ÖZYEĞİN UNIVERSITY
FACULTY OF ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE

CS 402

2022 Spring

SENIOR PROJECT REPORT

Analyzing Transaction Graphs for Price Prediction of Bitcoin

By
Peker Çelik & Eray Erdoğan & Umut Çırak

Supervised By
Dr. 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: Peker Çelik, Eray Erdoğan, Umut Çırak

Signature: 

Place, Date: 06.01.2022

Abstract

The crypto market is more volatile than stock markets and is rapidly affected by the news flow and politics. There are many complex indicators and methods used for technical analysis. However, with the gradual development of learning models and data science, large investors and corporate companies are securing their investments through more comprehensive and highly accurate indices developed [1].

Prediction tasks on financial time series are notoriously difficult, primarily driven by the high degree of noise and the generally accepted, semi-strong form of market efficiency [2]. In this paper, several prediction models integrating machine learning and statistical analysis tools are presented to predict the trend of bitcoin market and also price.

The main focused subject in this study is to obtain meaningful outputs from the historical bitcoin transaction database and increase the accuracy of prediction by using these results. While developing the model, two transaction datasets, weekly and monthly, were used. Transaction data covers the years 2010 to 2015 as monthly and 03-2011 to 09-2013 as weekly.

Recurrent neural networks (RNN) are one of the most powerful models for sequential data and Long Short-Term is one of the most successful RNNs architectures. LSTMs are less commonly applied to financial time series predictions, yet inherently suitable for this area [3].

One of the efficient ways to analyze the transaction dataset is the graph format. In this format, many characteristic features of graphs can be obtained as a single output. In our project, we measured whether the transaction graphs are effective in price prediction. The contribution of these datasets to the accuracy rate in many different learning algorithms such as LSTM, KNN, SVM, Random Forest, Decision Tree within the project was measured and compared.

Contents

I. Introduction

Bitcoin has become increasingly popular for its simplicity, transparency, and innovations to the financial system, and this is reflected in its market value. Bitcoin is a cryptocurrency and global payment system powered by its blockchain network and sees over 400,000 transactions taking place per day.

Blockchain is a distributed database where all public transactions and transactions are recorded [4]. The information accumulated in this database is verified by the consensus of the majority of the devices (which are called mining devices) operating in the blockchain network, and the transaction added to the database remains there forever.

In the Bitcoin network, transactions are not confirmed or secured by a third mechanism, cryptographic proof is used. Each user has one private and one public key in their wallet in the bitcoin network. In a transaction between two users, the recipient's wallet address, the sender's wallet address, and the BTC amount are transparently displayed on the network.


```
{'amount': 23.15627,
 'amount_usd': 970860.8,
 'blockchain': 'bitcoin',
 'from': {'address': 'Multiple Addresses',
          'owner': '',
          'owner_type': 'unknown'},
 'hash': 'f633af19c9035689462f2aeee7f70089eebf60b0f1539938a86e8141c219c01b',
 'id': '1777782196',
 'symbol': 'BTC',
 'timestamp': 1641583823,
 'to': {'address': 'bc1qq2kn8pddlwkypmxtsvhm0zq72hqqx56ha37z94',
        'owner': '',
        'owner_type': 'unknown'},
 'transaction_count': 1,
 'transaction_type': 'transfer'},
```


Fig. 1: Bitcoin Transaction Structure

One of the useful artificial neural networks for us to test the effectiveness of graphs in price prediction is LSTM. The model that makes time-series price forecasting by utilizing historical Price data and transaction metrics. In our LSTM model, we did not use the results from the transaction graphs in our training split at first, and we measured our price accuracy. Then we added our graph parameters, observed that our error rates decreased and the prediction accuracy increased.

Outside of the graph metrics, relational vector representations make it easy to apply data analysis and machine learning approaches to the structures[5]. In the machine learning and knowledge representation literature, a variety of methods for generating such embeddings have been studied.

Graph embeddings which is basically a way of representing a graph so you can leverage that in a machine learning model to calculate similarity. Graph embeddings are specific types of embedding that translate graphs to fixed length tensors, basically encode the entire graph into single vectors. Node2Vec is an algorithm for mapping nodes in a graph to an embedding space. In general, the embedding space has fewer dimensions than the original graph's number of nodes.

II. Background

NetworkX:

NetworkX is a package for the Python programming language that's used to create, manipulate, and study the structure, dynamics, and functions of complex graph networks.

Keras:

Keras is usually used for small datasets. Neural layers, cost functions, optimizers, initialization schemes, activation functions, and regularization schemes are all standalone modules that you can combine to create new models. New modules are simple to add, as new classes and functions. Models are defined in Python code, not separate model configuration files.

Sklearn:

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistent interface in Python.

TensorFlow:

It is an open-source artificial intelligence library, using data flow graphs to build models. It allows developers to create large-scale neural networks with many layers. TensorFlow is mainly used for: Classification, Perception, Understanding, Discovering, Prediction and Creation.

Numpy:

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms basic linear algebra, basic statistical operations, random simulation and much more.

III. Problem Statement

Project Scope

In this project, our aim is to build an improved model by performing a comprehensive bitcoin on-chain (transaction) analysis together with traditional time-series forecasting methods used in bitcoin price forecasting in academia.

Models based on on-chain analysis in published studies have a low success rate and are mostly based on transaction analysis only. We have built a bridge between bitcoin price data and transaction data and obtained an improved model.

Engineering Problem

The crypto market is known for being volatile, dynamic, and nonlinear[6]. Accurate bitcoin price prediction is extremely challenging because of multiple (macro and micro) factors, such as politics, global economic conditions, unexpected events, a company's financial performance, and so on.

Predicting modeling is one of the most popular mathematical methods in many fields. One of the most important studies among these areas is stock price estimation.

Most traditional studies on price prediction to date are based on Regression models, Time series models, ARIMA, LSTM. Projects that do extensive analysis on transaction information between wallets for Bitcoin price prediction are very few and have a low success rate. In addition, these studies are mostly based on transaction analysis only.

In this project, we aimed to establish a more comprehensive and multi-parameter LSTM model and supervised learning models such as KNN, SVN, Decision Tree, Random Forest by combining embeddings of transaction graphs and bridging the developed models used in stock price estimation and models that perform transaction analysis. We aimed to contribute to the insufficient number of studies in the academy by conducting a study on the correlation between transaction analysis and bitcoin price data.

Assumptions

- We will have free and complete access to the Bitcoin Transaction Data Set.
- We will have mostly enough processor power and ram capacity to process transaction graphs.
- The metrics we obtain from the transaction graphs will yield meaningful results that will help us improve our price prediction model.
- Our model will be minimally affected by unexpected news flows, political events, and global economic change processes.

Constraints

- Even when we split the transaction dataset into graphs on a monthly basis, we will never have enough computing power and storage capacity to process it holistically, as it contains billions of transactions and millions of nodes in a set.
- Since the crypto market is much more volatile than the stock market and is much more affected by the news feed, the accuracy of our model remained lower than the target rate.
- Since transactions between wallets are in many cases emotionally driven by people's panic and complacency conditions in the market, and also because bitcoin transactions are transparent, these transactions made by large investors to mislead small investors may compromise the tangibility of the analysis we are trying to do and may not yield any meaningful conclusions.
- Most of the transactions that take place in the Bitcoin network are done by a method called coinjoin. The coinjoin method reduces multiple transactions to one, and we see this transaction in the network as multi-wallet to multi-wallet. For this reason, we need to use the coinjoin parser to make the transaction singular, and this is a difficult process.

