. Bitcoin: 2008, A peer-to-peer electronic cash system.

[5] Martina Matta, Ilaria Lunesu, Michele Marchesi, 2015, Bitcoin Spread Prediction Using Social And Web Search Media, Università degli Studi di Cagliari.

[6] Ali Hasan, Sana Moin, Ahmad Karim and Shahaboddin Shamshirband, 16 January 2018: Machine Learning-Based Sentiment Analysis for

Twitter Accounts. Licensee MDPI, Basel, Switzerland.

[7] Ifigeneia Georgoula, Demitrios Pournarakis, Christos Bilanakos, Dionisios N. Sotiropoulos, George M. Giaglis, 2015: Using Time-Series and Sentiment Analysis to Detect the Determinants of Bitcoin Prices. Mediterranean Conference of Information System (MCIS).

[8] Jerman C. Kaminski, 2016: Nowcasting the Bitcoin Market with Twitter Signals.MIT Media Lab.

[9] Evite Stenqvist and Jacob Lönnö, 2017 Predicting Bitcoin price fluctuation

with Twitter sentiment analysis. KTH Royal Institute of Technology School of Computer Science and Communication. Stockholm Sweden.

[10] Sean McNally, Jason Roche, Simon Caton, 2018, Predicting the Price of Bitcoin Using Machine Learning, School of Computing, National College of Ireland, Dublin 1, Ireland.

[11] Url-1 <*https://trends.google.com.tr/trends/?geo=TR*/>, date retrieved 10.03.2019.

[12] Url-2 < *https://developer.twitter.com/en/docs*/>, date retrieved 25.04.2019.

[13] Url-3< *https://en.wikipedia.org/wiki*/>, date retrieved 25.04.2019.

[14] Url-4 < *https://api.coinmarketcap.com/v1/ticker/*>, date retrieved 23.04.2019.

[15] Url-5 < *https://developer.twitter.com//*>, date retrieved 25.04.2019.

[16] Url-6 < *https://docs.mongodb.com/*>, date retrieved 25.04.2019.

[17] Url-7 < *kaynak. https://hackernoon.com/what-steps-should-one-take-while-doing-data-preprocessing-502c993e1caa />,* date retrieved 25.04.2019.

[18] Url-8 < https://pandas.pydata.org/>, date retrieved 09.09.2019.

[19] Url-9<*https://machinelearningmastery.com/classification-versus-regression-in-machine-learning/>,* date retrieved 21.04.2019.

[20] Url-10 <*https://scikit-learn.org/stable/documentation.html*>, date retrieved 23.04.2019.

[21] Url-11 <*https://whatis.techtarget.com/definition/extrapolation-and-interpolation*> date retrieved 23.05.2019.

[22] Url-12 <*https://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/*> date retrieved 29.05.2019.

[23] Url-13 < *https://towardsdatascience.com/5-types-of-regression-and-their-properties-c5e1fa12d55e/>,* date retrieved 03.06.2018.

[24] Url-14 < *https://blockgeeks.com/guides/what-is-cryptocurrency/>,* date retrieved 03.03.2019.

APPENDICES

APPENDIX A: Codes

1. Cryptocurrency data collect code

```
import json
import pymongo

import requests
from time import sleep

def mongo_insert(rec):

    """Insert record into MongoDB
         """
         client = pymongo.MongoClient('localhost', 27017)
         db = client.btc
         collection = db['ticks']

         try:
             result = collection.insert_many(rec)
             rec_id = result.inserted_ids
         except pymongo.errors.ConnectionFailure as err:
 print('Connection Error: ', err)

     def get_tick(url):
         """Get tick data from BTCMarkets API

         Returns:
             dict: Tick data as a dictionary
         """
         req = requests.session()
         res = req.get(url)
         res_text = json.loads(res.text)
         return res_text

     if __name__ == '__main__':

         tick_urls = ['https://api.coinmarketcap.com/v1/ticker/']
     #    For each ticker URL in the tick_urls list
     #    get the data and then commit to MongoDB
         while True:
             try:
                 for ticker in tick_urls:
                     rec = get_tick(ticker)
                     mongo_insert(rec)
                 sleep(300)
             except:
            Pass
```

2. Google trend data collect code

```
from pytrends.request import TrendReq
from pymongo import MongoClient
import json
from time import sleep


pytrend = TrendReq()
kw_list = ["Bitcoin","Ethereum"]
pytrend.build_payload(kw_list, timeframe='now 1-d', gprop='news')
#pytrend.build_payload(kw_list, timeframe='2018-08-01 2019-04-17')

interest_over_time_df = pytrend.interest_over_time()
interest_over_time_df.drop('isPartial',axis=1,inplace=True)
interest_over_time_df.reset_index(level=0, inplace=True)
print (interest_over_time_df)

connection = MongoClient('localhost', 27017)
db = connection.GoogleTrend
collection = db.Google

if __name__ == '__main__':

    while True:

        try:
            records = json.loads(interest_over_time_df.T.to_json()).values()
            print(records)
            collection.insert(records)
            sleep(1440)
            print('sleep')
        except:
            Pass
```

