

ÖZYEĞİN UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE

**CS 402**

**2021 Spring**

**SENIOR PROJECT REPORT**

**Crypto-currency Price Prediction Using News and Social Networks Data**

**By**

**Barış Karaer**

**Ertan Ayanlar**

**Supervised By**

**Emre Sefer**

**Declaration of Own Work Statement/ (Plagiarism Statement)**

Hereby I confirm that all this work is original and my own. I have clearly referenced/listed all sources as appropriate and given the sources of all pictures, data etc. that are not my own. I have not made any use of the essay(s) or other work of any other student(s) either past or present, at this or any other educational institution. I also declare that this project has not previously been submitted for assessment in any other course, degree or qualification at this or any other educational institution.

|  |  |
| --- | --- |
| **Student Name and Surname:** | Barış Karaer, Ertan Ayanlar |
| **Signature:** | <Signatures of the group members > |
| **Place, Date:** | Özyeğin University, 07.01.2021 |

**Abstract**

Crypto currencies are digital assets that are created to work as a medium of exchange where each record of the ownership of these coins are stored on a digitalized ledger that uses strong cryptography. These crypto currencies are mostly not controlled by anyone (decentralized control) as opposed to central banking systems. Since it is decentralized, the biggest drivers of volatility in the crypto market are because of the speculation of the investors. These speculations and the volume of attention the crypto market gets, heavily moves the price of crypto currencies. The goal of our project is to design and implement a machine learning model which achieves good price predictions for Bitcoin, the crypto currency that was first created by Satoshi Nakamoto in a white paper which is called “Bitcoin: A Peer-to-Peer Electronic Cash system”, and Ethereum, a decentralized open source blockchain with smart contract functionality that was developed by Vitalik Buterin. The predictions are made using Twitter sentiment, Google Trends data. The Google Trends values are collected as weekly points and Twitter sentiment analysis is collected as hourly data. These values are also preprocessed and used in various ways to help the time series model make better predictions. Different models are created in order to show the significance of shifting data and choosing different features. This project uses different machine learning methods to predict the prices such as Linear Regression, Time Series predictions. In detail, LSTM and XGBOOST is used as time series machine learning models. In addition to the train test splitting technique, different cross validation techniques such as rolling forecast origin and sliding window are used to tune our hyperparameters and in the end, to get the best prediction results out of the time series machine learning models. In depth analysis using metrics and comparisons are made between predicting Ethereum and Bitcoin prices. Then, hourly predictions are made using Twitter sentiment analysis using Linear Regression and Time Series models.

**Contents**

[**I.**](#_3znysh7) **INTRODUCTION 4**

[**II.**](#_3dy6vkm) **BACKGROUND 5**

[**III.**](#_1t3h5sf) **PROBLEM STATEMENT 6**

[**IV.**](#_4d34og8) **SOLUTION APPROACH 7**

[**V.**](#_2s8eyo1) **RESULTS AND DISCUSSION 8**

[**VI.**](#_17dp8vu) **RELATED WORK 9**

[**VII.**](#_3rdcrjn) **CONCLUSION AND FUTURE WORK 10**

[**ACKNOWLEDGEMENTS 11**](#_26in1rg)

[**REFERENCES 12**](#_lnxbz9)

[**APPENDIX 13**](#_35nkun2)

1. **Introduction**

Crypto Currency Price Prediction Using Social Networks and News Data is a new project for us but many academics have implemented predictions on the future prices of Bitcoin. Our approach is to collect data from many different social media / market channels and estimate the market speculation and then, use these data to predict accurate Bitcoin and Ethereum prices using different machine learning algorithms. We will be using different time series algorithms to compare the accuracies and scores and make assumptions on which of them are better to predict the future prices. In our case, we have used linear regression and time series algorithms (LSTM and XGBOOST). The data that is given as independent or dependent variables to the machine learning algorithms are collected by us, the independent variables are Google Trends and Twitter Sentiment analysis scores. For predicting with Google Trends, the dependent variables are weekly closing prices for Bitcoin and Ethereum. For predicting with Twitter Sentiment Analysis, the dependent variables are hourly closing prices for Bitcoin. Each of them will be provided in detail in the results and discussion chapter. For further inspection, the data used, and the code repository can be found on our GitHub repository which is provided in the appendix.

We have used only one programming language, “Python”, for machine learning related codes, and used many libraries for visualizing, acquiring and predicting data. Azure Sentiment Analysis tool is used to give scores to certain tweets and assess them. Specifics are provided more in detail in the background chapter.

After predicting with appropriate hyperparameters, we have used different cross validation techniques like rolling forecast origin and sliding window to tune our hyperparameters. After tuning, new models are created with the best hyperparameters and analyzed with metrics like Root Mean Squared Error and Mean Absolute Error.

The Google Trends values were acquired weekly because the data was weighted differently if daily values were searched. Due to this problem weekly values are acquired from Google Trends and the average of closing prices per week is used as a dependent variable. These kinds of issues that we faced will be explained in detail in the results and discussions chapter. Especially, because of the big volume of twitter data, we had to collect only 17 million tweets between 2017 and 2019. Although it is better for the algorithm to have a lot of data regarding the Tweets, it was not possible for us to collect large amounts of data due to time concerns.

1. **Background**

Various kinds of tools and techniques are used for this project and the main ones are as follows:

* **Python**: We have used the Python programming language to use the machine learning libraries and make good predictions.
* **Visualizing Libraries**: We have used many visualizing libraries while using Python. These libraries are, MatPlotLib for plotting data, SeaBorn for visualizing the data, Pandas for manipulating data and preprocessing it.
* **SciKit Learn**: This is a library we used to create different machine learning models for our data.
* **Tweepy**: Tweepy library is used to provide access to the Twitter API and handle rate-limiting limitations of Twitter API.
* **Various Open Source API’s**: We have used many API’s to collect data from social media. The CoinMarketCap API is used to collect the prices of crypto currencies. The Twitter API is used to collect tweets from the Twitter application about selected crypto currencies.
* **Jupyter Notebook**: The Jupyter Notebook tool is used for code implementations. A visualization can be seen at appendix-3.
* **IntelliJ PyCharm**: This IDE is used in the development of the data collection module. A visualization can be seen at appendix-2.
* **Google Trends**: This was used to access the popularity of some keywords that are searched throughout the internet. For Bitcoin, the keyword “Bitcoin” was used to collect the popularity data.
* **Azure Sentiment Analysis**: This tool was used to predict if a certain Tweet is a positive tweet about a certain crypto. If the outcome of this analysis is negative or positive, it labels our data accordingly.
* **Azure Language Detection**: This tool was used to detect the languages of fetched tweets.

1. **Problem Statement**

Crypto Currency Price Prediction Using Social Networks and News Data is a new project. The main goal of our senior project is to create a good machine learning model that predicts the future prices of a selected crypto currency (mainly Bitcoin). This project is aimed to create different machine learning models and lastly acquire a good accuracy score while doing this prediction. This project was offered by the faculty member, Dr. Emre Sefer. Mr Sefer will be our supervisor throughout the whole project.

The volatility of the crypto currency market is widely known but the speculation of the investors and people all around the world majorly steers the prices of these currencies. Our assumption is that after analyzing the data from the social networks, past prices, trends in the web and the sentiment of the tweets that are sent in the Twitter platform, we can make a prediction about the future prices of these currencies. Our constraints for this project is that we will be using different types of machine learning techniques to make predictions, these techniques are called Linear Regression and Time Series. Furthermore, there are some constraints on the data that we collected. The Bitcoin closing prices have been collected for the past 5 years. The previous years are not used because they are not necessary. Similarly the google trends data are collected for the past 5 years. The Tweets that are collected from the Twitter application will be for the past 2 months, it is nearly impossible to collect all the Tweets about Bitcoin for the past 5 years. Also, the amount of tweet volumes will be difficult to acquire. This way, whether good or bad we can make deductions of the attention of the market to the crypto currency market.

1. **Solution Approach**

**DESIGN DECISIONS**

Since our project is separated into two modules, one that is responsible for data collection and other that is responsible for price prediction, they should be examined separately.

**Part 1: Data Collection Module**

Data collection module is responsible for collecting the required data to feed the machine learning model that is used for predicting the Bitcoin price. Tweets from twitter, past Bitcoin and Ethereum price data, google trends data, news and social media data for a specific cryptocurrency is collected through this module. This module is coded in Python.

In our first iteration, we decided to collect the tweet ourselves with the help of Tweepy module and run sentiment analysis on them using Azure’s Sentiment Analysis Module. We filter the tweets based on the language (‘EN’) and a date interval. We do not have to specify a language since we are using Azure’s Text Analytics’ language detection endpoint to figure out the language of the tweet that is needed for the sentiment analysis later on. We run sentiment analysis over the collected tweets using the Microsoft Azure Cloud services’ Text Analytics API, by sending a POST request. To achieve this, first we need to determine the language of each tweet via Azure’s Language Detection service. Tweets that are currently stored as an array needs to be converted to a format that Azure’s Language Detection endpoint can understand. We convert by placing the tweet array in a python dictionary with key ‘data’ and pass this dictionary to a function that organizes the contents in a format Azure’s Language detection endpoint understand. Now that we have the request body of the Azure’s Language Detection endpoint ready, we send a POST request to that endpoint and get a JSON object as response containing the detected languages of the tweets and their corresponding id’s. The request body for Azure’s Sentiment Analysis endpoint is prepared in the same fashion by placing tweets array under a ‘documents’ key of a dictionary. A POST request is made to the sentiment analysis endpoint and we get a JSON object as response containing each tweets sentiment score. A mean sentiment score is calculated and used on our price prediction module. However, because of the size limitations of POST requests and Azure’s Text Analytics endpoint, we are not able to send all tweets for analysis at once. To overcome this, we are splitting the calls to this endpoint in separate requests containing the half of the total tweets. Sadly, Twitter API only allows its users to only access past 7 days historical data at most and because of this limitation we decided to collect 1000 tweets daily starting from January 2021 and store them as CSV. Past prices of bitcoin are collected using the CoinMarketCap website and stored in a CSV file. Google Trends data is downloaded directly from Google’s official website.

