Final Project Report

Stock Wizard Program

Wisam Matlob, Aye Swe, Nikita Voloshenko, Marcus Zhou

San Jose State University

CMPE 188- 01

March 28, 2019

Spring 2019

Team MSVZ

Table of Content

1. Abstract ……………………………………………………………….3

2. Project Introduction……………………………………………………...4

3. Related Work……………………..……………………………………...4

4. Project Considerations…………………………………………………...6

5. Data and tools……………………………………………………………7

5.1 Data used

5.1.1 Selection of Dataset

5.2 Data Gathering

5.3 Data Preprocessing and Preparation

6. Technical Approach.…………………………………………………….11

6.1 Tools, Language and Library

6.2 Algorithms use

6.3 Evaluation and Errors

7. Conclusion & Evaluation..………………………………………………15

8. References……………………………………………………………….16

9. Appendix………………………………………………………………...17

Abstract

Our project, Stock Wizard application, is an application to predict the stock prices by analyzing the live stream of the publicly available data by using past data. In this project, different data sources and third party APIs were used to conduct the process of data collection along with data processing. As data cleaning process is completed, we split the data into training and testing data sets in order to train the model we used to make predictions of the bitcoin price. Supervised and unsupervised machine learning algorithms were applied on the cleaned data sets. After this process, we compared the predictions with the actual results. Finally, to visualize and analyze the data and the predicted data, we used different types of plots, and applied the dimensionality reduction techniques to reduce the number of random variables. Furthermore, feature selection and feature extraction.

2. Project Introduction

For decades, the stock market has been the core mechanism that shapes how the financial market operates. In fact, stock market prices are largely influenced by various factors such as the developing news in addition to several unknown possibilities regarding the stock price fluctuations. Therefore, it is worthwhile to explore the insightful relationship between the stock price and factors that have significant impacts on it.

As stated in the project proposal, ordinary citizens traditionally relied on financial experts, news outlets, social media and other public information about the companies; to determine what stock price movement is in their favor. Even the most experienced financial advisors are constantly encountered challenges regarding the accuracy and reliability of the news affecting the stock market prices. As a result, having such dependencies during the calculations means nothing is promised on the forecasts because stock prices continue to fluctuate based on the developing trends. To maximize the financial reward, there are many stock analysis tools available in the market used in the past decade but the field is still at large. The project's intention is to remove these dependencies and transform the way of the current analysis towards the uncertainties by training live stream data using machine learning algorithms. This document is structured into several components: project considerations, challenges and technical difficulties, lesson learned, and project outcomes.

3. Review of Related Work

In this section, we explored the related work done by others using learning algorithms to make predictions of stock market price. Our project is inspired by the proceedings of the Australasian Computer Science Week Multiconference. The article *Stock market analysis using* *social networks* was co-authored by several researchers where they used social media for the prediction of the stock prices. In the study, researchers gather Twitter tweets and various stock market exchanges and prepare their data by analyzing the raw data before feeding to the classifiers. In the beginning of the article, they discussed their related research work about neural network classifiers where they processed 12 millions weights before outputting to the 129 neurons with four hidden layers. They learned that using such a classifier requires intensive computational power.

The researchers then discussed their experiment process where they built the framework for prediction by following five steps that contains data-preprocessing, data alignment, rationale of sentiment analysis, evaluation of the classifier, and analysis of the results. The collection of data comes from the Twitter API where they aligned the tweets with prices. Authors livestreamed the tweets from the Twitter API as JSON format and gathered features are studied for reductended features, missing data for the analysis. Researcher used Linear regression and SVM with PCA applied data as their main two classifier for their data training and prediction. With the variation of data set, the accuracy fluctuated between 70% to 85%. They also learned that data preparation and normalization will significantly increase the computational efficiency. The result analysis section discussed that the Bernoulli NB provided 81% of accuracy while SVM performed at 85% concluding that SVM outperformed Bernoulli NB (Li, Yang, Zhang & Puthal etc., 2018).

The study gave our team insightful prediction for our technical difficulty and our choice of classifiers for better performed accuracy. The problem of making prediction toward constant fluctuating data based on training the historical pattern to the model is not brand new to the field of Computer Science. As discussed in class, different models have different accuracy rates based on the train and test data as well as veracity of the data itself. We used Twitter API in the first part of the project to capture the real-time tweets to extract the bitcoin stock related tweet. However, we were not able to further analyze the tweets as sentimental analysis for the limited time budgets the team have. Also we didn't have proper server to stream one month of Twitter tweets. Therefore, we had to abandon the Twitter features from our project.

4. Project Considerations

In this component, several areas were determined as a foundation to ensure the success of the project. These areas includes the scope of the project where the limitation of the projects were determined as the tools to implement the research materials and data source that were used to train the model to learn the historical stock prices pattern for bitcoins and make prediction.

4.1 Scope of the Project

Determining the scope of the Stock Wizard project was the initial phase of the entire research process. Initially, the scope of the project was to explore the bitcoin stock prices using Twitter live stream data. However, the data source utilized for the latter part of the project was Yahoo Financial APIs. The reason that we did not use Twitter APIs to complete the project can be found in the section 4.0, Challenges and Technical Difficulties.

5. Data and tools

5.1 Data Sources

We live stream our Bitcoin Price Index Prices from the CoinDesk API, Quandl Platform API, and Yahoo finance API. Bitcoin currency prices from those API were steamed from the January,2016 to January 2019, about 3 year of daily prices.

5.1.1 Selection of Dataset

Dataset from the Quandl and CoinDesk APIs are streamed as JSON and .csv type datasets streamed from the Yahoo finance API. After carefully studying the live streamed sample data from the three APIs, we learned that Quandl API has rate limit, so that we couldn't complete the live streaming for 3 year worth of data. Various datasets on Quandl require licenses that data feed were intended for their business partners and institutions use only. Although there are sample data available for testing purpose, the amount of data is not substantial for our project to get the accurate prediction. Even after API key was requested, we had limited data streaming authorization each day. On the other hand, CoinDesk API provide only closing prices type from historic data. However, we can get a full analysis of the bitcoin index price from the Yahoo finance API. Our dataset have 1098 tuples with the feature columns with Date, High, Low, Open, Close, Volume and Adj Close.

5.2 Data Preprocessing and Preparation

After we gathered the required dataset, we extract the needed features from the JSON file to .csv format files. We visualized the raw dataset using Boxplots, Histograms and Scatter plot for further understanding of the dataset by using the standard pandas library. After studying the raw dataset, we decided to eliminate some features such as closing price and Adjusted closing price, as they were redudented features. In order to make data more meaningful and compressed, we reduced the Open price, Close price, High price, and Low price features to percentage change. We finalized our raw dataset as independent features of High and Low percentage change, Open and Close percentage change, and Volume; and for our dependent features we choose Closing price. The following tables display the tables of raw datasets we collected.