3. Twitter data collect code

```
def import_mongo():
    connection = MongoClient('localhost', 27017)
    db = connection.TwitterStream
    collection = db.tweets
    fname='Bitcoin_'
    Twitter_file_yesterday = format_filename_yesterday(fname)
    filename=os.path.abspath(Twitter_file_yesterday)
    cleanpath = os.path.abspath(filename)
    fh = open(cleanpath, 'r')
    for line in fh:
        try:
            t = json.loads(line)
            tweet_id = t['id_str']  # The Tweet ID from Twitter in string format
            screen_name = t['user']['screen_name']  # The username of the Tweet author
            user_id=t['user']['id']
            followers = t['user']['followers_count']  # The number of followers the Tweet author has
            friends = t['user']['friends_count']
            location=t['user']['location']
            listed_count=t['user']['listed_count']
            following=t['user']['following']
            statuses_count=t['user']['statuses_count']
            timezone=t['user']['time_zone']
            text = t['text'] # The entire body of the Tweet
            clean_tweet_txt=clean_text(t['text'])
            hashtags  =  t['entities']['hashtags']    #  Any  hashtags
used in the Tweet
            created = t['created_at']   #  The  timestamp  of  when  the
Tweet was created
            language = t['lang']  # The language of the Tweet
            timestamp=t['timestamp_ms']
            quote_count=t['quote_count']
            reply_count=t['reply_count']
            retweet_count=t['retweet_count']
            favorite_count=t['favorite_count']
            coordinates=t['coordinates']
            text1=TextBlob(t['text'])
            polarity=text1.sentiment.polarity

            tweet                        =                          {
'tweet_id':tweet_id,'screen_name':screen_name,'user_id':user_id,'lis
ted_count':listed_count,
                 'followers':followers,                  'text':text,
'hashtags':hashtags,'following':following,'statuses_count':statuses_
count,
                 'language':language,             'created':created,
'friends':friends,'location':location,
                 'timestamp':timestamp,         'timezone':timezone,
'quote_count':quote_count,'reply_count':reply_count,

'retweet_count':retweet_count,'favorite_count':favorite_count,'coord
inates':coordinates,'clean_tweet_txt':clean_tweet_txt,
                 'sentiment_polarity':polarity}
            print(tweet)
            try:
                collection.insert(tweet)
            except pymongo.errors.ConnectionFailure as err:
                print('Connection Error: ', err)
        except :
            pass
if __name__ == '__main__':
    keywords=['Bitcoin',    'Ethereum',    'Ripple',    'Litecoin',
'Cardano', 'Stellar', 'Monero', 'IOTA', 'Dash', 'TRON', 'Tether',
              'Qtum','OmiseGO', 'Zcash',  'Bytecoin',
             'altcoin','BTC',  'ETH',  'XRP',  'BCH',  'LTC',  'XLM',
'XMR', 'DASH', 'XEM', 'TRX', 'USDT']
    query_fname='Bitcoin_'
    scheduler=BackgroundScheduler()
    scheduler.add_job(compress,   'cron',   hour='10',   minute='56',
second='50')
    scheduler.add_job(delete,    'cron',    hour='23',    minute='50',
second='50')
    scheduler.add_job(import_mongo, 'cron', hour='12', minute='25',
second='10')
    scheduler.start()
    while True:
        try:
            auth=get_twitter_auth()
            twitter_stream=Stream(auth,CustomListener(query_fname))
            twitter_stream.filter(track=keywords,
languages=['en'],is_async=True)
        except:
            pass
```

4. Twitter text clean code

```
def clean_text(text):

    emoji_pattern = re.compile("["
    u"\U0001F600-\U0001F64F"  # emoticons
    u"\U0001F300-\U0001F5FF"  # symbols & pictographs
    u"\U0001F680-\U0001F6FF"  # transport & map symbols
    u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
                       "]+", flags=re.UNICODE)

    clean_tweet = re.sub("&amp", "", text)
    clean_tweet    =    re.sub("(RT|via)((?:\\b\\W*@\\w+)+)",    "",
clean_tweet) #Cleans RT
    clean_tweet = re.sub("@\\w+", "", clean_tweet)
    clean_tweet = re.sub("[[:punct:]]", "", clean_tweet)
    clean_tweet = re.sub("[[:digit:]]", "", clean_tweet)
    clean_tweet = re.sub("[ \t]{2,}", "", clean_tweet)
    clean_tweet = re.sub("^\\s+|\\s+$", "", clean_tweet)
    clean_tweet    =    re.sub(r'[.,"!]+',    '',    clean_tweet,
flags=re.MULTILINE)
    clean_tweet    =    re.sub(r'[:]+',    '',    clean_tweet,
flags=re.MULTILINE)
    clean_tweet        =        re.sub(r'\w+:\/{2}[\d\w-]+(\.[\d\w-
]+)*(?:(?:\/[^\s/]*))*', '', clean_tweet)
    clean_tweet = re.sub(r"http\S+", '', clean_tweet) #Cleans URL
    clean_tweet = re.sub("#\\w+", "", clean_tweet) # clean hastag
    clean_tweet= re.sub("[^\w ]", '', clean_tweet)
    clean_tweet=emoji_pattern.sub(r'', clean_tweet)
    return(clean_tweet)
```

5. Data merge code

```
tweet = pandas.read_pickle('tweet.pkl')
tweet.index = pandas.to_datetime(tweet.index)

tweet_15min=tweet.resample('15T').agg({'text':     np.count_nonzero,
'favorite_count': np.mean,
'followers':       np.mean,'friends':       np.mean,'listed_count':
np.mean,'statuses_count':                  np.mean,'quote_count':
np.mean,'retweet_count':                   np.mean,'reply_count':
np.mean,'sentiment_polarity':np.mean})
tweet_15min                                                       =
tweet_15min[~tweet_15min.index.duplicated(keep='first')]

btc = pandas.read_pickle('btc.pkl')
btc = btc[~btc.index.duplicated(keep='first')]
btc_15min=btc.resample('15T').mean()

eth = pandas.read_pickle('eth.pkl')
eth = eth[~eth.index.duplicated(keep='first')]
eth_15min=eth.resample('15T').mean()
```

```
GoogleA = pandas.read_pickle('GoogleAll.pkl')
GoogleA = GoogleA[~GoogleA.index.duplicated(keep='first')]
GoogleA_15min=GoogleA.resample('15T').mean()

df_btc=pandas.merge_asof(btc_15min.sort_values('date'),eth_15min.sor
t_values('date'),on=['date'],direction                              =
'nearest').set_index('date')
df_btc_tweet=pandas.merge_asof(df_btc.sort_values('date'),tweet_15mi
n.sort_values('date'),on=['date'],
left_index=True,tolerance=pandas.Timedelta('4m'),direction          =
'backward').set_index('date')
df_tweet_btc_google=pandas.merge_asof(df_btc_tweet.sort_values('date
'),GoogleA_15min.sort_values('date'),on='date',left_index=True,direc
tion ='backward').set_index('date')
df_tweet_btc_google.price_btc_usd                                   =
df_tweet_btc_google.price_btc_usd.shift(periods=1)
```