IV. Solution Approach

4.1. Experimental Setup

The experimental environment of this study was prepared in the following three stages,

The first step is collecting transaction datasets and splitting these sets into weekly and monthly. This stage is a very important and meticulous stage because the data is very large and combining data into learning models is a more time-consuming process compared to training. When choosing the date range of the transaction data set we used, we paid attention to the fact that it was independent of external factors and that the price volatility was relatively low. That's why we started at the beginning of 2010, a date when bitcoin started to be recognized and priced after its invention. As we get closer to the present, the increase in bitcoin transactions has made the dataset very large and we used data up to 2015 because there was not enough cpu power to process this data.

As a second step, research was conducted on choosing the right methods to study the efficiency of transaction data. The architecture has been tuned with the right parameter setting for LSTM and other supervised machine learning models to work best. Measurement metrics were determined in order to compare the models.

The usability of transaction data in predicting the price of cryptocurrencies was evaluated together with the results obtained as a third step.

4.2. Data Preparation

Data preparation in the scope of this project involves graphing, graph sampling, obtaining graph embeddings, consolidation and data division for test and training.

1. Graphing

A graph database stores nodes and relationships instead of tables. It Navigates deep hierarchies and shows inter-relationships. This process is one of the most important processes in building our model.

	addressIn	addressOut	Date	Bitcoin
200020	7edb88cbe187f0e5ac5b0cefd4c69c3a7963abf36fb279...	6aa54fb5186e4f6cf95342d6d43fd787617fb7909cf4db...	12/28/2010 7:06	0.05
200021	0730bf7e6ff0aae5f73f18e675054c7098000f20b01d88...	86f155c96a72fcd1fc5eaa423bebf059a7ca740644dece...	12/28/2010 7:45	50.00
200022	57aa731309c3ce68669730c5452cc867856b31ce22a7ff...	4ddc92c14484dbbfbc285e65171944a59a10540691a201...	12/28/2010 7:45	346.05
200023	57aa731309c3ce68669730c5452cc867856b31ce22a7ff...	bfad39aa79284337ade44153ca366c95d2c61887bec4a4...	12/28/2010 7:45	0.05
200024	845bf8bc6de76a5857ad925fb34dad02657ec25664798d...	04000fc6050fca089d81dd9c27d475347af8a5b4a38ba8...	12/28/2010 7:45	144.56
...	...	...	...	...
249995	38fafce6c5bbadadd5875575dadf421f667b21095f4759...	61e53f75def3a12cec9732c5cf243160ae877089d19887...	1/31/2011 16:25	100.00
249996	fa7d841a7c1791735bcebb801157e01cb332550e4ec145...	1c307212794066081b7141cd1af6c022ad367059dfe5f6...	1/31/2011 16:25	26.00
249997	45d1bf58421ecc773dfc975089b2ddcab82fb37a6005f9...	b02fb0b5f2a5fbdd6f1ae485b79cd95cb5c848a734c953...	1/31/2011 16:25	210.70
249998	81514ac138f4f514e8d62122c4e95eceb2cf42e6b4b6b5...	62d588c0cde51f0a0780f5a458fe51b200b85cd9935030...	1/31/2011 16:25	5.01
249999	81514ac138f4f514e8d62122c4e95eceb2cf42e6b4b6b5...	c6f0ba286134619da6069fc0888871c8ee3456d82c7932...	1/31/2011 16:25	12.21

Fig 2: Part of Transaction Dataset

Network graphs show interconnections between a set of entities[7] where entities are nodes and the connections between them are represented through links or edges. In the graphs we created in Bitcoin transactions, each node represents the wallet address, and the edge between the two nodes represents the transactions processed.

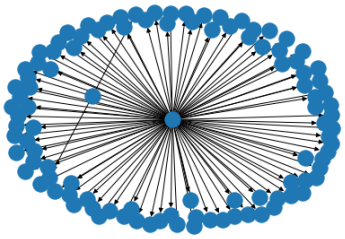

Fig 3: Daily Graph

We turned these transactions into graph networks in two different ways. We first analyzed the transactions that took place on the network as daily graph networks, and then we converted them into weekly graphs.

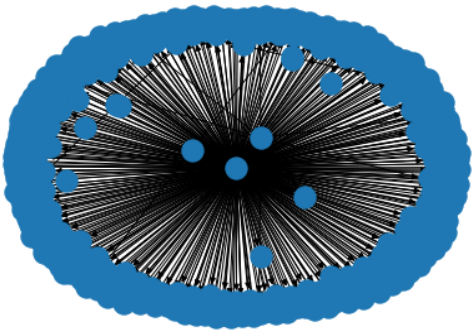

Fig 4: Weekly Graph

When we compared the model daily and weekly graphs, it gave a lower error rate with the weekly data. That's why we continued to use weekly graphs.

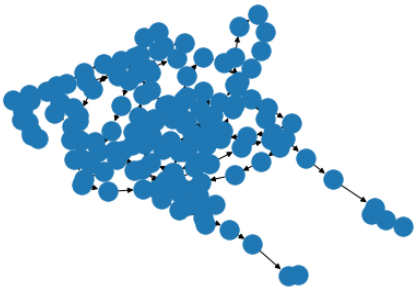

Fig 5: Transaction Graph[Closer Look]1

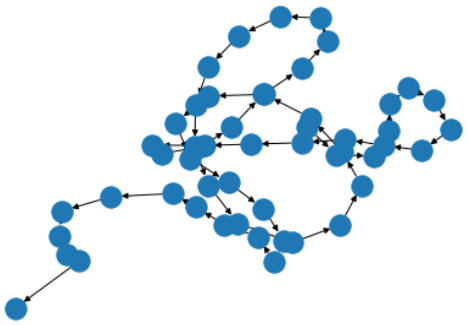

Fig 6: Transaction Graph[Closer Look][2]

2. Graph Sampling

Graph sampling is an approach to sampling from nodes and edges without destroying the characteristic features of the original graph with an algorithm. Commonly used techniques are Vertex Sampling, Edge Sampling, and Traversal Based Sampling. Exploration or traversal (also called topology-based) approaches are based on the idea of randomly selecting one node and then exploring its neighborhood.

Traversal (topology-based) approaches are based on the idea of randomly selecting one node and then exploring its neighborhood [8]. Sampling algorithms based on these techniques are Simple Random Walk Sampling (SRW), Random Walk Sampling with Fly Back Probability (RWF), Induced Subgraph Random Walk Sampling (ISRW), Snowball Sampling (SB), forest fire Sampling (FF), and more.

Another method, which is edge sampling, focuses on the selection of edges rather than nodes to populate the sample. Thus, the node selection step in the edge sampling algorithm proceeds by just sampling edges, and including both nodes when a particular edge is sampled.

Instead of using these algorithms as the graph sampling method in our project, we took a random sample containing 50000 nodes from each graph. The reason we prefer this method is that transaction graphs are too large to sample and we do not provide enough processing power and RAM capacity to use these methods.

3. Obtaining Graph Embeddings

In the machine learning literature, a variety of strategies for producing embeddings have been investigated. Embedding is a method of converting a complex dataset into a simple fixed-length vector (tensor) that captures important information while lowering dimensionality. Distances between embedded vectors can capture important notions in the original input by capturing similarity between various data points.

The translation of a graph into a vector or a group of vectors is known as graph embedding. The graph topology, vertex-to-vertex connection, and other essential information about graphs, subgraphs, and vertices should be captured during embedding.

Graph embeddings are frequently utilized as input features in machine learning studies including link prediction, community recognition, classification, and other similar tasks. For this reason, it was a method that we included in our study, we used the embeddings of our weekly and monthly graph data in machine learning models. Graph embeddings can be generated through various algorithms, in this project the Node2Vec algorithm is preferred.

Node2Vec is a method for mapping nodes in a graph to an embedding space. The algorithm aims to keep the original graph's structure; comparable nodes in the network will produce similar embeds in the embed region. Each node in the network is represented by a vector in these embedding spaces.