Date	Close	HighVsLow_change	OpenVsClose_change	Volume
2016-01-01	433.9900	2.838961	0.939644	20334286
2016-01-02	433.7200	1.555499	-0.062213	14566820
2016-01-03	430.7000	2.611737	-0.696302	23490162
2016-01-04	433.3200	1.553896	0.608312	22943968
2016-01-05	431.2000	1.487630	-0.489246	19499588
2016-01-06	430.8200	1.672941	-0.088126	18702334
2016-01-07	457.0500	6.772146	6.088390	58253615
2016-01-08	452.8700	3.675170	-0.914561	41594025
2016-01-09	448.3100	2.169027	-1.006911	17967113
2016-01-10	446.1900	2.339155	-0.472887	19778456
2016-01-11	447.7200	2.107431	0.342903	25358480
2016-01-12	445.0400	1.189240	-0.598588	19913400
2016-01-13	432.1800	5.549609	-2.889628	39525195
2016-01-14	429.1300	1.576076	-0.705724	19948730
2016-01-15	372.2600	17.943023	-13.252394	107060613
2016-01-16	385.0400	10.368447	3.433084	68297105
2016-01-17	382.4700	3.444442	-0.667463	25464382
2016-01-18	384.4000	3.772675	0.504615	26546465

Figure 1: A sample of the bitcoin price prediction dataset.

	High	Low	Open	Volume	close
Total	1099.000000	1099.000000	1099.000000	1.099000e+03	1099.000000
mean	4159.237073	3868.430695	4025.061016	4.148110e+08	4028.143024
SD	4076.959663	3708.378505	3916.688545	6.112649e+08	3915.154320
minimal	375.890000	350.390000	368.020000	7.377921e+06	368.020000
25%	673.215000	650.900000	662.210000	3.015572e+07	663.150000
50%	2676.040000	2506.300000	2583.750000	2.083496e+08	2590.050000
75%	6730.049800	6474.350100	6601.744900	5.392349e+08	6601.584950
max	19870.619100	18750.910200	19346.599600	6.245732e+09	19345.490200

Table 2. Statistical summary of the dataset.

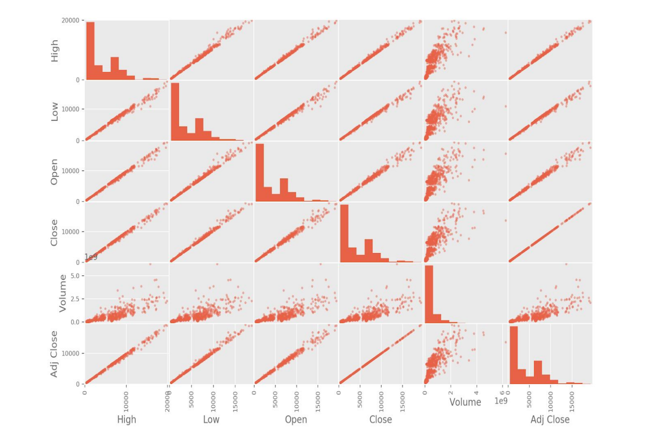

Figure 4: Data Visualization before preprocessing on the dataset to study the dataset.

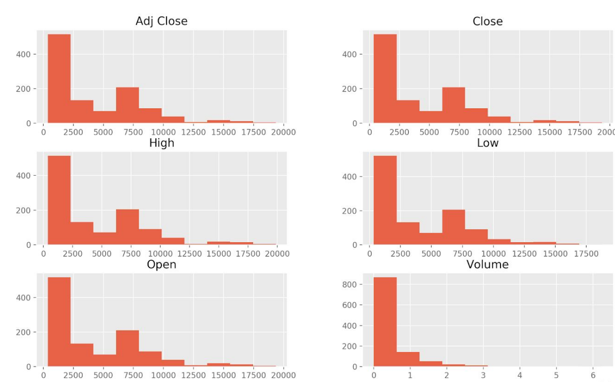

Figure 5: Features Visualization of the raw dataset.

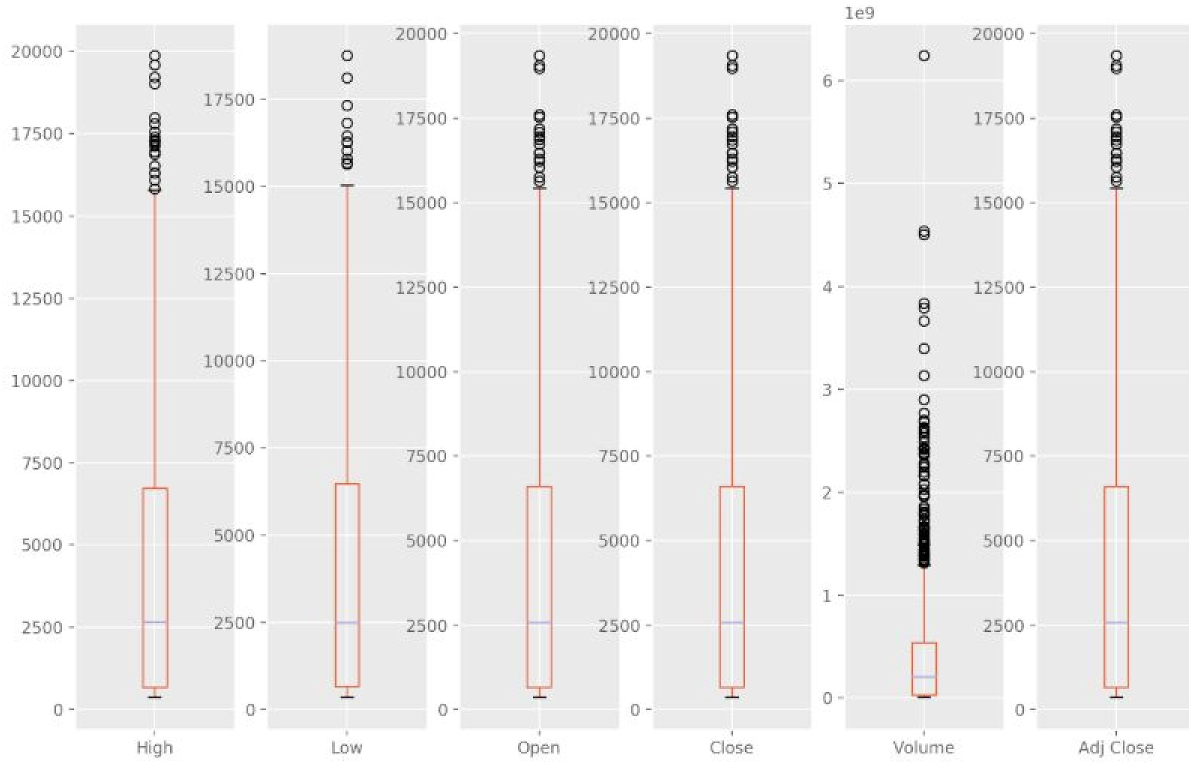

Figure 6: Raw data visualization with boxplot with analysing the basic statistics of the dataset.

We also applied the Principal Component Analysis on the collected data to reduce the attributes of the each dataset by transforming related data into each separate data point.

$$\min(n-1, p)$$

Figure 7 and 8 in the following page display the comparison between the two datasets that PCA is applied on, before and after the application. As displayed, the attributes seem to become guiding to one direction after PCA is applied while the direction of the attributes was scattered.

