Installation 04

Installation04

Cryptocurrency price movement prediction project using machine learning.

By Aws Abu Obaid

Awsahmad157@gmail.com

TABLE OF CONTENTS

I. Abstract 5

II. Introduction 5

2.1 What is cryptocurrency? 5

2.2 Difficulties predicting cryptocurrencies 5

2.3 Cryptocurrency future importance 5

III. Professional Practice Constraints 6

IV. System Architecture and Design 7

A. Preliminary Design 7

B. Web Application Design 8

V. Software Implementation 12

1. Data Collectors 13

2. Models Training 16

2.1 Feature Selection 17

2.2 Data Collection 18

2.3 Cleaning Datasets 19

2.4 Feature Extraction 21

2.5 Building and Training Models 23

3. Web Application 26

3.1 Frontend (Client Side) 26

3.2 Backend (Server Side) 27

3.2.1 CryptoDao 29

3.2.2 Predictors 30

3.2.3 Database 31

3.2.4 Server Routes 33

3.2.5 Running the Application 34

VI. Results 35

VII. Conclusion & Future Work 35

VIII. Bibliography 36

TABLE OF FIGURES

Figure 1 System Design....7
Figure 2 Web Application Design....8
Figure 3 Prediction Cards....9
Figure 4 Coins prices on a graph.... 10
Figure 5 Prediction History.... 10
Figure 6 models info card.... 11
Figure 7 Project Architecture.... 12
Figure 8 Data Collectors Functions.... 13
Figure 9 Clark Moody website.... 13
Figure 10 Yahoo Finance Data.... 14
Figure 11 Satoshi.info TPS.... 14
Figure 12 Bitcoin Number of tweets.... 15
Figure 13 bitstamp.net Data.... 15
Figure 14 Models Training Processes.... 16
Figure 15 Features Categories.... 17
Figure 16 Websites with available datasets.... 18
Figure 17 Data Scrapers.... 19
Figure 18 Tweets Count Data.... 19
Figure 19: Constant Interpolation.... 20
Figure 20: Linear Interpolation.... 20
Figure 21: Matching timeframes.... 21
Figure 22 Extracting a sample.... 22
Figure 23: NN Nodes Example.... 24
Figure 24 Dumping model and scaler.... 24
Figure 25 Flask Logo.... 27
Figure 26 Flask installation.... 28
Figure 27 Minimal Flask App.... 28
Figure 28 CryptoDao functions.... 29
Figure 29 getPrices function content.... 29
Figure 30 Predicting Process.... 30
Figure 31 CryptoDB schema.... 31
Figure 32 Class Tech.... 32
Figure 33 PriceVolume Table.... 32
Figure 34 Routes Functions.... 33
Figure 35 Run.py Functions.... 34

LIST OF TABLES

Table 1 NN on Bitcoin 25
Table 2 NN on Litecoin 25
Table 3 NN on Ripple 25
Table 4 schedule .... **Error! Bookmark not defined.**

I. Abstract

The goal of this project is to build machine and deep learning models to analyse and predict the movement of cryptocurrency prices. Focusing on: Bitcoin (BTC), Litecoin (LTC) and Ripple (XRP) cryptocurrencies. We will be testing many different algorithms and features to find the best preforming ones, while reporting our findings on feature importance and model accuracy. This project's implementation can be thought of as a proof of concept, to show how different algorithms can be used to make predictions. The constructed models are deployed onto a web application software running over a server that will be fetching data through different APIs to use them to make the desired prediction in real-time and display the outcome on the webpage. This contribution to the literature on machine learning cryptocurrency prediction, will hopefully add to the collective knowledge and help show the predictability of certain crypto markets and whether it can be used to make consistent profits and or give insights for investors.

II. Introduction

2.1 What is cryptocurrency?

Cryptocurrency is known as a digital coin existing virtually on the internet, decentralized from authority and central banks which is designed for the purposes of exchanging and trading. Their importance has been increasing lately, as they make fast and secure digital transactions, companies are allowing them to be used to buy their products. Furthermore, a lot of business companies are creating their own cryptocurrencies. Cryptocurrency prices are volatile and change consistently due to many factors which has made them a trading asset worth billions of dollars.

2.2 Difficulties predicting cryptocurrencies

Due to the numerous factors regulating cryptocurrency pricing and market changes, it is nearly impossible to precisely predict future price movements. One of the fundamental issues with many bitcoin price predictions is that they lack any analytical support to back up their assertions. As many investors tend to trade in cryptocurrencies, they seek to make a profit in the market. For example, someone owning a cryptocurrency now priced at $0.01 can easily be persuaded that the token will rise to $10,000 simply because they want it to be true. The problem is that many predictions are made without data or research to back them up. Unlike the human ability to analyse, machine learning can take massive amount of data to make predictions to a good degree of accuracy. Therefore, using machine learning can overcome this difficulty and eases the investing in the market.

2.3 Cryptocurrency future importance

In 2008, an individual or group, named Satoshi Nakamoto constructed the very first blockchain called Bitcoin. Its value has skyrocketed to absurd levels: the internet was filled with articles like "If I had brought $100 of bitcoin back in 2010, I'd have almost US$100 million now" or "Bitcoin's First Billionaires" floating around the Internet. Bitcoin is being accepted as a form of payment by a growing number of businesses and online vendors. Ethereum is comparable to Bitcoin in that it may be used for more than just financial purposes, such as mining, and can also be used to provide services on its own blockchain. Ethereum has built-in software programming languages that may be used to create smart contracts, which can be used for a variety of things, including the transfer, and mining the Ether -company's own tradeable digital asset-. However, Cryptocurrency is significant, and it will not go away or be restricted to 100 years as some have speculated: transactions are rapid, digital, safe, and global, allowing for the maintenance of records without the risk of data piracy. Fraud is reduced to a minimum. Also, many prominent banks are increasingly investing in current crypto customers (JPMorgan with Zcash) or establishing their own cryptocurrency (such as Bank of America).

According to analysts, the global cryptocurrency industry would more than triple in value by 2030, reaching roughly $5 billion. Investors, businesses, and brands can't ignore the growing wave of crypto for long time, whether they want it or not. Experts believe that cryptocurrencies are the way of the future. They'll be able to express our economy far faster than letters. As a result, a growing number of e-commerce entrepreneurs are adopting virtual money into their businesses.

III. Professional Practice Constraints

1) **Efficient Market Hypothesis**: bitcoin is hard to predict because it mostly follows the efficient market hypothesis, it states that all available information reflects on the current market price and can’t be used to predict the future price, as that will be decided by future revealed information, a direct implication of that it is impossible to beat the market consistently. Thankfully Bitcoin and other cryptocurrencies are not fully efficient, some more than others. Which may give us a chance to make note worthy predictions.
2) **Randomness**: The fluctuations of cryptocurrency market make them extremely hard to predict.
3) **Deep Learning models**: Some models are difficult to build specially the Neural Network ones. For example, RNNs are complex and hard to train because they are not feedforward neural networks. Signals in feedforward neural networks only go one way. The signal travels from an input layer to numerous hidden layers and then to the system's output layer. On the other hand, Recurrent neural networks and other forms of neural networks feature more complex signal motions. RNNs can send and receive signals and may comprise numerous "loops" in which numbers or values are sent back into the network. This is linked to the memory-related element of recurrent neural networks, according to experts. Lastly training Neural Networks in general, requires a lot of computational powers.
4) **Data collection:** Since not all data were available on internet, collecting data was bothersome, as we had to use various websites in which each website required different methods to interface and obtain the data.
    - Blockchain.com data had different time frames for different time periods.
    - Stock data was daily.
    - Minute-price data had large gaps.

IV. System Architecture and Design

A. Preliminary Design

Figure 1 is an overview of the preliminary system design concept. First, we train the Data we collected, then we deploy our models on the webapp to make their predictions and then the outcome is displayed to the clients who visit the page.

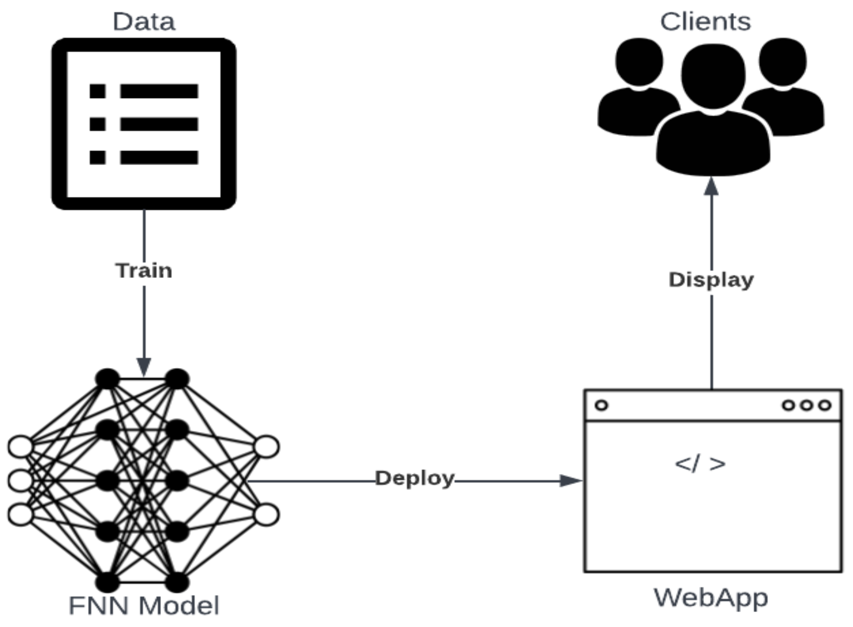
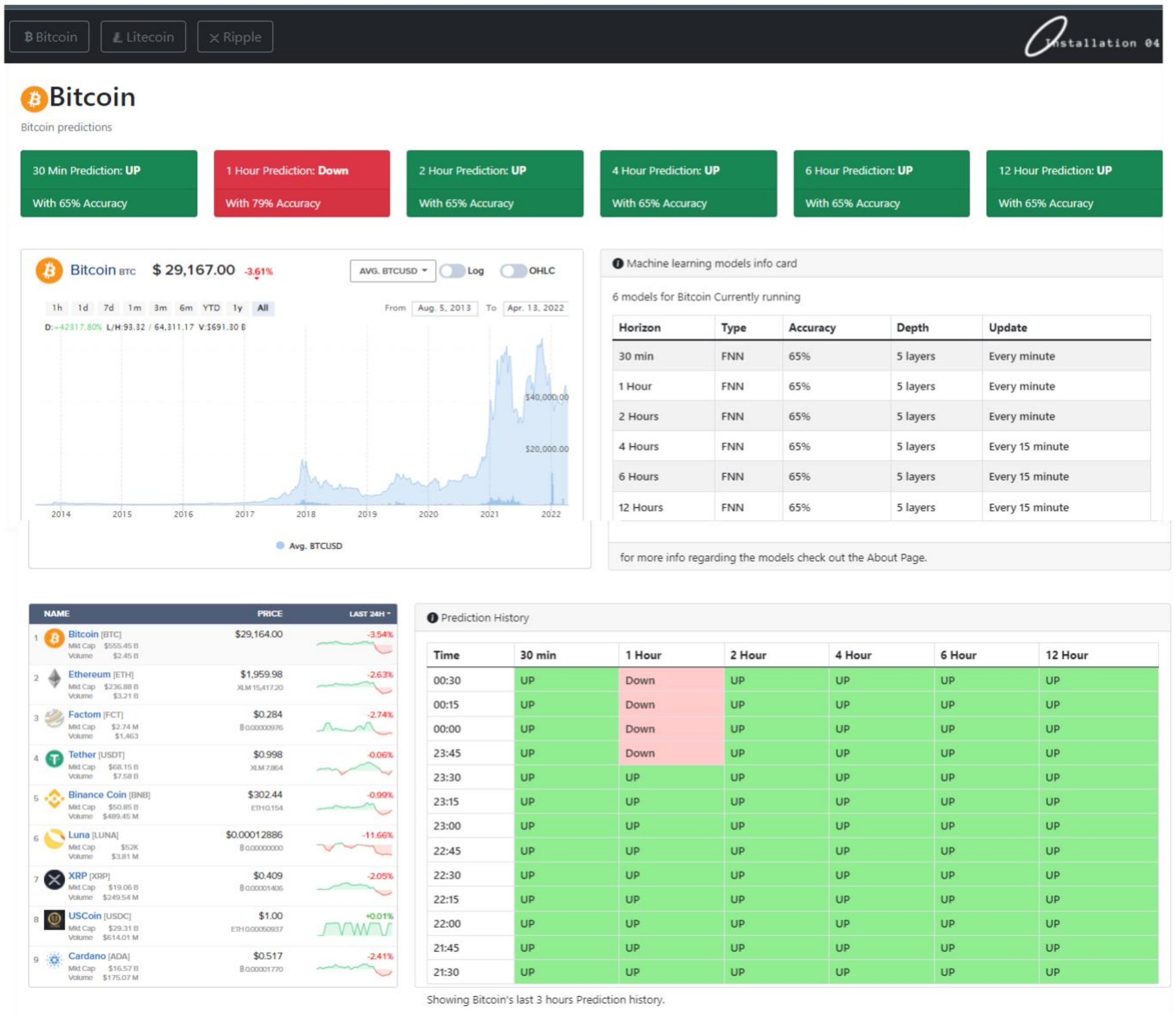