Node2vec has random walks in its working mechanism. If every step is calculated probabilistically, this means that every time the index is shifted in a certain direction depending on a probabilistic outcome[9]. This method explores the link between each step made and its distance from the beginning position. The following formula is introduced by Node2Vec for estimating the chance of migrating to node x if you were previously at node v:

$$P(c_i = x \mid c_{i-1} = v) = \begin{cases} \frac{\pi_{vx}}{Z} & \text{if } (v, x) \in E \\ 0 & \text{otherwise} \end{cases}$$

Fig 7: Probability of moving (retrieved from [10])

The unnormalized transition probability between nodes x and v [10] is vx, and Z is the normalization constant. For unweighted graphs, the Node2vec approach would be a better alternative since if each edge had a weight, it would be easier to generate a bias that could affect random walks.

The Node2Vec algorithm implies a directed random walk with two parameters, p and q. The likelihood of a random walk returning to the previous node is p, while the probability of a random walk passing through previously unknown parts of the graph is q.

$$\alpha_{pq}(t, x) = \begin{cases} \frac{1}{p} & \text{if } d_{tx} = 0 \\ 1 & \text{if } d_{tx} = 1 \\ \frac{1}{q} & \text{if } d_{tx} = 2 \end{cases}$$

Fig 8: Random Walk, (retrieved from [10])

Where dtx is the shortest path between nodes t and x.

The SkipGram model has an important place in the node2vec algorithm. The skip-gram model is a basic neural network with one hidden layer that predicts the likelihood of a given word being present when an input word is present. [11] The target word's pairings with all other words make up the training data.

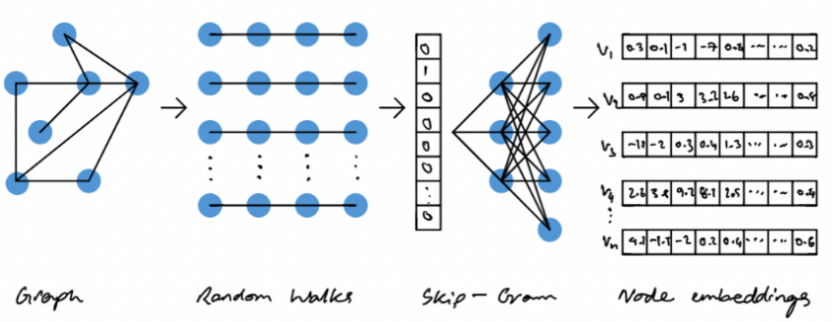

Fig 9: Node2Vec Architecture (Image taken from [11])

Node2vec starts by taking a graph as input and extracting a collection of random walks from it. The walks are then represented as a word sequence, with each node representing a word. After that, the produced random walks are fed into the skip-gram model then skip-gram produces an embedding for each node.

4. Consolidation

Consolidation is the process of combining all features used and graph features as a single data frame. In this stage, the historical bitcoin transaction data is collected from https://padlab.ir/data/bitcoin-mtg.tar.gz, https://ieee-dataport.org/open-access/bitcoin-transactions-data-2011-2013#files as monthly and historical bitcoin price data is collected from https://www.investing.com/crypto/bitcoin/historical-data. Weekly transaction data collected from Google Cloud's BigQuery service. We combined the graph metrics we obtained from the graphs we created from the transaction dataset with another dataset containing the price, volume, change(%) data for the LSTM model.

Date	Close	Open	High	Low	Vol	Change %	average_clustering	density	number_of_edges	number_of_nodes	degree_assortativity_coefficient
7-Nov-10	0.3	0.4	0.5	0.1	341460.0	-0.29	0.0074548658777515	2.9012294078308782e-05	83901	53777	-0.0342585661394527
14-Nov-10	0.3	0.3	0.3	0.2	146410.0	0.0	0.0050571396101364	3.131308710620072e-05	81415	50991	-0.0147054715207112
21-Nov-10	0.3	0.3	0.3	0.3	73380.0	0.0	0.0043213514039311	3.126867610145768e-05	76046	49316	0.0141049336366575
28-Nov-10	0.2	0.3	0.3	0.2	136930.0	-0.28	0.0068882426087726	2.7828533289361982e-05	90315	56969	-0.0540242954167851
5-Dec-10	0.2	0.2	0.2	0.2	66890.0	0.0	0.0064790288552684	2.731113660556557e-05	90881	57686	-0.0282287908286876
12-Dec-10	0.2	0.2	0.3	0.2	37500.0	0.0	0.0055697249101868	2.938139011669976e-05	76156	50912	-0.000225776409106
19-Dec-10	0.2	0.2	0.3	0.2	40100.0	0.0	0.0063743921357602	2.693356267294541e-05	85118	56217	-0.0096989907020586
26-Dec-10	0.3	0.2	0.3	0.2	62380.0	0.2	0.0045836144898547	2.8235704288134143e-05	79169	52952	-0.0086964978702854
2-Jan-11	0.3	0.3	0.3	0.3	56700.0	0.0	0.0049278816703327	2.3585236189484776-05	92645	62675	0.0401496697250584

Fig 10: Consolidated data frame with graph metrics

	Date,	Price	Open	High	Low	Vol.	Change %	0	1	2	...	246	247	248	249	250	251	
0	7/3/2011	14.4	15.4	16.5	11.0	366200.0	-6.62	0.143607	-0.070265	0.278090	...	0.362808	0.368988	0.364313	0.352094	0.355708	0.344235	0.
1	7/10/2011	13.7	14.4	15.7	13.5	190880.0	-4.59	0.089774	-0.207905	0.256683	...	0.344774	0.349917	0.348386	0.340110	0.360732	0.347816	0.
2	7/17/2011	13.7	13.7	14.7	12.5	263670.0	0.00	0.086321	-0.093323	0.228172	...	0.340600	0.362379	0.346005	0.337340	0.351533	0.349064	0.
3	7/24/2011	13.5	13.7	14.7	13.3	132600.0	-1.10	-0.004186	-0.129215	0.149419	...	0.298482	0.305089	0.290398	0.289368	0.307518	0.313389	0.
4	7/31/2011	6.6	13.5	14.9	5.7	398270.0	-51.59	0.047130	-0.083402	0.200965	...	0.254703	0.275766	0.265648	0.259244	0.257194	0.265005	0.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
110	8/11/2013	112.8	103.0	115.0	102.7	139760.0	9.47	0.110399	-0.196330	0.266526	...	0.319754	0.347547	0.350595	0.333103	0.334136	0.345477	0.
111	8/18/2013	119.6	112.8	125.0	111.8	160000.0	6.08	0.104036	-0.236846	0.293611	...	0.314435	0.343113	0.337746	0.333709	0.325222	0.329960	0.
112	8/25/2013	141.0	119.6	148.7	119.1	170740.0	17.89	0.000947	-0.145668	0.324759	...	0.287848	0.308039	0.298714	0.274369	0.292085	0.297317	0.
113	9/1/2013	129.0	141.0	148.9	121.3	130890.0	-8.52	0.123649	-0.091705	0.309119	...	0.326863	0.339807	0.325321	0.316037	0.326532	0.330963	0.
114	9/8/2013	136.7	129.0	145.9	124.0	137010.0	5.98	0.097686	-0.140164	0.285660	...	0.331062	0.322227	0.322605	0.324599	0.322069	0.315098	0.

Fig 11: Consolidated data frame with vector embeddings

Graph vectors were combined with other features for use in supervised machine learning models.

5. Train and Test Split

Train test split of the dataset is slicing the data set according to some ratio with or without randomly having two split datasets with the purpose of the train, validate and test the model. It is important because depending on the characteristics of the dataset and the topic, one method may have a significant advantage over the others.

We split the dataset into two different techniques. First, we split randomly with the ratios of 80% training and 20% test. Secondly, we split the dataset with a sliding window perspective. At that point, we tried different variants and one of the best solutions is training for 2 months and testing for 1 day so that we divide it in that order.