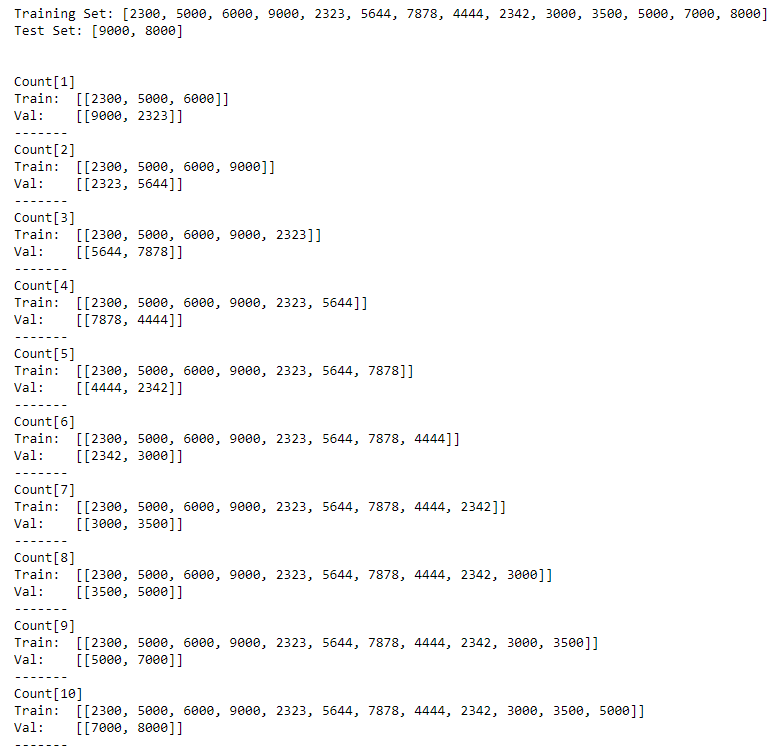
After we collected 2 months of tweet data and ran sentiment analysis on them we decided to go with another approach to obtain our Twitter data and sentiment analysis since we figured it out that 1000 tweets daily were not enough to come up with a ‘correct’ sentiment score to determining the overall sentiment of the day and the Azure’s Free Tier services were not enough to process all this data. At first we replaced Azure’s Sentiment Analysis with VaderSentiment, an open source sentiment analysis library but the end results were not encouraging still because of the low volume of tweets we were able to collect from Twitter. Then we decided to use datasets from Kaggle containing Tweets regarding to bitcoin. First dataset we used contains sentiment scores of the tweets but the values of the sentiment scores are very small values like 0.09 and it did not have a correlation with the Closing price of the bitcoin over those dates. So we decided to recalculate the sentiment scores using VaderSentiment. The first data set contains 17.7 million Tweets over the time period of 01/08/2017 and 21/01/2019. It is also an hourly dataset, reporting tweet stats for that hourly period like total volume of negative, positive, neutral tweets and the highest price bitcoin had during the period and the last price during the period. On this dataset we found the reported total volume of positive tweets on the dataset had the best correlation with the closing prices of the Bitcoin and we decided to use it as feature both for our hourly and daily price prediction analysis. We tried another data set from Kaggle, which only contained Tweets and ran sentiment analysis over those 16 min tweets with VaderSentiment again. This time correlation of our sentiment analysis scores were low, around 0.30 so we decided not to use this data set.

Our final and most recent approach was to discover how popular cryptocurrency analytics websites like LunarCrush work and we extracted their analytics data and decided to use it in our Price Prediction module. LunarCrush is a website targeting social media analysis and its impacts on the cryptocurrencies. We found out that they retrieve the data for the charts on their website by sending REST requests to their API and we analyzed their API endpoints and managed to come up with a request that let us to retrieve 720 days of data containing price data and social scores for the social platforms twitter, reddit and news. For twitter, the retrieved data had a ‘tweets’ field reporting the volume of the tweets for that day and 5 different sentiment scores for that day under ‘tweet\_sentiment1’, ‘tweet\_sentiment2’ and so on. Other social data contained in the response of this API request were number of reddit posts, reddit comments, social score and contributors and a ‘news’ field reflecting count of the news for the requested cryptocurrency on that day. We found high correlation of these features with the closing prices of Bitcoin and Ethereum so we used this data for our Twitter feature on LSTM and XGBoost models.

**Part 2: Price Prediction Module**

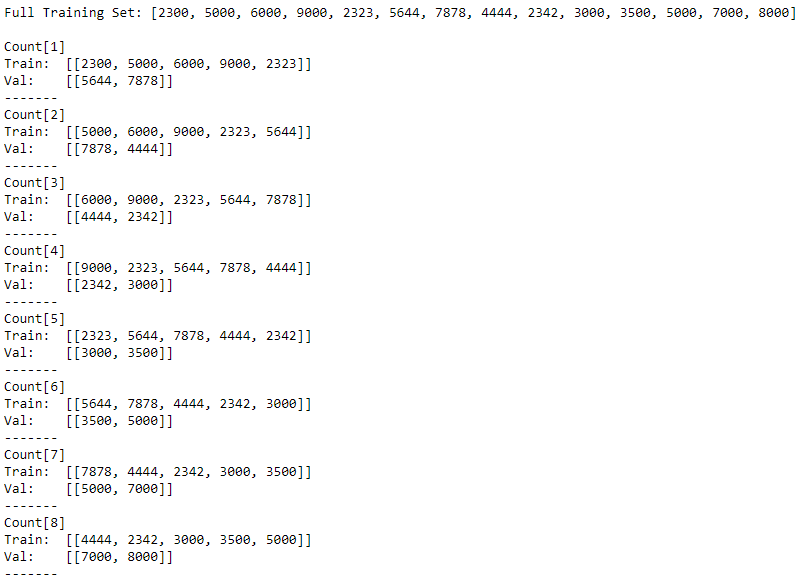
The Price Prediction Module is responsible for predicting the Bitcoin and Ethereum price with different machine learning models. We have used Linear Regression, LSTM and XGBOOST machine learning time series models to predict the prices. The closing prices of these cryptocurrencies are collected from the CoinMarketCap website. We have tried collecting Twitter data with the Twitter API and analyze them using Azure Sentiment analysis but due to API and budget limitations we have collected the Twitter data from a Kaggle project (<https://www.kaggle.com/jaimebadiola/bitcoin-tweets-and-price>). This way, it was much easier to get access to an analyzed dataset, we had 17 million tweets to use when we downloaded this dataset. We can separate the price prediction to 2 subparts. First subpart uses Google Trend values as independent values and predicts based on a mixture of these values and shifted weekly prices. Second subpart uses hourly sentiment analysis dataset mentioned earlier and predicts based on these sentiment values.

In the first subpart, the Google Trends data gives only specific scores to weeks and it does not give daily scores, because of this we have decided to calculate the weekly average closing prices of Bitcoin and Ethereum to use in our dataset. After the predictions for the Linear Regression are made and different metrics are calculated, LSTM time series model is used to make predictions for the Bitcoin and Ethereum closing prices. Different hyper parameters are used to create different results and find the most optimal one. After these predictions, mean absolute error and root mean squared error is used to find the most optimal model and compare between different features. Same process is applied to a different time series model called XGBOOST for Bitcoin and Ethereum. Different graphs and plots are given to further compare the results of these models. Additionally, 2 different cross validation techniques are used. These techniques are implemented from scratch in Python, which the code repository can be found in the Appendix part of this paper. First one is rolling forecast origin and the second one is sliding window. These techniques are used to find the right hyperparameters for LSTM or XGBOOST models. Different combination of parameters is used for the time series models. The results will be explained in detail in the Results and Discussion part of this paper. To briefly explain the cross-validation techniques, rolling forecast origin is an evaluation technique that updates the forecast origin successively and the forecasts are produced from each origin. The important parameters for the rolling origin function are minimum train size and horizon which affects our train and validation set. These parameters of the rolling forecast origin are fixed to make the comparison between different models easier. The minimum train size is set as 10 and the horizon is set as 2. As an example, Figure 1 shows the training set, test set and the train-validation pair that is created from the training and test set.



**Fig. 1.** Rolling forecast origin training set, test set, train-validation pair

The sliding window is a very well-known time series cross validation technique where a dataset is split into several pairs and these pairs are created by the sequences taken at a successive equally spaced points in time. Important parameters that affect our train-validation pairs are window size and horizon. These parameters are fixed just like rolling forecast to compare our results between different models. The window size is set as 5 and the horizon is set as 2 for all our models. Figure 2 explains the sliding window technique briefly.



**Fig. 2.** Sliding window training set, train-validation pair

LSTM and XGBOOST are known to be overfitting while training the dataset. These 2 cross validation techniques are used to prevent the overfitting problem that may occur while predicting the Bitcoin and Ethereum prices using these time series models.

Ikinci subpartı açıkla

**Usage of Tools and Techniques**

All the tools and libraries that are mentioned on the Background section were used. Our code base is fully in Python. Our project is separated in two modules, the data collection module that is used for retrieving past price of bitcoin and the tweets on the #bitcoin hashtag. Tweepy module is used for retrieving the tweets. Azure’s Text Analytics service is used for sentiment analysis and language detection of the tweets. Tensorflow, Scikit Learn and Keras library are used for price prediction through machine learning.

**Technical, Operational and Financial Feasibility**

**Financial Feasibility**

Our implementation only requires a mid-tier computer and an internet connection to fetch and process the required data and machine learning operations. We don’t need to use cloud computing to obtain our data and do our machine learning.

**Technical Feasibility**

Our implementation depends on the Tweepy module for twitter data collection, Coinbase API for past prices of the bitcoin and Google Trends for the trends data. Tweepy module can be easily replaced and twitter’s API can be directly used, another provider can be found for past prices of bitcoin and for replacing Google Trends other analytics tools can be used.

**Engineering Standards**

Modularization approach is used so that our implementation is divided into two modules; data collection and price prediction module. Processed data is stored as CSV.

**Knowledge and Skill Set**

We benefited from various courses from Özyeğin University. Main and the most important courses are as follows:

• CS 454 - Introduction to Machine Learning

• CS 452 - Data Science

• CS 320 - Software Engineering

• CS 321 - Programming Languages

• CS 202 - Database Management

• CS 201 - Data Structure

• CS 102 - Object-Oriented Programming

All Computer Science courses helped us to learn another subject but the key lessons are Data Science and Machine learning courses we take at Özyeğin University.

1. **Results and Discussion**

In our first iteration, Tweepy library was used in order to access Twitter API and fetch tweets containing a hashtag, #bitcoin in our case. This library allowed us to fetch large numbers of tweets since we configured it to wait when we are rate-limited by Twitter and then continue its job to fetch the tweets, this way we did not have to manage the rate limitations of the Twitter API. On these tweets, Microsoft Azure’s Cognitive Services, the Text Analytics service is used to analyze the positiveness of the tweets (sentiment analysis) to get a general idea about what people on Twitter think about the target cryptocurrencies. This way, the speculation of the market is analyzed. Then the tweets are formatted and converted to JSON for Azure Text Analytics service to understand, languages for these tweets are detected by the Azure Language Endpoint, then the formatted tweets are combined with the language data and sent to the Azure Sentiment Analysis endpoint. By doing so, we acquire a list of tweets, each tweets sentiment score and a mean sentiment score of all the tweets that are fetched for the specified time range and then we output them to an CSV file. Below some sample data can be examined.

**Mean Sentiment Score between dates 2020-12-01 - 2020-12-09 with data consisting of 9000 tweets: 0.549054**

**Positive sentiment scores:**

|  |  |  |  |
| --- | --- | --- | --- |
| id | date | text | score |
| 1.34E+18 | 2020-12-08-23:31:38 | The year is 2020 the month is December and you could be earning triple interest paid in BitcoinEarn X3 in BTC | 0.999962568 |
| 1.34E+18 | 2020-12-08-23:49:47 | Happy to see this tall green line for btc Lets keep the buy volume comingcrypto btc bitcoin omg link ren | 0.999139607 |
| 1.34E+18 | 2020-12-08-23:07:33 | buy the dip What a blessing to buy bitcoin on discount right before christmas | 0.994126081 |

**Negative sentiment scores:**

|  |  |  |  |
| --- | --- | --- | --- |
| 1.34E+18 | 2020-12-08-22:28:46 | Sometimes Bitcoin price movement make me tired | 0.092111319 |
| 1.34E+18 | 2020-12-08-23:22:30 | Sell all Bitcoin now dip is coming | 0.091784805 |

Sadly, Twitter API limits fetching of historical data (tweets) are older than 7 days. To overcome this, we have started fetching 1000 tweets each day from Twitter and storing them as CSV since January 2021. After we had enough data collected, we ran sentiment analysis on them using VaderSentiment library, however the results were not good enough to use as a feature because we were only able to collect a very low volume of tweets when compared with the total volume of tweets regarding bitcoin, potentially tens of thousands every day (show the results).

