THE UNIVERSITY OF MELBOURNE

Cryptocurrency Analytics Based on Machine Learning

COMP90055 COMPUTING PROJECT

Software Development Project

Tzu-Tung HSIEH (818625)
Yizhou WANG (669026)
Yunqiang PU (909662)

Supervisor: Prof. Richard Sinnott

Master of Information Technology
School of computing and information systems
University of Melbourne

Semester 2, 2019

Declaration

We certify that

1. All of this project from the drafting topic to the last finishing this essay are done by ourselves without any help from the previous job or similar work; all the citation and data retrieval we took follow the University's Academic Misconduct policy; and that to the best of my knowledge and belief it does not contain any material previously published or written by another person where due reference is not made in the text.

2. We packaged all the data, code, and generated models used in this project and uploaded them with this paper to verify the independent completion of the project.

3. This thesis are 7825 words in total (excluding text in images, table, references and appendices).

Abstract

Cryptocurrency is now becoming a buzzword in recent years. As the technology of blockchain is introduced with the notion of decentralization, the financial transactions can be made through peer-to-peer networks without the third-party organization [1]. Among the several cryptocurrencies, the most representative and well known one is Bitcoin (₿), and in this project, a price prediction system is built to explore bitcoin price fluctuations and to see if there are some patterns hidden in between.

Furthermore, simply utilizing the statistical methodology isn't enough and accurate to make the predictions. To be more specific, twitter data is additionally extracted to be supportive, providing external information to improve the models. The emotions from the public and official accounts are differentiated from each other to form helpful features when forecasting the price movement.

In general, over 2.5 million collected tweets are analyzed to measure the sentiment from period to period. The sentiment change is measured in daily basis and visualized through the web application. Besides change rates in public or government sentiment, the model is able to forecast the price with root mean sum error at 451, making great progress by approximately 12 times compared to the one in the first place and at least 20% improvement after adding the sentiment features.

Key word: Cryptocurrency, Machine Learning, Sentiment Analysis, Twitter Data Analysis, Web application

Acknowledgement

We would like to express our deepest gratitude to all those who have helped us in this project.

First of all, we shall pass the greatest gratitude to the supervisor Prof. Richard Sinnott who provided us with the valuable opportunity to do this project and gave us a lot of vital help continuously through the project. We also feel grateful to all of the teachers who imparted uncountable knowledge and essential professional skills. Last but not the least, our thankfulness will go to our beloved family and friends.

Table of Content

List of Figures

List of Tables

Chapter 1 Introduction

Cryptocurrency as an emerging product continuously attracts public attention. Satoshi Nakamoto, the inventor of bitcoin argues that cryptographic proof could replace the traditional trusted third parties (financial institutions) to ensure transaction cost reduction, flexible transaction size and highly security [1]. Although it seemingly gets lots of advantages, it causes great disputes at the same time such as money laundering or illicit goods purchase [2]. In this way, it has not been recognized globally. However, it has to be admitted that cryptocurrency is the trending topic in past decades.

In this project, our group attempt to analyze cryptocurrency price fluctuations and find out the potential patterns hidden in the variation. Furthermore, tweet data are used as the informative clue helps with the price prediction. As it is commonly known that Twitter is filled with the various emotions in terms of the specific topic so any voices on it, no matter where they come from, would be an intuitive signal of future trends.

In general, our work is organized together to deliver several price predictions of cryptocurrency on Nectar. It conducts sentiment analysis to analyze the relationship between price movement and user emotions. To be more specific, tweets are fetched through Streaming and Search API and after applying the VaderSentiment, they are combined with the historical price. Then machine learning comes to the stage, a total six methods are implemented and finally demonstrated via web.

Before getting to the topic, there is a point needed to be clarified, the cryptocurrency contains not only the bitcoin but also the other ones such as ETC, ETH and etc. The reason that this project is only focus on the bitcoin is explained below.

Figure 1: Ten Cryptocurrencies Movements

All of the cryptocurrencies (Log transformation) are plotted in Figure 1 for the comparison purpose. It is interesting to notice that most of them have similar trends, particularly starting from January 2018. It is more

obvious to see from the correlation matrix. For the BTC row, 7 out of 9 have the positive correlation above 0.3 and 4 of them are above 0.6! The higher the correlation, the similar behavior it shares. In other words, these data explicitly tells that they are pretty much the same. Therefore, it motivates us to solely focus on the bitcoin.

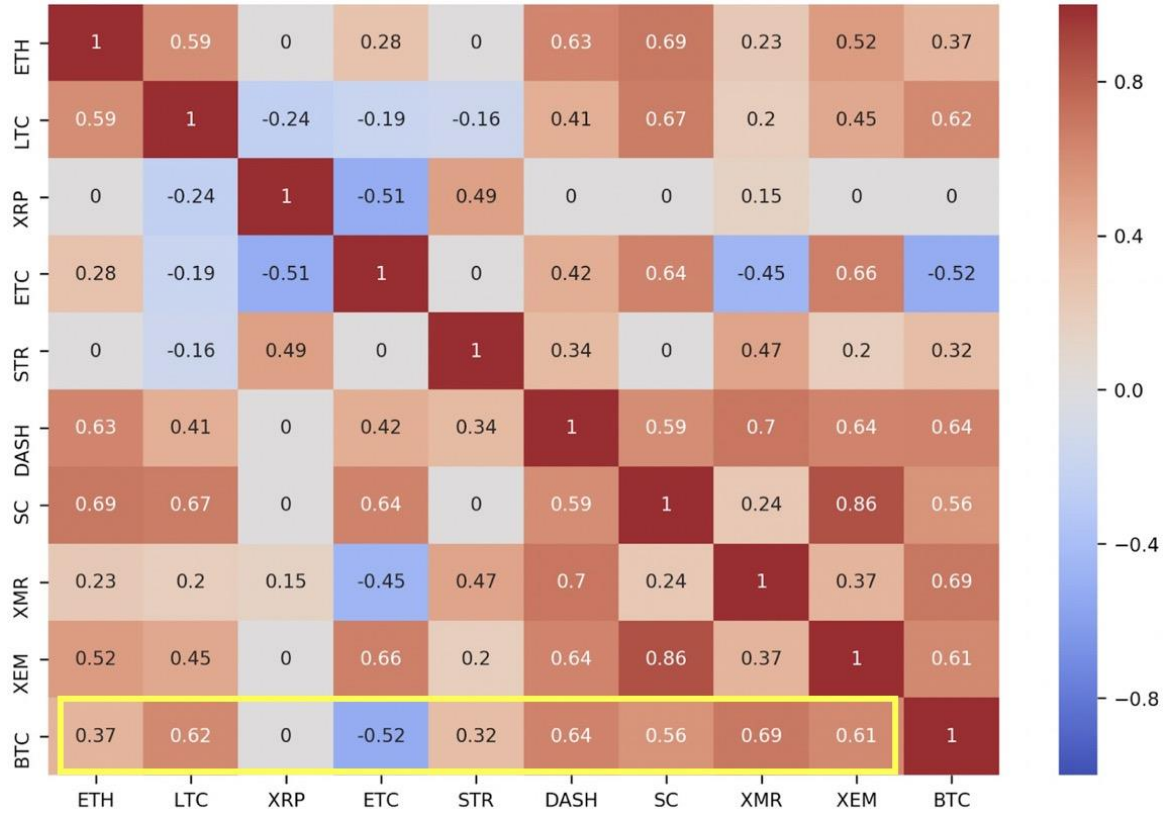

Figure 2: Correlation Matrix

This project can be divided into six parts as follows: Chapter 2 summarizes the architecture design for the system; Chapter 3 illustrates the model evaluation and metrics; Chapter 4 then talks about data analysis; Eventually, Chapter 5 and 6 mention about further improvement and making an overall conclusion.

Chapter 2 System Architecture and Design

2.1 System Layout

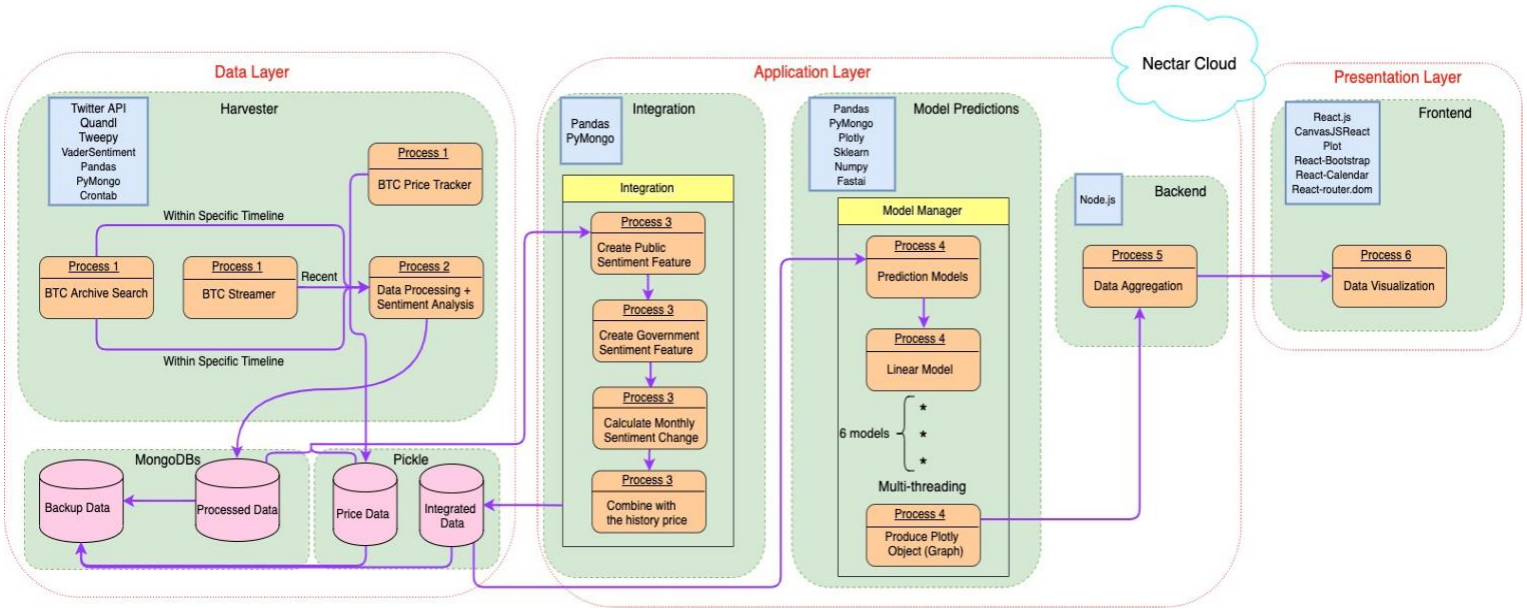

Figure 3: System Architecture

It is shown in Figure 1, the entire system is basically a three-tier model, covering the data layer, application layer and presentation layer. Each component at different hierarchies is responsible for its featured functions. To categorized, the data layer consists of search API (archive search) and streaming API. They serve totally different purposes, getting the historical and recent tweets. However, before they go into the database, they are preprocessed to remove any irrelevant information and retain 19 digits for unique indexing. To accelerate processing, VaderSentiment is applied for the sentiment analysis because it is lightweight and time-saving. Meanwhile, the price records is also extracted through quandl API and then saved into the database together. To get fast access, local pickle file is created as well as for the fault tolerance.