4.3. LSTM Model

Long Short Term Memory networks are a special kind of RNN, effective at capturing long-term temporal dependencies.

LSTMs on the other hand are both general and effective at capturing long-term temporal dependencies [5]. They minimize the optimization problems that affect the operation of simple recurrent networks (SRNs). Gates in the LSTM structure determine what will be remembered and what will be forgotten. That is, if the incoming input is unimportant, it is forgotten, if it is important, it is transferred to the next stage. It does this with the help of Gate and Cell State.

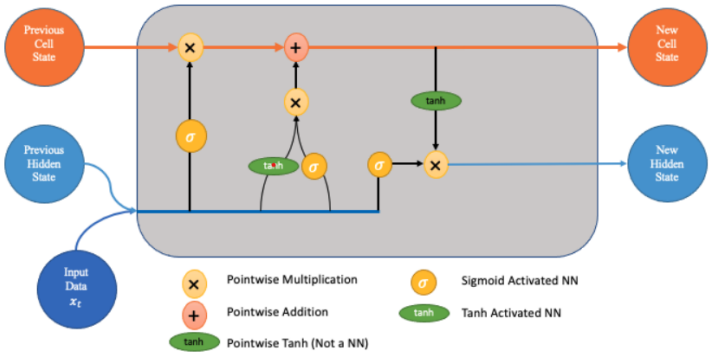

Fig 12: Long Short Term Memory
Retrieved from: https://towardsdatascience.com/lstm-networks-a-detailed-explanation

1. Graph Metrics

Metrics characterize the structure of the graph, the graph connection, and its components, the distance between nodes, various definitions of centrality, etc. we can reach knowledge that the node depends on its neighbor [9].

The NetworkX library supports graphs like these, where each edge can have weight. We use some of the metrics to determine and analyze the graph features, for that reason we use the NetworkX library. These metrics are respectively, Density, Assortativity Coefficient, Average Clustering Coefficient.

1.1 Density

Simply we can define graph density. The density of a graph is a measure of ties comparing how many ties among nodes there have been and how many ties between nodes are possible.

The density of the undirected and directed graph is quite simply calculated as in figure 4

$$UndirectedNetworkDensity = \frac{TotalEdges}{TotalPossibleEdges} = \frac{Cardinality}{Size} = \frac{m}{n(n-1)/2} \quad (1)$$

$$DirectedNetworkDensity = \frac{TotalEdges}{TotalPossibleEdges} = \frac{Cardinality}{Size} = \frac{m}{n(n-1)} \quad (2)$$

1.2 Assortativity Coefficient

Assortativity measures the tendency of a peak during a network being connected by other peaks and vertex-specific features. Normally, assortativity coefficients are described for unweighted and undirected networks [10]. Simply we can define the assortativity coefficient as measuring the correlation between every pair of nodes that are connected with one another [11].

1.3 Average Clustering Coefficient

Many networks show a tendency between neighbors deviating from uncorrelated random networks in which triangles are rare. We can call this tendency clustering and this shows us the clustering of edges into firmly connected neighborhoods [12].

$$C = \frac{1}{n} \sum_{v \in G} c_v,$$

2. LSTM Model Architecture

Model architecture is important to complete the task. Choosing architecture accordingly increases the success rate. The main reason for choosing stacking LSTM is to allow for greater model complexity. Since our object has much more complexity, we get much more accuracy in our forecasting model with Stacked-LSTM.

Our LSTM model is a four stacked model. The architecture is complex enough to obtain accurate conclusions. In our experimental setup, we have multiple models with the same layers but with different numbers of parameters.

Model: "sequential"

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 1, 30)	4680
dropout (Dropout)	(None, 1, 30)	0
lstm_1 (LSTM)	(None, 1, 40)	11360
dropout_1 (Dropout)	(None, 1, 40)	0
lstm_2 (LSTM)	(None, 1, 50)	18200
dropout_2 (Dropout)	(None, 1, 50)	0
lstm_3 (LSTM)	(None, 60)	26640
dropout_3 (Dropout)	(None, 60)	0
dense (Dense)	(None, 1)	61

Total params: 60,941
Trainable params: 60,941
Non-trainable params: 0

Model: "sequential"

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 1, 30)	4320
dropout (Dropout)	(None, 1, 30)	0
lstm_1 (LSTM)	(None, 1, 40)	11360
dropout_1 (Dropout)	(None, 1, 40)	0
lstm_2 (LSTM)	(None, 1, 50)	18200
dropout_2 (Dropout)	(None, 1, 50)	0
lstm_3 (LSTM)	(None, 60)	26640
dropout_3 (Dropout)	(None, 60)	0
dense (Dense)	(None, 1)	61

Total params: 60,581
Trainable params: 60,581
Non-trainable params: 0

Fig 13: LSTM Configuration of classic splitting

The models that use sliding window data also have the same layer architecture. However, it has more parameters.

Model: "sequential_1"

Layer (type)	Output Shape	Param #
lstm_4 (LSTM)	(None, 60, 50)	12000
dropout_4 (Dropout)	(None, 60, 50)	0
lstm_5 (LSTM)	(None, 60, 50)	20200
dropout_5 (Dropout)	(None, 60, 50)	0
lstm_6 (LSTM)	(None, 60, 50)	20200
dropout_6 (Dropout)	(None, 60, 50)	0
lstm_7 (LSTM)	(None, 50)	20200
dropout_7 (Dropout)	(None, 50)	0
dense_1 (Dense)	(None, 1)	51

Total params: 72,651
Trainable params: 72,651
Non-trainable params: 0

Model: "sequential_2"

Layer (type)	Output Shape	Param #
lstm_8 (LSTM)	(None, 60, 50)	11200
dropout_8 (Dropout)	(None, 60, 50)	0
lstm_9 (LSTM)	(None, 60, 50)	20200
dropout_9 (Dropout)	(None, 60, 50)	0
lstm_10 (LSTM)	(None, 60, 50)	20200
dropout_10 (Dropout)	(None, 60, 50)	0
lstm_11 (LSTM)	(None, 50)	20200
dropout_11 (Dropout)	(None, 50)	0
dense_2 (Dense)	(None, 1)	51

Total params: 71,851
Trainable params: 71,851
Non-trainable params: 0

Fig 14: LSTM Configuration of sliding window splitting

4. Analysis of LSTM Model

In the LSTM model we built, commonly used error metrics to measure the accuracy of the price prediction were used. These are RMSE, MAE, and MAPE.

RMSE: For analyzing the accuracy of the model we use the Root Mean Square Error(RMSE).
The difference or the difference between the real and the predicted price value is normalized by using the RMSE value. RMSE is the square root of the mean average of the square of all of the errors. RMSE produces larger error results compared to Mean Absolute Error.

$$\mathrm{RMSE} = \sqrt{\frac{1}{\mathrm{N}}\sum\left(\hat{\mathrm{Y}}_{\mathrm{i}} - \mathrm{Y}_{\mathrm{i}}\right)^2}$$

MAE: The mean absolute error is a measure of the difference between two continuous variables. MAE is also the average horizontal distance between each data point and the best-fit line. Since the MAE value is easily interpretable, it is frequently used in regression and time series problems. The MAE is a linear score that measures the mean size of errors in a set of predictions without considering their direction, with all individual errors weighted equally on the mean. The MAE value can range from 0 to ∞. Negatively oriented scores, that is, predictors with lower values, perform better.

$$\mathrm{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$

MAPE: In regression and time series models, the mean absolute percent error is frequently used to measure the accuracy of predictions. If there are zeros among the actual values, MAPE cannot be calculated because they will be divided by zero. The percentage error cannot exceed 100% for very low forecast values, but there is no upper limit to the percent error for very high forecast values. When MAPE is used to compare the accuracy of estimators, it is biased in that it systematically chooses a method with very low estimates.

$$M = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right|$$

Our error rate tables are given below. We have seen that the sliding window increases the error rates of the model with graph metrics. On the other hand non-random division of

	Without graph metrics		With graph metrics	
	Train	Test	Train	Test
RMSE	0.50	0.97	0.23	0.56
MAE	0.34	0.79	0.17	0.48
MAPE	0.14	0.07	0.10	0.05

Fig 15: Error Rates Comparison

The error rates we have shared above are split classically. Hence the sliding window has higher error rates, we do not continue on that aspect.