Then we decided to continue with open source Kaggle datasets like (4) “Bitcoin 17.7 million Tweets and price” by Jaime Badiola, although their correlation score was better we researched for better datasets used by popular analytics websites that can be found online, like LunarCrush (show that the correlation score was better).

LunarCrush’s dataset were collected through their API, by modifying API request parameters we were able to get social sentiment data of 2 years, from current day to past 720 days. We achieved extremely good corrrelation results from this data, some features hitting upto 84 % correlation score for Bitcoin and XX % for Ethereum. We decided to use this dataset on our Price Prediction library to support Twitter and other social sentiment scores. Figure i shows news and some social media features, figure ii shows the Twitter volume and sentiment features we retrieved from LunarCrush. In detail, the features for social media and news are reddit posts, reddit comments, social score, social contributors and news. The features for twitter are the twitter volume and different sentiment analysis results done by LunarCrush.

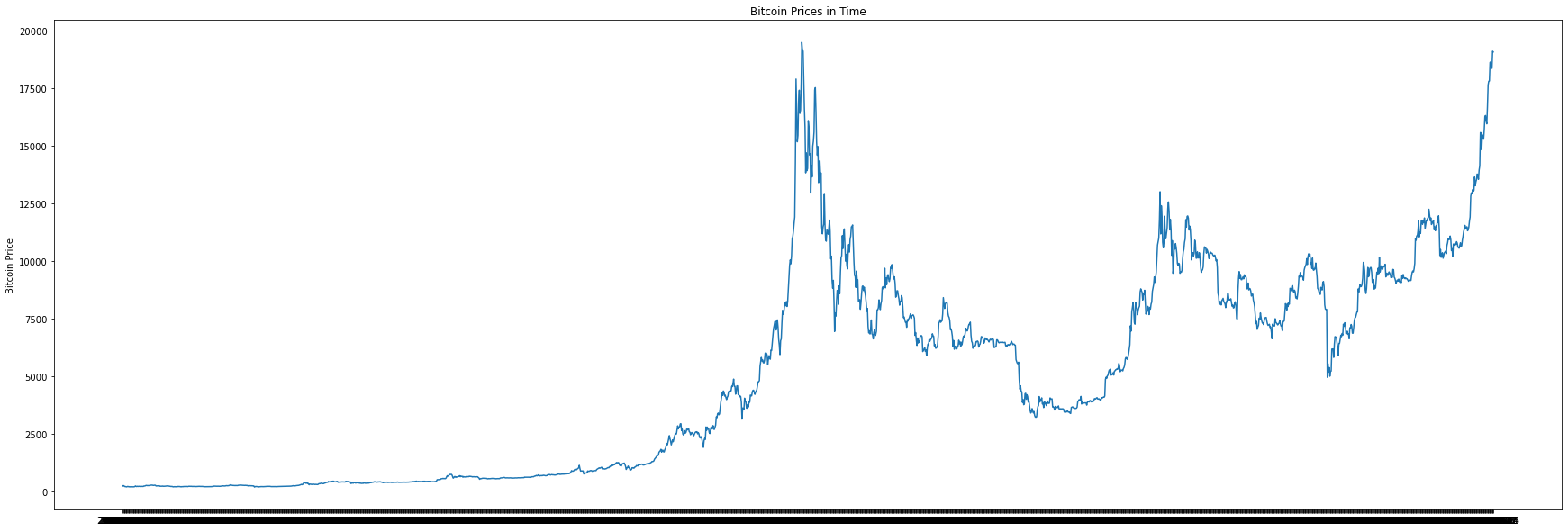
Graphical user interface, application

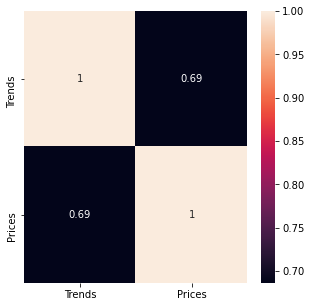
Description automatically generated

Graphical user interface, application

Description automatically generated

To further analyze the market speculation in a graph, Google Trends was used to find out and analyze the popularity of search queries in google search. Using the “Bitcoin” keyword a graph can be acquired to visualize the popularity of this keyword. Although there are many parameters we could use while using Google Trends, only the amount of volume for the entire world is significant for us. We didn’t use specific country data. For reference, here is the bitcoin price graph:

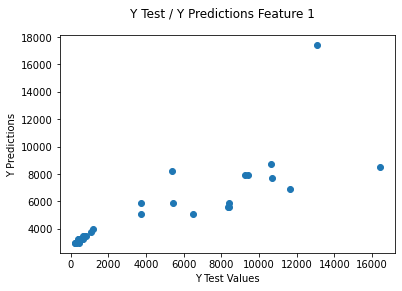


The future price predictions of bitcoin are made using the Python programming language with the help of many libraries such as NumPy, Pandas, Seaborn and MatPlotLib. These were the libraries that helped us visualize the data. Using the SciKit Learn and Keras library we implemented machine learning algorithms to predict the future prices. Our aim is to implement many machine learning algorithms and acquire different accuracies to compare them using the data that was collected from the social media and the market speculation.

The correlation of the Google Trends data is 0.69. The lack of daily google trends data is very bad for the prediction algorithm but the correlation of these data is quite high. We have splitted the data into test and train sets and because of the lack of daily data we had to use 40 % of our data for the test set to make predictions.

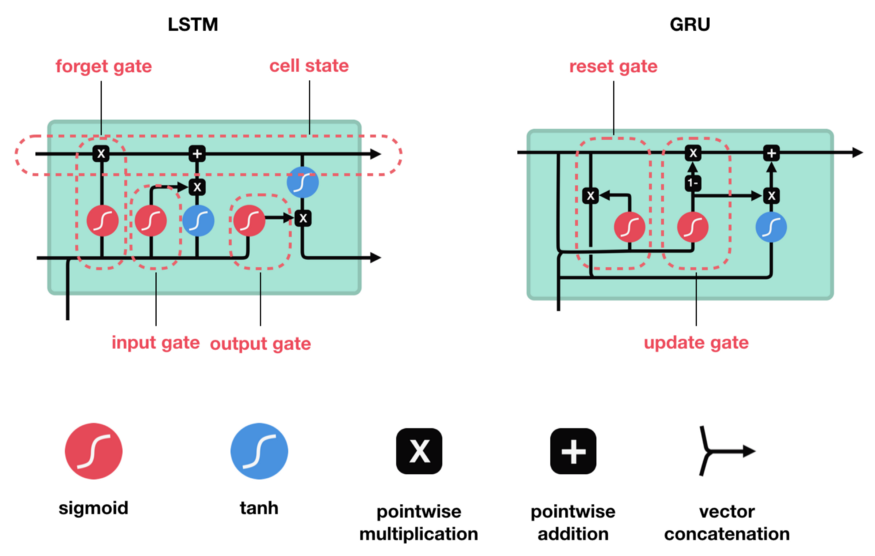
**Prediction with Linear Regression using Google Trends Data**

We have used the Linear Regression machine learning algorithm. Linear Regression is a Linear Model that assumes a linear relationship between the independent variables (x) and a single output variable (dependent variable y). In this case the independent variable is the google trends data and the dependent variable is the closing price of Bitcoin. We have already splitted the data into training and test sets. We have used metrics to analyze this model and here are the plots and graphs implemented to further analyze this prediction. The root mean squared error that is calculated is 3009.659205024709. The absolute mean squared error is 2725.539799221666. The visualization can be also seen in appendix-5.

****

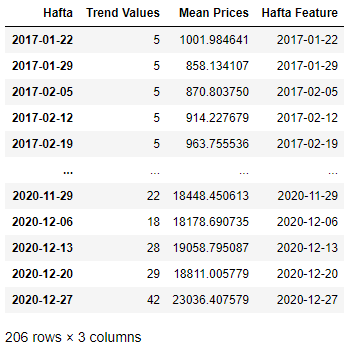
**Time Series Prediction with LSTM**

We chose the Time Series machine learning algorithm LSTM (Long-short term memory) to predict prices of Bitcoin and Ethereum. This artificial RNN (Recurrent Neural Network) architecture is used to process entire sequences of data and make predictions. LSTM networks are widely known to be suited for classifying, processing and making predictions based on time series data, so this kind of a model is very well suited for predicting stock market prices or crypto currency prices where speculations are always occurring in the market and the data is based on time.

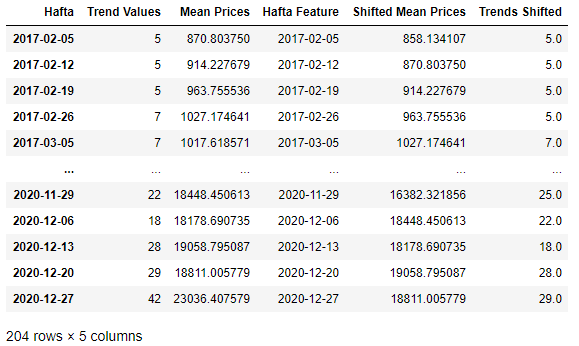
Although RNN's are very useful, it suffers from short term memory. If a data is in a very long sequence, then it will have a very hard time carrying information from earlier time steps to later ones. So, predicting over long periods of time would not end in an accurate prediction. On the other hand, LSTM was created as a solution to this short-term memory problem. The internal gates of the LSTMs can regulate the flow of information. These gates allow the model to learn which data in the sequences are important so it can keep or throw away the unnecessary ones. A simple representation of LSTM and GRU which is like an LSTM can be seen below.

**Bitcoin Closing Price Prediction with LSTM**

In this project, we acquired the data of the closing prices of Bitcoin between the beginning of 2017 and the end of 2020. Specific dates can be seen in Figure 1 and 2. Keras library is used to implement LSTM. Using the train test splitting technique, we have split the data into 126 weeks of train data and 80 weeks of test data. Different features are used to train the model. The month, year, day of month, Google Trend values, shifted price values and shifted Trend values are used as independent variables for the LSTM model. These independent variables for Bitcoin can be seen below:

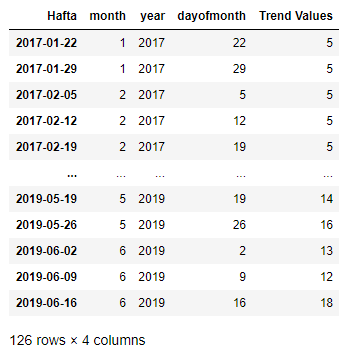


**Fig. 1.** Bitcoin Dataset

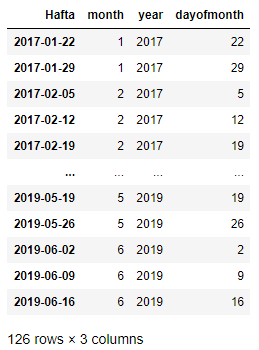


**Fig. 2.** Bitcoin Dataset with shifted values

The default hyperparameter we used for prediction is 50 units for the LSTM, 1 Dense unit, batch size of 5 and 100 epochs. These values for the hyperparameter were used to predict Bitcoin before hyperparameter tuning with cross validation. We will first analyze the prediction of Bitcoin prices using month, year, day of month and Google Trend values. This dataset can be seen in Figure 3. Then, we will analyze the predictions of Bitcoin prices by omitting the Trend values and using only month, year, day of month as features. This specific dataset can be seen in Figure 4. After analyzing the scenarios mentioned earlier, we will be showing the predictions of data that were shifted. At total 126 rows (weeks) are used for training and 80 rows (weeks) are used for testing our model. The dependent values of the model are the corresponding average weekly price of Bitcoin.