6. Grid search hyperparameter optimization code

```
import pandas
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn import preprocessing
from sklearn.svm import SVR
from sklearn.svm import LinearSVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import Lasso
from sklearn.linear_model import Ridge
from sklearn.tree import DecisionTreeRegressor
from sklearn.neural_network import MLPRegressor
from sklearn.model_selection import KFold

#import sys

#orig_stdout = sys.stdout
#f1 = open('GridSearchforClassification.txt', 'a+')
#sys.stdout = f1

def standartdeviation (df):
    df1=df.resample('W',                               closed='right',
convention='start').agg({'price': np.std})
    df=df1.resample('15T').pad()
    return(df['price'])

def logfunction(data):
    if data==0:
        data=0
    else:
        data=np.log10(data)
    return data



#tweet=pandas.DataFrame(tweet1,columns=['date','text','favorite_coun
t','followers','friends','listed_count','statuses_count','quote_coun
t','retweet_count','reply_count','sentiment_polarity'],dtype=float).
set_index('date')
tweet = pandas.read_pickle('tweet.pkl')
```

```
tweet.index = pandas.to_datetime(tweet.index)

tweet_5min=tweet.resample('5T').agg({'text':       np.count_nonzero,
'favorite_count': np.mean,
'followers':       np.mean,'friends':       np.mean,'listed_count':
np.mean,'statuses_count':                      np.mean,'quote_count':
np.mean,'retweet_count':                       np.mean,'reply_count':
np.mean,'sentiment_polarity':np.mean})
tweet_5min = tweet_5min[~tweet_5min.index.duplicated(keep='first')]

#btc=pandas.DataFrame(btc1,columns=['date','price_btc_usd','percent_
change_1h','percent_change_24h','percent_change_7d',
#
'h24_volume_usd','total_supply','market_cap_usd'],dtype=float).set_i
ndex('date')
btc = pandas.read_pickle('btc.pkl')
btc = btc[~btc.index.duplicated(keep='first')]


#eth=pandas.DataFrame(eth1,columns=['date','price_eth_usd','percent_
change_1h','percent_change_24h','percent_change_7d',
#
'h24_volume_usd','total_supply','market_cap_usd'],dtype=float).set_i
ndex('date')
eth = pandas.read_pickle('eth.pkl')
eth = eth[~eth.index.duplicated(keep='first')]

#GoogleA=pandas.DataFrame(GoogleA,columns=['date','btc_search_perc']
,dtype=float).set_index('date')
GoogleA = pandas.read_pickle('GoogleAll.pkl')
GoogleA = GoogleA[~GoogleA.index.duplicated(keep='first')]


df_btc=pandas.merge_asof(btc.sort_values('date'),eth.sort_values('da
te'),on=['date'],direction = 'nearest').set_index('date')
df_btc_tweet=pandas.merge_asof(df_btc.sort_values('date'),tweet_5min
.sort_values('date'),on=['date'],
left_index=True,tolerance=pandas.Timedelta('4m'),direction          =
'backward').set_index('date')
df_tweet_btc_google=pandas.merge_asof(df_btc_tweet.sort_values('date
'),GoogleA.sort_values('date'),on='date',direction
='nearest').set_index('date')
#df_tweet_btc_google=pandas.merge_asof(df_btc_tweet.sort_values('dat
e'),GoogleA.sort_values('date'),on='date',direction
='nearest').set_index('date')
df_tweet_btc_google.price_btc_usd                                   =
df_tweet_btc_google.price_btc_usd.shift(periods=1)

#df_tweet_btc_google.fillna(1, inplace=True)
#df_tweet_btc_google.dropna(inplace=True)
df_tweet_btc_google.interpolate()
df_tweet_btc_google.dropna(inplace=True)

print('-------------------------data columns and raws----------------
---------')

print(df_tweet_btc_google.shape)

X=df_tweet_btc_google[df_tweet_btc_google.columns.difference(['price
_btc_usd','date'])].values
```

```
y= df_tweet_btc_google['price_btc_usd'].ravel()
cv_test= KFold(n_splits=5)

####################################################################
############
print('-------------------------Lasso-------------------------')
Lasso=Lasso()

parameters = {
           'alpha':[0.1,0.5,1.0,10]

}
grid_search                                                         =
GridSearchCV(Lasso,parameters,cv=cv_test,scoring='r2',n_jobs=-1)
grid_search.fit(X, y)
print ('Best score: %0.3f' % grid_search.best_score_)
print ('Best parameters set for Lasso Regression:')
best_parameters = grid_search.best_estimator_.get_params()
for param_name in sorted(parameters.keys()):
    print ('\t%s: %r' % (param_name, best_parameters[param_name]))
means = grid_search.cv_results_['mean_test_score']
stds = grid_search.cv_results_['std_test_score']
for      mean,      std,      params      in      zip(means,      stds,
grid_search.cv_results_['params']):
    print("%0.3f (+/-%0.03f) for %r"
          % (mean, std * 2, params))
####################################################################
############
print('-------------------------Ridge-------------------------')
Ridge=Ridge()

parameters = {
           'alpha':[0.01, 0.1, 1.0, 10.0]

}
grid_search                                                         =
GridSearchCV(Ridge,parameters,cv=cv_test,scoring='r2',n_jobs=-1)
grid_search.fit(X, y)
print ('Best score: %0.3f' % grid_search.best_score_)
print ('Best parameters set for Ridge Regression:')
best_parameters = grid_search.best_estimator_.get_params()
for param_name in sorted(parameters.keys()):
    print ('\t%s: %r' % (param_name, best_parameters[param_name]))
means = grid_search.cv_results_['mean_test_score']
stds = grid_search.cv_results_['std_test_score']
for      mean,      std,      params      in      zip(means,      stds,