Consolidated data frame is checked for correlation and the visualization is done. As you can see below the correlation matrix is helpful for us to see the relationship between market values and transactions.

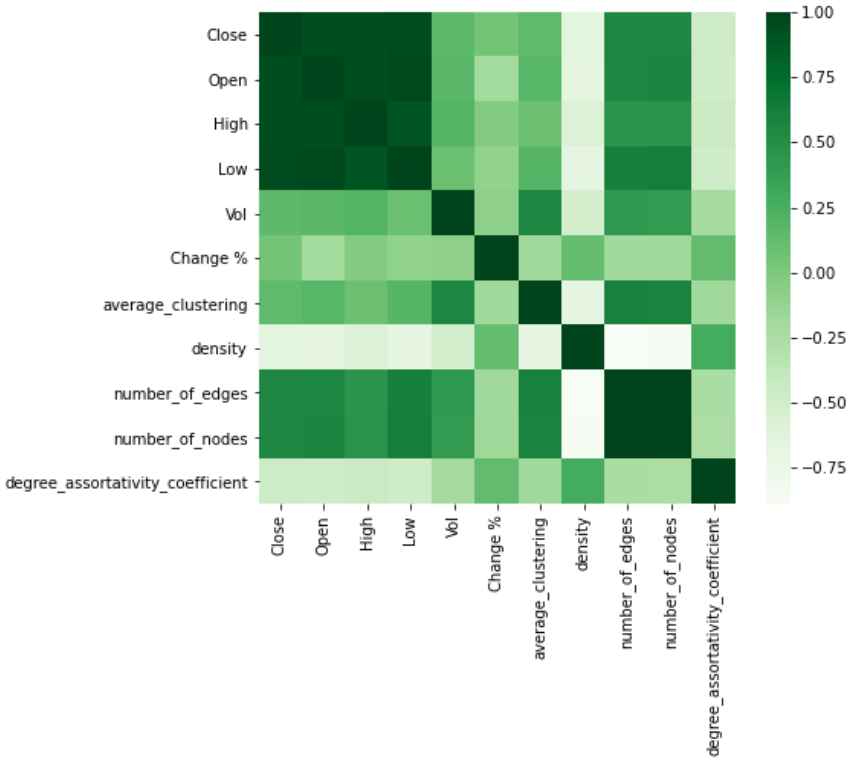

Fig 16: Correlation

5. Effectiveness of Embeddings

The main purpose of our project is to conduct a study on the usability of the transaction dataset to improve efficiency in the models used in price prediction. In the first stage, we showed that using graph metrics in our LSTM model increases the accuracy.

Embeddings can also be used as a method to benefit from transaction graphs. As already mentioned before, graph embedding is a fixed length vector representation that carries information about nodes, edges and their features. By this means, graphs are transformed into a format that can be computed easily and integrated into the models.

In the next phase of our study, we used some supervised machine learning algorithms such as Support Vector Machine(SVM), K-Nearest Neighbors(KNN), Random Forest(RF), Decision Tree(DT) to understand the effectiveness of embeddings by bitcoin trend forecasting. A training set is used in supervised learning to train models to produce the desired output. This training dataset contains features and correct outputs, so the algorithm learns over time. The features used in our models are price and volume related features and graph embeddings.

In this study we use seven features to predict a bitcoin trend – close price, open price, high price, volume, percentage of change and graph embeddings. Through these models, we measured the trend prediction accuracy by first using price related data and then adding transaction data(graph embeddings). Transaction data slightly increased the accuracy and monthly data produced a better result.

5.1 Hyperparameter Optimization

When building a machine learning model, there are design options for how to build the model architecture. To discover the optimal state of a model, we need to try a range of possibilities.

Hyper parameters are parameters that can be adjusted and fine-tuned in order to improve the performance of the machine learning model and the process of searching for the ideal model architecture is called hyperparameter tuning. These are not model parameters, hence they can not be learned directly from data. Train-test split ratio, learning rate, batch size, choice of activation function can be given as examples of the most known hyperparameters.

We made some parameter adjustments in each model to get better accuracy than the learning models we used. We did not handle this process manually because it is a very time-consuming and resource-intensive process. Instead, there are many functions that automate parameters for machine learning models, and we used one of them, grid search. The GridSearchCV function that comes in Scikit-learn is used for hyperparameter tuning of our models.

Grid Search assesses the performance of each combination of hyperparameters and their values, then chooses the optimum value for the hyperparameters.

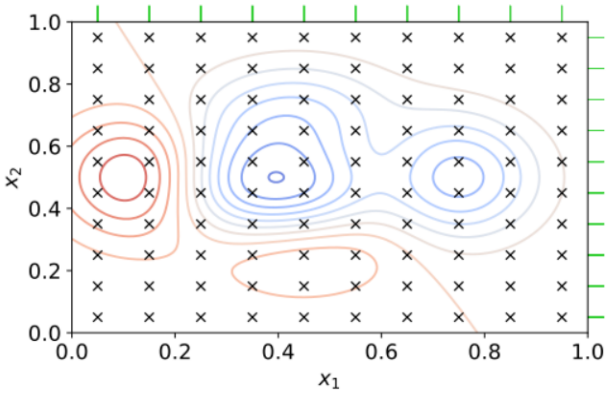

Fig 17: Grid Search Visualization on two parameters (Image from [28])

5.2 Principal Component Analysis (PCA)

Overfit problem in machine learning is more common with high dimensional datasets. This problem reduces the generalization capacity of the algorithm in the training set. To overcome this problem, dimensional reduction methods should be applied by reducing the number of features. Principal Component

Analysis (PCA) is one of these methods. While PCA tries to reduce the features in the dataset, it aims to preserve as much information as possible about the dataset.

While PCA is advantageous in some aspects, it also has some disadvantages. Finding correlated features increases power and time consumption, especially in datasets with a large number of features. PCA removes correlated features. Since it removes unnecessary features from the dataset, the overfit problem is reduced also. On the other hand, if there is not the right number of features to train, information loss may increase and this is a disadvantage.

When comparing the machine learning models we used in our study, we also considered different PCA values, and in some cases increasing PCA and in some cases decreasing PCA increased accuracy. The table below shows how the different PCA values we use with the weekly dataset in the KNN algorithm affect the results. Better results can be achieved as can be seen with a correct PCA adjustment.

PCA	F1	Precision	Accuracy
5	0.29131	0.239057	0.394737
10	0.450047	0.478225	0.473684
20	0.244617	0.202469	0.342105

Fig 18: PCA Tuning

5.3 Measurements

In order to decide which model should be the most accurate in machine learning, many measurements are needed to evaluate well. Model selection should not be made only on accuracy in project outputs. Three different metrics were used to compare the five different machine learning models used in this study and monitor the overall performance of the embeddings on models - F1 Score, Precision, Accuracy.

One technique to determine how often a machine learning classification system properly classifies a data point is to look at its accuracy. The number of correct predictions divided by the total number of samples is used to measure accuracy.It is calculated as the number of true positives and true negatives divided by the number of true positives, true negatives, false positives, and false negatives. A data point that the algorithm properly classifies as true or false is referred to as a true positive or true negative. A data point that was wrongly categorized by the algorithm is referred to as a false positive or false negative.

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

where TP = True Positives, TN = True Negatives,
FP = False Positives, and FN = False Negatives.