The general business logic is implemented by integration and model manager class. Initially, integration class retrieves price and tweet datasets, after doing the group analysis, which is to sum up the total negative, positive tweets within a specific timeline and categorized into public and government segment respectively. The details of the classification will be discussed in the next section. The integrated data will then be stored in the database and potentially be utilized in the next step, model building. Thanks to the multi-threading technique, there are six types of models running concurrently to deliver the answer, probably can save the time to some extent. Each model is also responsible for producing plotly objects in JSON format to give to the front end.

In terms of the presentation layer, it is React.js based system, which means it is not only developer, but also user-friendly for data demonstration and visualization.

The feature needs to be emphasized is that all of the employment is settled down on Nectar cloud, which means the whole system is extensible, timely updated and flexibly manipulated.

Our system would have some strengths considering the system architecture:

1. The overall design of the system architecture can significantly benefit from the independent update in stack of one tier without interfering with other areas of the application. In other words, the modular is easy to be extended and added with tons of new features.
2. It provides with the feature like automatic running and updating, which minimizes manual supervision as much as possible.
3. Encapsulation is another strength needed to be mentioned here, as user only gets access to the function demonstrated in the presentation layer, so they can't get pass through the application layer and see the data directly. It further ensures data security.
4. High fault tolerance is also achieved as not only the pickle file is produced, but also all the relevant data is backup in the MongoDBs.
5. Mostly importantly, all the components are deployed on the Nectar Cloud, which allows software to run continuously and efficiently.

2.2 Twitter Harvester

The twitter harvester is built with two distinct APIs. Search API is used to search for the tweet that meet the search criterion that already existed. While for streaming API, it crawls the one in near real-time [2].

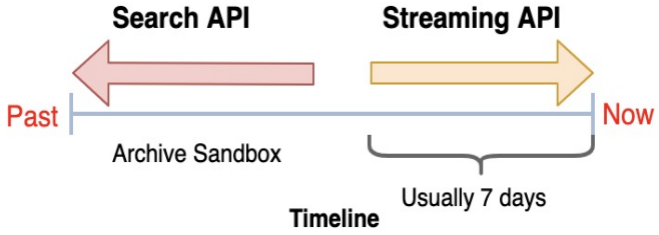

Figure 4: Twitter API Comparison

As archive search API has monthly flow control, it is then used for search twitter specifically from 2011 to 2018 (As bitcoin came to the market in 2011). For the most recent twitter data, it is crawled by streaming API starting since the end of August. Until 25th October, approximately 2.5 million tweets have been received, uniformly distributed within the timeline.

2.2.1 Search API

Search API is generally used to track the tweets posted in the specific period. All the search features are implemented with the help from another package named TwitterAPI. After utilizing the api.request function and filling in the label and product, any relevant tweets are returned that corresponds to the query and the timeline criterion.

```
1 PRODUCT = 'fullarchive'
2 LABEL = 'project'
3 r = api.request('tweets/search/%s/:%s' % (PRODUCT, LABEL),{'query': SEARCH_TERM,
4 'fromDate':'201310010000', 'toDate':'201310012359', "maxResults":100})
```

Figure 5: Code snippet for Search API

The figure above indicates the search timeline is between 01/10/2013 and 02/10/2013.
Unfortunately, the maximum archive search can be returned in a month is limited within 5k per account, so multiple twitter developer accounts have been applied to solve the problem.

2.2.2 Streaming API

Streaming API, in contrast, aims to harvest near real-time tweets that in the searching filter closely related to the bitcoin. The implementation inherits the super class in the tweepy module, known as StreamListener. Additionally, specific preprocessor functions are added in the class as well, trying to fetch and process the data simultaneously. The potential error such as rate limits or invalid requests are captured in the on_error function. The table in the next sections illustrates the main functions that have modified to serve the crawling purposes in details. For further information, to better fit for more accurate model analysis, the streaming API was almost kept running for two months. There is a hashtag list such as ‘bitcoin’, ‘cryptocurrency’ and ‘BTC’ are the keywords retrieved from both of the APIs in order to minimize the data bias and have a comprehensive coverage.

2.3 Processor

2.3.1 Preprocessor

In general, streaming returns detailed tweets but not all of the information is useful in this project, so it is recommended to reduce the redundancy and get rid of unfriendly content. There are 4 operations listed below added in the streamer class and tweets are customized into 12 key pairs to save the memory space while reading.

Function	Usage
removeWebsite()	With the help of Regex, annoying URLs that directs to the external sources are cleared.
sentimentAnalysis()	VaderSentiment scenarios to judge the tweet emotions when fetching the tweets.
addId()	Extract id from the original structure to form the unique index when inserting into MongoDBs.
on_error()	Handle total 7 errors such as authentication error or invalid request.

Table 1: Preprocessor Functions

2.3.2 Sentiment Analysis

Various texts, speeches and tweets can potentially reflect the attitudes, opinions and views of people, so the process is commonly known as sentiment analysis [3]. It is the key component built for the streaming class and closely associated with the model development in the next stage. To simplify the sentiment classification, lexicon-based is used, which basically judges the sentiment heavily based on the opinion words (the lexicon) in the dictionary [4]. Through the experiment, VaderSentiment is chosen as it tends to be the most impartial sentiment analysis system and the table below shows the results [5].

Method	Correct Predicted Positive	Correct Predicted Negative
IBM Waston	561	722
TextBlob	763	334

NLTK - VADER	710	432

Table 2: Sentiment Analysis Methodology Comparison [5]

IBM is inclined to make the correct decision on pessimistic text and TextBlob is exactly the other way around. Therefore, Vader seems to be the most suitable one for our task. Additionally, it contains proprietary dictionary of almost 4,500 words and reaches 85% classification accuracy, which further gives tons of confidence while using it [6].

```
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
sentiAnalyzer = SentimentIntensityAnalyzer()
print(sentiAnalyzer.polarity_scores('good'))
print(sentiAnalyzer.polarity_scores('bad'))
```

```
{'neg': 0.0, 'neu': 0.0, 'pos': 1.0, 'compound': 0.4404}
{'neg': 1.0, 'neu': 0.0, 'pos': 0.0, 'compound': -0.5423}
```

Figure 6: VaderSentiment Showcase

VaderSentiment evaluates the content to form four different keys. Obviously, it is understandable evidenced by the figure above. The first three demonstrates which emotions the content belongs to and compound measures the aggregate score ranging from -1 (extremely negative) to +1 (extremely positive), especially helpful to the identification in sentence level. In this way, it makes everything easier and becomes inspirational to simply treat compound as the key when doing the classification. Both streaming and search classes are using the VaderSentiment before they are saved into MongoDBs.

2.4 Historical Price Crawler

To develop the forecast model, it is crucial to get the historical price for the bitcoin. All of the price information is fetched through quandl API. The code for bitcoin is BCHARTS/BITSTAMPUSD and it retrieves the data since 2011. In our case, close price is chosen to be the predicted price.

```
import quandl
print(quandl.get('BCHARTS/BITSTAMPUSD'))
```

Date	Open	High	Low	Close	Volume (BTC)
2011-09-13	5.80	6.00	5.65	5.97	58.371382
2011-09-14	5.58	5.72	5.52	5.53	61.145984
2011-09-15	5.12	5.24	5.00	5.13	80.140795
2011-09-16	4.82	4.87	4.80	4.85	39.914007
2011-09-17	4.87	4.87	4.87	4.87	0.300000
2011-09-18	4.87	4.92	4.81	4.92	119.812800
2011-09-19	4.90	4.90	4.90	4.90	20.000000

Figure 7: Code Snippet for Price Crawler

2.5 Integration

The integration class is mainly responsible for the data manipulation and combination. First of all, it retrieves the tweet data from MongoDBs and calculates the negative and positive percentage for both the public and government sentiment. The optional other influence sentiment refers to the ones that has larger impact than normal individual user such as famous streamer but does not have authoritative power as the official twitter account. Then the sentiment statistic is resampled into daily basis to form the complete training set with price information for later model development.

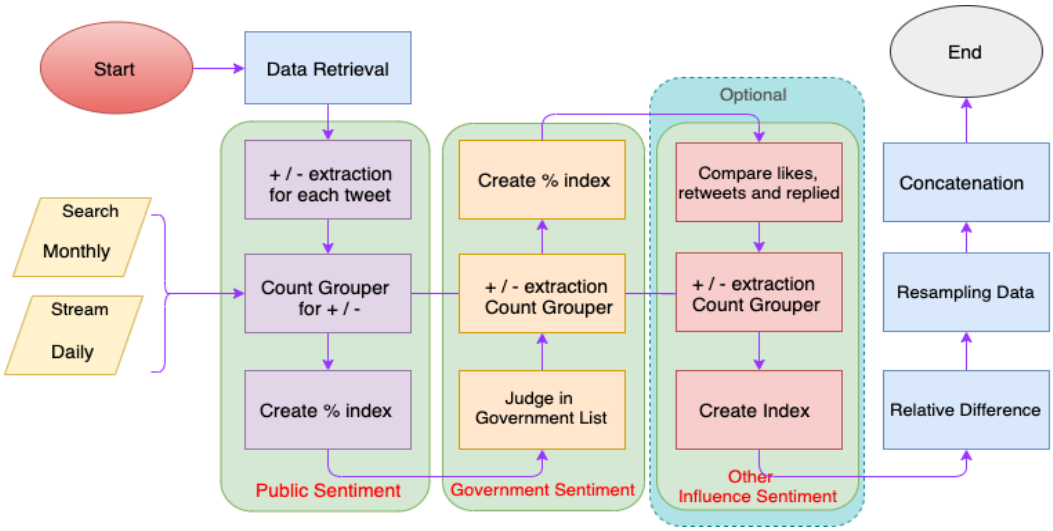

Figure 8: Integration Flowchart

It is noticed that count grouper has two options, as search API can't craw tweet in the continuous manner instead of stream API. Therefore, the tweets crawled by them has the different grouper frequency to make the data as accurate as possible

2.5.1 Sentiment Factor

1. General Sentiment

Public sentiment is basically derived from the general sentiment percentage among all of the tweet data. As each tweet instance is already labelled into positive '+', negative '-' or neutral ' ', summation for all of the categories is the only thing need to do.

```
In [8]: neu=df.groupby([pd.Grouper(key='timestamp', freq='M', label='left')])['New']
        .apply(lambda x: (x=='').sum()).reset_index(name='neu')
        pos=df.groupby([pd.Grouper(key='timestamp', freq='M', label='left')])['New']
        .apply(lambda x: (x=='+').sum()).reset_index(name='pos')
        neg=df.groupby([pd.Grouper(key='timestamp', freq='M', label='left')])['New']
        .apply(lambda x: (x=='-').sum()).reset_index(name='neg')
        total = df.groupby([pd.Grouper(key='timestamp', freq='M', label='left')]).size().reset_index(name='total')
```

Figure 9: Code Snippet for Grouper

After merging and applying the percentage calculation, the data is transformed into this format. The figure below is the showcase for the data crawled by search API (monthly)

	timestamp	neu	pos	neg	total	neu%	pos%	neg%
28	2018-05-31	113466	62634	20631	196731	0.576757	0.318374	0.104869
29	2018-06-30	20010	12202	4996	37208	0.537788	0.327940	0.134272
30	2018-07-31	82123	53822	18356	154301	0.532226	0.348812	0.118962

Figure 10: Result Showcase

2. Government Sentiment

In terms of the government sentiment, this feature is solely based on the government list selected manually. As it has no clear boundaries to verdict authoritative power so this decision is made subjectively. There are 48 international official accounts in total including Bitcoin, Coinbase, Altcoins and etc. These are generally official authorities and intuitively they possess authoritative power which could impact price in a distinctive way.