grid_search.cv_results_['params']):
    print("%0.3f (+/-%0.03f) for %r"
          % (mean, std * 2, params))
####################################################################
############
print('-------------------------KNeighborsRegressor------------------
-------')
KNeighborReg=KNeighborsRegressor()

parameters = {
           'n_neighbors':[1,2,3,5,7,10]

}
```

```
grid_search                                                         =
GridSearchCV(KNeighborReg,parameters,cv=cv_test,scoring='r2',n_jobs=
-1)
grid_search.fit(X, y)
print ('Best score: %0.3f' % grid_search.best_score_)
print ('Best parameters set for KNeighborsRegressor:')
best_parameters = grid_search.best_estimator_.get_params()
for param_name in sorted(parameters.keys()):
    print ('\t%s: %r' % (param_name, best_parameters[param_name]))
means = grid_search.cv_results_['mean_test_score']
stds = grid_search.cv_results_['std_test_score']
for     mean,     std,     params     in     zip(means,     stds,
grid_search.cv_results_['params']):
    print("%0.3f (+/-%0.03f) for %r"
          % (mean, std * 2, params))
#####################################################################
###########
print('------------------------MLPRegressor------------------------
')
MLP=MLPRegressor()

parameters = {
           'alpha':[0.00001,0.0001,0.001,1],
           'learning_rate_init': [0.0001,0.001,0.01],
           'shuffle': [True,False],
           'epsilon': [1e-7,1e-8,1e-9]

}
grid_search                                                         =
GridSearchCV(MLP,parameters,cv=cv_test,scoring='r2',n_jobs=-1)
grid_search.fit(X, y)
print ('Best score: %0.3f' % grid_search.best_score_)
print ('Best parameters set for MLP Regressor:')
best_parameters = grid_search.best_estimator_.get_params()
for param_name in sorted(parameters.keys()):
    print ('\t%s: %r' % (param_name, best_parameters[param_name]))
means = grid_search.cv_results_['mean_test_score']
stds = grid_search.cv_results_['std_test_score']
for     mean,     std,     params     in     zip(means,     stds,
grid_search.cv_results_['params']):
    print("%0.3f (+/-%0.03f) for %r"
          % (mean, std * 2, params))
#####################################################################
#############
print('------------------------DecisionTreeRegressor----------------
---------')
DesicionTree=DecisionTreeRegressor()

parameters = {
#           'max_depth':[1,4,8,10,16,32],
           'min_samples_split':[2,4,5,10,15,50,100]
}
grid_search                                                         =
GridSearchCV(DesicionTree,parameters,cv=cv_test,scoring='r2',n_jobs=
-1)
grid_search.fit(X, y)
#print (grid_search.cv_results_)
print ('Best score: %0.3f' % grid_search.best_score_)
print ('Best parameters set for DecisionTreeRegressor:')
best_parameters = grid_search.best_estimator_.get_params()
```

```
for param_name in sorted(parameters.keys()):
    print ('\t%s: %r' % (param_name, best_parameters[param_name]))
means = grid_search.cv_results_['mean_test_score']
stds = grid_search.cv_results_['std_test_score']
for      mean,      std,      params      in      zip(means,      stds,
grid_search.cv_results_['params']):
    print("%0.3f (+/-%0.03f) for %r"
          % (mean, std * 2, params))
#sys.stdout.close()


####################################################################
###########
print('------------------------LinearSVR-------------------------')
scaler = preprocessing.StandardScaler().fit(X)
X1 = scaler.transform(X)
#X1 = preprocessing.scale(X)
LinSVR=LinearSVR()

parameters = {
           'C':[0.001,0.01, 0.1, 1, 10,100],
           'tol': (1e-5,1e-4,1e-3),
           'epsilon': (0,0.01, 0.1,1)

}

grid_search                                                         =
GridSearchCV(LinSVR,parameters,cv=cv_test,scoring='r2',n_jobs=-1)
grid_search.fit(X1, y)
print ('Best score: %0.3f' % grid_search.best_score_)
print ('Best parameters set for Linear SVR Regression:')
best_parameters = grid_search.best_estimator_.get_params()
for param_name in sorted(parameters.keys()):
    print ('\t%s: %r' % (param_name, best_parameters[param_name]))
means = grid_search.cv_results_['mean_test_score']
stds = grid_search.cv_results_['std_test_score']
for      mean,      std,      params      in      zip(means,      stds,
grid_search.cv_results_['params']):
    print("%0.3f (+/-%0.03f) for %r"
          % (mean, std * 2, params))
#
####################################################################
###########
print('------------------------SVR-------------------------')
SVR=SVR()
parameters  =  {'C'  :  [0.001,0.01,  0.1,  1,  10,100],  'kernel':
('rbf','poly','sigmoid','linear')}
grid_search = GridSearchCV(SVR,parameters,cv=cv_test,scoring='r2')
grid_search.fit(X1, y)
print ('Best score: %0.3f' % grid_search.best_score_)
print ('Best parameters set for SVR_linear Regression:')
best_parameters = grid_search.best_estimator_.get_params()
for param_name in sorted(parameters.keys()):
    print ('\t%s: %r' % (param_name, best_parameters[param_name]))
means = grid_search.cv_results_['mean_test_score']
stds = grid_search.cv_results_['std_test_score']
for      mean,      std,      params      in      zip(means,      stds,
grid_search.cv_results_['params']):
    print("%0.3f (+/-%0.03f) for %r"
          % (mean, std * 2, params))
```