Precision shows how many of the values predicted by the algorithm as positive are actually positive. The precision value is especially important when the cost of False Positive estimation is high. Precision, therefore, calculates the accuracy for the minority class [25].

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

The number of positive class predictions made out of all positive examples in the dataset is measured by recall. Recall is a metric that indicates how well our model can detect relevant data.

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

The F1-score, which is defined as the harmonic mean of the model's accuracy and recall, is a technique of combining the model's precision and recall. It is possible to change the F-score to prioritize precision above recall, or vice versa. It is primarily used to compare the performance of precision and recall.

$$F1\text{-}score = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

6. Experimental Models

In our experiments we aimed to catch the trends that upwards, downwards and neutral. We used a threshold to decide is the change is a trend or not. We labeled the movements above the threshold as up or down and the ones under threshold are labeled as neutral.

6.1 Support Vector Machine (SVM)

Support vector machines are among the most basic and, perhaps, elegant algorithms for classification and regression[23]. It works well in high-dimensional spaces and when the number of dimensions exceeds the number of samples.

It is built on the concept of structural risk reduction.. SVMs have a superior generalization performance and are less prone to overfitting, making them ideal for stock market prediction applications [24]. For this reason, SVM became one of the algorithms we used in our project.

SVM categorizes data points by mapping them to a high-dimensional feature space, even if the data isn't linearly separable. After finding a separator between the categories, the data are converted so that the separator may be drawn as a hyperplane. New data features can be utilized to forecast which category a new record should belong to.

The support vector machine algorithm's goal is to identify a hyperplane in an N-dimensional space (N — the number of features) that distinguishes between data points[29]. There are several hyperplanes from which to choose to split the two groups of data points. The algorithm tries to find a plane with the maximum distance between the data points of both classes, so that future data points can be classified with more confidence.

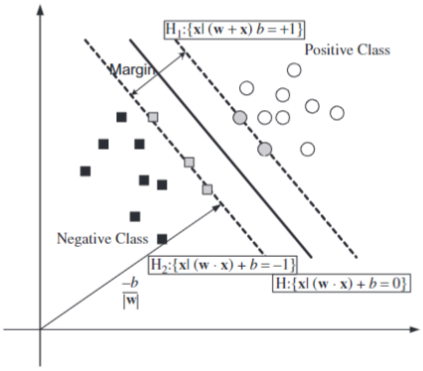

Fig 19: Classification of two classes using SVM (Image from [29])

As the number of features increases, the size of the hyperplane increases. For example, if the number of features is two, the hyperplane is just a line, if it is three, it is a two-dimensional plane.

SVM has been used to construct the forecasting model. The approach was tested to solve the prediction task of directional changes in Bitcoin price. The measurement values obtained as a result of training made with the SVM model only with price, volume-dependent features and then embedding features are in the table below.

Trend Interval	F1	Precision	Accuracy
Weekly	-0.07061	-0.10185	-0.09091
Monthly	0.039221	0.331871	0.00000

Fig 20: Increase in results of SVM.

The data in the table above shows the increase in measurement metrics when graph embeddings are added as parameters. The results of the SVM algorithm were worse compared to other algorithms, with a better performance in the monthly interval.

6.2 Decision Tree

A tree has numerous real-world analogues, and it turns out that it has affected a broad field of machine learning, including classification and regression [26]. In big datasets, decision trees can be used to find characteristics and extract patterns that are useful for predictive modeling. The interpretability of the created model is one benefit of decision tree modeling over other pattern recognition approaches [27].

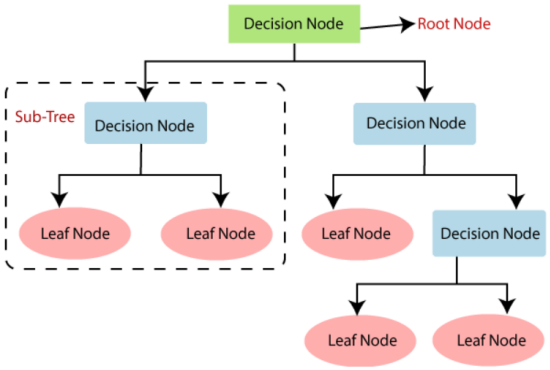

Fig 21: Decision Tree Classifier

The decision tree algorithm is useful for representing such mappings. Internal nodes have dataset properties, branches hold decision rules, and each leaf node has the outcome. The algorithm uses two node structures, a decision, and a leaf node. Decision nodes are used to make any decision and leaf nodes are the output of those decisions.

The method is non-parametric, which means it can handle huge, complex datasets without imposing a complex parametric framework. Therefore, this model is one the suitable models for this study, since the bitcoin transaction data is quite large.

Trend Interval	F1	Precision	Accuracy
Weekly	0.02117	0.01073	0.026316
Monthly	-0.01071	0.01123	0.000000

Fig 22: Increase in results of DT.

With the Decision Tree, we can see the increase in measurement metrics with embeddings in the trend forecast model from the table above. It achieved a very slight increase in accuracy weekly but did not make a difference monthly. Compared to other model results, DT does not perform well.

6.3 KNN

To begin, calculated parameter K, which is the number of closest neighbors. Determined the distance between the query instance and each of the training samples. Based on the K-th minimum distance, sort the distance and find the closest neighbors. Used the simple category of the nearest neighbors when we've gathered all of the data.

KNeighborsClassifier implements learning based on the nearest neighbors of each query point, which is an integer value specified by the user.

The optimal choice of the value is highly data-dependent: in general, a bigger k reduces the impacts of noise while blurring the classification boundaries.

Important parameters were used in the project:

leaf_size = 1:	Leaf size passed to BallTree or KDTree. This can affect the speed of the construction and query, as well as the memory required to store the tree. The optimal value depends on the nature of the problem.
n_neighbors =16:	Number of neighbors to use by default for k_neighbors queries.
p=1:	Power parameter for the Minkowski metric.

In the table below, we can see the performance increase data of graph embeddings in the KNN algorithm. We achieved the best result with the KNN algorithm and made a step forward in the usability of transaction data in bitcoin trend prediction.

Trend Interval	F1	Precision	Accuracy
Weekly	0.072327	0.207371	0.052632
Monthly	0.145471	0.158323	0.227273

Fig 23: Increase in results of KNN.

6.4 Random Forest

RandomTreesEmbedding performs an unsupervised data transformation. RandomTreesEmbedding encodes data by the indices of the leaves a data point ends up in, using a forest of fully random trees. The index is encoded in a one-of-K fashion, resulting in a sparse binary coding with high dimensionality. This code can be computed quickly and then utilized as a starting point for future learning exercises. The number of trees and the maximum depth per tree can be used to affect the code's length and sparseness.

A random forest is a meta-estimator that employs averaging to increase predicted accuracy and prevent overfitting by fitting a variety of decision tree classifiers on different subsets of the data.

Important parameters were used in the project:

max_depth:100:	The maximum depth of the tree.
max_features: 3:	The number of features to consider when looking for the best split.
min_samples_leaf: 3:	The bare minimum of samples must be present at a leaf node. A split point will be evaluated if it leaves at least min samples of leaf training data for each of the left and right branches, regardless of depth.

min_samples_split: 10: The minimum number of samples required to split an internal node
n_estimators: 200: The number of trees in the forest.

In the table below, we can see the performance increase data of graph embeddings in the RF algorithm. We have achieved an increase in accuracy of nearly 14% with the RF algorithm. Compared to SVM and DT algorithms, they performed very well. Compared to KNN, it is a very good result even though it has less performance.

Trend Interval	F1	Precision	Accuracy
Weekly	-0.07033	-0.08389	0.078947
Monthly	0.08106	0.12941	0.136364

Fig 24: Increase in results of RF.

Knowledge and Skill Set

While continuing this research, we benefited from the online resources as well as the courses we took.These courses helped us to design different areas of expertise. The courses stated have the most effect on our project.