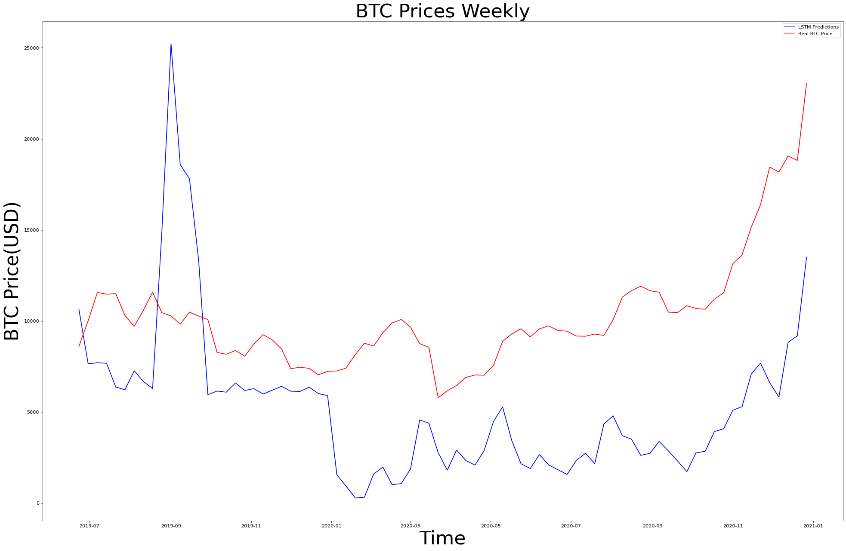


**Fig. 3.** The Training dataset for month, year, day of month and trend features

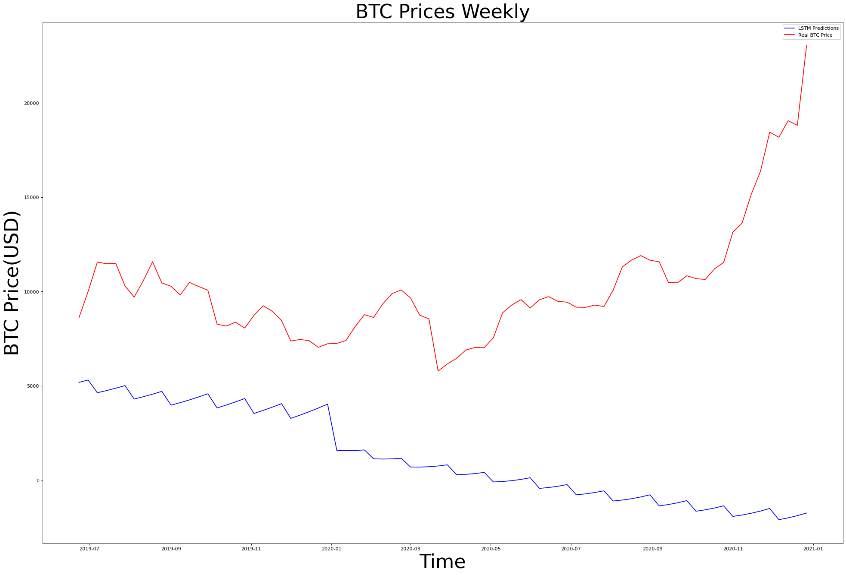


**Fig. 4.** The training dataset for only month, year and day of month features

After feeding our model with the independent variables mentioned earlier, we predict the prices based on the test data and show it in 2 graphs. Figure 6 is the prediction using only date values, figure 5 is the prediction using date and Google Trends values. The blue line is predicted, and the red line is real weekly prices of Bitcoin.



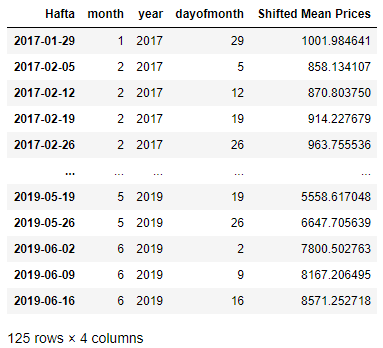
**Fig. 5.** The predictions for date and trend values



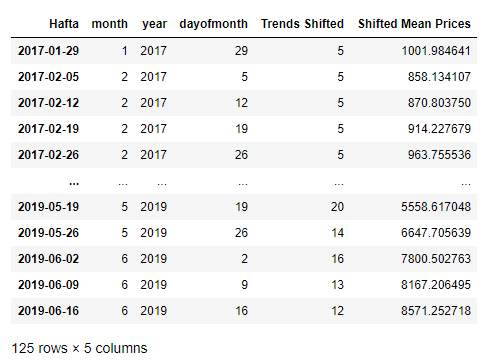
**Fig. 6.** The predictions for only date values

Clearly the values of Google Trends have significantly enhanced our model, despite the sudden spike of 2019-09 which is caused by the sudden increase of Trends value. We have used Mean Absolute error and Root Mean Squared error to further analyze and conclude our predictions. When we used only dates as features, the RMSE result is 10362.77, the MAE result is 9224.89. When dates and trend values are used as features, the RMSE result is 6555.80, the MAE result is 5843.18. Although LSTM models are prone to overfitting, this can be prevented by tuning hyperparameters and analyzing the model. Because of this, we have implemented 2 different cross validation technique to find the best hyperparameter. The detailed cross validation explanations will be explained after analyzing the shifted independent values prediction.

The values of prices and trend values are shifted to the right for 1 day, so our model can get the price and trend values from the day before and predict the corresponding row appropriately. After making these changes, 2 different train test splits are created. Both uses the year, month, day of month values. As an extra, the first one uses shifted trend values, the second one uses shifted trend values and shifted weekly prices of Bitcoin. Here are the data frames to summarize:

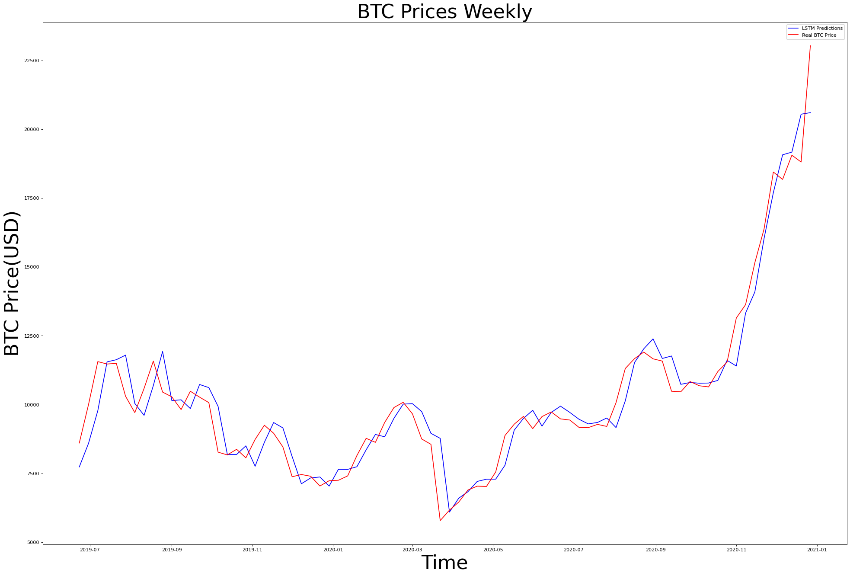


**Fig. 7.** The dataset for shifted mean price and date values

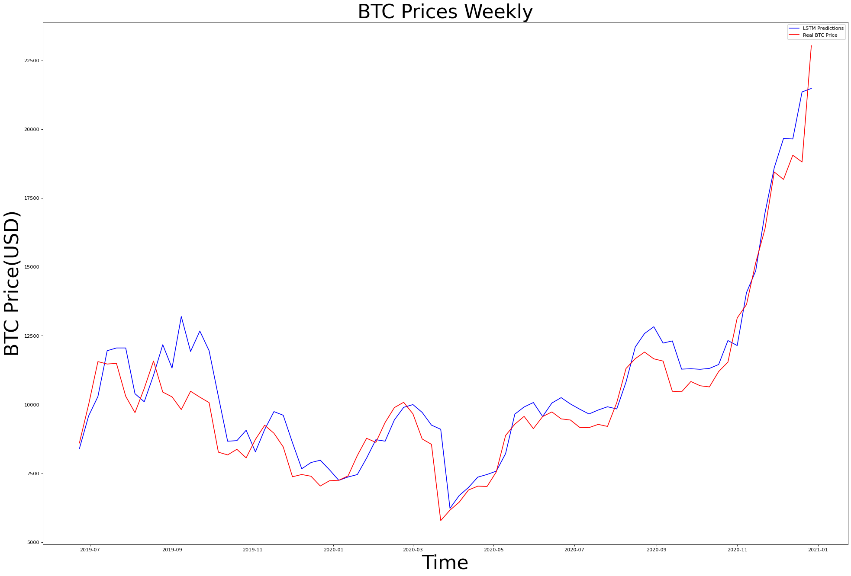


**Fig. 8.** The dataset for shifted mean price, shifted trends and date values

After feeding our model with the independent variables mentioned earlier, we predict the prices based on the test data and show it in 2 graphs. Figure 9 is the prediction using shifted price values, figure 10 is the prediction using shifted price and shifted Google Trends values. The blue line is predicted, and the red line is real weekly prices of Bitcoin.



**Fig. 9.** The prediction graph for shifted price and date values

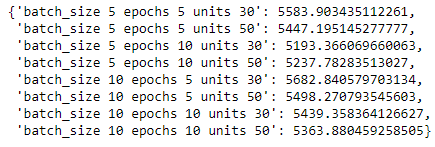


**Fig. 10.** The prediction graph for shifted price, date and trends values

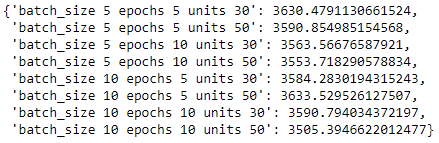
Unfortunately, the shifted Google Trends values did not affect the performance of our model when shifted average prices are used. When we used only shifted average prices as features, the RMSE result is 819.62, the MAE result is 587.59. When shifted average prices and shifted trend values are used as features, the RMSE result is 1024.85, the MAE result is 767.12.