7. Cross validation best score

```
import pandas
import numpy as np
from sklearn import preprocessing
from sklearn.svm import SVR
from sklearn.svm import LinearSVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import Lasso
from sklearn.linear_model import Ridge
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.neural_network import MLPRegressor
from sklearn.model_selection import KFold
from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.model_selection import cross_validate

#tweet=pandas.DataFrame(tweet1,columns=['date','text','favorite_coun
t','followers','friends','listed_count','statuses_count','quote_coun
t','retweet_count','reply_count','sentiment_polarity'],dtype=float).
set_index('date')
tweet = pandas.read_pickle('tweet.pkl')
tweet.index = pandas.to_datetime(tweet.index)

tweet_5min=tweet.resample('5T').agg({'text':        np.count_nonzero,
'favorite_count': np.mean,
'followers':        np.mean,'friends':        np.mean,'listed_count':
np.mean,'statuses_count':                          np.mean,'quote_count':
np.mean,'retweet_count':                           np.mean,'reply_count':
np.mean,'sentiment_polarity':np.mean})
tweet_5min = tweet_5min[~tweet_5min.index.duplicated(keep='first')]



#btc=pandas.DataFrame(btc1,columns=['date','price_btc_usd','percent_
change_1h','percent_change_24h','percent_change_7d',
#
'h24_volume_usd','total_supply','market_cap_usd'],dtype=float).set_i
ndex('date')
btc = pandas.read_pickle('btc.pkl')
btc = btc[~btc.index.duplicated(keep='first')]


#eth=pandas.DataFrame(eth1,columns=['date','price_eth_usd','percent_
change_1h','percent_change_24h','percent_change_7d',
#
'h24_volume_usd','total_supply','market_cap_usd'],dtype=float).set_i
ndex('date')
eth = pandas.read_pickle('eth.pkl')
eth = eth[~eth.index.duplicated(keep='first')]

#GoogleA=pandas.DataFrame(GoogleA,columns=['date','btc_search_perc']
,dtype=float).set_index('date')
GoogleA = pandas.read_pickle('GoogleAll.pkl')
GoogleA = GoogleA[~GoogleA.index.duplicated(keep='first')]


df_btc=pandas.merge_asof(btc.sort_values('date'),eth.sort_values('da
te'),on=['date'],direction = 'nearest').set_index('date')
```

```
df_btc_tweet=pandas.merge_asof(df_btc.sort_values('date'),tweet_5min
.sort_values('date'),on=['date'],
left_index=True,tolerance=pandas.Timedelta('4m'),direction          =
'backward').set_index('date')
df_tweet_btc_google=pandas.merge_asof(df_btc_tweet.sort_values('date
'),GoogleA.sort_values('date'),on='date',left_index=True,direction
='backward').set_index('date')
#df_tweet_btc_google=pandas.merge_asof(df_btc_tweet.sort_values('dat
e'),GoogleA.sort_values('date'),on='date',direction
='nearest').set_index('date')
df_tweet_btc_google.price_btc_usd                                   =
df_tweet_btc_google.price_btc_usd.shift(periods=1)

#df_tweet_btc_google.fillna(1, inplace=True)
#df_tweet_btc_google.dropna(inplace=True)
df_tweet_btc_google.interpolate()
df_tweet_btc_google.dropna(inplace=True)

X=df_tweet_btc_google[df_tweet_btc_google.columns.difference(['price
_btc_usd','date'])]
y= df_tweet_btc_google['price_btc_usd'].ravel()

cv_test= KFold(n_splits=5)

####################################################################
############
print("-------------------------------LinearRegressionModel---------
----------------------")

linear_model=LinearRegression()

scores   =   cross_validate(linear_model,   X,   y,   scoring='r2',
cv=cv_test, return_train_score=True)
print(scores['test_score'].mean())
print(scores)


##########################################################
print("-------------------------------LassoModel--------------------
-----------")

#lasso_model=Lasso(alpha=0.5,normalize=True)    #    alpha    prevent
overfitting
lasso_model=Lasso(alpha=0.1) # alpha prevent overfitting

scores = cross_validate(lasso_model, X, y, scoring='r2', cv=cv_test,
return_train_score=True)
#y_pred = cross_val_predict(lasso_model, X, y, cv=cv_test)
#accuracy_score1=accuracy_score(y, y_pred)
#print("accuracy score %s" %(accuracy_score1))
print("score_mean:%s" %(scores['test_score'].mean()))
print(scores['test_score'].mean())
print(scores)
##########################################################
print("-------------------------------RidgeModel--------------------
-----------")

#ridge_model=Ridge(alpha=10.0,normalize=True)    #    alpha    prevent
overfitting
ridge_model=Ridge(alpha=10.0) # alpha prevent overfitting
```

```
scores = cross_validate(ridge_model, X, y, scoring='r2', cv=cv_test,
return_train_score=True)
print("score_mean:%s" %(scores['test_score'].mean()))

print(scores['test_score'].mean())
print(scores)
##########################################################
print("-------------------------------GaussianProcesses-------------
------------------")

GaussianPR=GaussianProcessRegressor()
scores = cross_validate(GaussianPR, X, y, scoring='r2', cv=cv_test,
return_train_score=True)
print(scores['test_score'].mean())
print(scores)
##########################################################
print("-------------------------------KNeighborsModel---------------
----------------")

#KNeighborReg=KNeighborsRegressor(n_neighbors=5)
KNeighborReg=KNeighborsRegressor(n_neighbors=1)
scores    =    cross_validate(KNeighborReg,    X,    y,    scoring='r2',
cv=cv_test, return_train_score=True)
print("score_mean:%s" %(scores['test_score'].mean()))

print(scores['test_score'].mean())
print(scores)
##########################################################
print("-------------------------------MLPModel----------------------
---------")

#MLP=MLPRegressor(alpha=0.001,epsilon=1e-
09,learning_rate_init=0.001)
MLP=MLPRegressor()
scores   =   cross_validate(MLP,   X,   y,   scoring='r2',   cv=cv_test,
return_train_score=True)
print("score_mean:%s" %(scores['test_score'].mean()))
print(scores['test_score'].mean())
print(scores)
##########################################################
print("-------------------------------DesicionTree------------------
-------------")

#DesicionTree=DecisionTreeRegressor(min_samples_split=4)
DesicionTree=DecisionTreeRegressor(min_samples_split=50)
scores    =    cross_validate(DesicionTree,    X,    y,    scoring='r2',
cv=cv_test, return_train_score=True)
print("score_mean:%s" %(scores['test_score'].mean()))
print(scores['test_score'].mean())
print(scores)
####################################################################
###########
print("-------------------------------SVRModel----------------------
---------")
X = preprocessing.scale(X)
#regressor=SVR(C=100,epsilon=1,gamma='auto',tol=0.01)
regressor=SVR(C=10,kernel='linear')
scores    =    cross_validate(DesicionTree,    X,    y,    scoring='r2',
cv=cv_test, return_train_score=True)
print("score_mean:%s" %(scores['test_score'].mean()))
print(scores['test_score'].mean())
print(scores)
##########################################################
print("-------------------------------LinearSVRModel----------------
---------------")
#regressor=LinearSVR(C=0.1,epsilon=0,tol=1e-05)
regressor=LinearSVR(C=10,epsilon=0.01,tol=0.001)
scores = cross_validate(regressor, X, y, scoring='r2', cv=cv_test,
return_train_score=True)
print("score_mean:%s" %(scores['test_score'].mean()))
print(scores['test_score'].mean())
print(scores)
```