3. Other Influence Sentiment (Medium Sentiment)

While for the other influence sentiment, the classification rule is designed based on statistical information. If likes, retweets and replies are all exceed their average, then this particular tweet record can be classified as the one that has medium influence. Therefore, in our case, the judgement criterion is exactly whether likes, retweets, replies attributes are larger than 9, 8, 3 respectively. The average number is exactly calculated according to the whole dataset. The reason to put it into the optional area is due to the fact that the effects of this factor might be absorbed by the two sentiments mentioned before. It would cause the multicollinearity problem when adding this variable into the model and it is proof by the increased RMSE illustrated in the Chapter 4. However, the judgement method is still kept and maintained in the training set.

```
In [50]: x = df[df['likes']!=0]
         x['likes'].mean()
Out[50]: 8.671735569565076

In [13]: x = df[df['retweets']!=0]
         x['retweets'].mean()
Out[13]: 7.2887115696414995

In [14]: x = df[df['replies']!=0]
         x['replies'].mean()
Out[14]: 2.594119826515267
```

Figure 11: Mean for Likes, Retweets and Replies

2.5.2 Difference Calculation and Dummy Variable

It is calculated based on the difference between the previous and present period percentage. After getting the difference number, the specific function is applied to create a new index named Ispos, Isneg. These are dummy variables, representing categorical data with only two values, normally 1 and 0. 1 shows the presence of a qualitative attribute and 0 represents the absence [9]. In our context, the dummy variable Ispos is set to be 1 if this period has higher positive percentage than the last. Lastly, all the transformed data is concatenated with the price information and saved as pickle for convenient access.

```
1 history['pos_change'] = history['pos%'].diff()
2 history['neg_change'] = history['neg%'].diff()
3 history['Govpos_change'] = history['Govpos%'].diff()
4 history['Govneg_change'] = history['Govneg%'].diff()
```

Figure 12: Code Snippet for Comparative Difference

2.5.3 Final Integration and Demonstration

This step is to combine the price with the tweet information. The single instance from the integrated data is shown below to provide better understanding for this section.

Attribute	Value	Explanation
Open	8586.57	Open price
High	8666.84	The highest price in a day
Low	8450.61	The lowest price in a day
Close	8582.11	Close price
Volume(BTC)	7435.432	Volume in BTC unit
Volume(currency)	63608562.98	Volume in USD unit
Weighted Price	8555.79	Weighted price
Neg	121	Negative tweet count

Neg%	0.1249	Negative/Total
Pos	324	Positive tweet count
Pos%	0.3344	Positive/Total
Neu	524	Neutral tweet count
Neu%	0.5408	Neutral/Total
Total	969	Total tweet count
MInfluence	12	Total tweets which can be from streamer
GovNeg	23	Government negative tweet
GovNeg%	0.2987	Government negative/Gov total
GovPos	37	Government positive tweet
GovPos%	0.4805	Government pegative/Gov total
GovTotal	77	Government total tweet
Pos_change	0.0234	Positive tweet % difference from last day
Neg_change	-0.042	Negative tweet % difference from last day
GovPos_change	0.013	Gov positive tweet % difference from last day
GovNeg_change	-0.028	Gov negative tweet % difference from last day
Ispos	1	Pos_change larger than 0, yes 1, no 0
Isneg	0	Neg_change larger than 0
GovIspos	1	GovPos_change larger than 0, yes 1, no 0
GovIsneg	0	GovNeg_change larger than 0
Timestamp	2019-10-10T00:00:00Z	Time index

Table 3: Integration Data

2.5.4 Automatic Integration

As the whole system is running on a daily basis, it is necessary to implement timer to get the automatic updated. In general, Crontab Linux command line is used to set up the job running frequently. For easier implementation, the whole project is set in the virtual environment on Nectar to keep the environment variables consistent. The schedule time for the price crawler and integration is both at midnight in order to enhance the user experience.

2.6 Mongo Database

MongoDB has been chosen to our database to store raw tweet data. It is a documented-oriented database which basically in BSON format (JSON in binary) [10]. Considering the large size of tweet data, it has several advantages over traditional database such as strong scalability, schema less and deep query-ability.

There are two main types of collections in this project, covering the data fetched by search API and streaming API. They are separately existed because of their distinctive structures.

Timeline	Collection Name	Document Count	Explanation
Past Archive	Twitter_collection_14	5,500	They contain the tweet data from 2011 to 2019
	Twitter_collection_2016_2018	1,851,563	
	Twitter_collection_5_8_19	55,300	
Recent	Twitter_collection	984,591	It collects this semester's tweets crawled by streaming

Table 4: Mongo Database

Figure 13: Bitcoin

The reason to get so much tweet data between 2016 to 2018 than before this period is that the trend tells us there was a dramatic change. If there is no much fluctuations, it is reasonable to believe there is no much discussion as well. Therefore, this project focuses on data crawling starting from 2016.

Specific tweet structures are given below for better illustration.

Streaming API		Search API	
Key	**Description**	**Key**	**Description**
_id	Unique index	_id	Unique index
Created_at	Create time	user	username
Text	Tweet text	fullname	Account name

User: name	Account name	timestamp	Time index
reply_count	Reply no.	url	weblink
retweet_count	Retweet no.	likes	Like no.
favourite_count	Like no.	replies	Retweet no.
lang	Language	retweets	Retweet no.
timestamp_ms	Time index	text	Tweet text
Sentiment	Emotion score	Sentiment	Emotion score

Table 5: Tweet Structure for Streaming and Search

As we can notice that there are several differences existing between them. Firstly, the key word for likes, replies and retweets are different so the corresponding conditions must be adjusted in order to get the extraction. Then it comes to the name format, the one provided in search API is pretty straightforward instead of nested dictionary in streaming. Lastly, timestamps are in distinctive formats. To be more specific, one is 1572251561040 in milliseconds and the other one is standardized 2016-06-25 23:57:44. In this way, the transformation function datetime.fromtimestamp(float(row) / 1000.0) is applied.

2.7 Model Structure

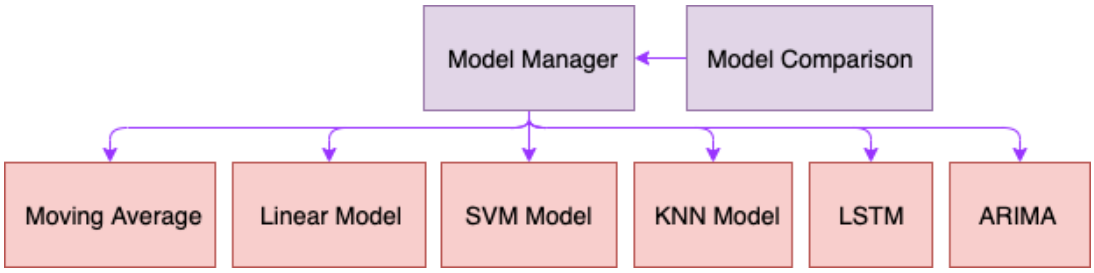

Figure 14: Model Structure

The whole structure is designed on the concept of multi-threading. The technology can simultaneously run six classes, saving a lot of processing time. Model manager plays the role of manager which to control six models as they have independent modules and data structures. As soon as these classes finish the job, Thread.join() is called to return the model statistics for further analysis. Finally, model comparison use the data produced by multiple threads to create a comparison table and saved into JSON format. The detailed model implementation is discussed in the next chapter.

2.8 Web Application

In order to achieve user-friendly environment, after data retrieval, our project uses the website through backend and frontend to present the analyzed result. In the presentation layer, React is utilized to build the project website. Firstly, according to StackOverflow's 2018 surveys [11], React has become one of the most prevalent applications in frontend framework. For instance, public companies such as Facebook, Netflix, BBC have been using React for years ago. Secondly, React is built with Javascript, CSS, HTML, so our team members can learn more from its variety. Lastly, React is embedded with the render function, it can speed up the presented efficiency. In the following section, this report will introduce the structure in the demonstrated order, involving backend

respects to how to make connection with data; frontend respects to the reaction between various components; and UI design.

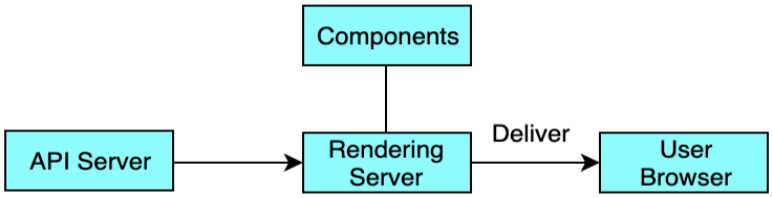

Figure 15: Structure of React Server

```
import React from "react";
import ReactDOM from "react-dom";
import App from "./containers/App";
import * as serviceWorker from "./serviceWorker";

ReactDOM.render(<App/> , document.getElementById('root'));
```

Figure 16: Code Snippet for render function

The concept referring to the website server is called server-side rendering, indicating the corresponding HTML would be mounted into browser. Figure 15 indicates that API server applies ReactDom packet, and it comes with the render function in server to deliver all components located in frontend to user's browser.