1. CS 201 (Computer Programming) : This course helped us to analyze graphs.
2. CS 320 (Software Engineering) : This course helped us to have an agile process and write the project report in the right format during our work.
3. CS 333 (Algorithm Analysis) : This course helped us to design paradigms, Graph algorithms
4. CS 440 (Machine Learning in Finance) : This course helped us to build machine learning models on finance time-series data and evaluate our models.
5. CS 452 (Introduction to data science) : This course taught us how to analyze data that we collected from transactions and the crypto market.

Results and Discussion

LSTM Results

From the result obtained in this research, This study shows the next day's bitcoin price forecasting method using different methods of time-series analyses such as the LSTM model. We split the Bitcoin data into training and testing. When we evaluate the results we have evidence of graph metrics are more accurate than without using graph metrics.

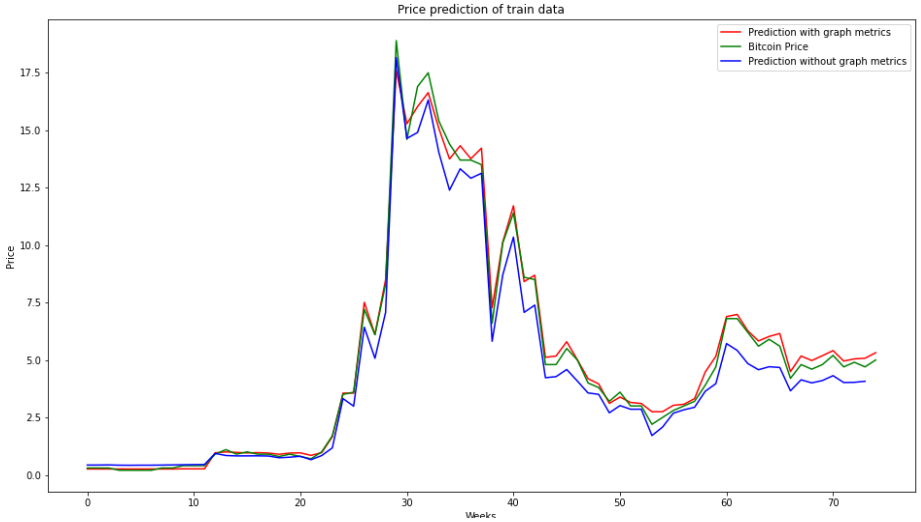

Fig 25: Training predictions.

In figure 25 above, you can see that peak times of the price out model can be closer to the real values. As it comes to the test, you can see below in figure 26 that also in test data graph metrics help the model to predict the price.

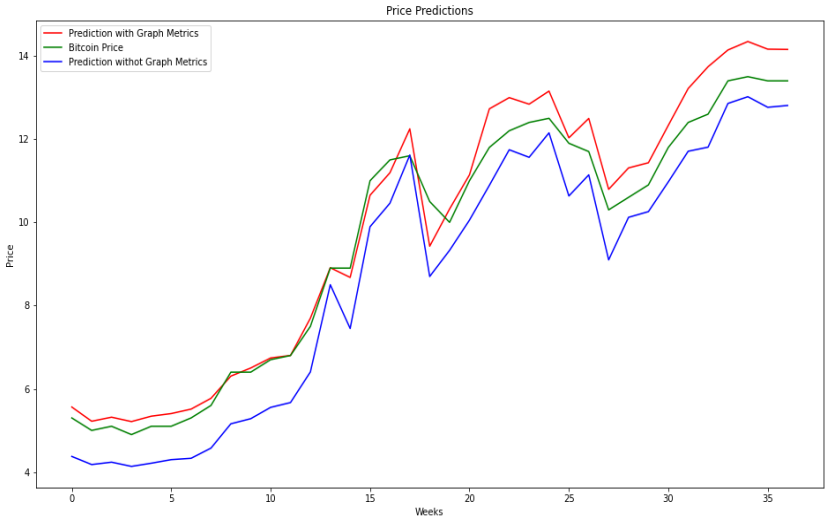

Fig 26: Test Predictions

Effectiveness of Graph Embeddings

A series of experiments have been conducted to examine the effectiveness of graph embeddings on proposed models. The embedding data we obtained from the transaction dataset increased the accuracy of the models we used within the scope of the project, as can be seen in the table below.

Algorithms	WITHOUT TRANSACTION DATA			WITH TRANSACTION DATA			IMPROVEMENT		
	F1	Precision	Accuracy	F1	Precision	Accuracy	F1	Precision	Accuracy
SVM	0.44335	0.53704	0.68182	0.37274	0.43519	0.59091	-0.07061	-0.10185	-0.09091
KNN	0.29131	0.23906	0.39474	0.36364	0.44643	0.44737	0.07233	0.20737	0.05263
DT	0.44722	0.47917	0.47368	0.46839	0.4899	0.5	0.02117	0.01073	0.02632
RF	0.26787	0.22424	0.34211	0.19753	0.14035	0.42105	-0.07033	-0.08389	0.07895

Weekly Results

Algorithms	WITHOUT TRANSACTION DATA			WITH TRANSACTION DATA			IMPROVEMENT		
	F1	Precision	Accuracy	F1	Precision	Accuracy	F1	Precision	Accuracy
SVM	0.19753	0.14035	0.42105	0.23675	0.47222	0.42105	0.03922	0.33187	0.00000
KNN	0.30702	0.30199	0.45455	0.45249	0.46032	0.68182	0.14547	0.15832	0.22727
DT	0.32353	0.34432	0.45455	0.31282	0.35556	0.45455	-0.01071	0.01123	0.00000
RF	0.33482	0.33333	0.5	0.41587	0.46275	0.63636	0.08106	0.12941	0.13636

Monthly Results

SVM and DT generally did not provide a noticeable performance increase in the accuracy of the models. We achieved a 22% increase in accuracy with KNN and an approximately 14% increase in accuracy with RF. Models perform better on a monthly basis. There may be two reasons for this. Firstly, while monthly data covers transaction data between 2010-2015, weekly data covers between 03-2011 and 09-2013. The second reason is that the data were taken from two different sources. Weekly data is collected through the BigQuery service provided by Google Cloud. In the transaction data obtained from here, CoinJoin data, which is the transaction aggregator method commonly used in the bitcoin network, is ignored. For this reason, the loss of information in the weekly transaction network may have prevented an effective result.

Discussion

The decentralized and transparent structure of Bitcoin, which is different from traditional currencies, and the fact that each transaction on the network is open to the public, provides an advantage in addition to the models that make a lot of work on fiat currency and stock predictions.

The list of weekly and monthly transactions was downloaded and compiled from the sources mentioned in the article, and the structure and development of the transaction network were investigated. We have performed detailed analysis of usability of Bitcoin transactions in machine learning projects.

Because the graphs' sizes were too large to process, they were sampled so that they could be applied to the models. Graph samples led to an increase in results even though fairly small portions of the actual data were

used. This brings an idea that better results can be obtained by using a more accurate sampling model or using all transaction data with sufficient CPU power.

On the other hand, the models used in our project are the models used in many stock market forecasting studies before. In this paper, we have shown that these models, which are widely used in the literature, can also be used in the crypto market.

V. Related Work

A. Specifically LSTM Approaches

Salim Lahiri et al.[13] used Deep Neural Network to predict the bitcoin's, ripple's, and Digital Cash's price prediction, They presented a comparative analysis of the predictability of deep learning neural networks (Long-short term memory) versus generalized regression neural networks (GRNN).When these results were observed, Long Short-term Memory (LSTM) neural networks results were found to be more consistent than Generalized Regression Neural Networks (GRNN) results. Trying and testing the value and consistency of Long-short term memory (LSTM) has also been studied in other articles working on this. Another example of work in this area is the implementation of a Bayesian optimized recurrent neural network (RNN) and a Long Short Term Memory (LSTM) network. In the case, specified study Bitcoin prediction with implementing the LSTM is more capable of recognizing longer-term dependencies [14].