**Bitcoin Cross Validation with LSTM**

2 different cross validation techniques are implemented to find the right hyperparameters for the LSTM model. These techniques are called rolling forecast origin and sliding window, they are explained in depth in the Solution Approach part of this paper. A combination of 5-10 batch sizes, 5-10 epochs, 30-50 units are used as hyperparameters to calculate the root mean squared errors and find the best hyperparameter among them. These combinations are used for all the implementations of the cross validation. More combinations for the hyperparameters could be used but because of the long runtime of the LSTM models we used 8 combinations at total. The rolling forecast origin results when date and trend features are used, can be seen in Figure 11 and the sliding window implementation results can be seen in Figure 12. The best hyperparameters that was found for rolling forecast origin are 5 for the batch size, 10 for the epochs and 30 for the units. The RMSE for this hyperparameter is 5193.36. The best hyperparameters found for sliding window are 10 for the batch size, 10 for the epochs and 50 for the units. The RMSE for this hyperparameter is 3505.

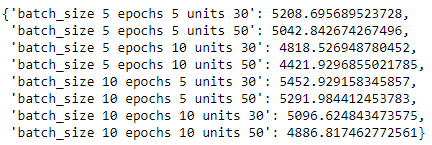


**Fig. 11.** Results for Bitcoin, rolling forecast origin, using date and trends features



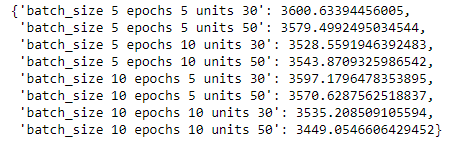
**Fig. 12.** Results for Bitcoin, sliding window, using date and trends features

Previously, the results for features of date and trend values are shown. Then, the same implementation is calculated for the features of date, shifted mean prices and shifted trends prices. The rolling forecast origin results can be seen in Figure 13 and the sliding window implementation results can be seen in Figure 14. The best hyperparameters that was found for rolling forecast origin are 5 for the batch size, 10 for the epochs and 50 for the units. The RMSE for this hyperparameter is 4421.92. The best hyperparameters found for sliding window are 10 for the batch size, 10 for the epochs and 50 for the units. The RMSE for this hyperparameter is 3449.05.



**Fig. 13.** Results for Bitcoin, rolling forecast origin, using date,

shifted trends and shifted mean price features

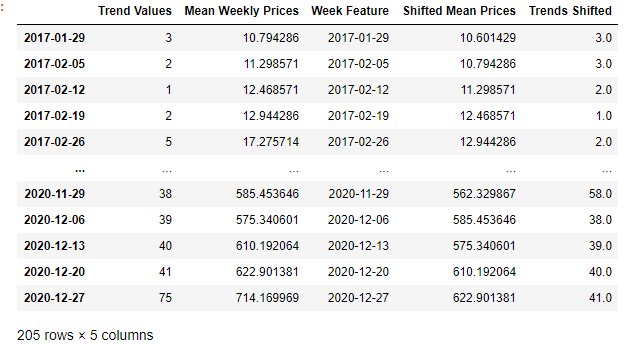


**Fig. 14.** Results for Bitcoin, sliding window, using date,

shifted trends and shifted mean price features

**Ethereum Closing Price Prediction with LSTM**

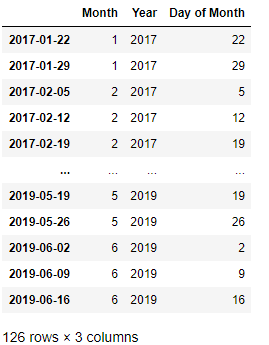
We acquired the data of the closing prices of Ethereum between the beginning of 2017 and the end of 2020 same as Bitcoin. The libraries that were used are as same as the Bitcoin implementation. Using the train test splitting technique, we have split the data into 126 weeks of train data and 80 weeks of test data. Different features are used to train the model. The month, year, day of month, Google Trend values, shifted price values and shifted Trend values are used as independent variables for the LSTM model. These independent variables for Ethereum can be seen below.



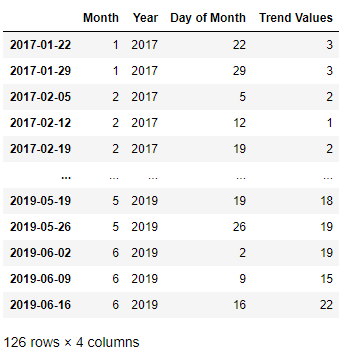
**Fig. 15.** The dataset used for Ethereum price prediction

The default hyperparameter we used for prediction is 50 units for the LSTM, 1 Dense unit, batch size of 5 and 100 epochs. These hyperparameters that were used are the same as Bitcoin to compare the results. These values for the hyperparameter were used to predict Ethereum before hyperparameter tuning with cross validation. We will first analyze the prediction of Ethereum prices using month, year, day of month and Google Trend values. Then, we will analyze the predictions of Ethereum prices by omitting the Trend values and using only month, year, day of month as features. After analyzing the scenarios mentioned earlier, we will be showing the predictions of data that were shifted.

At total, 126 rows (weeks) are used for training and 80 rows (weeks) are used for testing our model. The dependent values of the model are the corresponding average weekly price of Ethereum. Here are the features used for both scenarios:

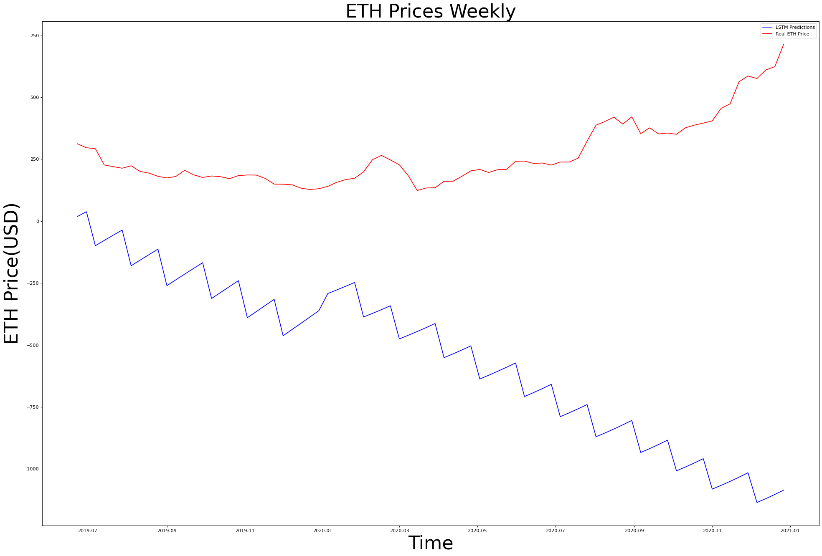


**Fig. 16.** The training dataset for month, year and day of month

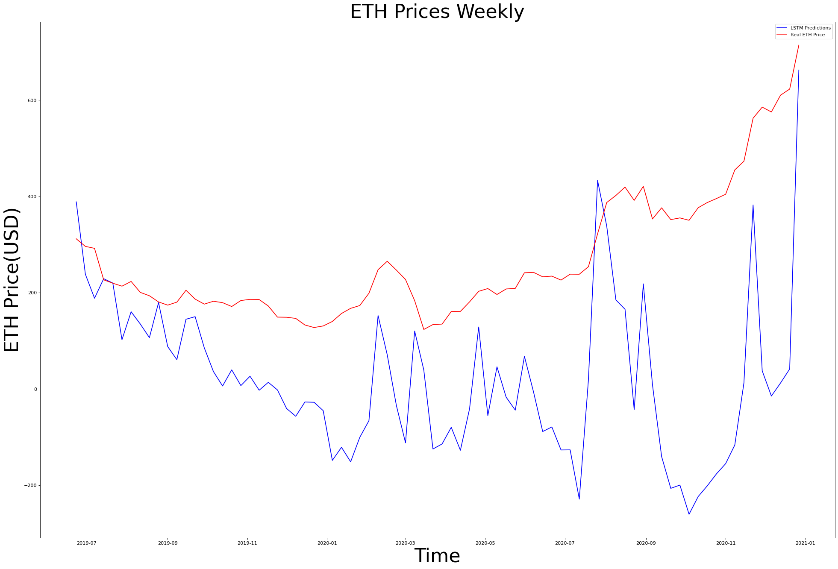


**Fig. 17.** The training dataset for month, year, day of month and trend values

After feeding our model with the independent variables mentioned earlier, we predict the prices based on the test data and show it in 2 graphs. Figure 18 is the prediction using only date values, figure 19 is the prediction using date and Google Trends values. The blue line is predicted, and the red line is real weekly prices of Ethereum.



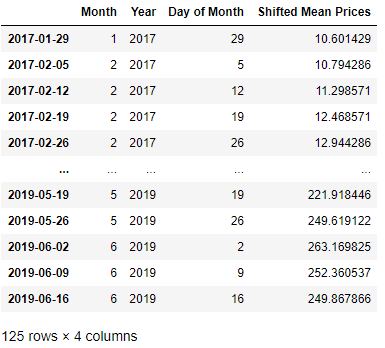
**Fig. 18.** The ETH prediction graph for using only date values



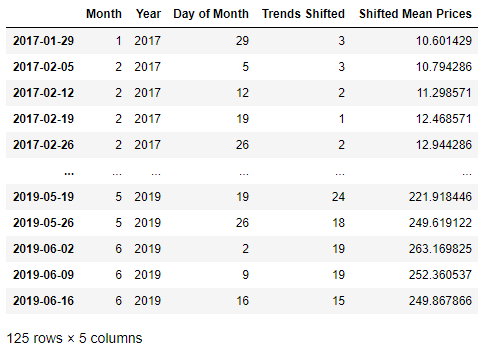
**Fig. 19.** The ETH prediction graph for using date and trends values

The values of Google Trends have significantly enhanced our model, figure 18 only uses the date features which did not help our model at all. Even though figure 19 does not make good predictions, it can be seen that the Trend values have helped our model significantly. We have used Mean Absolute Error and Root Mean Squared Error to further analyze and conclude our predictions. When we used only dates as features, the RMSE result is 921.50, the MAE result is 813.75. When dates and trend values are used as features, the RMSE result is 305.51, the MAE result is 250.91. Same as Bitcoin, we have implemented 2 different cross validation technique to find the best hyperparameter. The detailed cross validation explanations will be explained after analyzing the shifted independent values prediction.

The values of prices and trend values are shifted to the right for 1 day, so our model can get the price and trend values from the day before and predict the corresponding row appropriately. After making these changes, 2 different train test splits are created. Both uses the year, month, day of month values. As an extra, the first one uses shifted trend values, the second one uses shifted trend values and shifted weekly prices of Ethereum. Figure 20 and 21 shows the datasets that were used for our model.