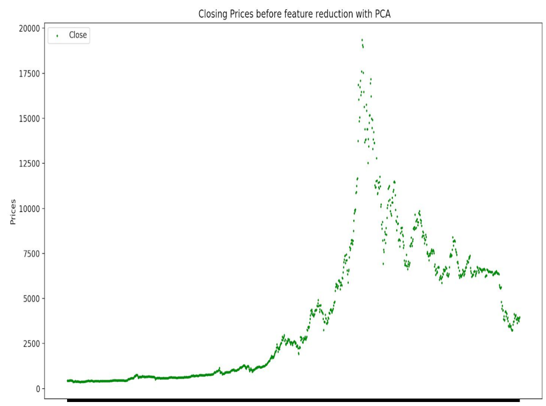	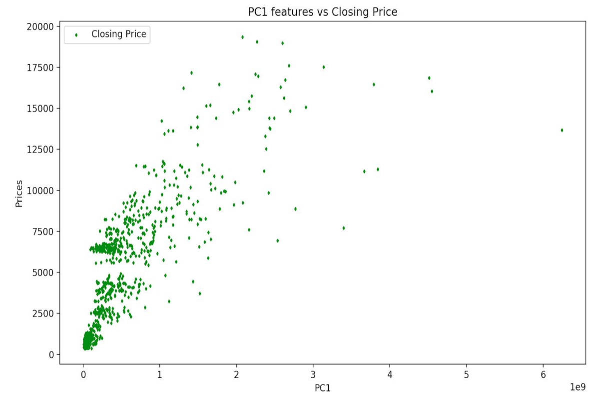
Figure 7: Three years closing stock price before the feature reduction using PCA.	Figure 8: Three years closing stock prices after the feature reduction using PCA.

5.3 Tools, Language and Libraries

In order to successfully predict the bitcoin stock prices, many developer tools were used during the project implementation. In this section, an overview of each tool used will be covered. The following are a list of tools that were used to implement the project: Python library packages, APIs from various data source, and the version control tools.

Libraries:

- Pandas - library is the main library that we used in our project for writing and editing data frames, reading from various file type and web data.
- Pickle - Project library used to save our trained and fitted data for later use in order to avoid refitting the 3 years of stock data.
- Request - request library is used in our project to make HTTP requests from the financial web APIs.

- Mathplotlib - library allowed us to visualize the data sets by plotting various graphs for different analysis such as plotting histogram, bar charts, and scatter plots.
- Sklearn - library is used for our Linear Regression, SVM, and to split our dataset into training and testing dataset and further data preprocessing such as scaling the data.
- Numpy - library package that enables us manipulating with array and numerical data analysis
- Tweepy - We used Tweepy library to handle authentication process for Twitter API and filtering of tweet's features from the streamed data from the twitter API.
- OAuthHandler- Our project use the library to authenticate the Twitter API credential keys.
- Python - Python is our development language for our project.
- Pycharm - Team use Pycharm IDE for the development which enables us to easily importing all necessary library and packages on the fly.
- Github - version control platform that allows us to do collaborative work

Our github repo is on **https://github.com/AyeSwe/Cleaned_Final.git.**

6. Technical Approach

6.1 Algorithms use and process

We used Simple Linear Regression model and Decision Tree Regression Model to predict the closing price of the bitcoin stock. Our dataset independent features are open, high, low, and volume. The target dependent feature is the daily closing price. This project streamed the history bitcoin price index data, from 2016 January to 2019 January from the yahoo finiciance API. Before we apply the data set into the Linear Regression, we evaluate the relationship of each features correlation to our target label by performing the linear regression and study their intercepts and coefficient for associated relation.

$$y = b_0 + b_1 x$$

y as the target label(dependent features), b0 as intercept value and b1 as coefficient of the feature, and x as our independent feature.

Figure 9: Real Price vs Predicted Price from Linear Regression model and Decision Tree Regression model.

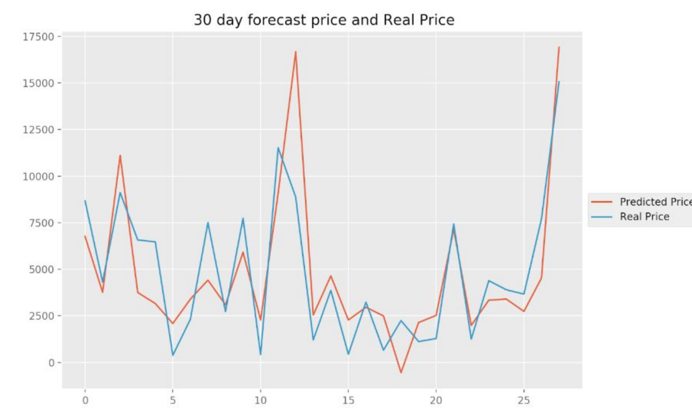

Figure 10: Forecasted price and predicted price with open/close % change, low/high % change, and Volume features.

Figure 11: Forecasted price and predicted price with open, high, low, and volume features.

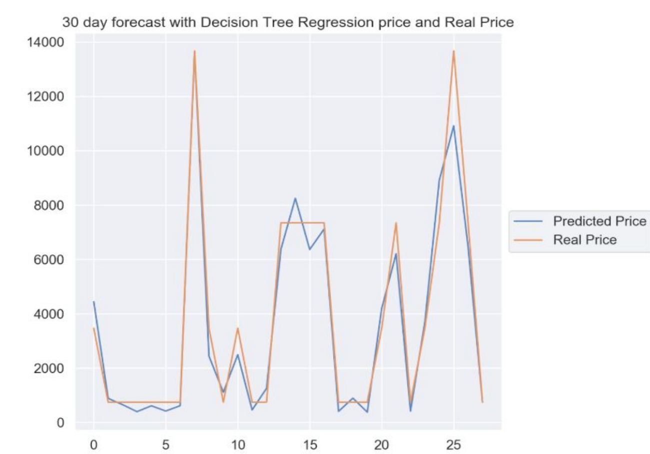

Figure 12: 30 day prediction with Decision Tree Regression model.

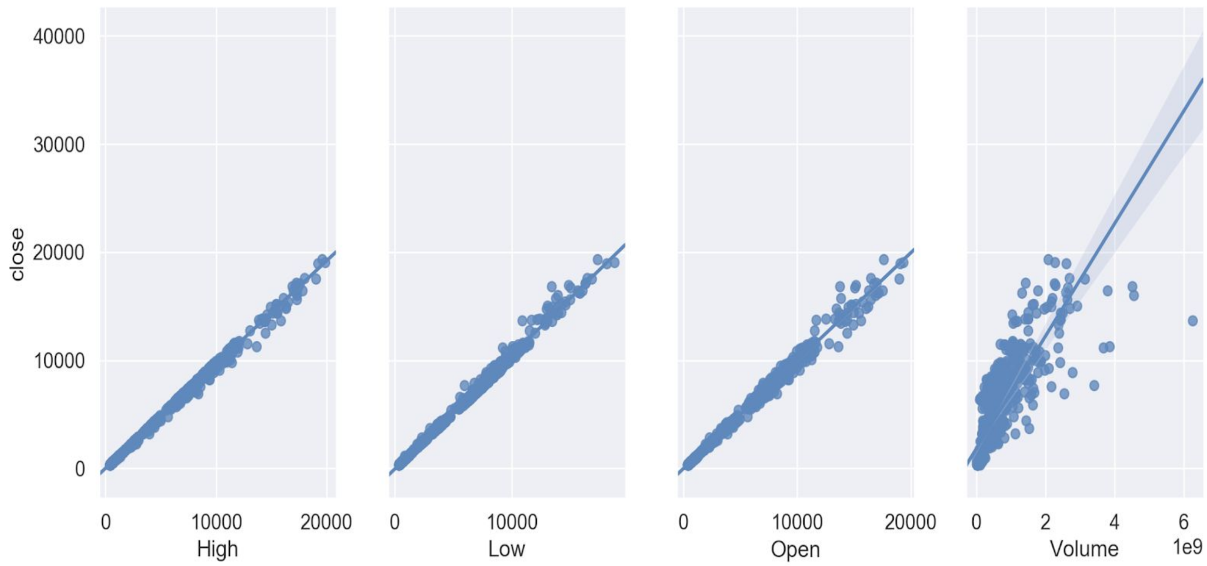

Figure 13: Least Squares Line for each features and our target label.