APPENDIX B: Scores

Table 5.1 Regression model R² scores.

Index	Linear Regression	Lasso	Ridge	K_Neighbors	Decision Tree	SVR	Linear SVR
5 Shift	0.9965	0.9972	0.9965	0.9527	0.9456	0.9366	0.9967
5*2 Shift	0.9950	0.9958	0.9950	0.9440	0.8931	0.9192	0.9951
5*3 Shift	0.9938	0.9945	0.9938	0.9281	0.8750	0.8754	0.9937
5*4 Shift	0.9927	0.9937	0.9928	0.9148	0.8485	0.8289	0.9927
15 Shift	0.9867	0.9873	0.9868	0.5255	0.2624	0.4729	0.9743
15*2 Shift	0.9807	0.9798	0.9808	0.4898	0.0291	0.1606	0.9687
15*3 Shift	0.9859	0.9843	0.9858	0.4587	0.3648	0.1626	0.9717
15*4 Shift	0.9936	0.9918	0.9932	0.4415	0.2033	0.2381	0.9893
30 Shift	0.9848	0.9838	0.9848	0.5066	0.2837	0.3439	0.3335
30*2 Shift	0.9942	0.9923	0.9942	0.4590	0.2556	0.1138	0.3537
30*3 Shift	0.9746	0.9731	0.9746	0.4391	-0.0610	-0.025	0.2919
30*4 Shift	0.9377	0.9377	0.9380	0.4110	0.1373	0.1129	0.0488
45 Shift	0.9917	0.9896	0.9917	0.5046	-0.0216	-0.107	-0.600
45*2 Shift	0.9777	0.9772	0.9777	0.4613	0.0714	0.0337	-0.998
45*3 Shift	0.9304	0.9316	0.9307	0.4136	0.0437	-0.006	-1.211
45*4 Shift	0.8795	0.8984	0.8816	0.3690	-0.2641	-0.042	-1.769
60 Shift	0.9966	0.9950	0.9966	0.4509	-0.2275	0.0540	-2.398
60*2 Shift	0.9608	0.9604	0.9612	0.4074	0.3443	0.1928	-2.057
60*3 Shift	0.8707	0.9005	0.8744	0.3719	0.0175	-0.052	-2.950

60*4 Shift	0.8216	0.8609	0.8287	0.3809	0.2076	0.1605	-4.290
1 Day Shift	0.8854	0.9441	0.9279	0.4794	-0.3657	-0.223	-8.87
1Day*2 Shift	0.4237	0.6492	0.6495	0.3235	-0.1212	-0.529	-14.36
1Day*3 Shift	0.3870	0.6136	0.5248	0.0701	0.2661	0.1487	-13.24
1Day*4 Shift	-0.9611	-0.006	-0.607	-0.8313	-1.5054	-1.776	-21.12

Table 5.2 Classification model accuracy scores.

Index	Random Forest	K-neighbors	Desicion Tree	Gaussian NB	Bernoulli NB	MLP	Gaussian Process	Linear SVC	SVC	Locistig Regression
5 Shift	0.528	0.508	0.547	0.499	0.515	0.491	0.499	0.532	0.542	0.497
5*2 Shift	0.576	0.500	0.591	0.501	0.583	0.508	0.501	0.562	0.574	0.497
5*3 Shift	0.560	0.496	0.569	0.505	0.570	0.494	0.501	0.562	0.560	0.498
5*4 Shift	0.556	0.498	0.567	0.508	0.578	0.496	0.501	0.569	0.554	0.498
15	0.536	0.540	0.531	0.499	0.519	0.4	0.512	0.500	0.5	0.472