2.8.1 Backend Design

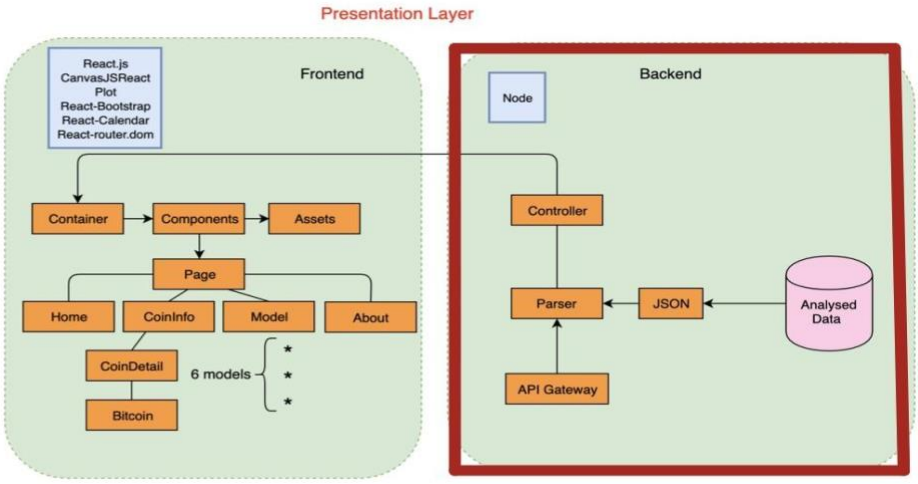

Figure 17: Presentation Layer - Backend

The presentation layer is composed of frontend and backend. In the backend, this project utilizes Node.js to parse API and deliver the data to frontend. The package.json shows the responsibilities of Node.js, and the critical functions will be explained in the following.

- Key name: dependencies
  Functions: It lists all of the APIs used in the website, such as bootstrap, plotly, and canvas.
- Key name: scripts

Functions: Npm package stalled makes easier for operator manipulation and web server controlling. For example, just inputting the command “npm run” in the terminal, API server will be built and started immediately.

Furthermore, React is good at displaying via browsers but it is difficult to handle the data processing. Therefore, our project delivers JSON file through Node.js to frontend.

```
"dependencies": {
  "@progress/kendo-date-math": "^1.5.0",
  "@progress/kendo-react-common": "^3.6.0",
  "@progress/kendo-react-dateinputs": "^3.6.0",
  "@progress/kendo-react-dropdowns": "^3.6.0",
  "@progress/kendo-react-intl": "^3.6.0",
  "@progress/kendo-react-tooltip": "^3.6.0",
  "axios": "^0.19.0",
  "bootstrap": "^4.3.1",
  "canvasjs": "^1.8.1",
  "components": "^0.1.0",
  "luxon": "^1.19.3",
  "moment": "^2.24.0",
  "normalize.css": "^8.0.1",
  "plotly.js": "^1.49.5",
  "react": "^16.9.0",
  "react-bootstrap": "^1.0.0-beta.12",
  "react-calendar": "^2.19.2",
  "react-dom": "^16.9.0",
  "react-plotly.js": "^2.3.0",
  "react-router-dom": "^5.0.1",
  "react-scripts": "3.1.1",
  "styled": "^1.0.0",
  "styled-components": "^4.3.2",
  "update": "^0.7.4"
},
```

```
"scripts": {
  "precommit": "pretty-quick --staged",
  "start": "react-scripts start",
  "build": "react-scripts build",
  "test": "react-scripts test",
  "eject": "react-scripts eject",
  "flow": "flow",
  "update-flowtyped": "flow-typed update"
},
```

Figure 18: Code Snippet for Dependencies and Scripts

2.8.2 Frontend Design

When API server started, it would call rendering server to present all components from exported modules, which is dominated by React.js. The advantage of React is that it can use render function to update data automatically. In addition, React consists of various independent components and it can establish complicated user interface via JavaScript and CSS. Therefore, each component achieves high reusability, and the code in frontend does not need to repeat.

The structure of the front side is divided into three parts, which are container, components, and assets.

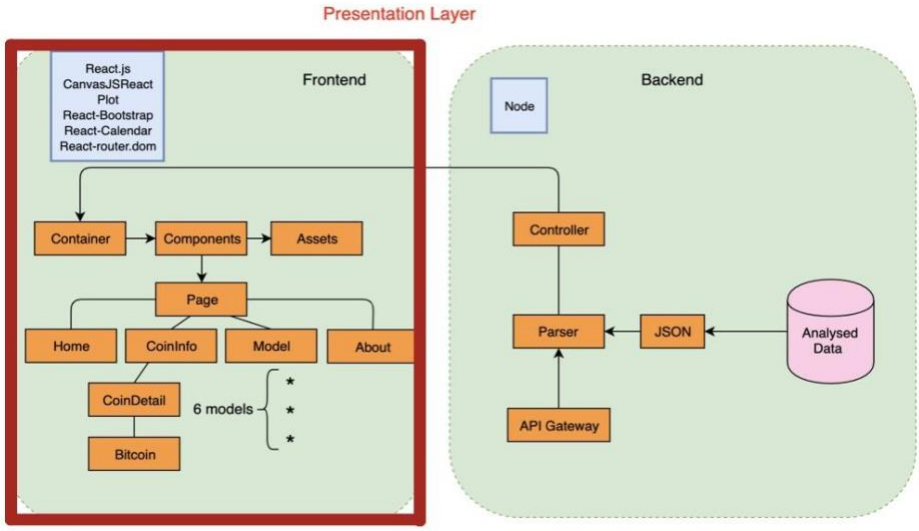

Figure 19: Presentation Layer - Frontend

1 Container

The container is the entrance of frontend, and it provides the link address for each page. Each page has become the components in the container, so it includes 20 components (pages).

Figure 20: Code Snippet for Container

```
<React.Fragment>
  <NavigationBar />
  <Jumbotron />
  <Layout>
    <Router>
      <Switch>
        <Route exact path='/' component={Home}/>
        <Route exact path='/coinInfo' component={CoinInfo}/>
        <Route exact path='/model' component={Model}/>
        <Route exact path='/about' component={About}/>
        <Route exact path="/detail/bitcoin" component={Bitcoin} />
        <Route exact path="/detail/bitcoin/twitter" component={BitcoinTwi} />
        <Route exact path="/model/KNN" component={KNN} />
        <Route exact path="/model/SVM" component={SVM} />
        <Route exact path="/model/Linear" component={Linear} />
        <Route exact path="/model/MovingAverage" component={MovingAverage} />
        <Route exact path="/model/ARIMA" component={ARIMA} />
        <Route exact path="/model/LSTM30" component={LSTM30} />
        <Route exact path="/model/LSTM/LSTM60" component={LSTM60} />
        <Route exact path="/model/LSTM/LSTM90" component={LSTM90} />
        <Route exact path="/model/KNN/KNNTwitter" component={KNNTwitter} />
        <Route exact path="/model/SVM/SVMTwitter" component={SVMTwitter} />
        <Route exact path="/model/Linear/LinearTwitter" component={LinearTwitter} />
        <Route exact path="/model/KNN/KNNGov" component={KNNGov} />
        <Route exact path="/model/SVM/SVMGov" component={SVMGov} />
        <Route exact path="/model/Linear/LinearGov" component={LinearGov} />
      </Switch>
    </Router>
  </Layout>
</React.Fragment>
```

2 Components

The component not only contains each page presented on the website, but also includes the element that is utilized in each page. For example, table is responsible for listing the sentiment variation, Jumbotron can be applied in each page to adjust the page exchange.

3 Assets

Assets store all figures that is used in the project evidenced by portfolio, model predictions.

2.8.3 JavaScript Libraries

React can embed significant JavaScript Libraries for building user interface, which specifies the framework and layout. Next part the libraries that have adapted in our project will be introduced.

1 React.js

React depends on JavaScript to support visualization, which provides flexible writing styles to combine with JavaScript, CSS, and HTML. When the developer needs to construct a complicated application, they can rely on the critical benefit of React components. Therefore, all presented pages in our project inherits React Component architecture.

```
export class CoinInfo extends React.Component{
    constructor(props){
        super(props);
        this._onButtonClick = this._onButtonClick.bind(this);
    }
```

Figure 21: Code Snippet for Container

2 React-router.dom

This function is called a Router, it is switched from React-router.dom to cover all page components in the container file and it supports for page switch function, and trace for every address used in our web application.

3 React-Bootstrap

In the beginning, all of the traditional layouts, buttons, and tables are designed by CSS. However, React-Bootstrap plays a more important role, supplying more efficient and modern way to establish the UI design. It does not need the dependencies like jQuery, so the programmer can follow the structure of Bootstrap to adjust the requirements easily. This project adapts the components from React-Bootstrap: Layout, Button, Card, and Table.

Figure 22: Code Snippet for Original Table

```
const Table = ({ columns, data }: Prop) => {
    const headerColumns = () =>
        Object.keys(columns).map(key => (
            <Th align={columns[key].align} width={columns[key].width}>
                {columns[key].label}
            </Th>
        ));
    const cell = (item, key) =>
        <Td>
            {item[key]}
        </Td>
    const row = (item) =>
        <Tr>
            {Object.keys(columns).map(key => cell(item, key))}
        </Tr>;

    return(
        <T>
            <thead>
                <tr>{headerColumns()}</tr>
            </thead>
            <tbody>
                {data.map(item => row(item))}
            </tbody>
        </T>
    );
}

export default Table
```

```
type Prop = {
    columns: {
        [key: string]: {
            key: string,
            label?: string,
            align?: "center" | "right" | "left",
            width?: number,
        }
    },
    data: Array<Object>
};

const T = styled.table`
    width: 100%;
`;

const Th = styled.th`
    color: ${textColor};
    font-size: 18px;
    font-weight: 700;
    text-align: ${props => props.align ? props.align : 'left'};
    width: ${props => props.width && `${props.width}px`}
`;

const Tr = styled.tr`

`;

const Td = styled.td`

`;
```

```
<Table striped bordered hover variant="dark">
    <thead>
        <tr align="center">
            <th>Name</th>
            <th>Period</th>
            <th>Twitter Variation</th>
            <th>Government Variation</th>
        </tr>
    </thead>
    <tbody>
        <tr align="center">
            <td>
                <Button  variant="outline-warning">
                    <a href={'/detail/Bitcoin'} >Bitcoin</a>
                </Button></td>
            <td>Today</td>
            <td>{this.todayTwiVariation()}</td>
            <td>{this.todayGovVariation()}</td>
        </tr>
        <tr align="center">
        <td>
        </td>
            <td>One Week</td>
            <td>
                {this.oneWTwiVariation()}
            </td>
            <td>{this.oneWGovVariation()}</td>
        </tr>
    </tbody>
</Table>
```

Figure 23: Code Snippet for Bootstrap Table

The figure 21 indicates the original code that the developer needs to complete a table, but the figure 22 applies Bootstrap structure to reduce inherited actions as much as possible.

4 React-Plotly

React-plotly.js is a library that is commonly used to present graph, analysis, and statistics, and it is based on Python Django framework [12]. Recently, plotly begins to support the popular frameworks such as React. When plotly applies in React, the JSON format contains chart type, data and style are transferred through React component. That is the technique has been used in our project to draw price movement, represent different models, and demonstrate sentiment variation.