Figure 1 Preliminary Design

B. Web Application Design

Figure 2 shows the design of the web application.. There are 3 buttons on the web header, you can use them to visit other coins pages. The webapp is split into multiple sections: The webapp is split into multiple sections:

- Predictions Cards
- Bitcoin & Litecoin & Ripple graphs for prices.
- Model’s Info Card
- Predictions History
- Other coins prices in a widget



Bitcoin

Bitcoin predictions

30 Min Prediction: **UP** With 65% Accuracy

1 Hour Prediction: **Down** With 79% Accuracy

2 Hour Prediction: **UP** With 65% Accuracy

4 Hour Prediction: **UP** With 65% Accuracy

6 Hour Prediction: **UP** With 65% Accuracy

12 Hour Prediction: **UP** With 65% Accuracy

Machine learning models info card

6 models for Bitcoin Currently running

Horizon	Type	Accuracy	Depth	Update
30 min	FNN	65%	5 layers	Every minute
1 Hour	FNN	65%	5 layers	Every minute
2 Hours	FNN	65%	5 layers	Every minute
4 Hours	FNN	65%	5 layers	Every 15 minute
6 Hours	FNN	65%	5 layers	Every 15 minute
12 Hours	FNN	65%	5 layers	Every 15 minute

for more info regarding the models check out the About Page.

	NAME		PRICE		LAST 24H
1	Bitcoin [BTC]	Mkt Cap $555.45 B Volume $2.45 B	$29,164.00		-3.54%
2	Ethereum [ETH]	Mkt Cap $236.88 B Volume $3.21 B	$1,959.98	XLM 15,417.20	-2.63%
3	Factom [FCT]	Mkt Cap $2.74 M Volume $1,463	$0.284	Ƀ 0.00000976	-2.74%
4	Tether [USDT]	Mkt Cap $68.15 B Volume $7.58 B	$0.998	XLM 7.864	-0.06%
5	Binance Coin [BNB]	Mkt Cap $50.85 B Volume $489.45 M	$302.44	ETH 0.154	-0.99%
6	Luna [LUNA]	Mkt Cap $52K Volume $3.81 M	$0.00012886	Ƀ 0.00000000	-11.66%
7	XRP [XRP]	Mkt Cap $19.06 B Volume $249.54 M	$0.409	Ƀ 0.00001406	-2.05%
8	USCoin [USDC]	Mkt Cap $29.31 B Volume $614.01 M	$1.00	ETH 0.00050937	+0.01%
9	Cardano [ADA]	Mkt Cap $16.57 B Volume $175.07 M	$0.517	Ƀ 0.00001770	-2.41%

Prediction History

Time	30 min	1 Hour	2 Hour	4 Hour	6 Hour	12 Hour
00:30	UP	Down	UP	UP	UP	UP
00:15	UP	Down	UP	UP	UP	UP
00:00	UP	Down	UP	UP	UP	UP
23:45	UP	Down	UP	UP	UP	UP
23:30	UP	UP	UP	UP	UP	UP
23:15	UP	UP	UP	UP	UP	UP
23:00	UP	UP	UP	UP	UP	UP
22:45	UP	UP	UP	UP	UP	UP
22:30	UP	UP	UP	UP	UP	UP
22:15	UP	UP	UP	UP	UP	UP
22:00	UP	UP	UP	UP	UP	UP
21:45	UP	UP	UP	UP	UP	UP
21:30	UP	UP	UP	UP	UP	UP

Showing Bitcoin's last 3 hours Prediction history.

Figure 2 Web Application Design

Figure 3 shows the prediction cards. The built-in models are plugged into these cards, and when they make predictions, the cards display the prediction along with its accuracy. 30Min, 1 Hour, and 2 Hour cards are updated every one-minute, other cards are updated every 15 minutes. The cards are coloured green when the prediction is UP and red when the Prediction is DOWN.

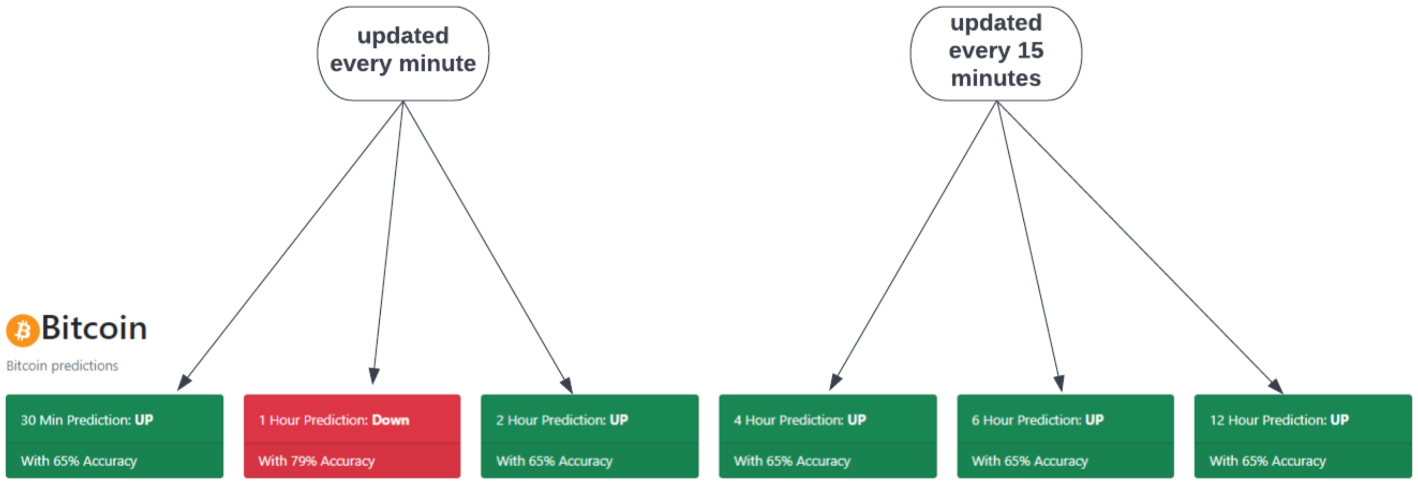


Figure 3 Prediction Cards

The live prices of each coin in each page will be displayed on a graph as shown in figure 4. This graph is a widget taken from this website [https://coinlib.io/coin/BTC/Bitcoin].

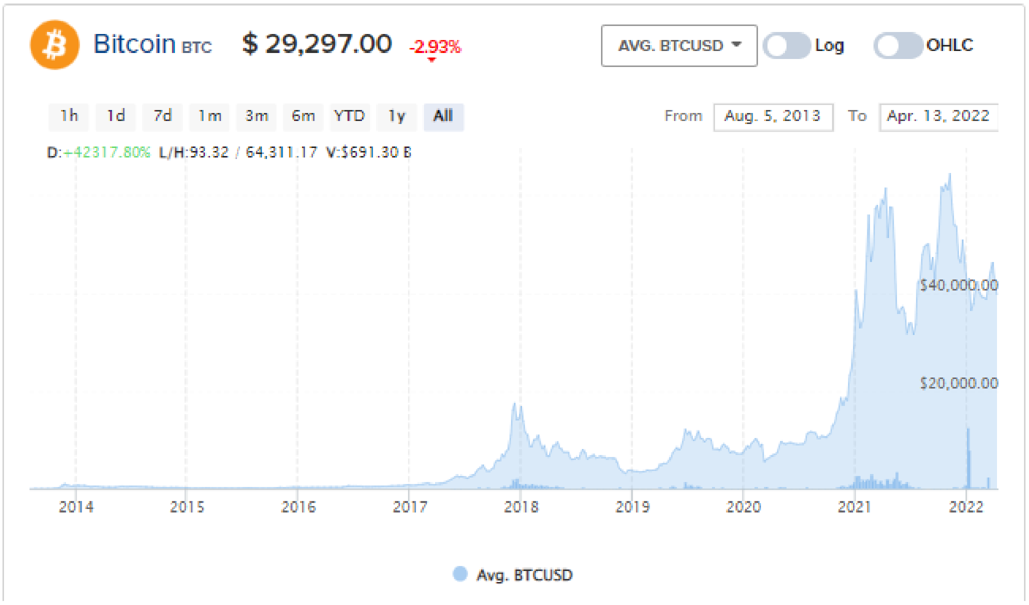
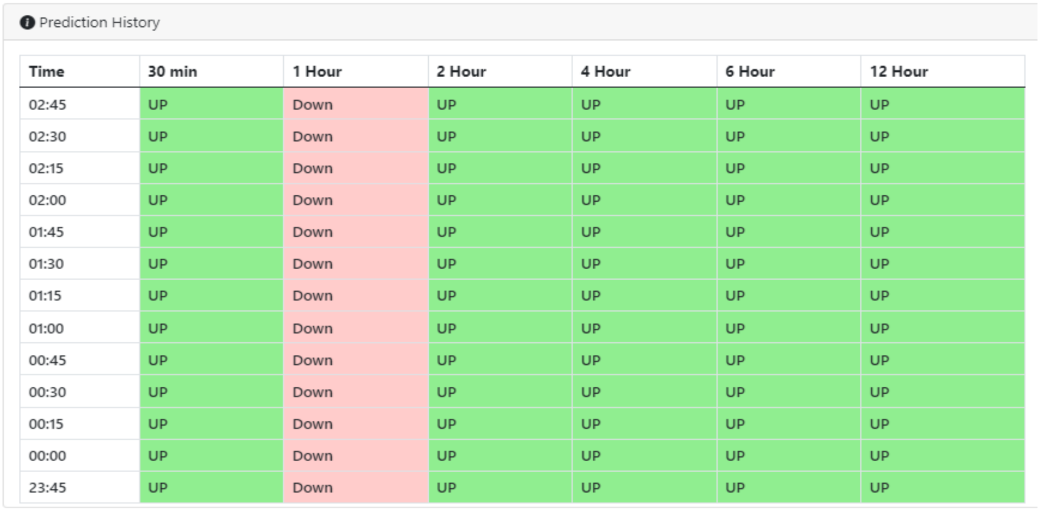