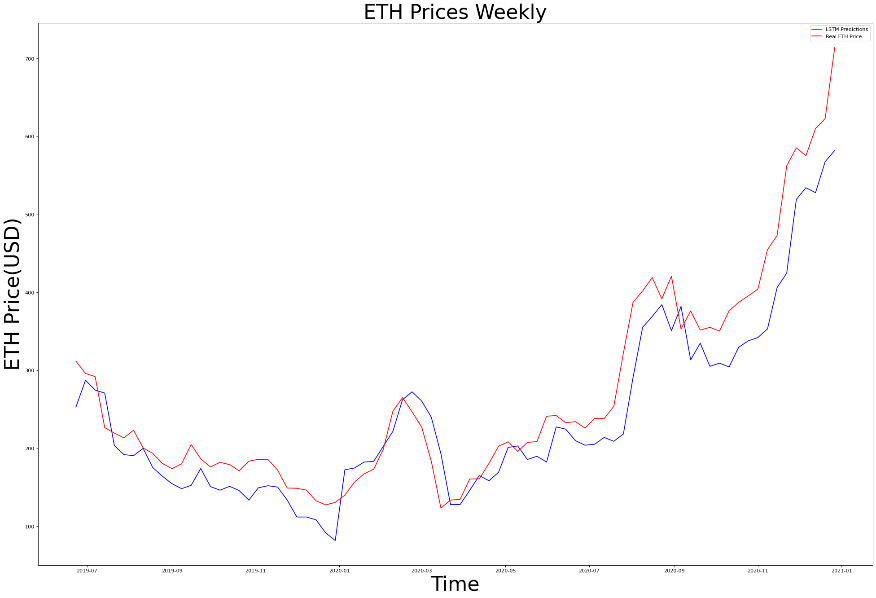


**Fig. 20.** The dataset for shifted mean price and date values

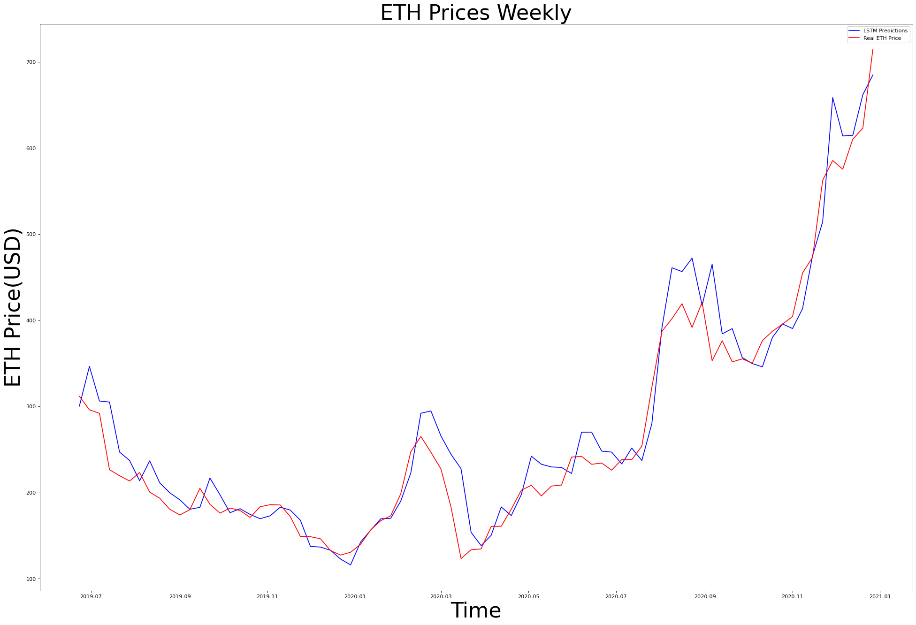


**Fig. 21.** The dataset for shifted mean price, shifted trends and date values

After feeding our model with the independent variables mentioned earlier, we predict the prices based on the test data and show it in 2 graphs. Figure 22 shows the prediction using shifted price values, figure 23 shows the prediction using shifted price and shifted Google Trends values. The blue line is predicted, and the red line is real weekly prices of Ethereum.



**Fig. 22.** The dataset for shifted mean price and date values

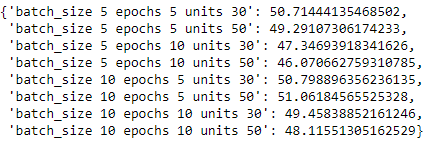


**Fig. 23.** The dataset for shifted mean price and date values

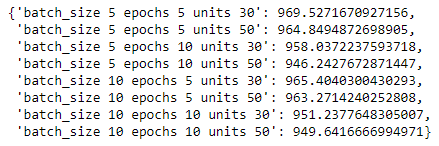
We calculated the Root Mean Squared Error and Mean Absolute Error of these graphs. When we used only shifted average prices as features, the RMSE result is 46.57, the MAE result is 37.17. When shifted average prices and shifted trend values are used as features, the RMSE result is 32.71, the MAE result is 23.25. If we observe both graphs, they look quite similar but after analyzing the metrics we concluded that when we used both shifted Trend and shifted price values, it has helped our model to predict better prices compared to using only shifted price values. We can see the enhancement in these Ethereum predictions where we use the shifted trend values as an extra feature for the LSTM model.

**Ethereum Cross Validation with LSTM**

Same techniques as Bitcoin Cross Validation with LSTM are used, they are explained in depth in the Solution Approach part of this paper. A combination of 5-10 batch sizes, 5-10 epochs, 30-50 units are used as hyperparameters to calculate the Root Mean Squared Errors and find the best hyperparameter among them. These combinations are used for all the implementations of the cross validation. More combinations for the hyperparameters could be used but because of the long runtime of the LSTM models we used 8 combinations at total. The rolling forecast origin results when date and trend features are used, can be seen in Figure ii and the sliding window implementation results can be seen in Figure iii. The best hyperparameters that was found for rolling forecast origin are 5 for the batch size, 10 for the epochs and 50 for the units. The RMSE for this hyperparameter is 46.07. The best hyperparameters found for sliding window are 5 for the batch size, 10 for the epochs and 50 for the units. The RMSE for this hyperparameter is 946.24.



**Fig. 13.** Ethereum, rolling forecast origin results using date and trends features



**Fig. 13.** Ethereum, sliding window results using date and trends features

(Show the results for shifted values)

Previously, the results for features of date and trend values are shown. Then, the same implementation is calculated for the features of date, shifted mean prices and shifted trends prices. The rolling forecast origin results can be seen in Figure 13 and the sliding window implementation results can be seen in Figure 14. The best hyperparameters that was found for rolling forecast origin are 5 for the batch size, 10 for the epochs and 50 for the units. The RMSE for this hyperparameter is 4421.92. The best hyperparameters found for sliding window are 10 for the batch size, 10 for the epochs and 50 for the units. The RMSE for this hyperparameter is 3449.05.

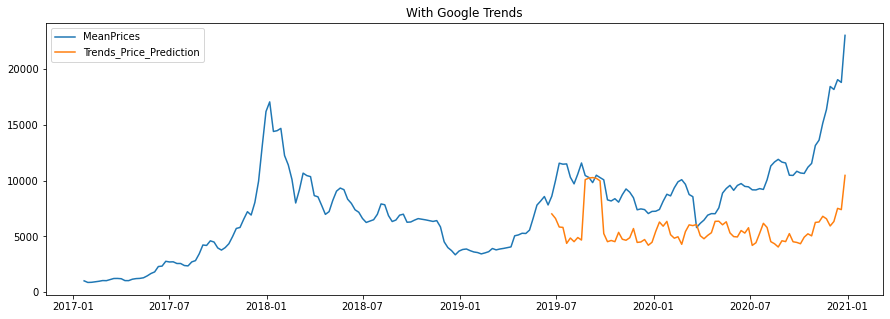
**Time Series Prediction with XGBOOST**

As an alternative to LSTM, we chose the Time Series machine learning algorithm XGBOOST to predict prices of Bitcoin and Ethereum. This machine learning algorithm is an efficient implementation of stochastic gradient boosting for classification and regression problems. XGBOOST is also known to be used for time series forecasting, so this kind of model is well suited for predicting stock market prices or crypto currency prices where speculations are always occurring in the market and the data is based on time.

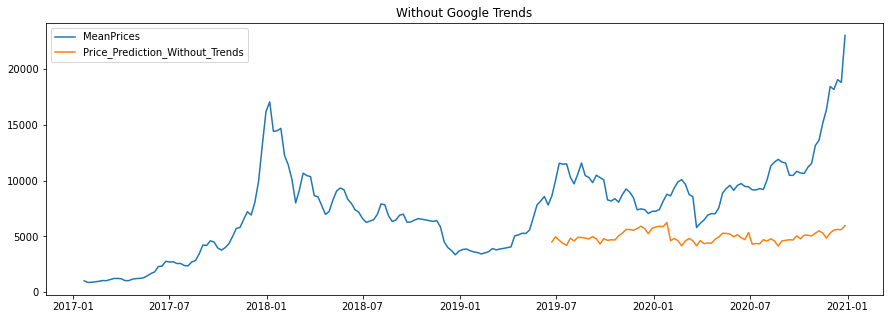
Give some information on XGBOOST here.

**Bitcoin Closing Price Prediction with XGBOOST**

The data that is used in the XGBOOST implementation is the same as the LSTM implementation. Also, the train test splitting ratio (126 weeks of train data and 80 weeks of test data) and different features used for XGBOOST is the same explained as the LSTM model mentioned earlier. The dates of the closing prices are between 2017 and 2020. The xgboost library for Python is used for using the time series model. The default hyperparameter is 50 units for the LSTM, 1 Dense unit, batch size of 5 and 100 epochs. Again, the dataset that will be used for predicting is the same as the LSTM dataset used before. The dependent values of the model are the corresponding average weekly price of Bitcoin. After feeding our model with the independent variables we predict the prices based on the test data and show it in 2 graphs. Figure x is the prediction using date and trends values, figure y is the prediction using only date values. The blue line is predicted, and the red line is real weekly prices of Bitcoin.



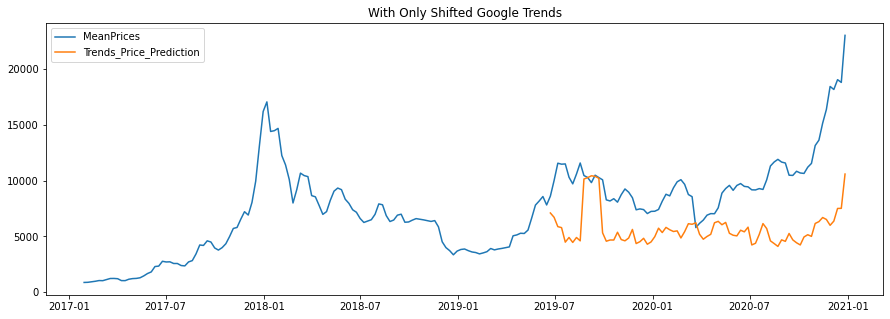
**Fig. x.** The prediction with XGBOOST using Trend and date values

****

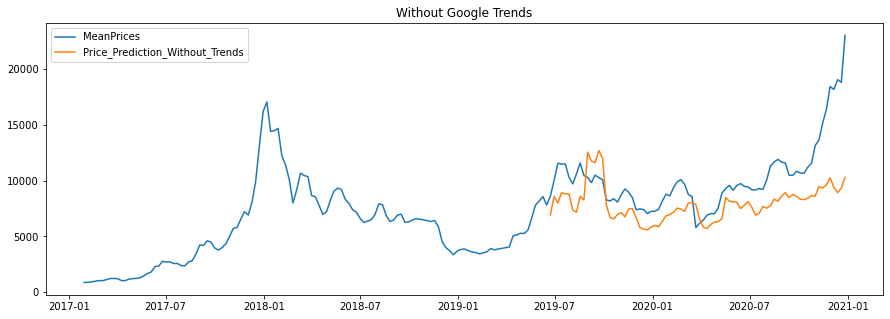
**Fig. y.** The prediction with XGBOOST using only date values

The values of Google Trends have significantly enhanced our model. The sudden spike of 2019-09 is caused by the sudden increase of Trends value. We have used Mean Absolute error and Root Mean Squared error to further analyze and conclude our predictions. When we used only dates as features, the RMSE result is 6052.07, the MAE result is 5256.78. When dates and trend values are used as features, the RMSE result is 5371.73, the MAE result is 4544.24. When we compare these error results it is very clear that the trend values affect the predictions positively. We have implemented 2 different cross validation technique for XGBOOST as well. The detailed cross validation explanations will be explained after analyzing the shifted independent values prediction.