5 CanvasJSReact

CanvasJS is introduced by Apple company in 2004, and it defines drawable region in HTML code [13]. It uses JavaScript code to access the drawing functions. In order to get access to the data and style, CanvasJSReact is a feasible way for plotly to provide components properties in JSON format. CanvasJSReact is intensively applied in sentiment variation, sentiment proportion, and model comparison sections.

6 React-Calendar and moment

There are many types of calendars that are support for React, and the simplest style named React-Calendar API is chosen. The other reason is that this type of calendar supplies various properties, such as onChange, returnValue, and view. This project requires users to pick the specific date to show the relative sentiment variation, or price variation via calendar. Therefore, just inheriting these functions can easily parsed the date value and even possible for the different types of analyzed data.

2.9 User Interface Design

Figure 24: Code Snippet for Original Table

Figure 22 presents the Navigation Bar of website, and these four components named Home, List, Model, and About are packaged as a component Navigation Bar applied in each page. Therefore, each page can get access to other tabs easily.

1 Home

Figure 25: Snapshot for Home Page

Home page contains a line chart listing ten kinds of cryptocurrencies, and it is implemented by plotly. The user has the choice to zoom into the specific timeline. Additionally, it provides options like double clicking on the particular one type of the cryptocurrency, the result will present as the figure 24.

Figure 26: Snapshot for Double Clicking

2 List

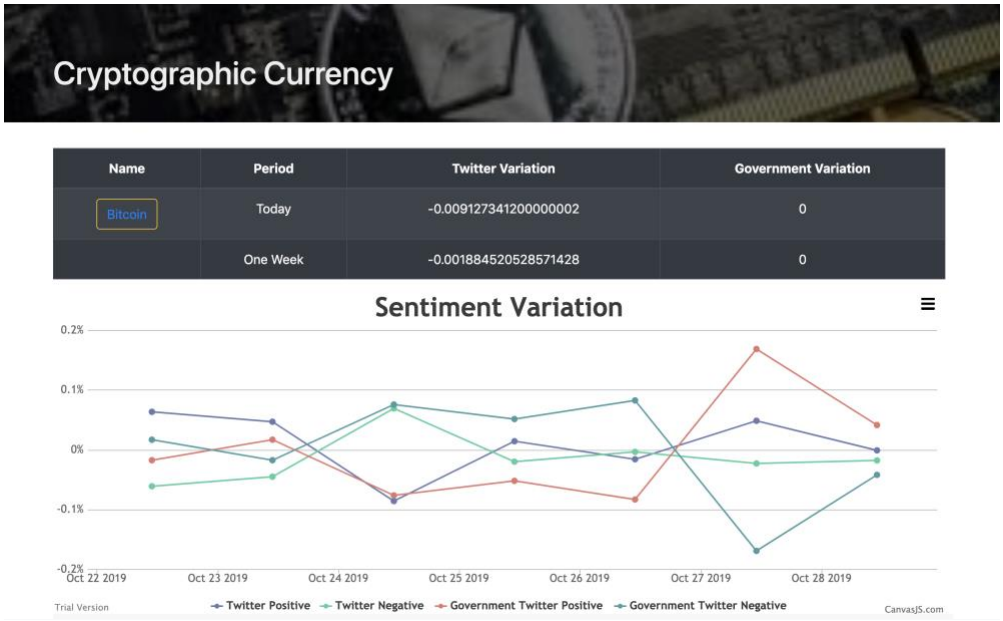

Figure 27: Snapshot for List Page

List page demonstrates today and one week variation in public and government sentiment in order to provide timely information for any users. The line chart below uses canvas component to illustrate one week variation in the positive and negative perspectives.

3 Coin Detail (Bitcoin)

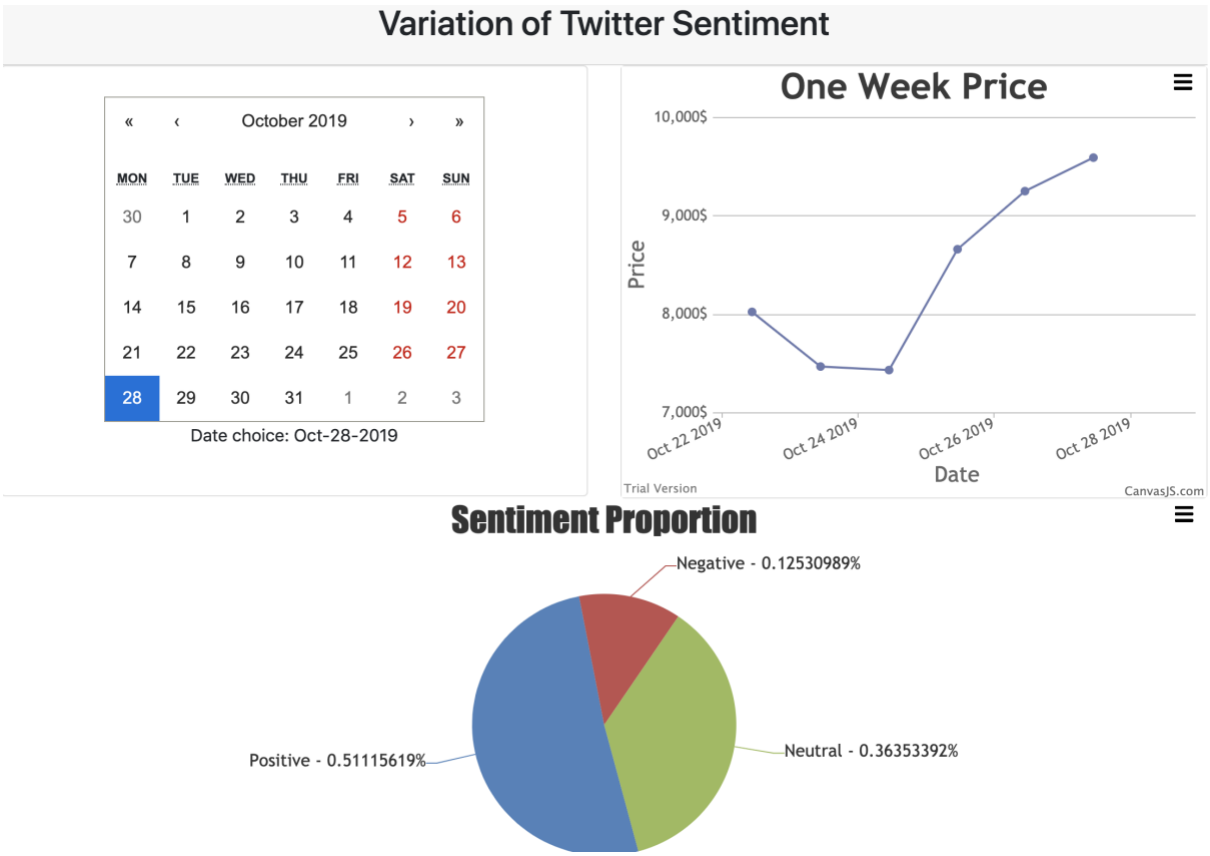

Figure 28: Snapshot for Bitcoin Page

After selecting the date, one week price movement and the pie chart will be popped out. This page utilizes react-calendar and canvas components to achieve the goals. There are nearly one-month's data available for the user to get access to.