Figure 4 Coins prices on a graph



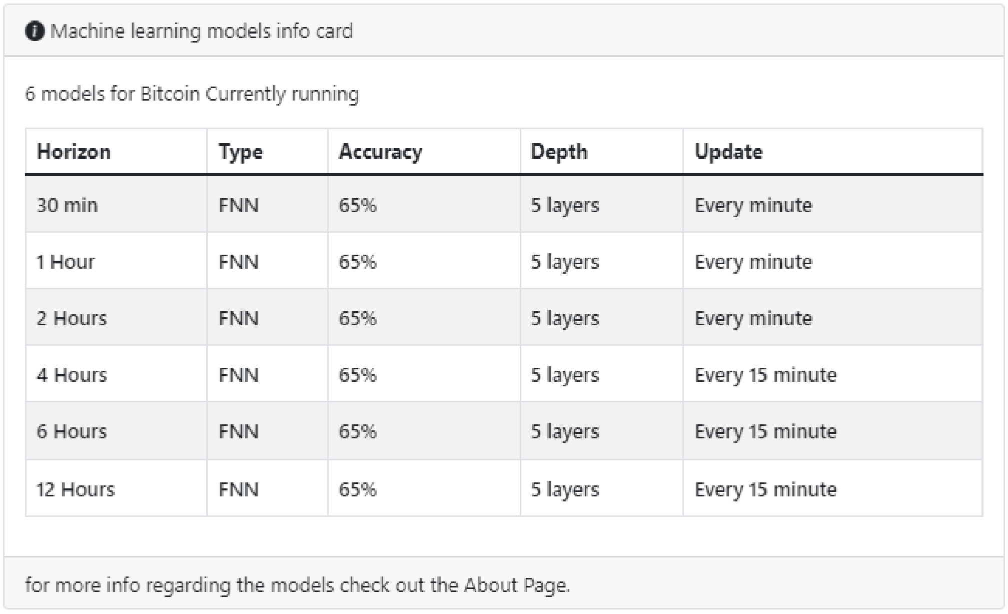
Prediction History

Time	30 min	1 Hour	2 Hour	4 Hour	6 Hour	12 Hour
02:45	UP	Down	UP	UP	UP	UP
02:30	UP	Down	UP	UP	UP	UP
02:15	UP	Down	UP	UP	UP	UP
02:00	UP	Down	UP	UP	UP	UP
01:45	UP	Down	UP	UP	UP	UP
01:30	UP	Down	UP	UP	UP	UP
01:15	UP	Down	UP	UP	UP	UP
01:00	UP	Down	UP	UP	UP	UP
00:45	UP	Down	UP	UP	UP	UP
00:30	UP	Down	UP	UP	UP	UP
00:15	UP	Down	UP	UP	UP	UP
00:00	UP	Down	UP	UP	UP	UP
23:45	UP	Down	UP	UP	UP	UP

Figure 5 Prediction History

The last 3 hours predictions made are displayed on a table for each horizon as shown in figure 5. It's a good view to give insights on past predictions.

This table in figure 6 shows the deployed models with their info. This table can change, depending on the deployed models. It shows the prediction Horizon, the used algorithm, the accuracy of the algorithm, the layers depth used in the algorithm, and the Update.


Machine learning models info card

6 models for Bitcoin Currently running

Horizon	Type	Accuracy	Depth	Update
30 min	FNN	65%	5 layers	Every minute
1 Hour	FNN	65%	5 layers	Every minute
2 Hours	FNN	65%	5 layers	Every minute
4 Hours	FNN	65%	5 layers	Every 15 minute
6 Hours	FNN	65%	5 layers	Every 15 minute
12 Hours	FNN	65%	5 layers	Every 15 minute

for more info regarding the models check out the About Page.

Figure 6 models info card

V. Software Implementation

Figure 7 shows the detailed layout of the project design and architecture.

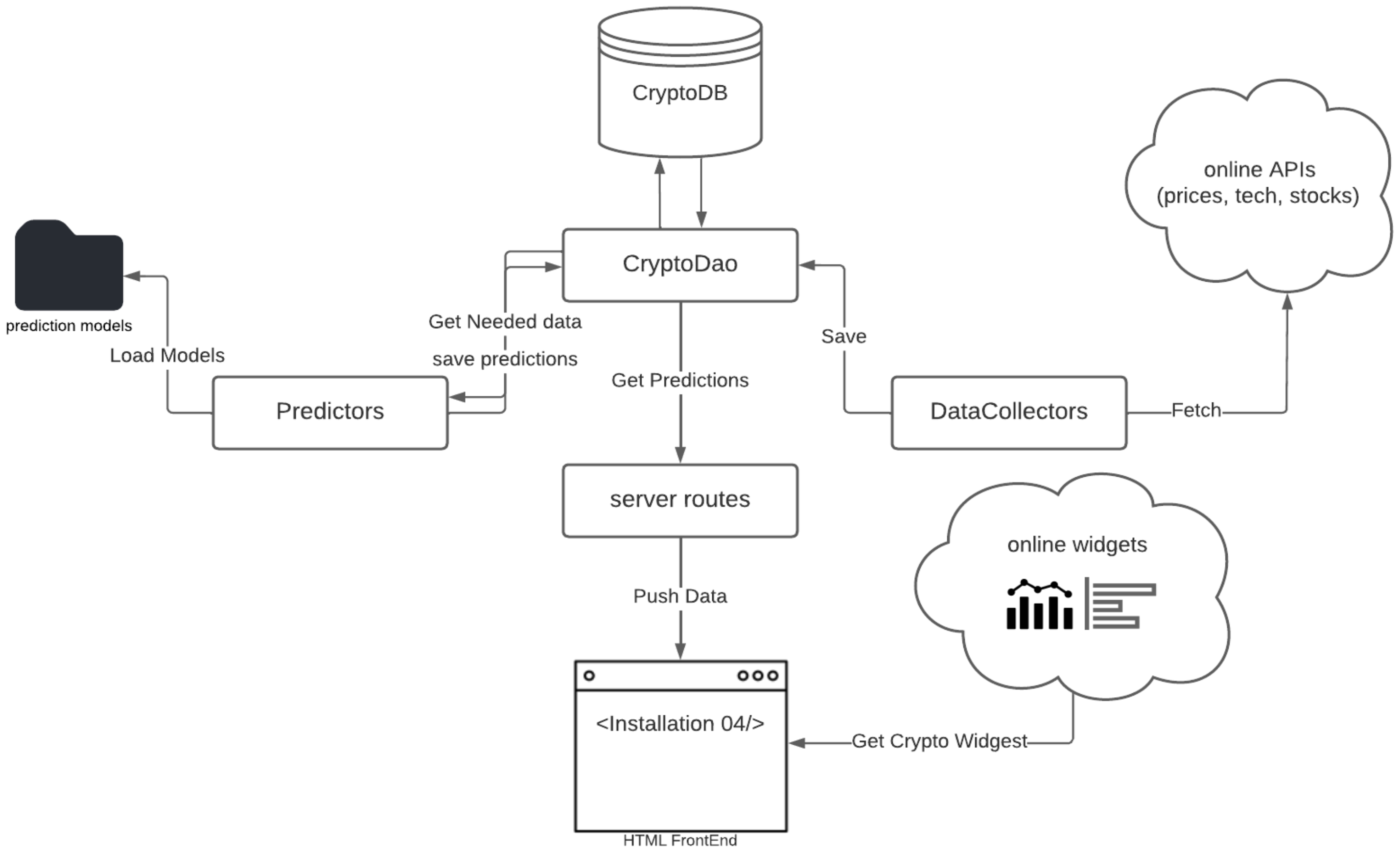
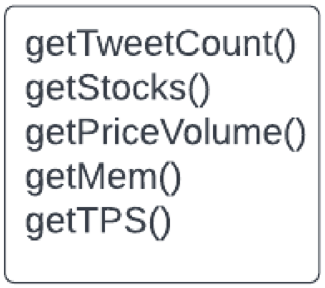


Figure 7 Project Architecture

1. Data Collectors

Data collectors contains all the functions that are responsible to bring the data through online APIs. The codes were optimized and built in a consistent way in order make the code run continuously without any interruptions or exceptions. Also, a Log file was created to keep track of the ongoing processes, the log file tells us the current state of the processes and if the data is being fetched correctly. it was useful to help make any maintenance required when captured.

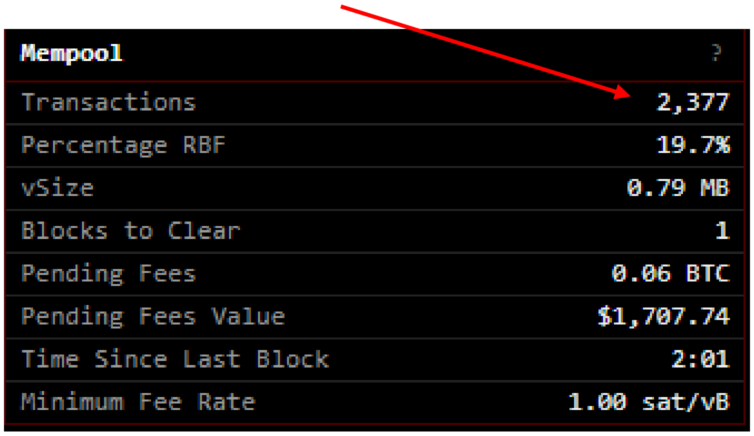


```
getTweetCount()
getStocks()
getPriceVolume()
getMem()
getTPS()
```

Figure 8 Data Collectors Functions

Figure 8 shows the functions responsible of fetching data, all these functions return -1 if they couldn't get the data, so we know how to deal with it**.**

1. **Memory pool:** The code creates a connection with Clarkmoody website and uses its socket to send requests to fetch the memory pool transactions then the server responses with the Transactions value. the value is saved in the Database.

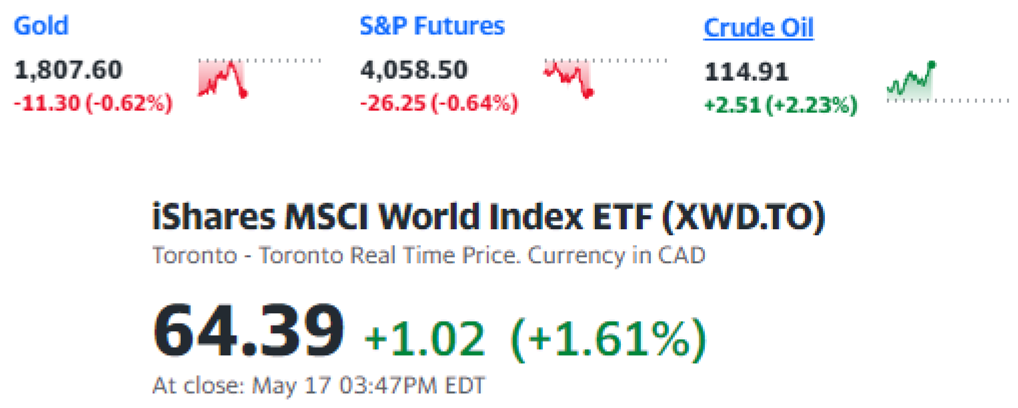


Mempool	?
Transactions	2,377
Percentage RBF	19.7%
vSize	0.79 MB
Blocks to Clear	1
Pending Fees	0.06 BTC
Pending Fees Value	$1,707.74
Time Since Last Block	2:01
Minimum Fee Rate	1.00 sat/vB

Figure 9 Clark Moody website

Figure 9 shows the location of Mempool data we collected; this is where the code takes the data from. You can visit https://bitcoin.clarkmoody.com/dashboard/

2. **Stock Collector:** Yahoo Finance offers real time prices. So, the code uses YF API to request GOLD, S&P, OIL, and MSCI prices, then it saves them in a csv file. Figure 10 shows the prices displaying on the website.



Gold
1,807.60
-11.30 (-0.62%)

S&P Futures
4,058.50
-26.25 (-0.64%)

Crude Oil
114.91
+2.51 (+2.23%)

iShares MSCI World Index ETF (XWD.TO)
Toronto - Toronto Real Time Price. Currency in CAD
64.39 +1.02 (+1.61%)
At close: May 17 03:47PM EDT

Figure 10 Yahoo Finance Data

3. **Transactions Per Second (TPS):** The code uses selenium web driver that has an API in which it accesses Satoshi.info and fetches the transactions per second. Figure 11 shows how Satoshi.info offers its real time data.