Shift						76			29	
15*2 Shift	0.569	0.512	0.584	0.487	0.648	0.515	0.511	0.583	0.565	0.474
15*3 Shift	0.626	0.527	0.683	0.488	0.688	0.519	0.511	0.626	0.601	0.481
15*4 Shift	0.604	0.516	0.636	0.489	0.690	0.478	0.512	0.635	0.586	0.482
30 Shift	0.480	0.531	0.537	0.511	0.522	0.513	0.469	0.485	0.537	0.480
30*2 Shift	0.737	0.490	0.629	0.510	0.753	0.532	0.457	0.671	0.620	0.464
30*3 Shift	0.716	0.483	0.670	0.513	0.740	0.496	0.458	0.616	0.643	0.494
30*4 Shift	0.539	0.517	0.534	0.512	0.534	0.473	0.457	0.499	0.539	0.472
45 Shift	0.485	0.518	0.523	0.477	0.491	0.480	0.499	0.501	0.523	0.469
45*2 Shift	0.822	0.492	0.770	0.478	0.889	0.514	0.486	0.786	0.781	0.478
45*3 Shift	0.528	0.507	0.461	0.482	0.585	0.485	0.485	0.507	0.499	0.482
45*4 Shift	0.495	0.484	0.495	0.503	0.546	0.503	0.484	0.465	0.486	0.486
60 Shift	0.543	0.486	0.532	0.543	0.543	0.475	0.468	0.539	0.561	0.486
60*2 Shift	0.939	0.491	0.932	0.552	0.957	0.487	0.459	0.907	0.860	0.486
60*3	0.522	0.489	0.518	0.547	0.547	0.5	0.457	0.500	0.5	0.486

Shift						00			11	
60*4 Shift	0.484	0.466	0.524	0.509	0.556	0.498	0.458	0.567	0.534	0.486
1 Day Shift	0.555	0.487	0.658	0.446	0.692	0.609	0.508	0.643	0.616	0.569
1Day* 2 Shift	0.931	0.507	0.979	0.458	0.945	0.486	0.507	0.909	0.916	0.532
1Day* 3 Shift	0.552	0.621	0.637	0.462	0.637	0.414	0.510	0.615	0.601	0.502
1Day* 4 Shift	0.626	0.564	0.641	0.481	0.633	0.474	0.508	0.508	0.584	0.476

APPENDIX C: Optimum Hyperparameters

Table 5.3 Grid search optimum parameters for 15 min interval regression model.

Parameters	Lasso	Ridge	K-Neighbors	MLP	Decision Tree	Linear SVR	SVR
alpha	0.1	0.1	-	1	-	-	-
epsilon	-	-	-	1e-07	-	0	-
kernel	-	-	-	-	-	-	linear
C	-	-	-	-	-	100	100
gamma	-	-	-	-	-	-	-
tol	-	-	-	-	-	0.0001	-
max_dept	-	-	-	-	-	-	-
min_sample	-	-	-	-	50	-	-
n_neighbours	-	-	3	-	-	-	-
shuffle	-	-	-	False	-	-	-
learning_rate_init	-	-	-	0.01	-	-	-

Table 5.4 Grid search optimum parameters for 30 min interval regression model.

Parameters	Lasso	Ridge	K-Neighbors	MLP	Decision Tree	Linear SVR	SVR
alpha	0.1	0.1	-	1	-	-	-
epsilon	-	-	-	1e-08	-	0	-
kernel	-	-	-	-	-	-	linear
C	-	-	-	-	-	100	100
gamma	-	-	-	-	-	-	-
tol	-	-	-	-	-	1e-05	-
max_dept	-	-	-	-	-	-	-
min_sample	-	-	-	-	50	-	-

n_neighbours	-	-	3	-	-	-	-
shuffle	-	-	-	True	-	-	-
learning_rate_init	-	-	-	0.01	-	-	-

Table 5.5 Grid search optimum parameters for 45 min interval regression model.

Parameters	Lasso	Ridge	K-Neighbors	MLP	Decision Tree	Linear SVR	SVR
alpha	0.1	0.1	-	0.001	-	-	-
epsilon	-	-	-	1e-07	-	0.1	-
kernel	-	-	-	-	-	-	linear
C	-	-	-	-	-	100	100
gamma	-	-	-	-	-	-	-
tol	-	-	-	-	-	1e-05	-
max_dept	-	-	-	-	-	-	-
min_sample	-	-	-	-	5	-	-
n_neighbours	-	-	3	-	-	-	-
shuffle	-	-	-	False	-	-	-
learning_rate_init	-	-	-	0.01	-	-	-

Table 5.6 Grid search optimum parameters for 60 min interval regression model.

Parameters	Lasso	Ridge	K-Neighbors	MLP	Decision Tree	Linear SVR	SVR
alpha	0.1	0.1	-	1	-	-	-
epsilon	-	-	-	1e-07	-	1	-
kernel	-	-	-	-	-	-	linear
C	-	-	-	-	-	100	100
gamma	-	-	-	-	-	-	-
tol	-	-	-	-	-	1e-05	-
max_dept	-	-	-	-	-	-	-
min_sample	-	-	-	-	10	-	-
n_neighbours	-	-	2	-	-	-	-
shuffle	-	-	-	True	-	-	-
learning_rate_init	-	-	-	0.01	-	-	-

Table 5.7 Grid search optimum parameters for a 1-day interval regression model.

Parameters	Lasso	Ridge	K-Neighbors	MLP	Decision Tree	Linear SVR	SVR
alpha	10	10	-	0.0001	-	-	-
epsilon	-	-	-	1e-09	-	0.1	-
kernel	-	-	-	-	-	-	linear
C	-	-	-	-	-	100	100
gamma	-	-	-	-	-	-	-
tol	-	-	-	-	-	0.001	-
max_dept	-	-	-	-	-	-	-
min_sample	-	-	-	-	10	-	-
n_neighbours	-	-	5	-	-	-	-
shuffle	-	-	-	True	-	-	-

learning_rate_init	-	-	-	0.01	-	-	-

Table 5.8 Grid search optimum parameters for 15 min interval classification model.