4 Model

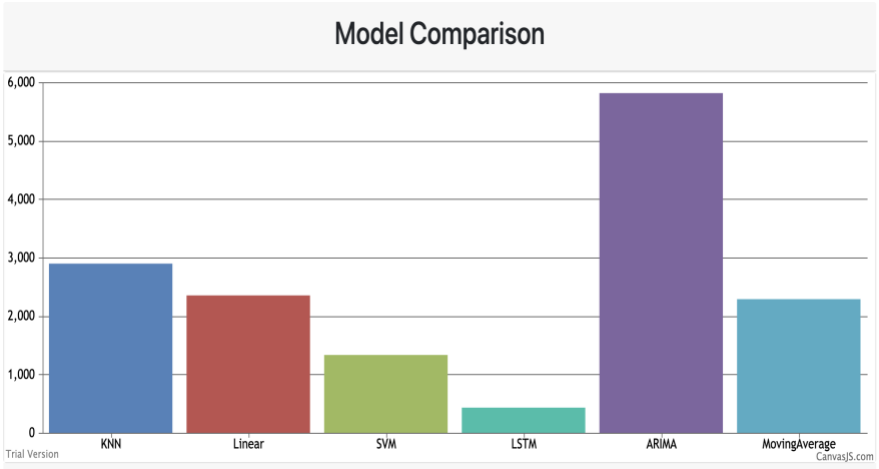

Figure 29: Snapshot for Model Comparison

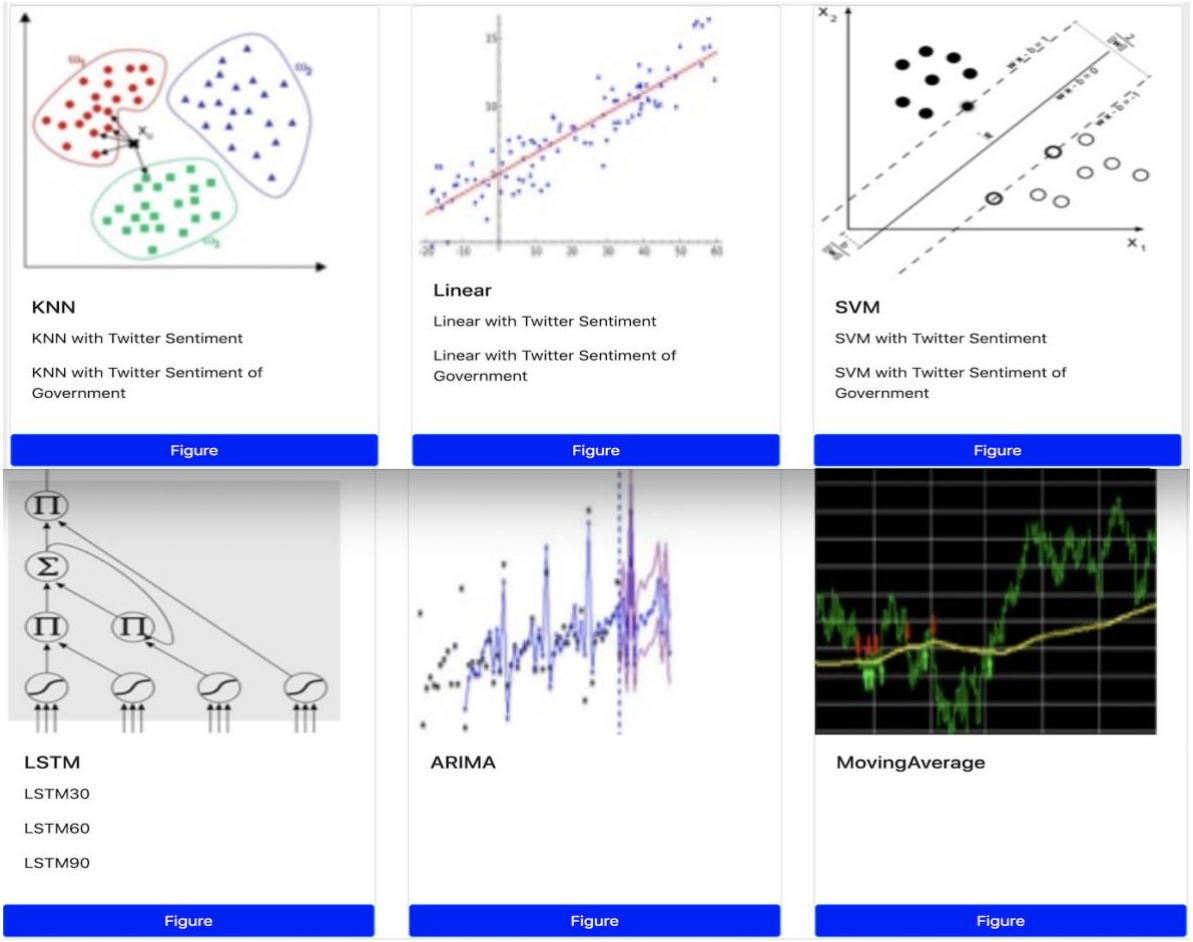

Figure 30: Snapshot for Distinct Models

Figure 27 and Figure 28 are contained in the Model Page. The model comparison points out LSTM (Long Short Term Memory) is the most accurate model. As they are in RMSE (Root Mean Squared Error) unit, the lower the RMSE, the better the model is. In addition, this bar chart is derived from the canvas component, and the layout in Figure 28 is applied in the card deck from bootstrap style.

5 Model Detail (KNN)

There are six different models experimented in the project, KNN model is used as an example to introduce the model page. The line chart generated via plotly demonstrates the variation referring to history, actual, and etc. The information tab below uses bootstrap card deck to fix the position. The button on the right hand side can change the data source, showing the result of public or government sentiments.

Figure 31: Snapshot for Model Detail (KNN)

6 About

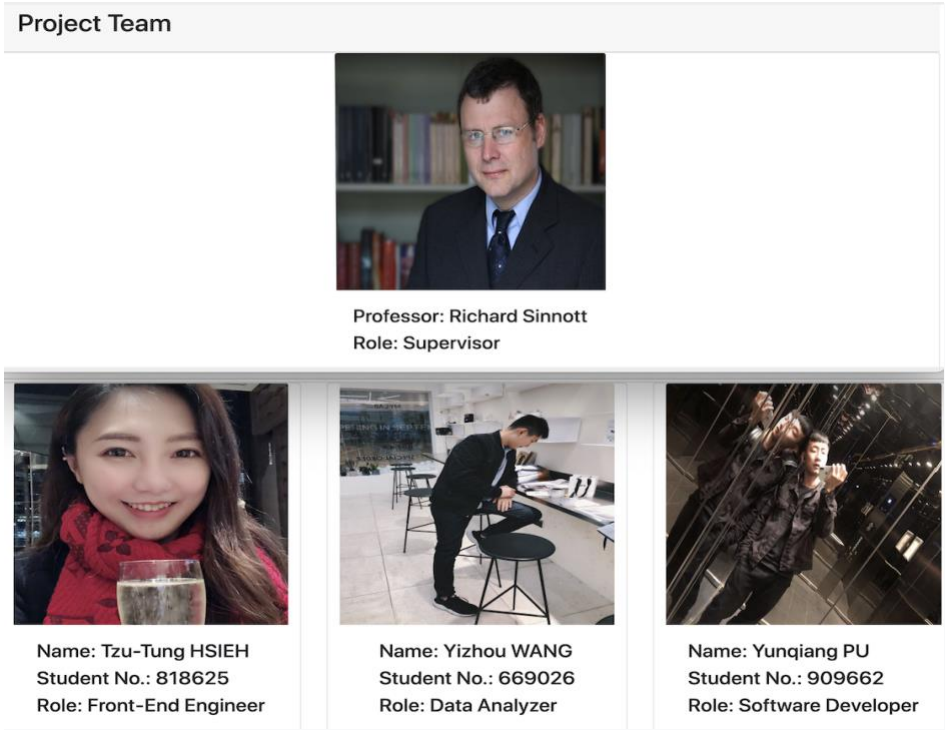

Figure 32: Snapshot for About us

About page is made up of a card desk from bootstrap style, it introduces the project supervisor and team members.

Chapter 3 Model Exploration and Metrics

3.1 Model Evaluation Metrics and Explanation

The goodness of the model is explained by two metrics through the model development.

First is more widely used, known as RMSE (root-mean-square-error). It quantifies the prediction errors as the standard deviation of the residuals. The smaller the RMSE, the more accurate the model is. This tool is particularly sensitive to any outliers and tends to show more accurate comparison from model to model.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(Predicted_i - Actual_i)^2}{n}}$$

Second one is called MAPE (mean-absolute-percentage-error). It considers the error percentage but it can't reflect the variance of the dataset and have a difficult time to deal with the zero in the denominator term. Therefore, the critical evaluation metric mainly uses RMSE and the table in Chapter 4 further proof the point mentioned here.

$$MAPE = \frac{100\%}{n}\sum_{i=1}^{N}\frac{Actual_i - Predicted_i}{Actual_i}$$

The model development used total data as the training set, and 20% of it is regarded as the test set. Several models used in this project have different tendencies. For instance, moving average is a typical mathematical model, which only considers the data relationship in the sense of arithmetic. While taking the linear model as an example, it is more extensible, features are easier to be appended and understand. Therefore, various types of models are tried to give different flavors. It needs to point out that twitter dummy variables created before (Isneg, GovIspos) are mainly implemented in last three methods due to the reason clarified just now.

3.1.1 Moving Average

The moving average method is one of the most popular tools in technical analysis. For further information, technical analysis uniquely employs the geometry and pattern recognition, trying to compete market [14]. It uses average price over some period of time to filter out the noise. To simplify the calculation, simple moving method is applied and the formula is shown below:

$$SMA = \frac{A_1 + A_2 + \cdots + A_n}{n}$$

Where A takes account for the price in the specific period and n indicates that there are n number of periods.

To better present the theory behind this methodology, the figure below explicitly show the relationship between observed and predicted dataset.

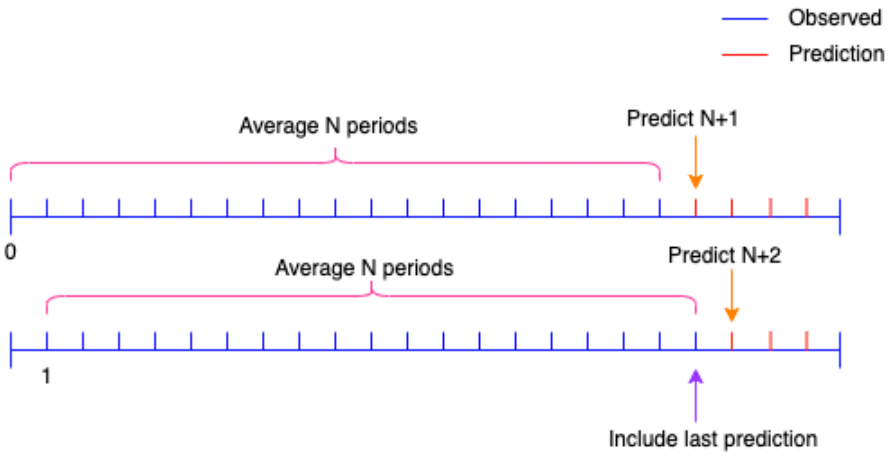

Figure 33: Moving Average

3.1.2 Autoregressive Integrated Moving Average

It is a traditional statistical model, the full name is autoregressive integrated moving average model. In the perspective of its name, it can be noticed that this is the model which combines the AR and MA methods. Integrated indicates it uses the difference between the present and previous value in order to achieve stationarity, which is an extremely important assumption in time series analysis. AR says that the present value is explained by its own lagged values while for MA, it uses the linear combination of error terms to get regression error. The flexibility this model has can easily transform into AR or MA when changing the coefficient to (1, 0, 0) and (0, 0, 1) respectively.

$$ARIMA(p,d,q) = \left(1 - \sum_{i=1}^{p} \emptyset_i L^i\right)(1-L)^d X_t = (1 + \sum_{i=1}^{q} \theta_i L^i)\varepsilon_t$$

3.1.3 Long Short Term Memory

In terms of the LSTM, it is a totally different story compared to the ones mentioned before. It is a recurrent neural network but extended with long term dependencies, offering a robust performance in long-term sequential time series [15]. Thanks to the sophisticated layers that could transform the data into the necessary keys to produce the model. Basically, the trick to behave well is that LSTM is capable of storing the past information that is important and wipe out the one that is not. It is achieved by three characteristics, forget gate to selectively abandon useless information; input gate to add valuable messages; and cell state to output state statistics. In this way, sequential data keep coming to establish long short term memory model.

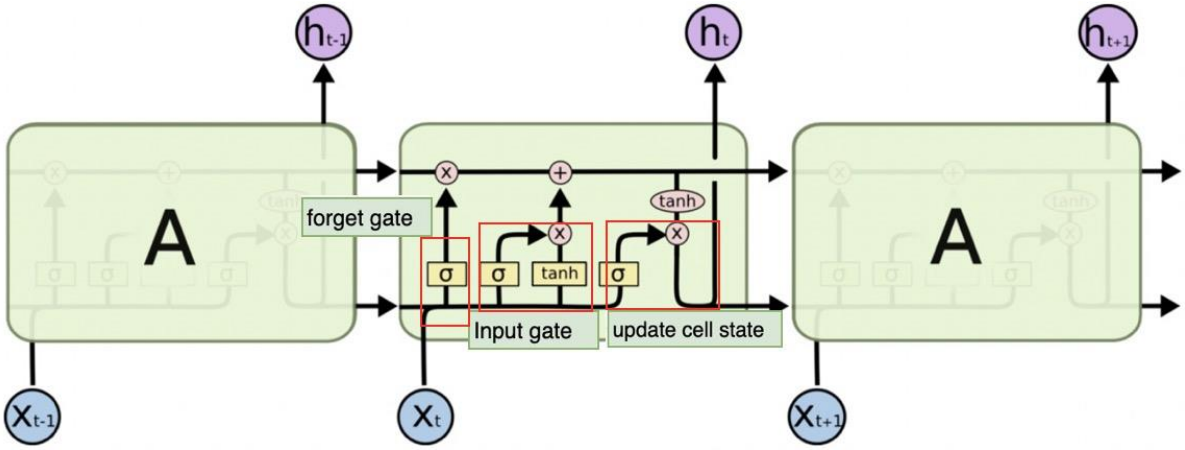

Figure 34: LSTM Architecture

3.1.4 Linear Regression

Linear Regression is a straightforward method. The formula is provided below and the aim is to get the weight matrix for all of the variables.