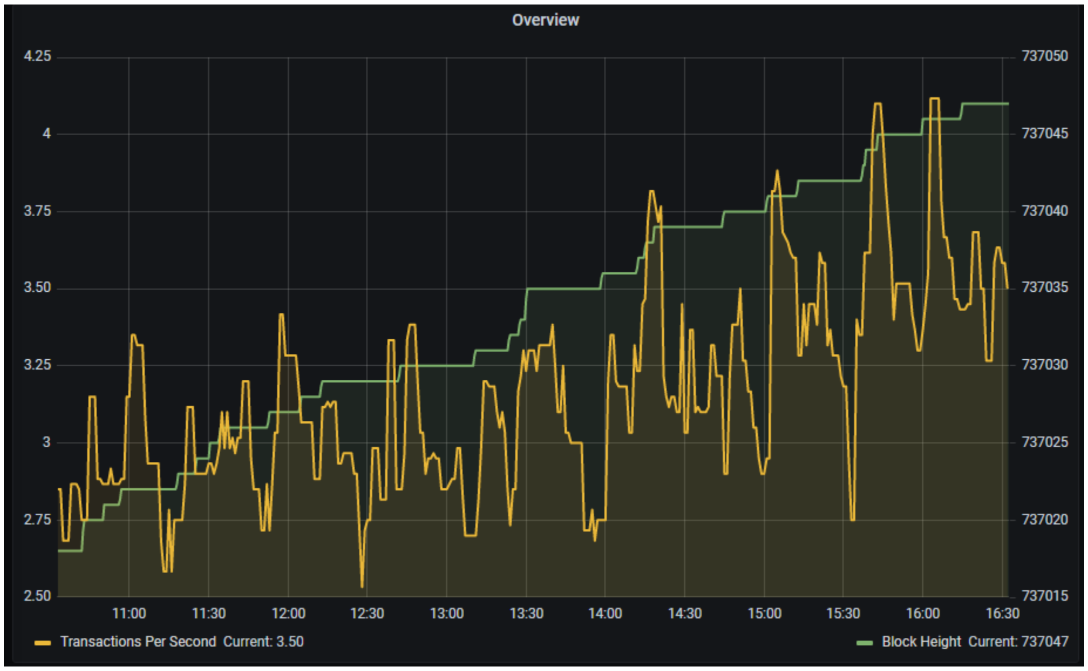


Figure 11 Satoshi.info TPS

4. **Tweets count**: The code uses beautiful-SOAP library that can access Bitinfocharts.com website that offers various data for many coins. Figure 12 shows how the site displays the data, beautiful-SOAP can fetch data from graphs. This graph is updated every one hour, so we fetch the latest data displayed on the graph. As the arrow shows.

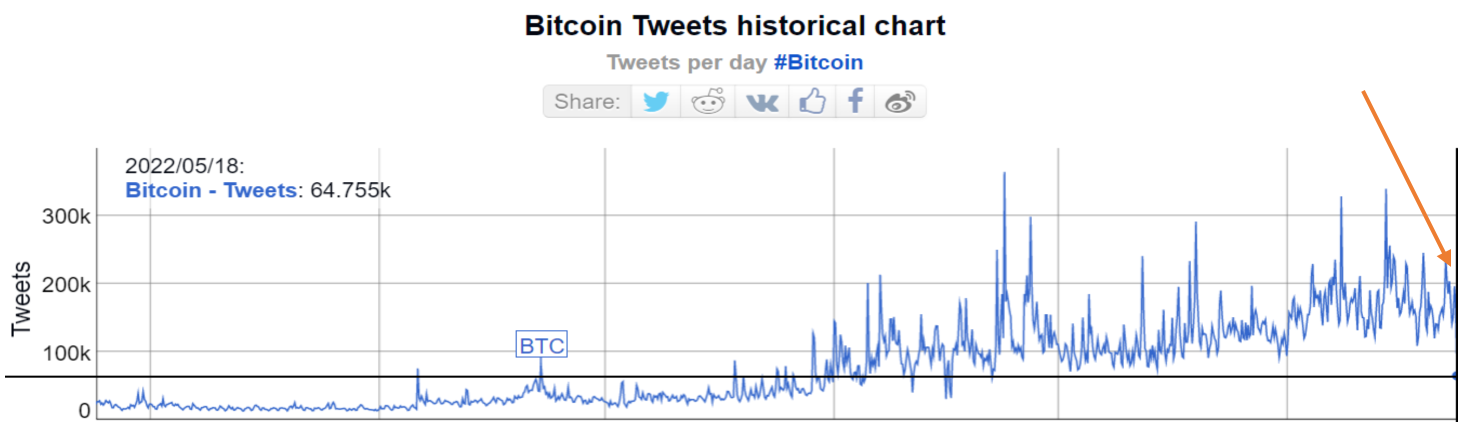
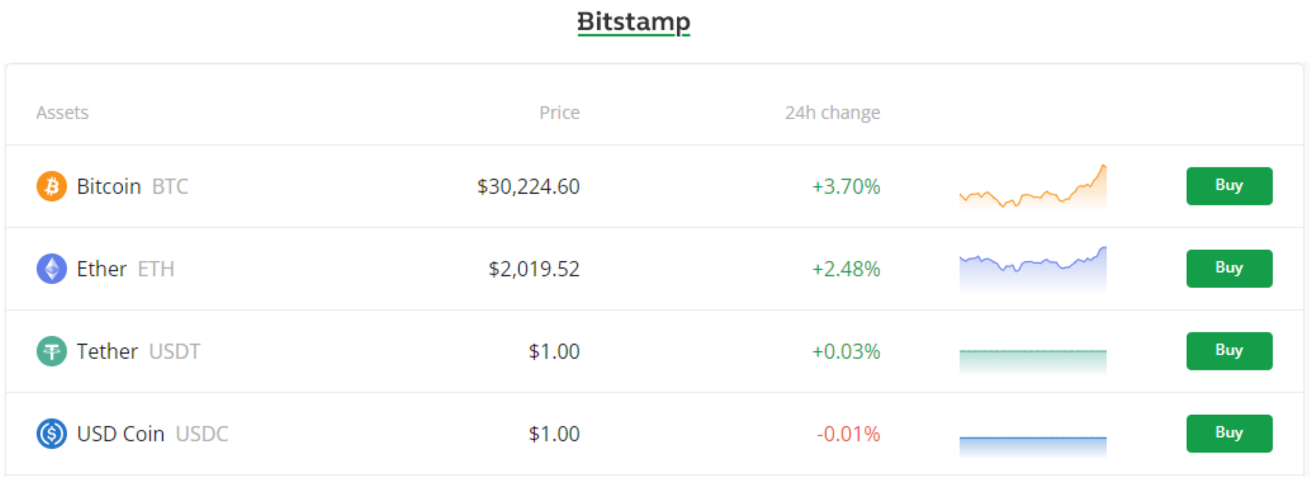


Figure 12 Bitcoin Number of tweets

5. **Prices & Volume:** bitstamp.net, This website offers minutely prices and volume data for the coins. So, our code uses it’s API to fetch the data. Figure 13 shows the data which the website offers.



Bitstamp

Assets	Price	24h change		
Bitcoin BTC	$30,224.60	+3.70%		Buy
Ether ETH	$2,019.52	+2.48%		Buy
Tether USDT	$1.00	+0.03%		Buy
USD Coin USDC	$1.00	-0.01%		Buy

Figure 13 bitstamp.net Data

2. Models Training

This training included several parts to be completed. Figure 14 summarizes the scheme we followed:

1. Determine the features that will be used in our models.
2. Collect data from various resources.
3. Clean up the datasets.
4. Extract features from datasets.
5. Build different models and train the data.

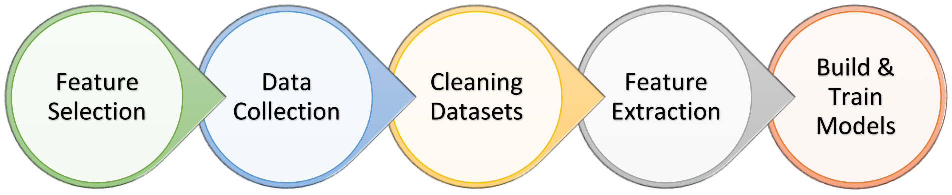


Figure 14 Models Training Processes

2.1 Feature Selection

The first step is to determine the data we need to use for our models and that will be the mostly effect the behaviour of cryptocurrency prices. We employ features from feature categories identified by Jaquart et al [1].
Figure 15 shows the categories of features used, and what we selected from each category, also based on what we found on the internet.

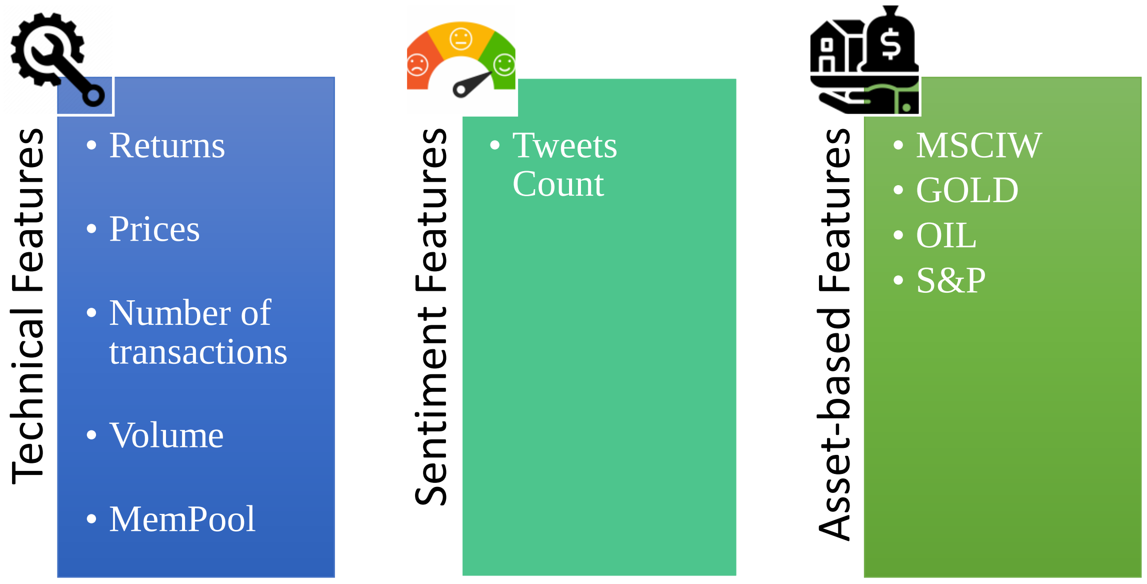


Figure 15 Features Categories

2.2 Data Collection

This part is divided into two parts:

2.1 Available Datasets

These are the datasets we found on a certain website and downloaded them right away. Figure 16 shows the websites used.

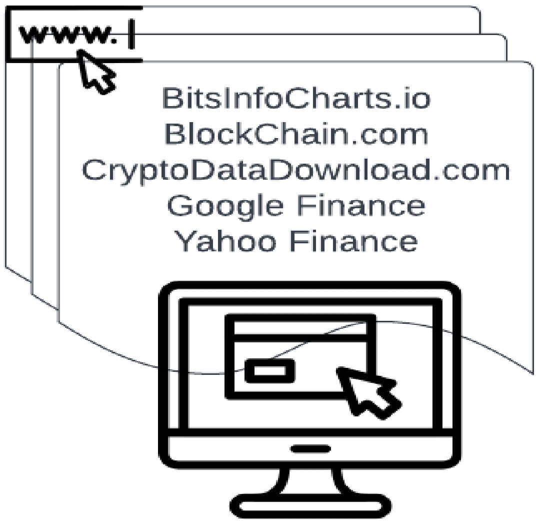


Figure 16 Websites with available datasets

2.2 Scraping our own datasets

Most of these websites gave incomplete data, also most of the data required a lot of pre-processing and interpolation which will in turn affect accuracy.

So, the best way to get the data we want is to scrape it ourselves from different websites and APIs. We used various websites that required different methods to interface and obtain the data.

We programmed 5 different processes using python that collect data points every 15 minutes and saved them into datasets. Figure 17 shows the processes that were used to collect data.