Our linear Regression model Intercept at 1.50 and their coefficients are 0.8.39 for High, 0.601 for Low, -0.45 for Open and 0.0000004 for Volume. According to the visual analysis of Least Square Line for each features, we can see that we have low variance nature of the sample which stay in approximately in the same line. We can also observed that Volume and open features have a bit of high variance than that of other features which is explained by the their low coefficient and negative coefficient. We can also study this behavior from our linear regression model algorithm for sklearn as the confidence level as following.

The confidence level of the each features with target label prediction has approximately 95% to 99 % so that we have true value of target label around 97% ± 2% true value .

We use the hypothesis testing and P-value to further evaluate the Linear Regression model and P-value analysis. Hypothesis testing on alternative hypothesis has relationship between features. All the coefficient of each feature are positive so that all features are positively correlated to our target label.

While Linear Regression model can predict our dataset with 0.99% accuracy score, Decision Tree Regression model has 0.94% accuracy score. In order to avoid the overfitting of the Decision Tree maximum depth level is controlled.

6.2 Evaluation and Errors

We used Root Mean Squared Error(RMSE) to evaluate our Linear Regression model and Decision Tree Regression by observing the square root of the variance of the residuals as standard variation between our predicted data and real value. Our RMSE for the linear Regression model is 65.41 and Decision Tree Regression model is 797.47. We performed the outlier removal by removing data which are out 2 standard variation from every features rows. After repeated testing with different standard variation, we concluded that Decision tree Regression model are more sensitive the outlier present in the dataset. Next, we apply PCA to our dataset's independent features and used the Decision Tree Regression model to predict the target data. We observed that accuracy score rise to 99.4% and and we are able to take down the RMSE value to 353.84 from 797.47. However, Linear Regression model prediction with PCA dataset accuracy and RMSE are unchanged.

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2$$

7. Conclusion & Evaluation

Since initial phase, this project involves exploration of the few research paper to define the problem statement and possible solutions that contains complex concepts of machine learning algorithm analysis to tackle the existing pain point in the stock market; price prediction. First, we had hard time defining the data sources from Twitter API to Quandl, Coindesk, and Yahoo Financial APIs. During the data processing, we also had difficulty for access to the live-stream data as well as computational limitation. Feature reduction technique we used was the Principal Component Analysis (PCA) to reduce the variables across the spectrum to train the model that we compared the result of before and after technique being applied. While, toward the final phase of the project, we received an unexpected result of 99% accuracy with Linear Regression model, we had to progressed from 80% accuracy. On the other hand, just to compare the accuracy of other machine learning model, we experimented with SVM which gave us quite lower accuracy results than Linear Regression model. For the third experiment, we used decision tree to make the prediction that results at 94% accuracy.

References

Man Li, Chi Yang, Jin Zhang, Deepak Puthal, Yun Luo, and Jianxin Li. 2018. Stock market analysis using social networks. In Proceedings of the Australasian Computer Science Week Multiconference (ACSW '18). ACM, New York, NY, USA, Article 19, 10 pages. DOI: https://doi.org/10.1145/3167918.3167967

T. Bodnar, C. Tucker, K. Hopkinson, & S. G. Bilén. (2014). Increasing the veracity of event detection on social media networks through user trust modeling. Paper presented at the *2014 IEEE International Conference on Big Data (Big Data),* pp. 636-643. doi:10.1109/BigData.2014.7004286

https://pdfs.semanticscholar.org/7fe7/fd71e0d9fa4dbf8423fa1c872a5966545985.pdf

Appendix

Source Code

Also Available at : https://github.com/AyeSwe/Cleaned_Final.git

1. **All_features_prediction-Yahoo.py**

```
"""
This program will predict the 30 days stock close price with all of the features (Open, high, low, Volume)
"""
import seaborn as sns; sns.set(font_scale = 1.2)
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import seaborn as sns; sns.set(font_scale = 1.2)
from sklearn import metrics
yahoo= pd.read_csv('./Data/yahoo.csv')
# testing for dropping date column, Adj Close column
#df_Forcast.drop(["a"], axis=1, inplace=True)
yahoo.drop(['Date','Adj Close'], axis =1, inplace= True)
yahoo['close']= yahoo['Close']
yahoo.drop(['Close'], axis =1, inplace= True)
yahoo = yahoo.dropna()
x = yahoo.iloc[:,: -1]# evey row and every column other than the last column
print (x.shape)
y = yahoo.iloc[:,-1]# only the last column : this is class data
# get the testing data out of x and y
x_train_data, x_test_data, y_train_data, y_test_data = train_test_split(x,y, random_state=17, train_size=0.975)# predit about 10 day
```

```
#  # Linear Regression model
# # #
linearRegression_model = LinearRegression()
# # training the High feature and close price
linearRegression_model.fit(x_train_data,y_train_data)
# getting the intercept point from
print ("Intercept: ", linearRegression_model.intercept_)
print ("Coefficienc", linearRegression_model.coef_)
y_prediction = linearRegression_model.predict(x_test_data)
print ('close prediction: ', y_prediction)
# # plotting the least Squares Line
#
# sns.pairplot(yahoo, x_vars=['High','Low','Open','Volume'], y_vars='close', size=4, aspect=0.7,
kind='reg')
# #plt.show()
#
print ('linearRegression_model.score() is :',
linearRegression_model.score(x_train_data,y_train_data) )
#
# # Model Evaluation Metric for LinearRegression
# # MSE (Mean Squared error checking for features)
print ("Root Mean Error is ",np.sqrt(metrics.mean_squared_error(y_test_data, y_prediction)))
#
print ("Test data is : ", y_test_data)
realPrice = np.array(y_test_data)
print ("Test data is : ", realPrice)
# sending to csv for further comparison
newDf = pd.DataFrame(columns=['Close'])
newDf['Close']= y_prediction # fill the column with Date
newDf.to_csv('./Data/LinearRegressionPrediction.csv')
```

```
plt.plot(y_prediction)
plt.plot(realPrice)
plt.title(" 30 day forecast price and Real Price")
# plt.legend()
# plt.show()
graph = plt.subplot(111)
box = graph.get_position()
graph.set_position([box.x0, box.y0, box.width*0.65, box.height])
legend_x = 1
legend_y = 0.5
plt.legend(["Real Price", "Predicted Price"], loc='center left', bbox_to_anchor=(legend_x,
legend_y))
plt.show()
```