$$y = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n + \varepsilon_t$$

Although it is too simple to fit the data, it is undeniable that it still delivers acceptable results within short time. In particular, it can be scaled up to certain orders or appended with multiple variables. Unlike previous methods, date as the only variable is not enough or even informative, so Fastai python module comes to the stage in order to map 12 new features in total such as 'Year', 'Month', 'Week', 'Day' etc. shown below.

timestamp	Close	Year	Month	Week	Day	Dayofweek	Dayofyear
2011-09-13	5.97	2011	9	37	13	1	256
2011-09-14	5.53	2011	9	37	14	2	257
2011-09-15	5.13	2011	9	37	15	3	258
2011-09-16	4.85	2011	9	37	16	4	259
2011-09-17	4.87	2011	9	37	17	5	260
2011-09-18	4.92	2011	9	37	18	6	261
2011-09-19	4.9	2011	9	38	19	0	262

Figure 35: Fastai Transformation

3.1.5 K-nearest neighbors

KNN algorithm's idea is to label the instance by its K nearest neighbors according to the distance. It has strengths in processing time, robust prediction and even comfortable with the classes that are not linearly separable [16]. Although KNN is generally applied in the classification problem, there are some insights showing that it can be applied in the time series problem as well [17]. K is the parameter specifically selected by GridSearch, ranging from 2 to 9. After five times cross validation, K is chosen with much higher confidence. In addition, the data used for KNN classifier is also transformed by Fastai.

```
1 params = {'n_neighbors': [2, 3, 4, 5, 6, 7, 8, 9]}
2 knn = neighbors.KNeighborsRegressor()
3 model1 = GridSearchCV(knn, params, cv=5)
4 model1.fit(x_train, y_train)
```

Figure 36: K GridSearch

3.1.6 Support Vector Machine

Support vector machine is known as another classic machine learning algorithm. It has powerful prediction ability in many areas. The aim is to find out the optimal hyperplane with the maximum margin to classify the data [17]. Instead of doing the GridSearch to find the best parameter, the penalty factor C is set to 1e3 and the kernel is customized into radial basis function to fit the training purpose according to the empirical evidence. The reason behind it that it will take a much longer time apparently. Similarly, Fastai transformation has been added as well to keep it consistent, especially necessary when doing the comparison in the later stage.

3.2 Possible Prediction

Prediction is not produced by all of the methods introduced in this section because the reason is mentioned before: different methods have different tendencies. The last three models are added with prediction functions. In addition, the maximum forecast timeline is set up to 1 year, providing multiple choices for the user such as daily, weekly, monthly etc. Together with the price change, 90% confidence boundary line are given as well, in order to be informative as much as possible. Furthermore, only the model with the smallest RMSE is implemented with prediction function as it has the most accurate predictive power compared to others. However, it is hard to make assumption that the tweet use crawled by now could present the emotion tomorrow, so the prediction is only reasonable within the specific timeline.

Figure 37: Prediction in 90% Confidence

3.3 Model Automatic

Model updated is also executed on a daily basis, sharing the similar update pattern with the integration class. The implementation also thanks to the Linux Crontab by simply adding all the class needed to be updated automatically in the sh command document.

Chapter 4 Data analysis

4.1 Model Accuracy

In this section, model accuracy is specifically evaluated with the two metrics mentioned before. Let's take a close look at the data related to the model.

Methods (Best)	RMSE	MAPE
Linear Model	2349.14	69.592%
KNN	3012.42	69.565%
SVM	1369.97	69.514%
Moving Average	2295.32	69.562%
ARIMA	5242.28	69.596%
LSTM	451.63	68.213%

Table 6: All Six Models Evaluation

It may not obvious to see the differences but after plugging the data into the bar chart, it is easy to notice that ARIMA produces the most error, next comes to KNN, closely followed by linear, moving average and SVM. While for long short term memory, it has the extraordinary performance because of the help from multiple hidden layers. It has dramatically 12 times improvement when making the comparison between the best and the worst model.

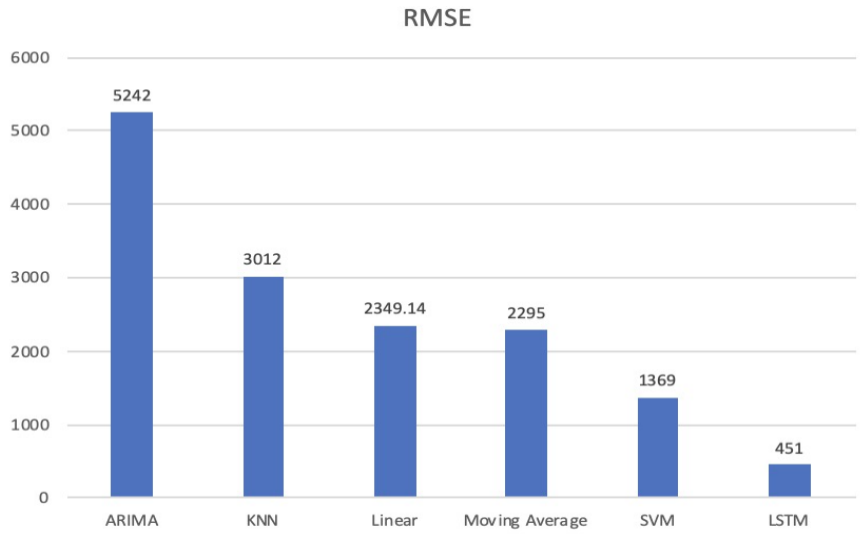

Figure 38: RMSE for Six Models

In the other hand, MAPE metrics cannot show much differences. Regardless of the model performance, it almost settles down the percentage roughly at 69%. Therefore, it is not an understandable metrics for the reader, which motivates to pay more attention on RMSE.

4.2 Tweet Sentiment Role

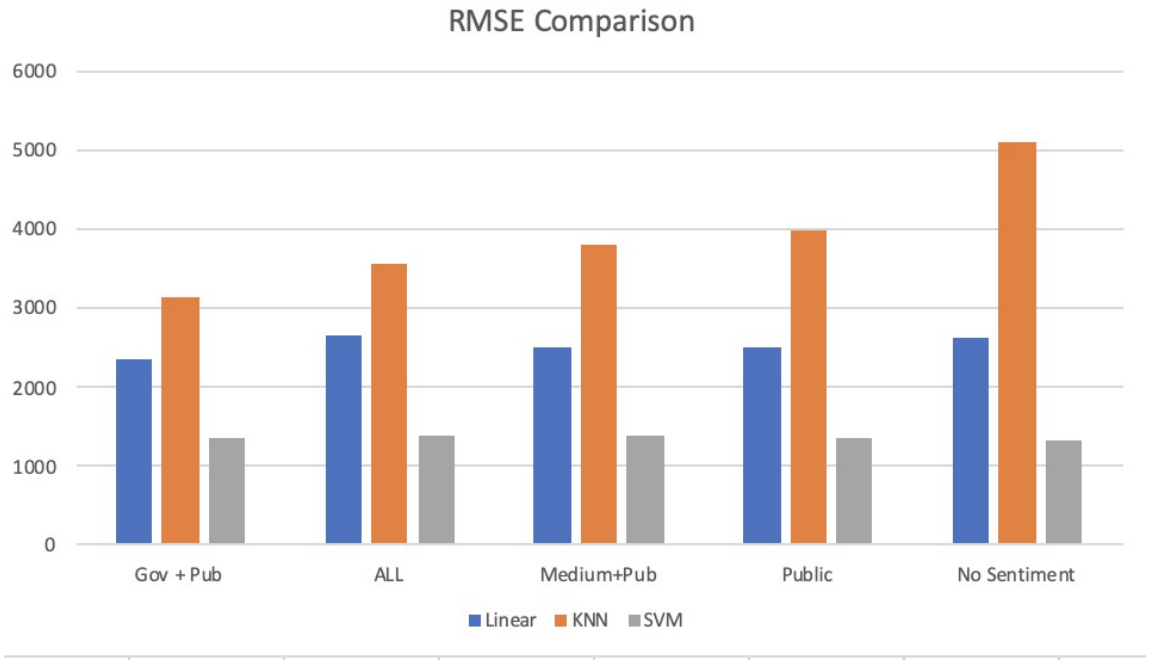

Figure 39: Sentiment Influence

The most interesting part is to see the model performance with different twitter sentiments. First of all, no-sentiment model performs very poorly no matter which algorithm is used. When adding the single tweet variable (public) into the training set, there are some enhancements, indicating the sentiment as the extra information actually reflects some latent patterns. In other words, sentiment does capture price change and capable of price prediction.

After proofing the meaningfulness of the sentiment factor, the combination of two factors varies performance in a good way. In particular, gov + public and medium + public, saying that public sentiment cannot capture the whole price fluctuation. The information missed by public sentiment is exactly filtered by more specific factors such as government and medium. However, different metrics between two of them gives us more insights that government is more valuable factor.

Moreover, when all of the sentiment factors have been applied into the models, surprisingly causing a decrease compared to the best model so far (gov + public) while an increase compared to the second best (medium + public). In this way, it is more confident to say government is a better factor. From the result above, it introduces a classic statistical problem (multicollinearity) mentioned in 2.5.1. Take the example in the linear model, the formula is shown below:

$$y = \beta_0 + Public_1 x_1 + Government_2 x_2 + Medium_3 x_3 + \varepsilon_t$$

Correlated

Multicollinearity refers to the situation that two variables are highly related. In our case, government is highly related with medium as intuitively if the particular tweet account belongs to government category, it also fall in the medium boundary automatically (as official twitter account should have the attribute likes, retweets, replies

larger than 9, 8, 3) but it doesn't work the other way around. Therefore, medium is derived from the government factor. The problem will result in a poor performance and usually overstate model metrics [19].

All of the discussions above concludes that medium is not a good indicator. Considering the bulky the model manager already is, medium is not added to the model eventually but the initial process is still kept in the integration class. The reason not to abandon government and public is due to information amount, the more information it contains, the larger contribution made to the model prediction.

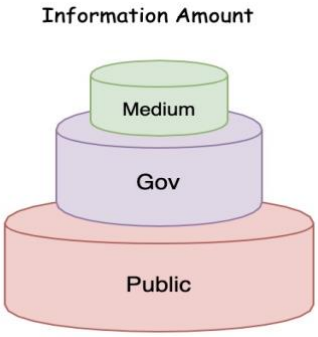

Figure 40: Sentiment Information Amount

4.3 Further Findings

After analyzing the bar chart, it gives some sense that sentiment is a useful factor. However, it is better to demonstrate in a more clear way by using the correlation matrix. The correlation matrix basically ranges from +1 to -1. '+1' indicates two variables have the perfect positive correlation, '0' means no relationship at all and '-1' tells about perfect inverse correlation.