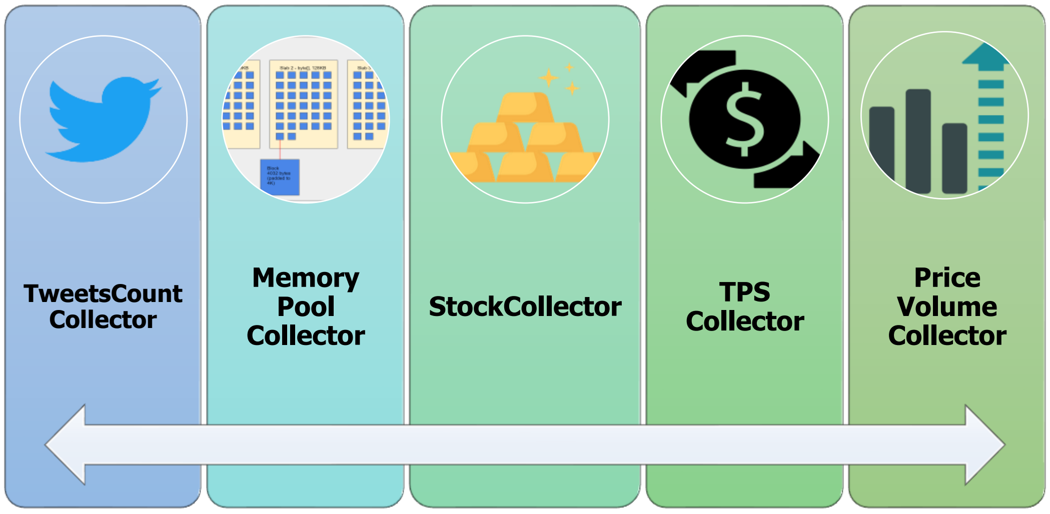
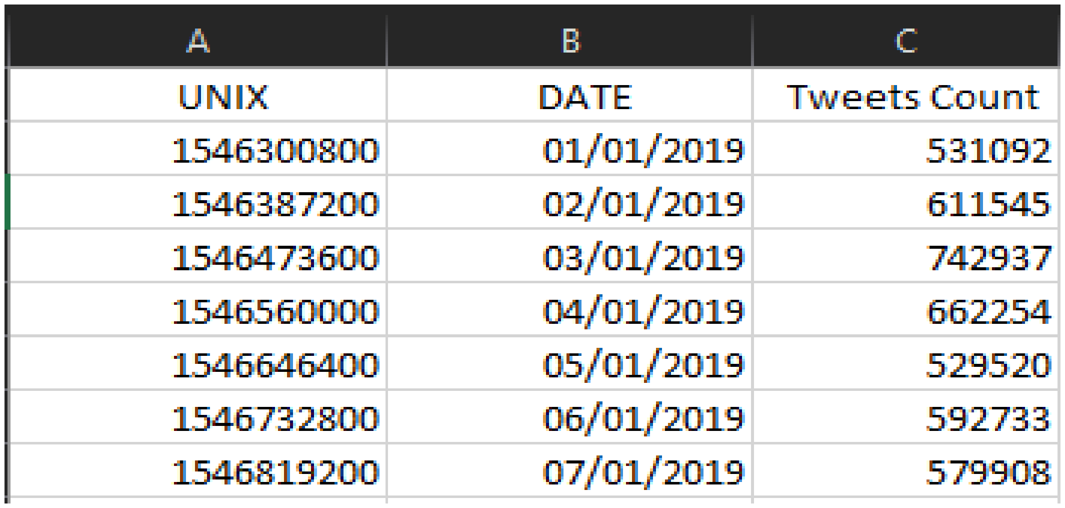


Figure 17 Data Scrapers

Figure 18 shows how Tweets count is scraped and saved into a CSV file; this data is extracted as one of the features to be part of the model training. Other scrapers do the same thing.



A	B	C
UNIX	DATE	Tweets Count
1546300800	01/01/2019	531092
1546387200	02/01/2019	611545
1546473600	03/01/2019	742937
1546560000	04/01/2019	662254
1546646400	05/01/2019	529520
1546732800	06/01/2019	592733
1546819200	07/01/2019	579908

Figure 18 Tweets Count Data

2.3 Cleaning Datasets

Finding and correcting inaccurate data in the dataset is a necessary step before feeding it to our models. Faulty or missing data might result in errors and may affect the accuracies of the model.

Our obtained data required a lot of work to prepare it for model training.

Almost all the data had jumps and skips or had different time frames, which we corrected mostly using linear and constant interpolations.

Figure 19 shows the constant interpolation used. Figure 20 shows the linear interpolation.

In case of jumps in the data, linear interpolation seemed to work better.

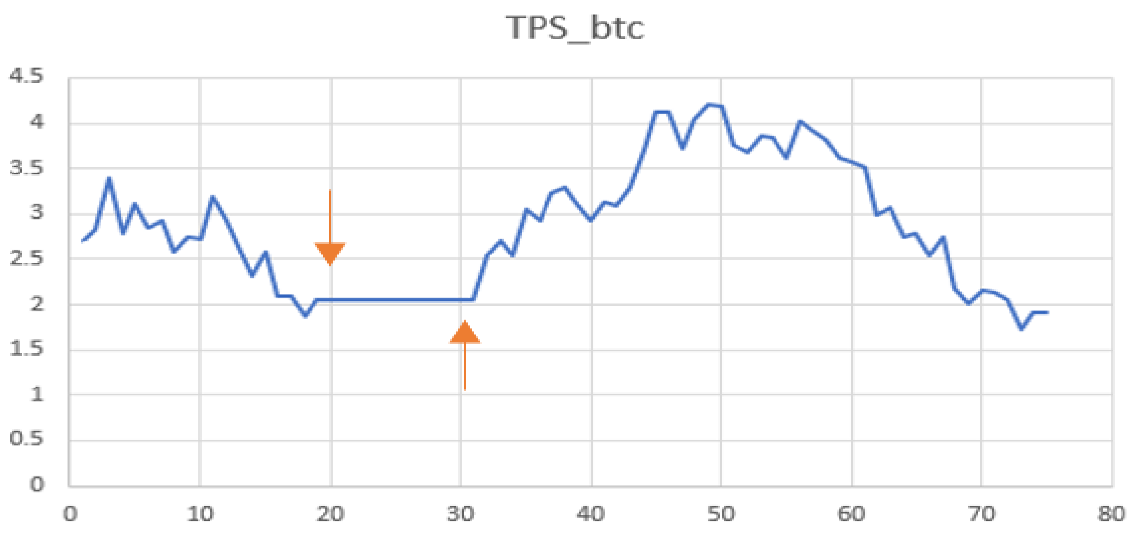


Figure 19**:** Constant Interpolation

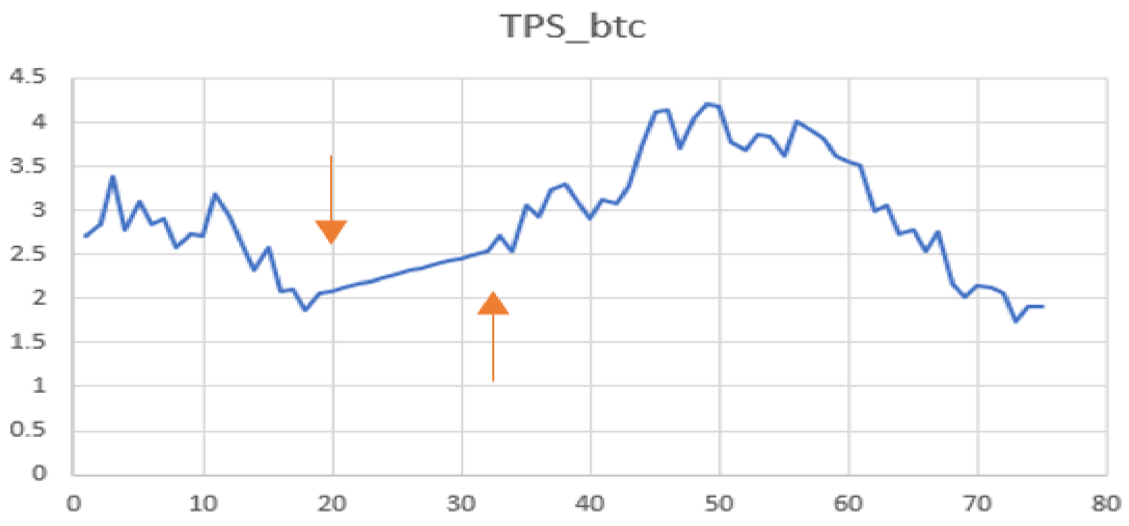


Figure 20: Linear Interpolation

In case of matching time frames:

The time frame we need from all features is 30-minutes. We used Unix time to mark timestamps.

Stock was daily data, therefore we used constant interpolation during the stock market is closed, and tried linear and constant, and polynomial interpolation for the rest when the market was open. Yahoo and google finance only gave daily data for stocks.Blockchain.com offers data every 30 minutes for the past month, every 60 minutes for the past 2 months and so on. So, we used interpolation to fill the gaps.

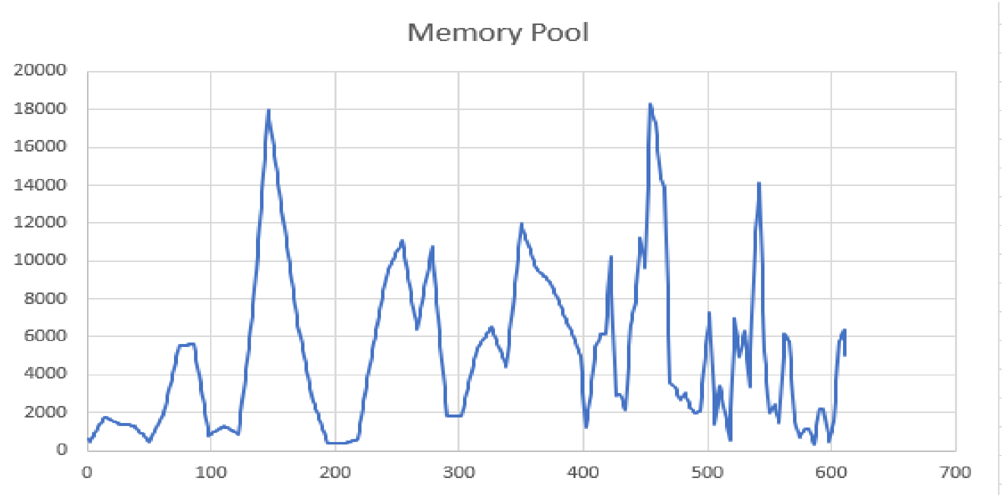


Figure 21: Matching timeframes

As shown in figure 21, the left half of the data has a sample every 60 minutes, while the right half is every 30 minutes.

2.4 Feature Extraction

For the models that have no memory like SVM and Logistic Regression, we made each sample have data regarding current and past prices and movements.

Each sample that will be fed to the model had the following structure:

[Current values], [Period 1 values], [Period 2 values], [Period 3 values], ...

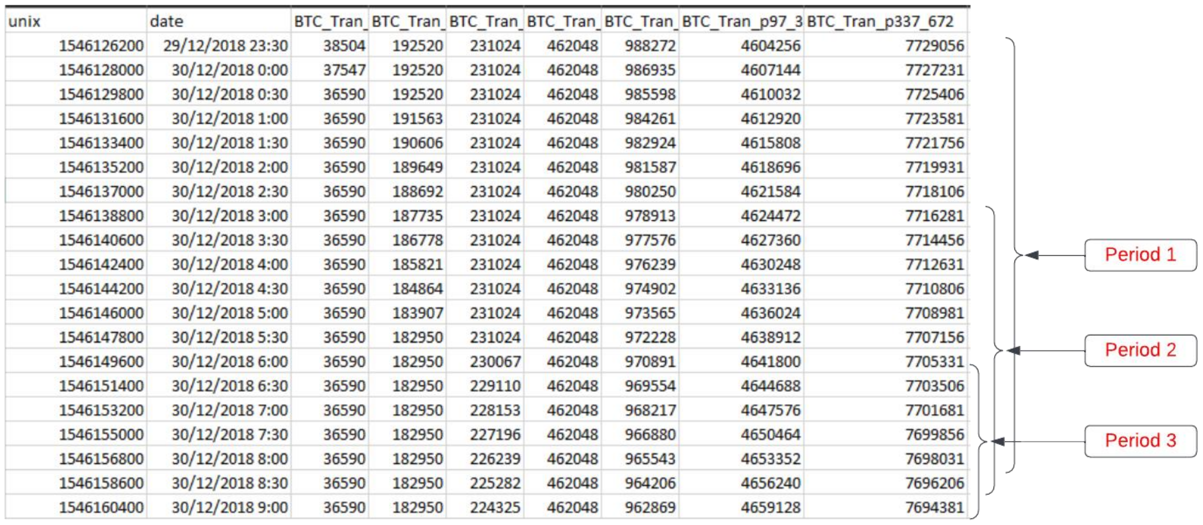
Where a period represents a past time interval. For example: **past half hour**. we tested different numbers of periods and different sizes for each period. For example:

Regarding bitcoin technical features: we use minute prices to extract returns, high and low in a period.

$$R = \frac{F-S}{S} \qquad (1)$$

Where, R= returns, F= final price, S= starting price.