Parameters	Random Forest	K-Neighbors	Decision Tree	MLP	Gaussian NB	Bernoulli NB	Linear SVC	SVR
max_depth	10	-	4	-	-	-	-	-
min_sample_split	4	-	15	-	-	-	-	-
n_neighbors	-	3	-	-	-	-	-	-
alpha	-	-	-	1e-05	-	0	-	-
epsilon	-	-	-	1e-08	-	-	-	-
learning_rate_init	-	-	-	0.0001	-	-	-	-
shuffle	-	-	-	True	-	-	-	-
var_smoothing	-	-	-	-	1e-10	-	-	-
C	-	-	-	-	-	-	10	10
multi_class	-	-	-	-	-	-	ovr	ovo
tol	-	-	-	-	-	-	0.001	0.0001
kernel	-	-	-	-	-	-	-	poly
gamma	-	-	-	-	-	-	-	auto

Table 5.9 Grid search optimum parameters for 30 min interval classification model.

Parameters	Random Forest	K-Neighbors	Decision Tree	MLP	Gaussian NB	Bernoulli NB	Linear SVC	SVR
max_depth	4	-	8	-	-	-	-	-
min_sample_split	100	-	10	-	-	-	-	-
n_neighbors	-	100	-	-	-	-	-	-
alpha	-	-	-	1e-05	-	0	-	-
epsilon	-	-	-	1e-07	-	-	-	-
learning_rate_init	-	-	-	0.0001	-	-	-	-
shuffle	-	-	-	False	-	-	-	-
var_smoothing	-	-	-	-	1e-07	-	-	-
C	-	-	-	-	-	-	100	1
multi_class	-	-	-	-	-	-	ovr	ovo
tol	-	-	-	-	-	-	0.0001	0.0001
kernel	-	-	-	-	-	-	-	poly
gamma	-	-	-	-	-	-	-	auto

Table 5.10 Grid search optimum parameters for 45 min interval classification model.

Parameters	Random Forest	K-Neighbors	Decision Tree	MLP	Gaussian NB	Bernoulli NB	Linear SVC	SVR
max_depth	10	-	16	-	-	-	-	-
min_sample_split	10	-	15	-	-	-	-	-
n_neighbors	-	1	-	-	-	-	-	-
alpha	-	-	-	0.000	-	0	-	-

				1				
epsilon	-	-	-	1e-08	-	-	-	-
learning_rate_init	-	-	-	1	-	-	-	-
shuffle	-	-	-	True	-	-	-	-
var_smoothing	-	-	-	-	1e-08	-	-	-
C	-	-	-	-	-	-	100	10
multi_class	-	-	-	-	-	-	ovr	ovo
tol	-	-	-	-	-	-	0.001	0.0001
kernel	-	-	-	-	-	-	-	poly
gamma	-	-	-	-	-	-	-	auto

Table 5.11 Grid search optimum parameters for 60 min interval classification model.

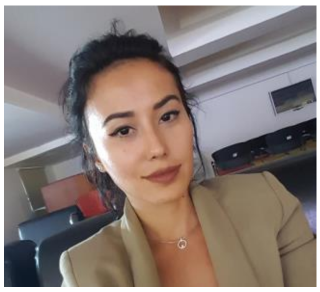
Parameters	Random Forest	K-Neighbors	Decision Tree	MLP	Gaussian NB	Bernoulli NB	Linear SVC	SVR
max_depth	4	-	32	-	-	-	-	-
min_sample_split	4	-	2	-	-	-	-	-
n_neighbors	-	100	-	-	-	-	-	-
alpha	-	-	-	1e-05	-	5	-	-
epsilon	-	-	-	1e-07	-	-	-	-
learning_rate_init	-	-	-	0.01	-	-	-	-
shuffle	-	-	-	True	-	-	-	-
var_smoothing	-	-	-	-	1e-10	-	-	-
C	-	-	-	-	-	-	100	0.1

multi_class	-	-	-	-	-	-	ovr	ovo
tol	-	-	-	-	-	-	0.01	0.1
kernel	-	-	-	-	-	-	-	linear
gamma	-	-	-	-	-	-	-	-

Table 5.12 Grid search optimum parameters for 1-day interval classification model.

Parameters	Random Forest	K-Neighbors	Decision Tree	MLP	Gaussian NB	Bernoulli NB	Linear SVC	SVR
max_depth	16	-	4	-	-	-	-	-
min_sample_split	15	-	50	-	-	-	-	-
n_neighbors	-	2	-	-	-	-	-	-
alpha	-	-	-	0.0001	-	0	-	-
epsilon	-	-	-	1e-09	-	-	-	-
learning_rate_init	-	-	-	10	-	-	-	-
shuffle	-	-	-	False	-	-	-	-
var_smoothing	-	-	-	-	1e-10	-	-	-
C	-	-	-	-	-	-	100	10
multi_class	-	-	-	-	-	-	ovr	ovo
tol	-	-	-	-	-	-	0.1	0.0001
kernel	-	-	-	-	-	-	-	linear
gamma	-	-	-	-	-	-	-	-

CURRICULUM VITAE



Name Surname	**:** Özlem Gül Pamuk
Place and Date of Birth	**:** Muğla 17.10.1988
Address	**:** Sevkibey Sokak. Moda/Kadıköy İstanbul
E-Mail	**:** ozlemgulpamuk@hotmail.com

EDUCATION:

B.Sc.	**:** Anadolu University/Electrical Electronic Engineer

PROFESSIONAL EXPERIENCE and REWARDS:

- Software Test Engineer at Huawei Technologies
- Software Development Engineer in Test at Siemens