2. **coindesk_bitcoin.py**

```
"""
#   This program read the streamed cleaned coindesk API stock prices for coindesk.csv
#   perform the necessary further datapreprocessing
#   analyst with Linear regression model , and predict the 1 month worth of price
#   perform the pickling for the model
#   perform the required Visual presentation for further analysis
"""
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import pandas as pd
from matplotlib import style
import matplotlib.pyplot as plt
import math
import numpy as np
import time
import datetime
```

```
import pickle
df = pd.read_csv('./Data/coindesk.csv')
df = df.set_index('Date')
#print (df.info())
style.use('ggplot')
#print (df.head())
df.rename({"Unnamed: 0":"a"}, axis="columns", inplace=True)
df.drop(["a"], axis=1, inplace=True)
#print (df.head())
#df.plot(kind= 'box',subplots=True, layout= (1,6),sharex=False,sharey=False)
# df['Close'].plot()
# plt.title("CoinDesk bitcoin price from 2016January To 2019January")
# plt.xlabel("Date")
# plt.ylabel("Price")
#plt.show()
forecast_col = 'Close'
# # # get the data set length percentage is 0.1 will be in the forecasted
forecast_out = int(math.ceil(0.1* len(df)))# 1 % of the data
 # preparing for the empty labels for the incoming forecast
df['label']= df[forecast_col].shift(-forecast_out)
#print (df)
df.dropna(inplace=True)
# #
#print (df.tail())
x = np.array(df.drop(['Close'],1)) # all columns, other than label column
y = np.array(df['Close'])# only label column
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.2)
# Linear regression model
# linear_regression = LinearRegression()
```

```
# linear_regression.fit(x_train,y_train)
# # ---------Pickling-----------
# # with open ('./Data/inearregressionCoinDeskFitted.pickle', 'wb') as f:
# #     pickle.dump(linear_regression,f)
pickled = open('./Data/linearregressionCoinDeskFitted.pickle','rb')
linear_regression = pickle.load(pickled)
LinearRegression(copy_X= True, fit_intercept=True,n_jobs=1, normalize=False)
accuracy = linear_regression.score(x_test,y_test)
# # #
print ('\nLinear_regression accuracy is :', accuracy,"\n")# possible from the not enough
information.
# # # New predit
X = x[:-forecast_out]
X_lately = x[-forecast_out:]
Forecast_set = linear_regression.predict(X_lately)
#print (Forecast_set)
# # just visualization
df['Forecast'] = np.nan
last_date = df.iloc[-1].name
# print ("last date is: " , last_date)
last_date = time.mktime(datetime.datetime.strptime(last_date,"%Y-%m-%d").timetuple())
one_day = 86400
next_unix = last_date + 86400
## just to show the forcast_set with Price values
label_arry = np.array(df['label'])
next_date_array= []
# # just visualization ( later get inside the
for i in Forecast_set:
    next_date = datetime.datetime.fromtimestamp(next_unix)# might be this one wrong
    next_date = str(next_date)
```

```
    #print ("String next date is:", next_date)
    next_date= str.split(next_date," ")
    #print (("Splited string next date is:", next_date[0]))
    next_date = next_date[0]
    next_date_array.append(next_date) # just to get an arrray for later use
    next_unix +=one_day
    df.loc[next_date] = [np.nan for _ in range(len(df.columns)-1)] +[i]
#print ("next_date array is: ", next_date_array)
# make a forcast vs nexdate dataset for Demo
newDf = pd.DataFrame(columns=['Date','Price'])
newDf = pd.DataFrame({'Date': next_date_array[:],'Price': Forecast_set[:]})
#print ("newDf is: ---->", newDf)
#---------------------------------------------------
df['Close'].plot()
df['Forecast'].plot()
plt.title("January 1st, 2016 To January 1st, 2019 bitcoin Stock at CoinDesk API price and 30 day
forecast")
plt.legend(loc=4)
plt.xlabel('Date')
plt.ylabel('Price')
plt.show()
```

3. **Data_streamming.py**

```
"""
#   This program live stream the bitcoin price from the coindesk API
#   3 years of daily stock information are gathered
#   perform some necessary data preparation before saving to the further analysis
#   streamed data are saved as coindesk.csv
This program will live stream from the yahoo financial analysis of bitcoin value in US dollar
"""
import matplotlib.pyplot as plt
```

```
import requests
import datetime as dt
import pandas as pd
import pandas_datareader.data as web
r =
requests.get('https://api.coindesk.com/v1/bpi/historical/close.json?start=2016-01-01&end=2019-
01-02').json()
#print(r)
#sending json file into the data frame
df = pd.DataFrame(r, columns=['bpi'])
#df.to_csv('./Data/coindesk.csv')
#print (df.info())
# drop the null values
df.dropna(inplace=True)
#split a column to two column
#since "bpi" is one columns with values and key dictionary type
# sending values of the bpi as price
Price = df['bpi'].values
#print (Price)
# sending key (Dates) of the bpi as Date
Date= df['bpi'].keys()
#print (Date)
# making a new dataframe with columns labels
newDf = pd.DataFrame(columns=['Date','Close'])
newDf['Date']= Date # fill the column with Date
newDf['Close']= Price# fill the column with Prices
newDf.to_csv('./Data/coindesk.csv')
#----------------------Streaming of coinDesk data end here and yahoo finance data streaming start
here----------------------
symbol= 'BTC-USD'
```

```
start= dt.datetime(2016,1,1)
end = dt.datetime.now()
df = web.DataReader(symbol, 'yahoo', start, end)
df.to_csv('./Data/yahoo.csv')
#print (df)
df = pd.read_csv('./Data/yahoo.csv',parse_dates=True,index_col=0 )
df = df.round(4)
df.to_csv('./Data/yahoo.csv')
```