Figure 22 illustrates the concept of periods, and how each period is extracted. When training our models, we coded it in a way that allows us to specify the number of periods and how much to increment with every period. Different tested combinations resulted in different outcomes and accuracies. The periods manipulation like period size or period increment ratio or period count, also played a good part in models training time, some models took hours to train, others took several minutes.



unix	date	BTC_Tran_	BTC_Tran_	BTC_Tran_	BTC_Tran_	BTC_Tran_	BTC_Tran_p97_3	BTC_Tran_p337_672
1546126200	29/12/2018 23:30	38504	192520	231024	462048	988272	4604256	7729056
1546128000	30/12/2018 0:00	37547	192520	231024	462048	986935	4607144	7727231
1546129800	30/12/2018 0:30	36590	192520	231024	462048	985598	4610032	7725406
1546131600	30/12/2018 1:00	36590	191563	231024	462048	984261	4612920	7723581
1546133400	30/12/2018 1:30	36590	190606	231024	462048	982924	4615808	7721756
1546135200	30/12/2018 2:00	36590	189649	231024	462048	981587	4618696	7719931
1546137000	30/12/2018 2:30	36590	188692	231024	462048	980250	4621584	7718106
1546138800	30/12/2018 3:00	36590	187735	231024	462048	978913	4624472	7716281
1546140600	30/12/2018 3:30	36590	186778	231024	462048	977576	4627360	7714456
1546142400	30/12/2018 4:00	36590	185821	231024	462048	976239	4630248	7712631
1546144200	30/12/2018 4:30	36590	184864	231024	462048	974902	4633136	7710806
1546146000	30/12/2018 5:00	36590	183907	231024	462048	973565	4636024	7708981
1546147800	30/12/2018 5:30	36590	182950	231024	462048	972228	4638912	7707156
1546149600	30/12/2018 6:00	36590	182950	230067	462048	970891	4641800	7705331
1546151400	30/12/2018 6:30	36590	182950	229110	462048	969554	4644688	7703506
1546153200	30/12/2018 7:00	36590	182950	228153	462048	968217	4647576	7701681
1546155000	30/12/2018 7:30	36590	182950	227196	462048	966880	4650464	7699856
1546156800	30/12/2018 8:00	36590	182950	226239	462048	965543	4653352	7698031
1546158600	30/12/2018 8:30	36590	182950	225282	462048	964206	4656240	7696206
1546160400	30/12/2018 9:00	36590	182950	224325	462048	962869	4659128	7694381

Figure 22 Extracting a sample

2.5 Building and Training Models

We used Juypter Notebook for the development and training of the models. Jupyter Notebook is Free software, open standards, and web services for interactive computing across all programming languages.

Libraries used:

- Scikit-learn open-source python package built for data analysis, integrated with many machine learning algorithms for supervised and unsupervised problems. This package is designed for non-specialists to help them create machine learning models. Focusing on performance, documentation, and API consistency. [2]
- Pandas used for analysing and structuring the data. It allows filtering and merging of data, and importing data from external sources like Excel, for instance.
- Keras used for deep learning. It's designed to preform fast calculations and prototyping, as it allows the use of GPU as well as the CPU of the computer.
- TensorFlow used for developing neural networks to work with large datasets. And deep learning applications.
- NLTK for working with computational linguistics, natural language recognition, and processing.
- Matplotlib used mainly for visualization like creating 2D plots, charts, and histograms.
- NumPy used for mathematical operations on arrays, also preforms efficient calculations on matrices.

✓ Picked Algorithm

During graduation project one, multiple algorithms were tested. We built Logistic Regression, SVM, NN and RNN. After comparing the performance and accuracies of these algorithms, we landed on Feedforward neural networks (FNN). Neural networks seem to work better and gave greater results.

Neural Networks: a collection of algorithms that copy the behaviour and connections of neurons of human brain to find correlations between large amounts of data. NN constitute from a circuit of nodes, it gets a signal then processes it and can affect other nodes connected to it as shown in figure 23. Can be used in areas like paraphrase detection, language generation, character recognition, spell checking, etc.

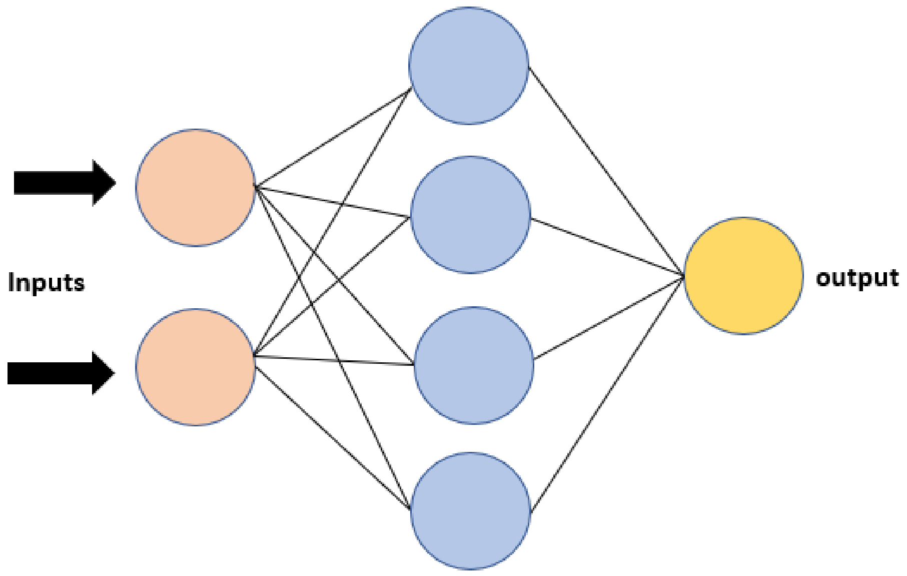
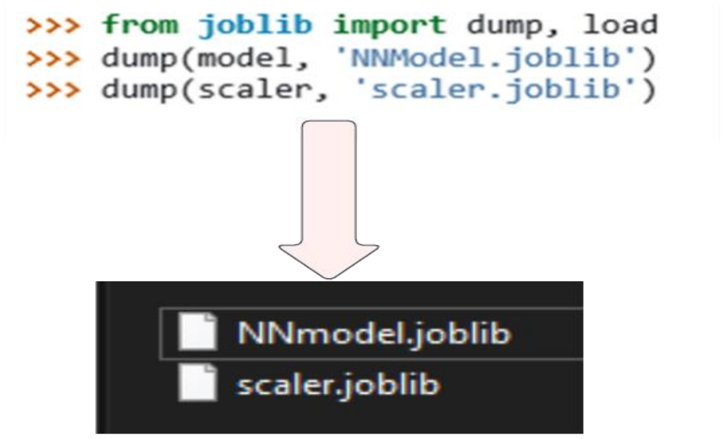


Figure 23: NN Nodes Example

Now after noting that Neural Networks are a good predictor. we used the newest data that's been being collected for six months to train NN models with newest data. This will allow the models to be more adaptable and accurate when making real-time predictions. After building these models, they're DUMPED along with their scaler into a file using joblib dump () function. Thus, like this we don't have to retrain the model again. They're saved in Models file and ready to be loaded when needed to make predictions. Figure 24 shows how the model is dumped along with it's scaler after training it. Every model we make will be saved in this way, inside the model's folder, so the **Predictor** loads the model it needs to make the prediction.



```
>>> from joblib import dump, load
>>> dump(model, 'NNModel.joblib')
>>> dump(scaler, 'scaler.joblib')
```

```
NNmodel.joblib
scaler.joblib
```

Figure 24 Dumping model and scaler

- Models Results

Table 1 NN on Bitcoin

horizon	30 min	1 hour	2 hours	4 hours	6 hours	12 hours
Accuracy	65%	79%	87%	83%	85%	85%
Price features	Past 2 years	Past 2 years	Past 2 years	Past 2 years	Past 2 years	Past 2 years
Technical features	non	non	non	#Tweets, memPool, TPS	#Tweets, memPool, TPS	#Tweets, memPool, TPS, Stocks

Table 2 NN on Litecoin

horizon	30 min	1 hour	2 hours	4 hours	6 hours	12 hours
Accuracy	65%	79%	87%	83%	85%	85%
Price features	Past 2 years	Past 2 years	Past 2 years	Past 2 years	Past 2 years	Past 2 years
Technical features	non	non	non	non	#Tweets, memPool, TPS	#Tweets, memPool, TPS

Table 3 NN on Ripple

horizon	30 min	1 hour	2 hours	4 hours	6 hours	12 hours
Accuracy	65%	79%	87%	83%	85%	85%
Price features	Past 2 years	Past 2 years	Past 2 years	Past 2 years	Past 2 years	Past 2 years
Technical features	non	non	non	non	#Tweets, memPool, TPS	#Tweets, memPool, TPS Stocks

3. Web Application

A web application is a browser-based interactive application created with web development technologies. The project is programmed on Visual Studio Code using python FLASK Framework, structured, and coded with object-oriented methods. The web application development consists of two main parts: **Frontend** and **Backend**.

3.1 Frontend (Client Side)

This part includes how we created our html pages and how we linked the frontend with the backend. Using Jinja templating.

- Client-side rendering (**frontend**): The frontend of a website is what you view and interact with. It contains everything the user sees and interacts with visually, from text and colours to buttons, graphics, and navigation menus.

  Essential Frontend Languages:
  - HTML: a markup language that is used to build and organize web content so that it can be displayed by a browser.
  - CSS: determines the look and design of a website's content, including layout, colours, and fonts.
  - JAVASCRIPT: a scripting language that is used to create interactive features such as drop-down menus, modal windows, and contact forms.

We used HTML tags to organize our contents and build the webpage. And **Bootstrap** as a CSS framework to style our pages. **Bootstrap** "Is a CSS framework that is free and open source. It includes design templates for typography, forms, buttons, navigation, and other interface elements in HTML, CSS, and JavaScript.".
Jinja is a web template engine for the Python programming language. Jinja is the default engine template for FLASK, and it allows the designers to call functions inside HTML code. And pass data between frontend and backend.

Turbo Flask an extension that integrates the turbo.js JavaScript library with Flask application. Containing functions that allow you to dynamically update the webpage with new data without having to reload the whole page. This was used to make the web application look more dynamic and smoother.

3.2 Backend (Server Side)

Server-side rendering (**backend**): The backend (known as "server-side") is the part of the website that you cannot view. It's in charge of storing and organizing data, as well as ensuring that everything on the client side works properly. The backend and frontend communicate through sending and receiving data that is presented as a web page. Your browser makes a request to the server-side whenever you fill out a contact form, enter in a web URL, or make a purchase. which returns information in the form of frontend code that the browser may read and display.

Python is a popular Backend programming language. And using a specialized framework can simplify the process of developing webapp.

- ### Framework

A framework "is a code library that makes a developer's life easier when building reliable, scalable, and maintainable web applications" by providing reusable code or extensions for common operations. There are several frameworks for Python, including **Flask**, Tornado, **Pyramid**, and **Django**. Every framework has its own approach of putting routes, models, views, database interaction, and overall application settings combined.