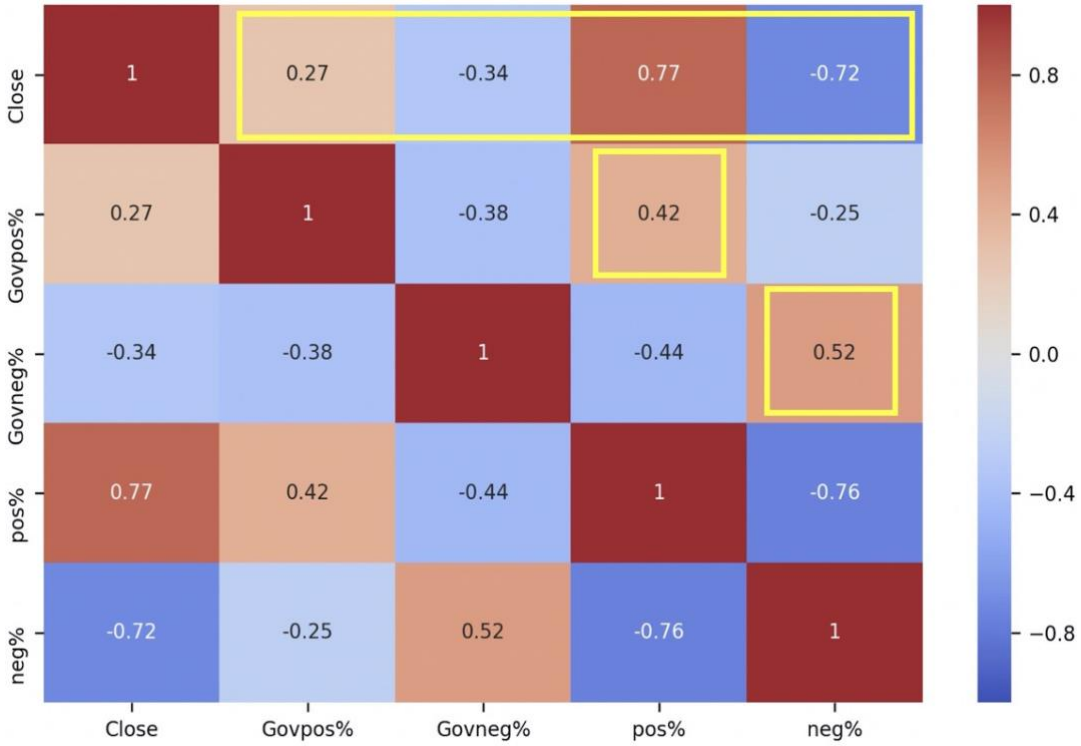

Figure 41: Correlation Matrix

From the matrix, it can be found out that close price has the positive correlation with Govpos% and pos%. Govpos% and pos% presents government and public sentiment before they are transformed into dummy variables respectively. Similarly, it has negative correlation with Govneg% and neg%. It really does make sense because when people are optimistic regards to the market, the general market price will go up and vice versa.

Furthermore, public and government sentiments have the consistent results, saying that when there is more positive information, most of the official accounts usually share the similar attitudes. Likewise, if the negative emotion surge, official accounts usually hold the negative standings.

4.4 LSTM Further Implementation

There is not much to say about the other five models but it needs a little bit polish for LSTM as it has the best performance.

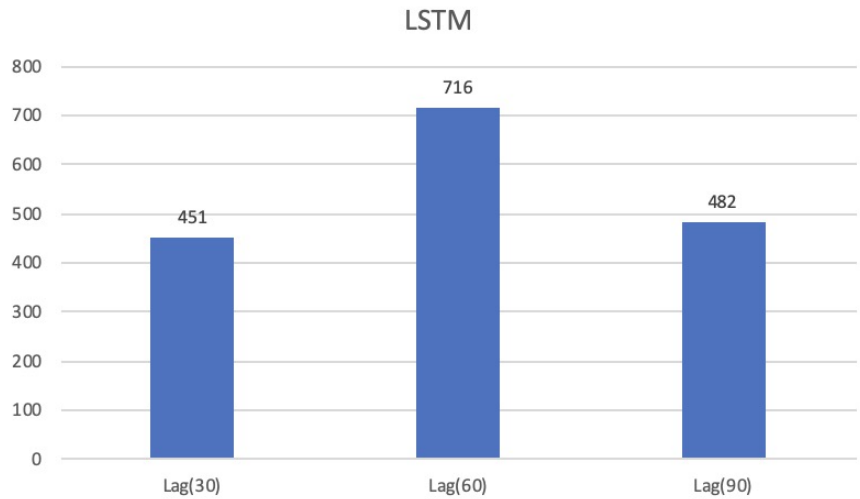

Figure 42: LSTM by trying Different Lags

Based on the current model, different timing lag features are added to get the most optimal performance. Usually, there is a lag between the information leakage and price change, which means the price may not respond to the information immediately. Therefore, it is reasonable to be tested by trying several lagging variables. After applying 1, 2 and 3 months lag factors, it is apparent to find out that one month lag is most accurate to describe the model, with RMSE at only 451. It has achieved great progress compared to the worst model by 12 times better.

Chapter 5 Future improvement

Our project is genuinely a comprehensive system, covering the data crawling, storage, processing, analyzing and visualization. It is really difficult to organize them together in such a limited time. However, there is something more could probably be done to enhance the overall performance.

First of all, only tweet data has been fetched, there are still a lot of sources on the market. For instance, Instagram, YouTube or Facebook. In addition, any valuable information in external annual report or relevant policy reviews could be used as the reference to signal price change. Therefore, all of them can be taken into account in the future work to improve the data variety.

MongoDBs is used for storing the initial raw tweets but later on the processed data or plotly JSON object is in pickle format in local directory. To enhance security, they should be stored in MongoDBs as well. Even though pickle format is convenient to access and is native to python object, the internal integrity and encapsulation is not well achieved.

The model section could be additionally extended by adding qualitative attributes to enhance the model accuracy or introduced with other types of techniques such as the sliding window on all of the models to adjust the information lag that impacts on the price. However, it will take much longer time in training, possibly lagging the user experience.

The Crontab leads to the update in the daily basis, so does the same to the price tracker, it would be more user-friendly if a real-time system will be delivered in the future.

Chapter 6 Conclusion

In conclusion, this project explores the tweets' predictive power in cryptocurrency. Various techniques have been utilized through the project in order to achieve the analysis purpose. Initially, tweet data is retrieved via Search and Streaming API, after doing the sentiment analysis (VaderSentiment), they are saved into non-relational database. Then, the tweet information is transformed into dummy variables and concatenated with the historical price as known as the training data. In the next stage, main thread named model manger runs six models concurrently to deliver the prediction results. As for the web visualization, Nodejs and Reactjs play important roles in server implementation and deliver comprehensive information. To keep it updated on a daily basis, it is set up onto the Nectar platform and with the help of python-crontab.

The overall results are fascinating, the best model has made 12 times performance improvement compared to the one designed in the first place. Furthermore, by analyzing the potential emotions, it is interesting to find out the strong correlation with the price whatever from the government or public perspective, which suggests that tweet data actually is a very useful data source for cryptocurrency analytics.

Reference

[1] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System (2011). Bitcoin.org. 1(1), 1.

[2] Federal Bureau of Investigation (FBI). Bitcoin Virtual Currency: Intelligence Unique Features Present Distinct Challenges for Deterring Illicit Activity (Apr 2012). *Directorate of Intelligence*. Retrieved from https://www.wired.com/images_blogs/threatlevel/2012/05/Bitcoin-FBI.pdf

[3] Peter D. D. An Analysis of Cryptocurrency, Bitcoin, and the Future (Sep 2016). *International Journal of Business Management and Commerce*. 1(2), 1.

[4] Twitter Firehose vs. Twitter API: What's the difference and why should you care? (https://brightplanet.com/2013/06/25/twitter-firehose-vs-twitter-api-whats-the-difference-and-why-should-you-care/).

[5] Vishal A. K. and S.S., Sonawane. Sentiment Analysis of Twitter Data: A Survey of Techniques (Apr 2016). *International Journal of Computer Applications*. 139(11), 5-6.

[6] Riddhiman G. , Mohamed D. , Meichun H. and Bing L. Combining Lexicon-based and Learning-based Methods for Twitter Sentiment Analysis (June 2011). *Hewlett-Packard Development Company, L.P*. Retrieved from https://www.hpl.hp.com/techreports/2011/HPL-2011-89.pdf

[7] AI Driven Enterprise solutions. VADER, IBM Watson or TextBlob: Which is Better for Unsupervised Sentiment Analysis? (Mar 2019). Retrieved from https://medium.com/@Intellica.AI/vader-ibm-watson-or-textblob-which-is-better-for-unsupervised-sentiment-analysis-db4143a39445

[8] C.J. Hutto. and Eric G. VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Retrieved from http://comp.social.gatech.edu/papers/icwsm14.vader.hutto.pdf

[9] Dummy Variable in Regression (https://stattrek.com/multiple-regression/dummy-variables.aspx).

[10] MongoDB documentation. (https://docs.mongodb.com/manual/)

[11] StackOverflow. Developer survey results 2018 (2018). Retrieved from

https://insights.stackoverflow.com/survey/2019 *(hämtad 2019-04-16)*.

[12] Plotly Data Science Platform Used by NASA and Google Raises $5.5 Million. *Techvibes* (Aug 2018). Retrieved from https://techvibes.com/2015/06/03/plotly-2015-06-03

[13] Mozilla Developer Connection. *HTML Canvas Element* (June 2011). Retrieved from https://developer.mozilla.org/en/DOM/HTMLCanvasElement

[14] Andrew W. L., Harry M. and Jiang W. Foundations of Technical Analysis: Computational Algorithms, Statistical Inference, and Empirical Implementation (Aug 2000). *The Journal of Finance*. 4(4), 1706

[15] Sepp H. and Jurgen S. Long Short-Term Memory (1997). *Neural Computation*. 9(8), 1735-1780.

[16] Deng, Z., Zhu, X., Cheng, D., Zong, M., and Zhang, S. Efficient KNN Classification Algorithm for Big Data (2016). *Neurocomputing*, 195, 143-148.

[17] Yanshan S. KNN predictability analysis of stock and share closing prices (Feb 2016). *Thesis for University of Leicester*. Retrieve from https://pdfs.semanticscholar.org/2002/2b45a0175e21189abe497b2dd4555e586065.pdf

[18] Scholkopf, B., and Smola, A. J. Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond (2001). *MIT press*.

[19] Farrar. D. E. and Glauber R. R. Multicollinearity in Regression Analysis: The Problem Revisited (Dec 1964). *Sloan School of Management MIT*.

Appendix

Video Demonstration: https://www.youtube.com/watch?v=vXfqrg6xQx8

Code used for this project

Analyzer: https://github.com/alanwangwyz/twitter-model

Web: https://github.com/bell0925/cryptographic.git

Web Application: http://45.113.234.255:3000