The values of prices and trend values are shifted to the right for 1 day, so our model can get the price and trend values from the day before and predict the corresponding row appropriately. The train test split ratio and the dataset are the same as the one used in LSTM. 2 train, test dataset is created and both uses the year, month, day of month values. As an extra, the first one uses shifted trend values, the second one uses shifted trend values and shifted weekly prices of Bitcoin. We predict the prices based on the test data and show it in 2 graphs. Figure XX is the prediction using shifted trends values, figure YY is the prediction using shifted price and shifted Google Trends values. The blue line is predicted, and the orange line is the real weekly prices of Bitcoin.



**Fig. XX.** The prediction with XGBOOST using dates and shifted trends values



**Fig. YY.** The prediction with XGBOOST using dates, shifted trends and shifted prices values

We used the same metrics that were used for LSTM. When we used dates and shifted trends as features, the RMSE result is 5347.77, the MAE result is 4532.11. When dates, shifted trends and shifted average prices are used as features, the RMSE result is 3381.60, the MAE result is 2559.94. If we observe both graphs, the predictions are quite different from each other. We can see the enhancement in these Bitcoin predictions when shifted prices are added as a feature.

**Bitcoin Cross Validation with XGBOOST**

The techniques explained in the Solution Approach part of this paper are used for this part. A combination of 0.20-0.25-0.30 learning rates, 3-4-5-6-8 max depths, 5-7 minimum child weights, 0.3-0.4 gamma, 0.5-0.7 colsample by tree are used as hyperparameters to calculate the Root Mean Squared Errors and find the best hyperparameter among them. These combinations are used for all the implementations of the XGBOOST cross validation. So at total 120 different models are created and their errors are calculated.

The rolling forecast origin results when date and trend features are used, can be seen in Appendix-10 and the sliding window implementation results can be seen in Appendix-11. The best hyperparameters that was found for rolling forecast origin are 0.2 for the learning rate, 4 for the max depth, 5 for the minimum child weight, 0.3 for the gamma and 0.5 for the colsample by tree. The RMSE for this hyperparameter is 4411.12. The best hyperparameters found for sliding window are 0.25 for the learning rate, 3 for the max depth, 5 for the minimum child weight, 0.3 for the gamma and 0.5 for the colsample by tree. The RMSE for this hyperparameter is 3322.24.

**Ethereum Closing Price Prediction with XGBOOST**

**Ethereum Cross Validation with XGBOOST**

**VI. Related Work**

We have used the Tweepy library on our data collection module for accessing Twitter API. Instead of using this library, we could directly access the Twitter API by sending POST/GET requests. However we would have to implement several methods for various endpoints of Twitter API and that would slow down the development process. Also Tweepy library has built-in support for handling rate-limits of Twitter API which has a great usage in our project since we have to pull large amounts of tweets from Twitter.

For language detection of the tweets and sentiment analysis we have used Microsoft Azure’s Text Analytics services. Instead of using this service for language detection we could have used TextBlob or Polyglot library. For more reliable language detection we have decided to go with Azure and decided to benefit from it’s sentiment analysis service as well thus only dealing with one provider on text analytics operations. It is worth noting that we have used Azure’s free plan that has limitations while other options we have mentioned are free to use although being less capable.

We have used the Scikit-Learn and Keras libraries for using machine learning models Linear Regression and LSTM. Instead of using these libraries we could have used other libraries like PyTorch or Theano. However, we have decided to use Keras and Scikit-Learn because they were easier for us to implement and we are very familiar with these libraries. Furthermore, there isn't a lot of difference in the models themselves. So if we used a different library for our machine learning models, the results would not change significantly. The closing price data we collected are from the official CoinMarketCap website. These prices do not change around the world, that's why there wouldn't be any change if we decided to take the closing prices elsewhere. For Linear Regression, Google Trends bitcoin data was used as an independent variable. Since Google is the most used search engine on the internet, we used Google’s Trends application to gain insight of the market speculation. We could have used Yandex or other search engines but Google would give us the most accurate scores for the market speculation. Because of this, we only used Google's data.

**VII. Conclusion and Future Work**

We have implemented Linear Regression and LSTM time series machine learning models on the Bitcoin closing prices. Different metrics, graphs, tables and plots are used to further analyze the predictions and the data. As it was mentioned in the solution approach part, the data was collected from different resources. The Azure Sentiment Analysis was done with collected tweets and the Google Trends data was downloaded from the official website. Very good accuracy results were found with the time series LSTM model. There are no social impacts we can think of but these outputs are very beneficial for investors and people who are new to the market. There are some economic issues that come with our work. Investors should always invest their money while assessing their own risk. These cryptocurrencies are very volatile and they can change negatively, positively in a small amount of time. Investors should research these cryptocurrencies and invest their money accordingly. In the second term, we are planning to create predictions with different time series models like XgBoost, Prophet and ARIMA. Furthermore, we will be using the Twitter Sentiment Analysis data to use as a feature for Linear Regression and calculate the accuracy reports, metric reports. Different metrics, parameters will be used for the previously mentioned time series models too.

**Acknowledgements**

We would like to thank Dr. Emre SEFER for guiding us through this project and providing valuable feedback. **References**

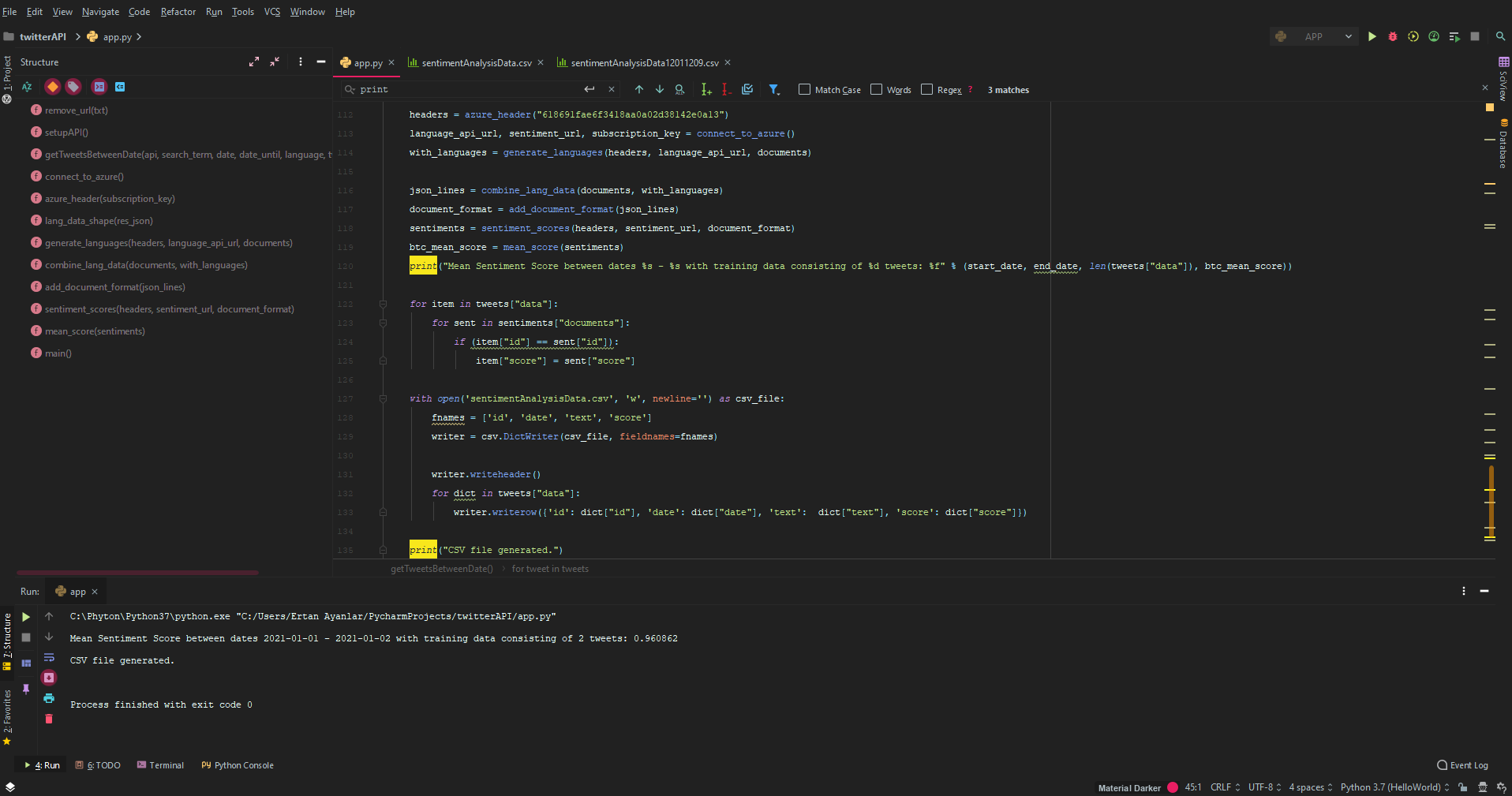
1. Satoshi Nakomoto “Bitcoin: A Peer to Peer Electronic Cash system”
2. Michael Phi (Illustrated Guide to LSTM and GRU: A step by step explanation)
3. SMUDataScience Review(<https://scholar.smu.edu/cgi/viewcontent.cgi?article=1039&context=datasciencereview>)
4. “Bitcoin 17.7 million Tweets and price” by Jaime Badiola (<https://www.kaggle.com/jaimebadiola/bitcoin-tweets-and-price>)