- ### Flask Framework

We decided to use **Flask** [3] as a Framework for our Backend. **Flask** is a Python web framework built with a small core and easy-to-extend philosophy.

Figure 25 Flask Logo

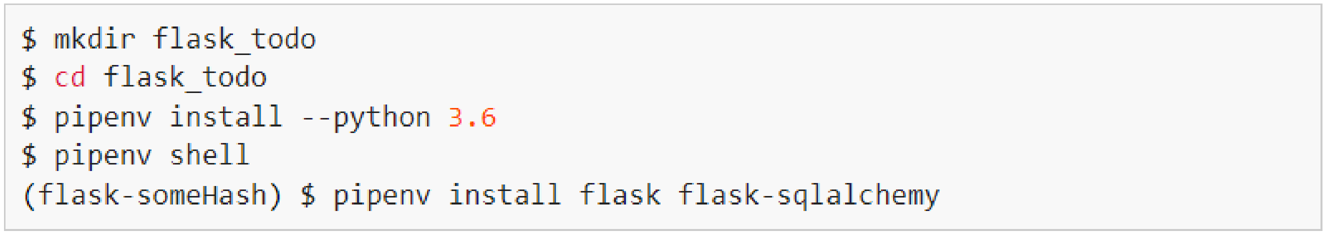
- ### Why Is Flask a Good Choice?

Since the similar **Flask** web application is clearer in many cases, **Flask** is considered more Pythonic than the Django web framework. **Flask** is very straightforward to learn as a novice due to the lack of complex code required to get a small project up and running. Advantages of Flask:

- Includes a built-in development server and a quick debugger.
- Light.
- Secure cookies are supported.
- Templating using Jinja2.
- Request dispatching using REST.
- Unit testing is supported by default.

Flask Installation is simple, can be done using Python Package Index (PPL).

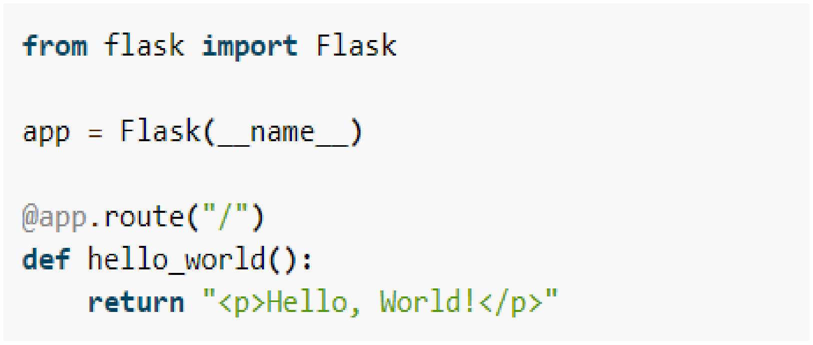
Start-up and Configuration is very easy, using Python 3 virtual Environment:



```
$ mkdir flask_todo
$ cd flask_todo
$ pipenv install --python 3.6
$ pipenv shell
(flask-someHash) $ pipenv install flask flask-sqlalchemy
```

Figure 26 Flask installation

A minimal Flask application looks like this:



```python
from flask import Flask

app = Flask(__name__)

@app.route("/")
def hello_world():
    return "<p>Hello, World!</p>"
```

Figure 27 Minimal Flask App

The backend development of this project consists of four main things:

1. **CrytoDao**
2. **Server Routes**
3. **Predictors**
4. **Database**

3.2.1 CryptoDao

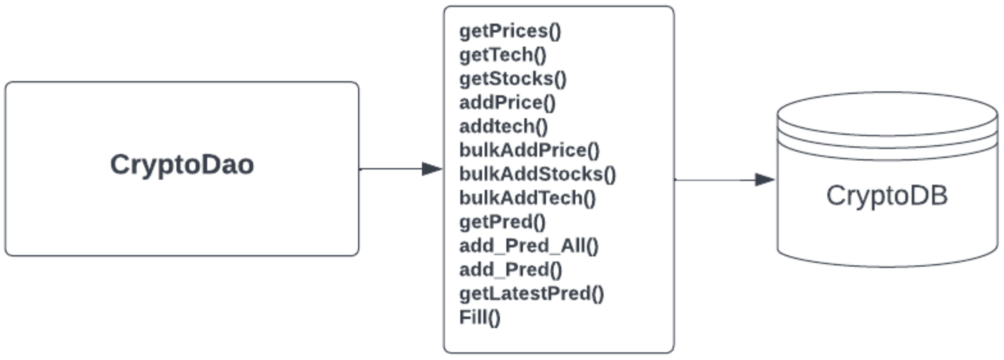
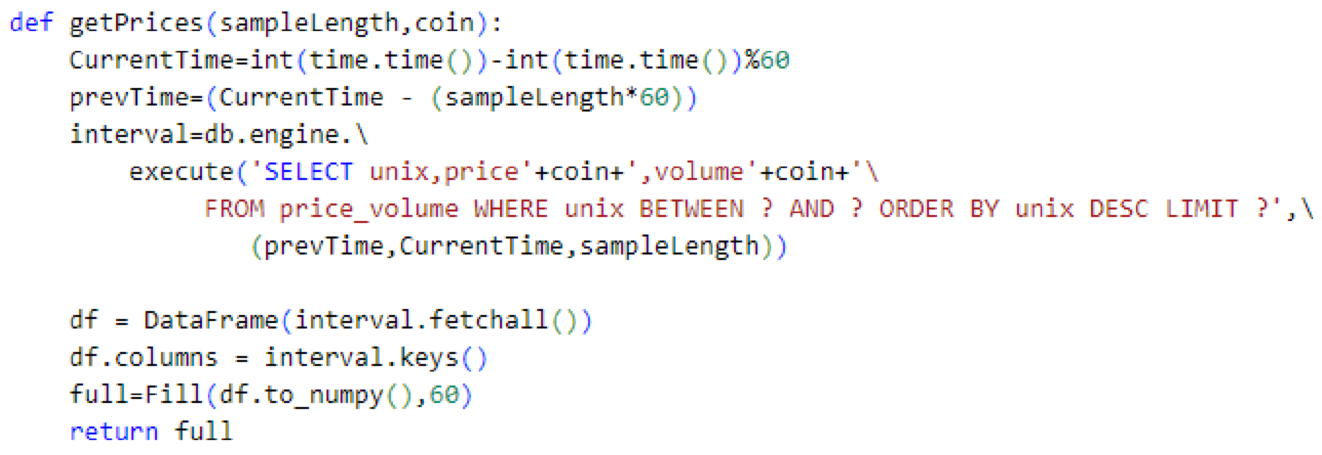


Figure 28 CryptoDao functions

Figure 28 shows the functions CryptoDao uses to invoke and deal with the database. We are going to explain some of these functions work. The rest uses same techniques.



```
def getPrices(sampleLength,coin):
    CurrentTime=int(time.time())-int(time.time())%60
    prevTime=(CurrentTime - (sampleLength*60))
    interval=db.engine.\
        execute('SELECT unix,price'+coin+',volume'+coin+'\
            FROM price_volume WHERE unix BETWEEN ? AND ? ORDER BY unix DESC LIMIT ?',\
               (prevTime,CurrentTime,sampleLength))

    df = DataFrame(interval.fetchall())
    df.columns = interval.keys()
    full=Fill(df.to_numpy(),60)
    return full
```

Figure 29 getPrices function content

- **getPrices ():** This function gets the prices and volume from the database, it takes two parameters Sample Length and Coin, it gets current time using **time.time().** Afterwards it calculates how much data it should fetch from the database using the sample length and selects what table to fetch from based on the chosen coin then it saves the 2D array in *interval* variable. Then it transforms the fetched interval of data into a NumPy array, then it checks of any gaps in the data and fills them using **Fill ()** function.
  For example, if sample length = 97200 and coin = BTC, it fetches past 97200 rows from Bitcoin table.

- **Fill ():** an important function that receives **rows** and **timeframe** as parameters**,** this function checks if there are any skips on the Unix time in the received rows, if so, it fills the gap between two rows with the previous value.

3.2.2 Predictors

Predictors are responsible of employing the models from the file to be used to make prediction. Two functions are being called continuously **update15MinModels ()** and **updateOneMinModels ().** These functions are in charge of loading the desired models from the file. Then CryptoDao is used to fetch data from the database and then the data is passed to the models. A sample of this data is constructed to feed it to the model to make the prediction. Once the predictions are made, they're saved into predictions table in the Database, and displayed on the webpages. Figure 30 illustrates these processes. **Models'** folder contains 18 models, 6 for each coin.

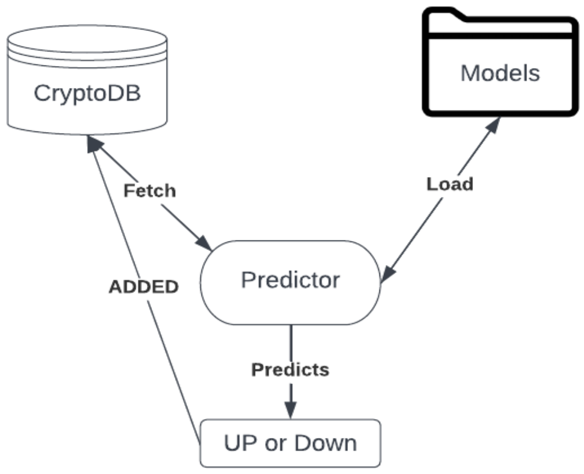


Figure 30 Predicting Process

3.2.3 Database

SQLite is one of the most used Database engines in the world and written with C language. Our CryptoDB uses SQLite engine. Figure 26 shows the database schema.

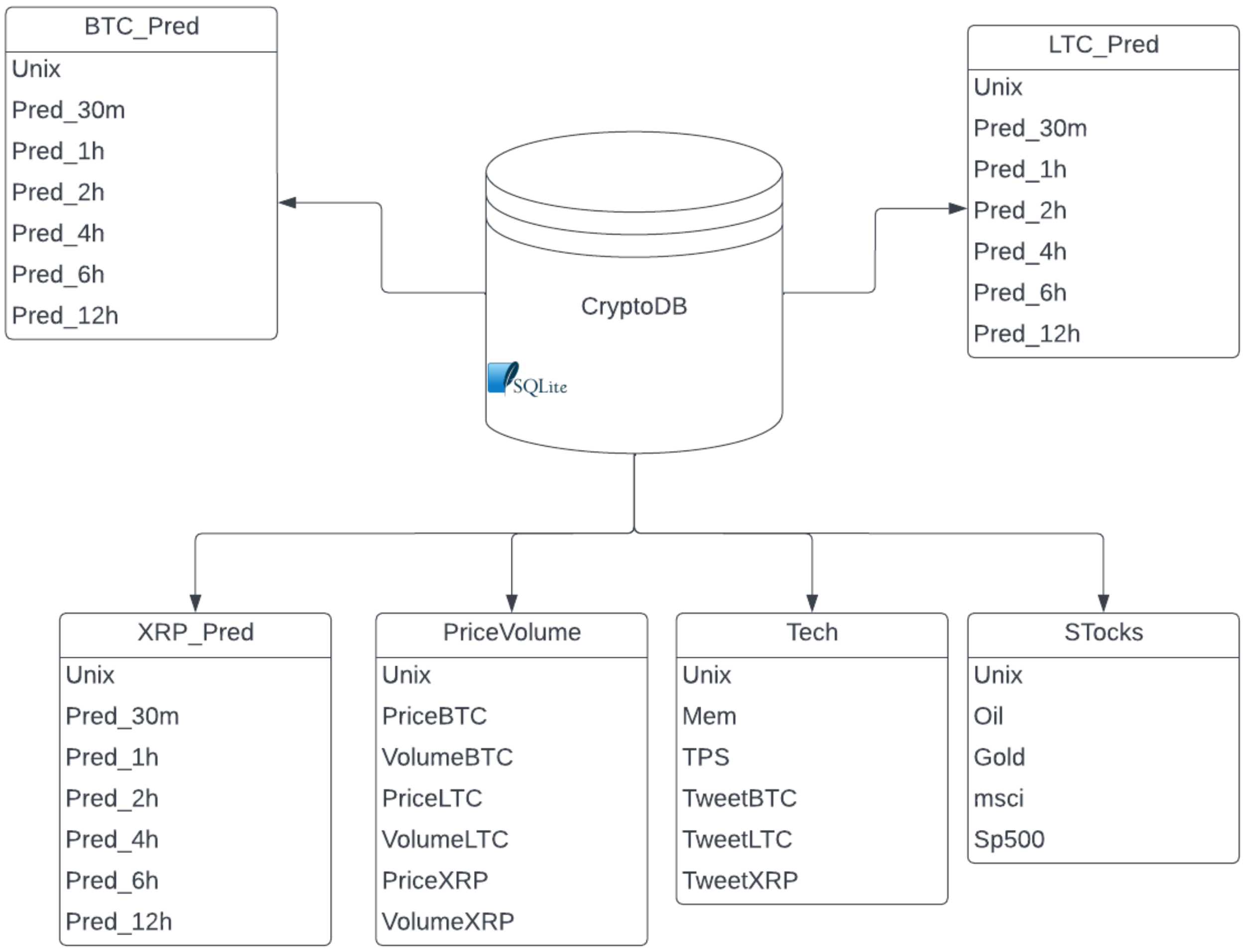
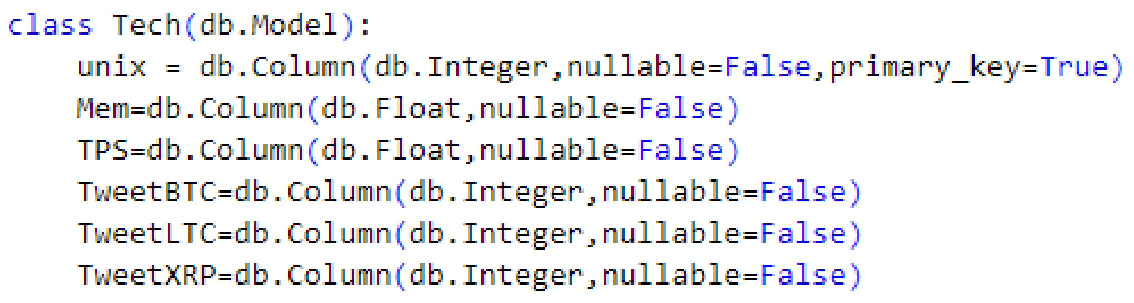