4.. **DecisionTree.py**

```
"""
This program will predict the 30 days stock close price with all of the features (Open, high, low, Volume)
"""
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from math import sqrt
from sklearn.model_selection import train_test_split
import seaborn as sns; sns.set(font_scale = 1.2)
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeRegressor as DecisonTree
from sklearn.metrics import mean_squared_error
yahoo= pd.read_csv('./Data/yahoo.csv')
# testing for dropping date column, Adj Close column
#df_Forcast.drop(["a"], axis=1, inplace=True)
yahoo.drop(['Date','Adj Close'], axis =1, inplace= True)
yahoo['close']= yahoo['Close']
yahoo.drop(['Close'], axis =1, inplace= True)
yahoo = yahoo.dropna()
```

```
print ("Data summary before outlier removal: ", yahoo.describe())
# dropping any outlier that out of 3 standard deviation from the column mean (99.7%)
yahoo = yahoo[np.abs(stats.zscore(yahoo)< 2).all(axis=1)]
print ("Data summary After outlier removal:")
print (yahoo.info())# wow 1099 - 1065 = 34 rows are deleted for outlier
print ("Data summary before outlier removal:")
print(yahoo.describe())
#split the dat to training data and testing data
x = yahoo.iloc[:,: -1]# evey row and every column other than the last column
print (x.shape)
y = yahoo.iloc[:,-1]# only the last column : this is class data
print (y.shape)
x_train_data, x_test_data, y_train_data, y_test_data = train_test_split(x,y, random_state=17,
train_size=0.975)# predit about 10 day
print ("x train is: ",x_train_data)
print ("y_train is: ",y_train_data)
# normalize the x train data and x testing data
scale = StandardScaler()
xtrain_scale= scale.fit_transform(x_train_data)
xtest_scale = scale.transform(x_test_data)
# # Decision Tree model
D_Tree_model = DecisonTree(max_depth=2) # to avoid outfitting max_depth is controlled.
# # train with scaled x train data and y train data
D_Tree_model.fit(xtrain_scale, y_train_data)
tree_predit = D_Tree_model.predict(xtest_scale)
print ("D_Tree Prediction is ", tree_predit)
D_treeAccurecy = D_Tree_model.score(xtest_scale,y_test_data)
#linearRegression_model.score(x_test_data, y_test_data)
print ("Decision Tree model accuracy is: ", D_treeAccurecy )
# # Evaluation of Decision tree model
```

```
DT_MeanError = mean_squared_error(y_train_data,D_Tree_model.predict(xtrain_scale))
Root_mean_square = sqrt(DT_MeanError)
print ("Mean square error the decision tree model", DT_MeanError)
#
# Now see how the behavior or mean square error on the test data.
D_treeTestMeanError = mean_squared_error(y_test_data, D_Tree_model.predict(xtest_scale))
Root_mean_square = sqrt(D_treeTestMeanError)
print ("Testing the testing data to see how model generalize the prediction: ",
Root_mean_square)
tree_predit_array = np.array(tree_predit)
y_test_data_array = np.array(y_test_data)
# sending to csv for further comparison
newDf = pd.DataFrame(columns=['Close'])
newDf['Close']= tree_predit_array # fill the column with Date
newDf.to_csv('./Data/DecisionTreePrediction.csv')
plt.plot(y_test_data_array)
plt.plot(tree_predit_array )
plt.title(" 30 day forecast with Decision Tree Regression price and Real Price ")
# plt.legend()
# plt.show()
graph = plt.subplot(111)
box = graph.get_position()
graph.set_position([box.x0, box.y0, box.width*0.65, box.height])
legend_x = 1
legend_y = 0.5
plt.legend(["Predicted Price","Real Price" ], loc='center left', bbox_to_anchor=(legend_x,
legend_y))
plt.show()
```

5. **Features_Analysing.py**

MSVZ

```
"""
This program analysis the yahoo finance dataset
features with LinearRegression model, intercepts, coefficients, and mean square of each features
"""
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import seaborn as sns; sns.set(font_scale = 1.2)
from sklearn import metrics
yahoo= pd.read_csv('./Data/yahoo.csv')
yahoo.drop(['Date','Adj Close'], axis =1, inplace= True)
yahoo['close']= yahoo['Close']
yahoo.drop(['Close'], axis =1, inplace= True)
yahoo = yahoo.dropna()
#print(yahoo.info())
print (yahoo.tail())
# this will do the linear regression performance for each features
feature_high = ['High'] # High, Low, Open, Volume,
x = yahoo[feature_high]
y = yahoo.iloc[:,-1] # target label
# get the testing data out of x and y
x_train_data, x_test_data, y_train_data, y_test_data = train_test_split(x,y, random_state=17,
train_size=0.975)# predict about 10 day
 # Linear Regression model
# #
linearRegression_model = LinearRegression()
# # training the High feature and close price
linearRegression_model.fit(x_train_data,y_train_data)
```

```
# getting the intercept point from
print (linearRegression_model.intercept_)
print (linearRegression_model.coef_)
# now predict clos price with the new High price (5901.36)
# make new dataframe
#new_High_data=pd.DataFrame({'High': [5901.36]})
#new_High = linearRegression_model.predict(new_High_data)
y_prediction = linearRegression_model.predict(x_test_data)
print ('close prediction: ', y_prediction)
# plotting the least Squares Line
sns.pairplot(yahoo, x_vars=['High','Low','Open','Volume'], y_vars='close', size=4, aspect=0.7, kind='reg')
plt.show()


print ('linearRegression_model.score() is :', linearRegression_model.score(x_train_data,y_train_data) )
# Model Evaluation Metric for LinearRegression
# MSE (Mean Squared error checking for features)
print("Root Mean Error is ",np.sqrt(metrics.mean_squared_error(y_test_data, y_prediction)))
```

6. LinearRegression_DecisionTree_RealPriceComparison.py

```
"""
This program will predict the 30 days stock close price with all of the features (Open, high, low, Volume)
"""
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeRegressor as DecisonTree
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.utils import resample
from sklearn import svm
from sklearn.linear_model import LinearRegression
import seaborn as sns; sns.set(font_scale = 1.2)
yahoo= pd.read_csv('./Data/yahoo.csv')
# testing for dropping date column, Adj Close column
#df_Forcast.drop(["a"], axis=1, inplace=True)
yahoo.drop(['Date','Adj Close'], axis =1, inplace= True)
yahoo['close']= yahoo['Close']
yahoo.drop(['Close'], axis =1, inplace= True)
yahoo = yahoo.dropna()
#print(yahoo.info())
#print (yahoo.tail())
#split the dat to training data and testing data
x = yahoo.iloc[:,: -1]# evey row and every column other than the last column
print (x.shape)
y = yahoo.iloc[:,-1]# only the last column : this is class data
print (y.shape)
x_train_data, x_test_data, y_train_data, y_test_data = train_test_split(x,y, random_state=17,
train_size=0.975)# predict about 30 day
# # Linear Regression model
#
linearRegression_model = LinearRegression()
linearRegression_model.fit(x_train_data, y_train_data)
liner_predict=linearRegression_model.predict(x_test_data)
accurency_linear = linearRegression_model.score(x_test_data, y_test_data)
# normalize the x train data and x testing data
scale = StandardScaler()
```

```
xtrain_scale= scale.fit_transform(x_train_data)
xtest_scale = scale.transform(x_test_data)
# # Decision Tree model
D_Tree_model = DecisonTree(max_depth=2) # to avoid outfitting max_depth is controlled.
# # train with scaled x train data and y train data
D_Tree_model.fit(xtrain_scale, y_train_data)
tree_predit = D_Tree_model.predict(xtest_scale)
print ("D_Tree Prediction is ", tree_predit)
D_treeAccurecy = D_Tree_model.score(xtest_scale,y_test_data)
print("Confident  of LinearRegression model is: ", accurency_linear)
print("Confident  of Decision Tree Regression model is: ", D_treeAccurecy)
LinearModel_prediction = np.array(liner_predict)
TreeeModel_predition = np.array(tree_predit)
real_data = np.array(y_test_data)
plt.plot(LinearModel_prediction)
plt.plot(TreeeModel_predition)
plt.plot(real_data)
plt.title(" 30 day forecast price from Linear Regression Vs Decision Tree Regression Vs Real
Price ")
graph = plt.subplot(111)
box = graph.get_position()
graph.set_position([box.x0, box.y0, box.width*0.65, box.height])
legend_x = 1
legend_y = 0.5
plt.legend(["TreeModel Price", "LinearModel Price","Real Price"], loc='center left',
bbox_to_anchor=(legend_x, legend_y))
plt.show()
```