B. Time Series Forecasting

These days, machine learning (ML) and neural network algorithms have gained much appreciation for implementing time series for Bitcoin price prediction. According to Roy's paper [15], They worked on time series data set and then forecast the future Bitcoin closing price using the previous data. They use Autocorrelation for applying time series. Autocorrelation is the correlation of a signal between its values at different times. In other words, it is the expression of similarity between observed values in a given time series. Autocorrelation helps to determine the optimal solution for a particular dataset and the forecasting method fits more precisely. Also, there is a research paper that worked on time series analysis for analyzing the history of Bitcoin transactions [16]. The purpose of the project is to detect changing points, namely anomaly detection, against a given (Bitcoin) address's transaction history.

C. Bitcoin Transaction

According to some related work on the bitcoin transaction, researchers use transaction graphs. They discovered that the network includes many small transactions, but also a subset of transactions that move large amounts of money. The analysis then focuses on large transactions to identify how quantities are piling up and spreading out [17].

VI. Conclusion and Future Work

A.Conclusion

In this study on Bitcoin transaction analysis, we showed the results that emerged with the data we collected, the tools we used and the machine learning models we build. As a result, transaction data can be used effectively to determine the direction in the crypto market. Transaction data, which we can access through the transparent structure of the crypto currencies, provides an advantage compared to the studies used for stock market forecasting.

B.Future Work

While working with the Transaction dataset, our biggest problem was the lack of processing power and capacity when converting to graph format and outputting it. To solve this problem, we will divide our dataset into parts under certain categories. To take advantage of edge features, we will create directed graphs and work on node embeddings by building a GNN model. Bitcoin and especially the crypto market gained great popularity after 2018, so volatility has increased and it has become more affected by external reasons. Since our study covers data between 2010 and 2015, we did not test the current effectiveness of this study. In the next phase of the project, we will test the current effectiveness of current models.

Acknowledgments

We would like to thank Dr. Emre Sefer, Ozyegin University for helping us during this research.

References

[1]Roondiwala, Murtaza, Harshal Patel, and Shraddha Varma. "Predicting stock prices using LSTM." *International Journal of Science and Research (IJSR)* 6.4 (2017): 1754-1756.

[2] Fama, E. F. (1970). Efficient capital markets: A review of theory and empirical work. The Journal of Finance, 25 (2), 383–417. doi: 10.2307/2325486 .

[3] Fischer, Thomas, and Christopher Krauss. "Deep learning with long short-term memory networks for financial market predictions." *European Journal of Operational Research* 270.2 (2018): 654-669.

[4] Crosby, Michael, et al. "Blockchain technology: Beyond bitcoin." *Applied Innovation* 2.6-10 (2016): 71.

[5] Grohe, Martin. "word2vec, node2vec, graph2vec, x2vec: Towards a theory of vector embeddings of structured data." Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems. 2020.

[6] Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., & Schmidhuber, J. (2016). LSTM: A search space odyssey. *IEEE transactions on neural networks and learning systems*, *28*(10), 2222-2232.

[7] Yousaf, Imran, Shoaib Ali, and Wing-Keung Wong. "An empirical analysis of the volatility spillover effect between world-leading and the asian stock markets: Implications for Portfolio management." *Journal of Risk and Financial Management* 13.10 (2020): 226.

[8] Data-To-Viz. (May 15, 2020). *Network Diagram* https://www.data-to-viz.com/graph/network.html

[9] Grover, A., & Leskovec, J. (2016). Node2vec: Scalable feature learning for networks. Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 13-17-August-2016, 855–864. https://doi.org/10.1145/2939672.2939754

[10] K. Toyoda, T. Ohtsuki and P. T. Mathiopoulos, "Time Series Analysis for Bitcoin Transactions: The Case of Pirate@40's HYIP Scheme," 2018 IEEE International Conference on Data Mining Workshops (ICDMW), 2018, pp. 151-155, doi: 10.1109/ICDMW.2018.00028.

[11] Grover, A., & Leskovec, J. (2016). Node2vec: Scalable feature learning for networks. Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 13-17-August-2016, 855–864. https://doi.org/10.1145/2939672.2939754

[12] Hu, Pili, and Wing Cheong Lau. "A survey and taxonomy of graph sampling." *arXiv preprint arXiv:1308.5865* (2013).

[13] Nicosia, V., Tang, J., Mascolo, C., Musolesi, M., Russo, G., & Latora, V. (2013). Graph Metrics for Temporal Networks. https://doi.org/10.1007/978-3-642-36461-7_2

[14] Yuan, Y., Yan, J., & Zhang, P. (2021). Assortativity measures for weighted and directed networks. http://arxiv.org/abs/2101.05389

[15] Learn About Assortativity Coefficient in Python With Data From UK Faculty Dataset (2008) Student Guide. (2019).

[16] Saramäki, J., Kivelä, M., Onnela, J.-P., Kaski, K., & Kertész, J. (2006). Generalizations of the clustering coefficient to weighted complex networks. http://www.arxiv.org

[17] S. Lahmiri and S. Bekiros, "Cryptocurrency forecasting with deep learning chaotic neural networks", Chaos Solitons & Fractals, vol. 118, pp. 35-40, 2019.

[18] McNally, S., Roche, J., & Caton, S. (2018). Predicting the Price of Bitcoin Using Machine Learning. Proceedings - 26th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, PDP 2018, 339–343. https://doi.org/10.1109/PDP2018.2018.00060

[19] S. Roy, S. Nanjiba and A. Chakrabarty, "Bitcoin Price Forecasting Using Time Series Analysis," 2018 21st International Conference of Computer and Information Technology (ICCIT), 2018, pp. 1-5, doi: 10.1109/ICCITECHN.2018.8631923.

[20] D. Di Francesco Maesa, A. Marino and L. Ricci, "Uncovering the Bitcoin Blockchain: An Analysis of the Full Users Graph," 2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA), 2016, pp. 537-546, doi: 10.1109/DSAA.2016.52.

[21] Grover, A., & Leskovec, J. (2016). Node2vec: Scalable feature learning for networks. Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 13-17-August-2016, 855–864. https://doi.org/10.1145/2939672.2939754

[22] https://arxiv.org/pdf/1607.00653.pdf

[23] https://towardsdatascience.com/node2vec-explained-db86a319e9ab

[24] Bratanic, Tomaz. “Complete Guide to Understanding NODE2VEC Algorithm.” Medium, Towards Data Science, 4 Sept. 2021, https://towardsdatascience.com/complete-guide-to-understanding-node2vec-algorit hm-4e9a35e5d147.

[25] Y. Lin, H. Guo and J. Hu, "An SVM-based approach for stock market trend prediction," The 2013 International Joint Conference on Neural Networks (IJCNN), 2013, pp. 1-7, doi: 10.1109/IJCNN.2013.6706743.

[26] Gujarati, D. N. (2003). Basic Econometrics. 4th. New York: McGrawHill.

[27] Davis, Jesse, and Mark Goadrich. "The relationship between Precision-Recall and ROC curves." *Proceedings of the 23rd international conference on Machine learning*. 2006.

[28] Myles, Anthony J., et al. "An introduction to decision tree modeling." *Journal of Chemometrics: A Journal of the Chemometrics Society* 18.6 (2004): 275-285.

[29] Quinlan, J. Ross. "Learning decision tree classifiers." *ACM Computing Surveys (CSUR)* 28.1 (1996): 71-72.

[30] https://www.analyticsvidhya.com/blog/2021/06/tune-hyperparameters-with-gridsearchcv/

[31] Noble, William S. "What is a support vector machine?." *Nature biotechnology* 24.12 (2006): 1565-1567.