**Appendix**

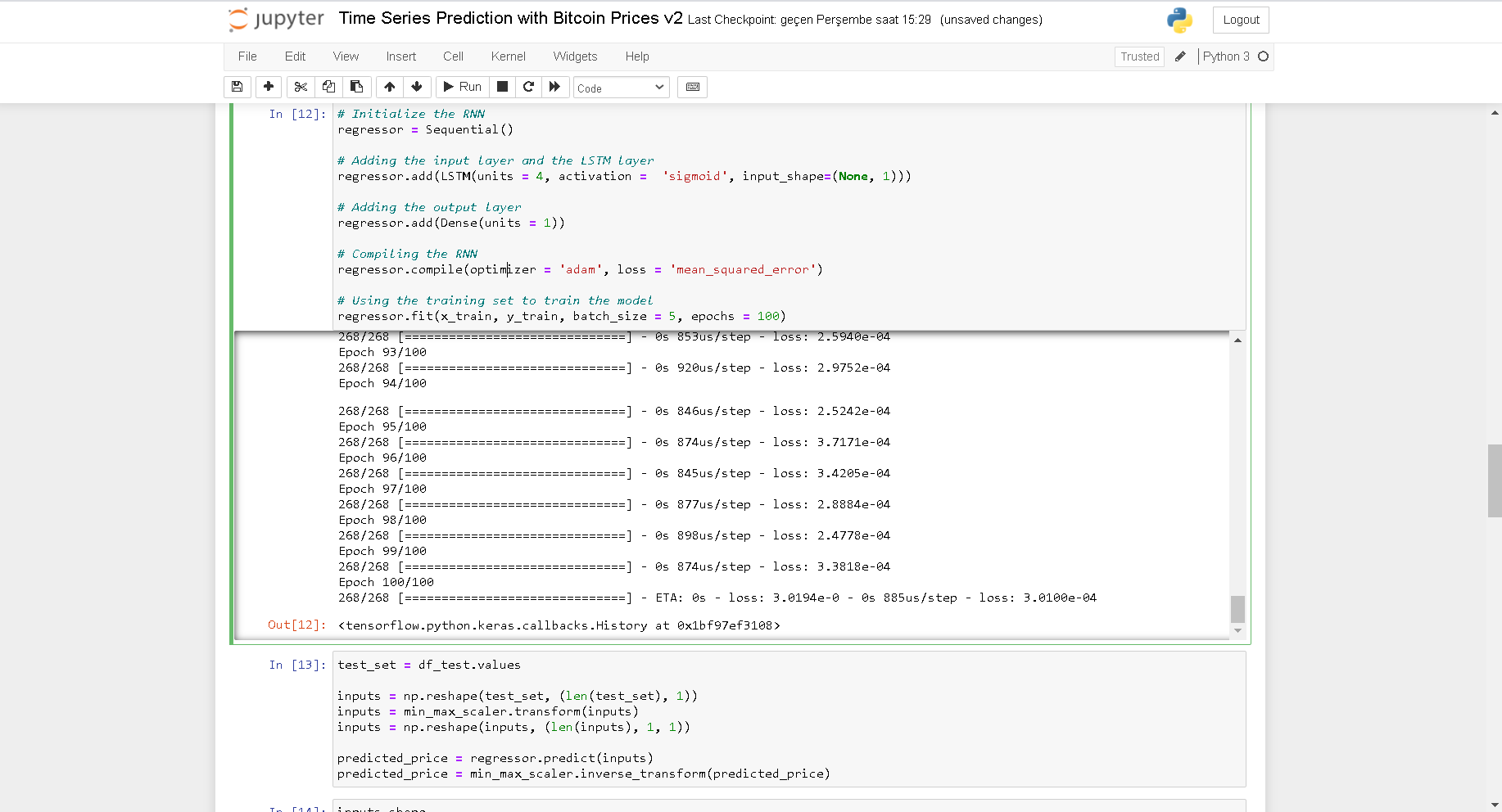
**1) Sample Sentiment Analysis Results**

****

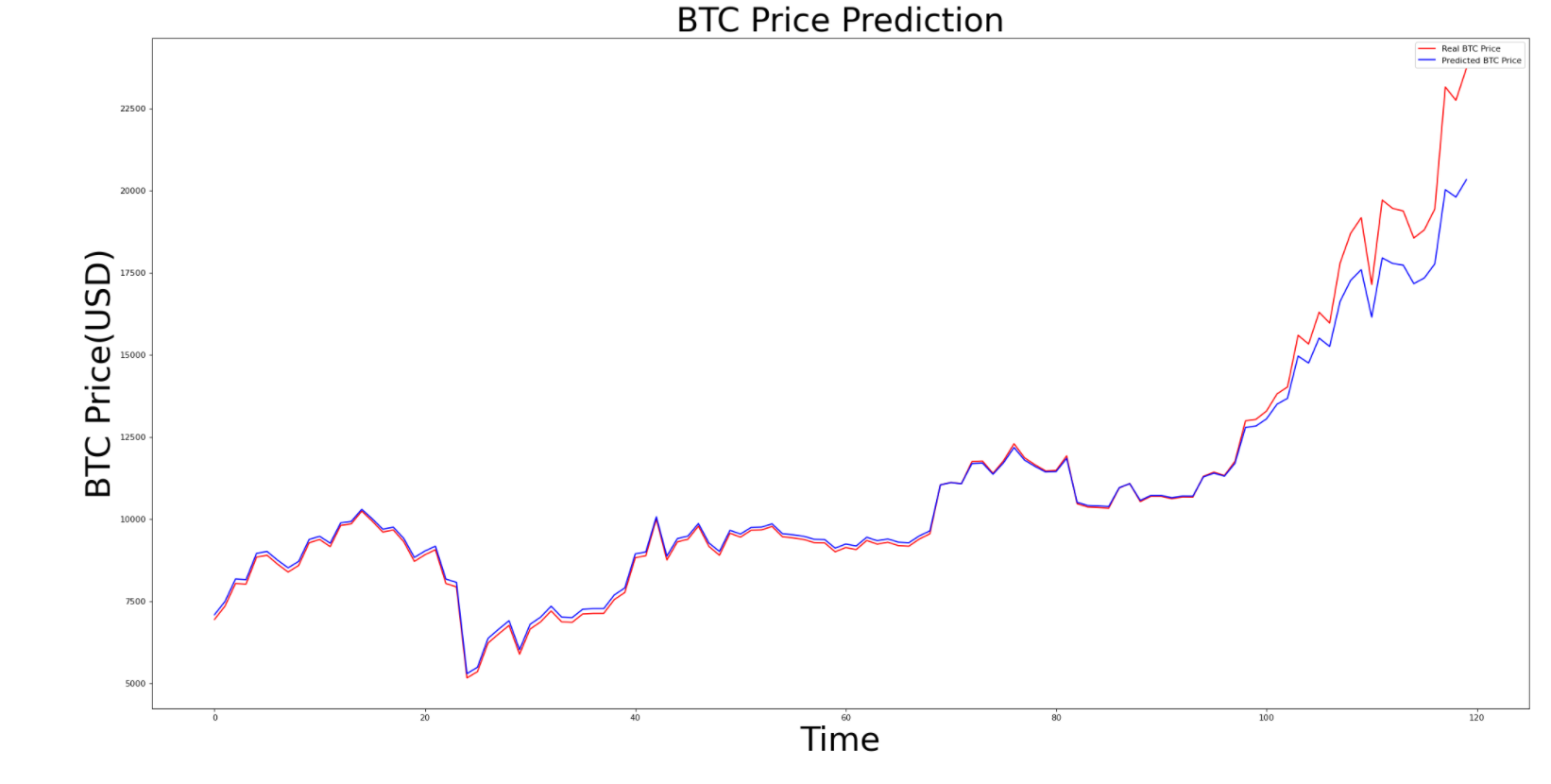
****

**2) Data Collection Module IntelliJ PyCharm Working Environment**

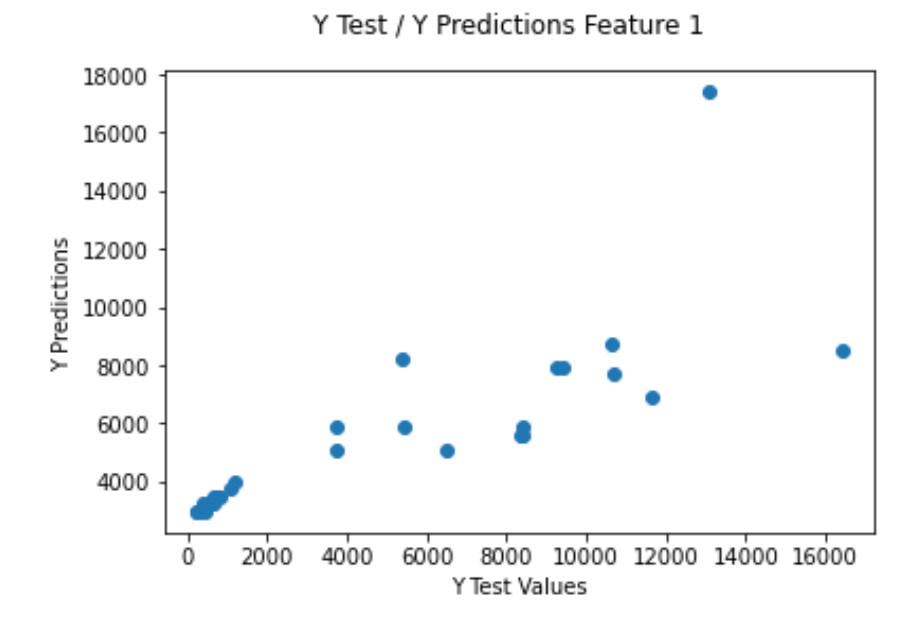
**3) Price Prediction Module Jupyter Notebook Working Environment**

****

**4) Price Prediction Module Time Series LSTM Prediction Graph**

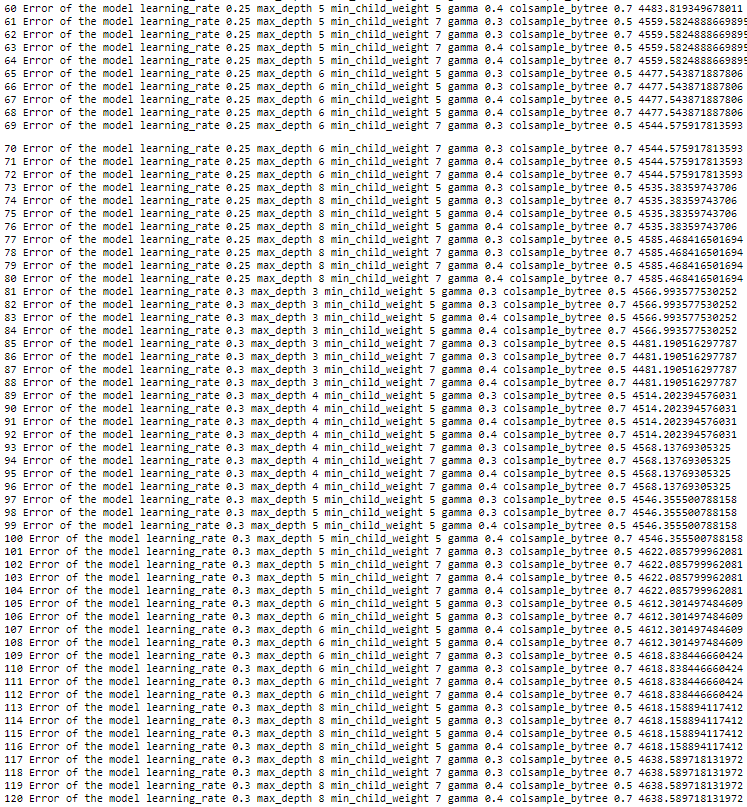


**5) Price Prediction ModuleLinear Regression with Google Trends Data Prediction Graph**

****

Table

Description automatically generated with low confidence**10) Bitcoin Rolling Forecast Cross Validation Results using XGBOOST**



**11) Bitcoin Sliding Window Cross Validation Results using XGBOOST**