7. **PCA_YahooFiniance.py**

```
"""
```

MSVZ

This program will reduce the dimension of yahoo finance 6 columns into two columns as Price and other features, to analyse if the

the behavior of the Linear regression, and svm model

dropped Adj Close because it is the same with Close column

```
"""

from sklearn.preprocessing import StandardScaler

import pandas as pd

import pandas_datareader.data as web

from matplotlib import style

import matplotlib.pyplot as plt

import math

import numpy as np


df = pd.read_csv('./Data/yahoo.csv')

df = df.set_index('Date')

print (df.head())

df.drop(["Adj Close"], axis=1, inplace=True)

print (df.head())

#-----------Data Preparation Section start herer -------------

#df = pd.DataFrame(df)

# separating the label column and named as label

label = df.iloc[:,3]

label = pd.DataFrame(label)

label.to_csv('./Data/PCA_label.csv')

#

df['Volume']= df.Volume.astype(float)

print (label)

# # separating the features columns in one dataframe

features = df.drop(df.columns[3],axis = 1)

#
```

```
print ("features are --->",features)
print (features.info())
# # standardized the all features in the dataset
standard_scaler = StandardScaler()
standard_scaler.fit(features)
# # this will transform to array
transformed_data = standard_scaler.transform(features)
print(transformed_data)
# # do the matrix Transform
Transformed_matrix = features.T
# # here finding the covariance matrix for the Eigenvectors  and Values
c_matrix = np.cov(Transformed_matrix)
# #
#print (c_matrix)
# # Now find the Eigenvalue
E_values, E_vector = np.linalg.eig(c_matrix)
#print ("Eigen Values \n", E_values)
# # got max eign value
max_Eigne = E_values.max()
# # get the percentage variance of the max Eigenvalue
sum_all_Eignen_values= sum(E_values)
#
PC1_variance_percentage = max_Eigne/sum_all_Eignen_values
PC1_variance_percentage=np.round(PC1_variance_percentage* 100, 1)
print ("\nPC1 variance percentage is ====>",PC1_variance_percentage, "%")# wow that is very
big percentage PC1
# # get PC2 percentage
second_max_eigne= E_values[1]
PC2_variance_percentage = second_max_eigne/sum_all_Eignen_values
PC2_variance_percentage=np.round(PC2_variance_percentage* 100, 1)
```

```
print ("PC2 variance percentage is ====> ",PC2_variance_percentage, "%")
# #now project the data point to the PC1
PC1 = features.dot(E_vector.T[0])
PC2 = features.dot((E_vector.T[1]))
PC1.to_csv('./Data/PC1.csv') # for later K_mean use, will save this file
#---------visualization-----------
s= 5
plt.scatter(PC1,label, s, c="g", marker='d', label="Closing Price")
plt.xlabel("PC1")
plt.ylabel("Prices")
plt.title("PC1 features vs Closing Price")
plt.legend(loc='upper left')
plt.show()
```

8. **Percentage_chage_Prediction_Yahoo.py**

```
"""
This program will predict the 30 day close price with open and close percentage change, high
and low percentage change
"""
from sklearn import svm
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import pandas as pd
import time
from matplotlib import style
import matplotlib.pyplot as plt
import math
import datetime
import numpy as np
import pickle
```

MSVZ

```
from sklearn import preprocessing
df = pd.read_csv('./Data/yahoo.csv')
df = df.set_index('Date')
style.use('ggplot')
# df.plot(kind= 'box',subplots=True, layout= (1,6),sharex=False,sharey=False)
# df.hist()
# scatter_matrix(df)
# plt.show()
#df.rename(column= {})
# percentage change for open vs close, and high vs low
df ['OpenVsClose_change'] = (df['Close']-df['Open'])/ df['Open'] * 100
#df ['HighVsLow_change'] = (df['High']-df['Low'])/ df['Low'] * 100
# # just change the data set
new_df = pd.DataFrame(df)
print("new_df is: --->", new_df)
new_df = new_df[['Close','HighVsLow_change','OpenVsClose_change','Volume']]
print (new_df)
# recording the real Closing price before testing for the forecasting the Price
original_DataFrame =pd.DataFrame(columns=['Date','Price'])
original_DataFrame =pd.DataFrame(new_df['Close'].values , columns=['Price'])
# there are 1098 tuples, - 30 is 1067
original_DataFrame = original_DataFrame[1068:1098] # This Before the 30 day of the end day
#print ("original_DataFrame is ----->", original_DataFrame)
#original_DataFrame = pd.DataFrame({'Date': next_date_array[:],'Price': Forecast_set[:]})
plt.plot (original_DataFrame['Price'])
plt.xlabel("Date")
plt.ylabel("Price")
plt.show()
#  # will forcast the Closing price
```

```
forecast_col = 'Close'
# # # get the data set length percentage is  0.1 will be in the forecasted
forecast_out = int(math.ceil(0.027* len(new_df)))# 30 days of the data out of 1098 days , accuracy with 75% to 83% swinging, +-8% change
#print ("Forcast_out is : ", forecast_out)
#
 # preparing for the empty labels for the incoming forecast
#
new_df['label']= new_df[forecast_col].shift(-forecast_out)
#print (new_df['label'])
# get x value and y value of as rest of the data column and label column
X = np.array(new_df.drop(['label'],1)) # all columns, other than label column
X = preprocessing.scale(X)
X = X[:-forecast_out]
X_lately = X[-forecast_out:]
new_df.dropna(inplace=True)
y = np.array(new_df['label'])# only label column
#print (len(X), len(y))
# # # get testing set and training set
# # #split the dataset with a random seed
# # # training size is the 90% of the data set
x_train, x_test, y_train, y_test = train_test_split(X,y,test_size=0.2)
# # Linear regression model
# linear_regression = LinearRegression()
# linear_regression.fit(x_train,y_train)
# with open ('./Data/linearregressionFitted.pickle', 'wb') as f:
#      pickle.dump(linear_regression,f)
pickled = open('./Data/linearregressionFitted.pickle','rb')
linear_regression = pickle.load(pickled)
# # not to come out the negative value in the accuracy score
```

```
LinearRegression(copy_X= True, fit_intercept=True,n_jobs=1, normalize=False)
accuracy = linear_regression.score(x_test,y_test)
# #
print ('\nLinear_regression accuracy is :', accuracy)
svm_modle= svm.SVR()
svm_modle.fit(x_train,y_train)
accuracy_SVR = svm_modle.score(x_test,y_test)
print ("svR_accurency:", accuracy_SVR)
 #predit the stock price for the bitcoin for next 0.1% of the day which is 4 day for here
Forecast_set = linear_regression.predict(X_lately)
print ("Forecast_set is :", Forecast_set)
new_df['Forecast'] = np.nan
last_date = new_df.iloc[-1].name
# print ("last date is: " , last_date)
last_date = time.mktime(datetime.datetime.strptime(last_date,"%Y-%m-%d").timetuple())
# print ("timestamp is: ",last_date)
one_day = 86400
next_unix = last_date + 86400
# print ("next unix is: ", next_unix)
## just to show the forcast_set with Price values
label_arry = np.array(new_df['label'])
next_date_array= []
# # just visualization ( later get inside the
for i in Forecast_set:
    next_date = datetime.datetime.fromtimestamp(next_unix)# might be this one wrong
    next_date = str(next_date)
    #print ("String next date is:", next_date)
    next_date= str.split(next_date," ")
    #print (("Splitted string next date is:", next_date[0]))
```

```
    next_date = next_date[0]
    next_date_array.append(next_date) # just to get an array for later use
    # this should be in function (change it later)
    next_unix +=one_day
    new_df.loc[next_date] = [np.nan for _ in range(len(new_df.columns)-1)] +[i]
print ("next_date array is: ", next_date_array)
# make a forcast vs nexdate dataset for Demo
newDf = pd.DataFrame(columns=['Date','Price'])
newDf = pd.DataFrame({'Date': next_date_array[:],'Price': Forecast_set[:]})
#original_Data= pd.DataFrame({'Date': next_date_array[:],'Price': original_DataFrame['Price']})
print ("newDf is: ---->", newDf)
# this just to get comparison price with Real Price
Forecast_DataFrame= pd.DataFrame(Forecast_set)
print ("Forecast DataFrame is ", Forecast_DataFrame)
Forecast_DataFrame.to_csv('./Data/Forecast.csv')
# let's do the same dataframe
# plt.plot(newDf['Price'])
# plt.title("Forcasted Price vs Original Price graph")
# plt.xlabel("Date")
# plt.ylabel("Prices")
# plt.show()
# print (len(next_date_array))
# print (len(Forecast_set))
#new_df['Date']= next_date_array
#new_df['Price']= Forecast_set
# print (new_df)
#-----------------------------------------------------
#new_df['Close'].plot()
new_df['Forecast'].plot()
```

```
plt.title("January 1st, 2016 To January 1st, 2019 bitcoin Stock price and 30 day forecast")
plt.legend(loc=4)
plt.xlabel('Date')
plt.ylabel('Price')
#plt.show()
```