Figure 31 CryptoDB schema

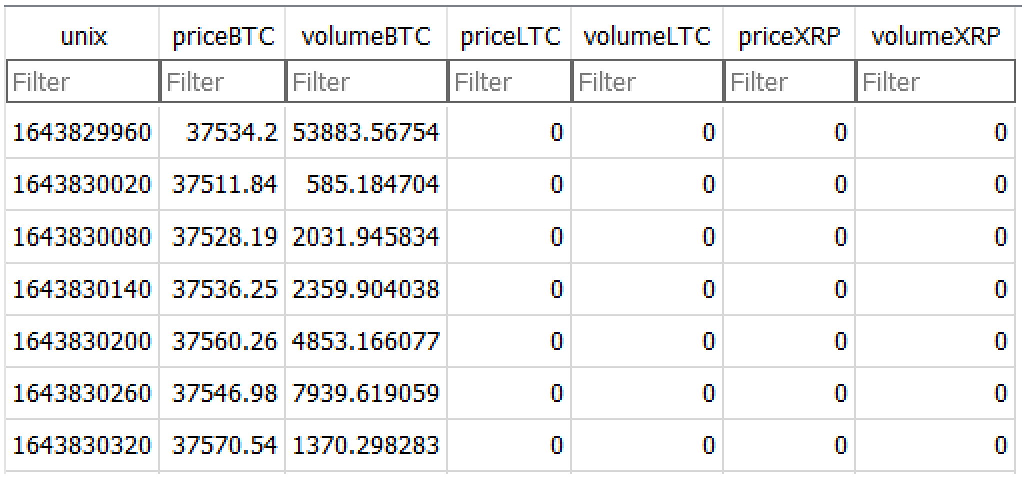
CryptoDB is split into six tables, where each table is identified by its class name as shown code snippet in figure 32. The Unix attribute is the Primary Key in each table. The columns are prevented from storing NULL values, by specifying nullable=False.



```
class Tech(db.Model):
    unix = db.Column(db.Integer,nullable=False,primary_key=True)
    Mem=db.Column(db.Float,nullable=False)
    TPS=db.Column(db.Float,nullable=False)
    TweetBTC=db.Column(db.Integer,nullable=False)
    TweetLTC=db.Column(db.Integer,nullable=False)
    TweetXRP=db.Column(db.Integer,nullable=False)
```

Figure 32 Class Tech

Figure 33 shows how PriceVolume table is saved into the database, and the attached tables inside it. See how the data is stored in cells.



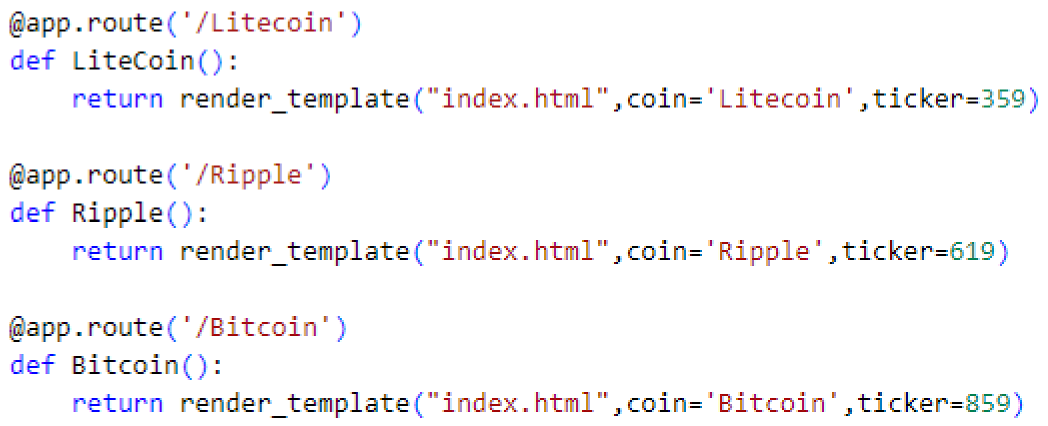
unix	priceBTC	volumeBTC	priceLTC	volumeLTC	priceXRP	volumeXRP
Filter	Filter	Filter	Filter	Filter	Filter	Filter
1643829960	37534.2	53883.56754	0	0	0	0
1643830020	37511.84	585.184704	0	0	0	0
1643830080	37528.19	2031.945834	0	0	0	0
1643830140	37536.25	2359.904038	0	0	0	0
1643830200	37560.26	4853.166077	0	0	0	0
1643830260	37546.98	7939.619059	0	0	0	0
1643830320	37570.54	1370.298283	0	0	0	0

Figure 33 PriceVolume Table

3.2.4 Server Routes

App Routing is the process of mapping URLs to a specific function that will deal with the URL's logic.

In our code we use three main routes. /Bitcoin /Litecoin /Ripple. Each route is responsible of directing the user to a web page.



```
@app.route('/Litecoin')
def LiteCoin():
    return render_template("index.html",coin='Litecoin',ticker=359)

@app.route('/Ripple')
def Ripple():
    return render_template("index.html",coin='Ripple',ticker=619)

@app.route('/Bitcoin')
def Bitcoin():
    return render_template("index.html",coin='Bitcoin',ticker=859)
```

Figure 34 Routes Functions

As shown in figure 3, every function under @app.route is considered a route, code logic is handled there too. Every function renders the HTML page for each coin.

3.2.5 Running the Application

Figure 35 shows the functions that are included in Run.py, This file is responsible of running the whole application. Firstly, it adds the needed functions to the scheduler to be scheduled on the specified times using **APScheduler** class imported from **flask_apscheduler**.

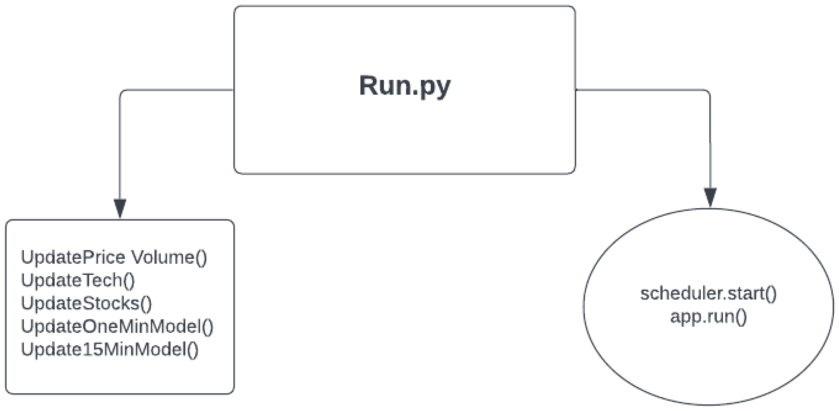


Figure 35 Run.py Functions

After that in the **main,** scheduler.start() and app.run() are called.

- scheduler.start() "This function starts the scheduler, and it runs the scheduled functions when the time is matched."

○ UpdatePriceVolume ()" scheduled to run every minute on second five."
○ UpdateTech () "scheduled to run 4 times in every hour, when the minute is exactly 0 ,15, 30, and 45."
○ UpdateStocks () "scheduled to run 4 times in every hour, when the minute is exactly 0 ,15, 30, and 45."
○ UpdateOneMinModel ()" scheduled to run every minute on second zero."
○ Update15MinModel () "scheduled to run 4 times in every hour, when the minute is exactly 0 ,15, 30, and 45."

- app.run () "Runs the flask web application on a local host server."

VI. Results

a) Summary of goals:

- We had successfully trained 16 Neural Network models.
- Developed a web application.
- Linked database with our web.
- Allowed the Data collectors to fetch the data and save them into the Database.
- Built a Dynamic webapp that's connected with the database.

b) Summary of constraints:

- Rnn models were meant to be used in our web and they had a promising accuracy, but training them took very long time, additionally tuning them needs time too. So, we couldn't rely on them entirely, that's why we used Neural Networks instead, and they went out better than we expected.
- Sentiment analysis were also meant to be used as a feature in our models training. But it has many challenges, so we didn't include it as a feature.

VII. Conclusion & Future Work

In our analytical experiments regarding models' creation and testing, we noticed that testing different algorithms can give different results, algorithms like SVM & Logistic Regression were simple to build and train but gave relatively good accuracies, around 55%-60% on 30 min-1hour. On the other hand, Feedforward Neural Networks were a bit harder to tune and train, because it required determining and tuning many variables in the model; but gave incredible accuracies on all horizons. RNNs are considered well-suited for predicting sequential events. but require a lot of time and processing power to train; also finding an effective way to structure the data for them is a tricky task, also optimizing them is tough too. Regarding the web application, It's built in a way that makes it scalable and maintainable. The main objective for it is to give insights to investors who visit the site; and give prediction insights to bots who use the API.

Future Work

Cryptocurrencies are inherently hard to predict, our market predictions and insights may and probably won't yield worthwhile profits, we already knew this going in but as the market is changing and a lot of factors may determine price movements. Our project will definitely be insightful and may contribute to any already existing trading strategies. It is challenging to enter the market with investments and expecting to make profits fast. Therefore, our web application will show users future market movements. Also, our web is built in a scalable way, it allows future additions and modifications. The creation of these models can be utilized for Trading-Bots to give them the ability to make successful transactions in the market and make noticeable profits, it can also be helpful for people who invest in this field like crypto miners.

VIII. Bibliography

[1] P. Jaquart, D. Dann, C. Martin
Machine learning for bitcoin pricing - a structured literature review
WI 2020 Proceedings, GITO Verlag (2020), pp. 174-188, 10.30844/wi_2020_b4-jaquart

[2] F. Pedregosa, G. Varoquaux, A. Gramfort, et al.
Others, scikit-learn machine learning in Python
J Mach Learn Res, 12 (2011), pp. 2825-2830

[3] Available online: https://flask.palletsprojects.com/en/2.1.x/