9. **PickleDataCollection.py**

```
"""
This program will take the linear Regression model fit as pickle from both of the API dataset
"""
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import pandas as pd
from matplotlib import style
import math
import datetime
import numpy as np
import pickle
df = pd.read_csv('./Data/yahoo.csv')
#print (df.info())
df = df.set_index('Date')
#print (df.head())
style.use('ggplot')
# df.plot(kind= 'box',subplots=True, layout= (1,6),sharex=False,sharey=False)
# df.hist()
# scatter_matrix(df)
# plt.show()
#df.rename(column= {})
# percentage change for open vs close, and high vs low
df ['OpenVsClose_change'] = (df['Close']-df['Open'])/ df['Open'] * 100
```

```
df ['HighVsLow_change'] = (df['High']-df['Low'])/ df['Low'] * 100
# # just change the data set
new_df = pd.DataFrame(df)
new_df = new_df[['Close','HighVsLow_change','OpenVsClose_change','Volume']]
#print("new_df is: --->", new_df)
# recording the real Closing price before testing for the forecasting the Price
original_DataFrame =pd.DataFrame(columns=['Date','Price'])
original_DataFrame =pd.DataFrame(new_df['Close'].values , columns=['Price'])
# there are 1098 tuples, - 30 is 1067
df_size = original_DataFrame.shape[0]
original_DataFrame = original_DataFrame[df_size - 30:df_size] # This Before the 30 day of the end day
#print ("original_DataFrame is ----->", original_DataFrame)
#original_DataFrame = pd.DataFrame({'Date': next_date_array[:],'Price': Forecast_set[:]})
#  # will forcast the Closing price
forecast_col = 'Close'
# # # get the data set length percentage's 0.1 will be in the forecasted
forecast_out = int(math.ceil(0.027* len(new_df)))# 30 days of the data out of 1098 days , accuracy with 75% to 83% swinging, +-8% change
#print ("Forcast_out is : ", forecast_out)
 # preparaing for the empty labels for the incoming forcast
new_df['label']= new_df[forecast_col].shift(-forecast_out)
#print (new_df['label'])
# # # get x value and y value of as rest of the data column and label column
X = np.array(new_df.drop(['label'],1)) # all columns, other than label column
X = X[:-forecast_out]
X_lately = X[-forecast_out:]
new_df.dropna(inplace=True)
y = np.array(new_df['label'])# only label column
#print (len(X), len(y))
```

```
# # # get testing set and training set
# # #split the dataset with a random seed
# # # training size is the 90% of the data set
# #
x_train, x_test, y_train, y_test = train_test_split(X,y,test_size=0.2)
# Linear regression model
linear_regression = LinearRegression(n_jobs = -1)
linear_regression.fit(x_train,y_train)
with open ('./Data/linearregressionFitted.pickle', 'wb') as f:
    pickle.dump(linear_regression,f)
# pickled = open('./Data/linearregressionFitted.pickle','rb')
# linear_regression = pickle.load(pickled)
# # not to come out the negative value in the accurency score
LinearRegression(copy_X= True, fit_intercept=True,n_jobs=1, normalize=False)
accuracy = linear_regression.score(x_test,y_test)
print ('\nLinear_regression accuracy is :', accuracy)
```

10. **TweetStream.py**

```
#tweepy.streaming import StreamListener
import Twitter_credential
from tweepy import OAuthHandler
from tweepy import Stream
#This is a basic listener that just prints received tweets to stdout.
class StdOutListener(StreamListener):
    def on_data(self, data):
        print (data)
        return True
    def on_error(self, status):
        print (status)
```

```
if __name__ == '__main__':
    #This handles Twitter authentication and the connection to Twitter Streaming API
    Listen = StdOutListener()
    auth = OAuthHandler(Twitter_credential.CONSUMER_KEY,
Twitter_credential.CONSUMER_SECRET)
    auth.set_access_token(Twitter_credential.ACCESS_TOKEN,
Twitter_credential.ACCESS_TOKEN_SECRET)
stream = Stream(auth, Listen)
stream.filter(track=['bitcoin','currency'])
# this streaming is captured from command line redirection to the "tweet_data_testing.txt file "
```

11. **Tweet_Data_Converter.py**

```
import datetime
import pandas as pd
import json
tweets_data_path = 'tweet_data_testing.txt'
tweets_data = []
tweets_file = open(tweets_data_path, "r")
# reading line by line from json file ( Remember Json are in dictionary format)
for line in tweets_file:
    try:
        tweet = json.loads(line)
        # changing Twitter time format to python yy,m,d
        tweet_daytime = datetime.datetime.fromtimestamp(int(tweet['timestamp_ms']) / 1000)
        tweet_day = tweet_daytime.strftime('%Y-%m-%d')
        #print(tweet_day)
        # appending to the tweets_data array
        tweets_data.append(tweet)
    except:
        continue
```

```
tweets = pd.DataFrame(tweets_data)
# this replace the Date Column of the tweets with converted Date format
tweets['Date'] = tweet_day
print(" tweets['Date']:is  ", tweets['Date'])
print(tweets.info())
#print(tweets)
```

12. **Read_Json_File.py**

```
import json
import pandas as pd
import matplotlib.pyplot as plt
import datetime
tweets_data_path = 'tweet_data_testing.txt'
tweets_data = []
tweets_file = open(tweets_data_path, "r")
# reading line by line from json file
for line in tweets_file:
  try:
    tweet = json.loads(line)
    # changing Twitter time format to python yy,m,d
    tweet_daytime = datetime.datetime.fromtimestamp(int(tweet['timestamp_ms']) / 1000)
    tweet_day = tweet_daytime.strftime('%Y-%m-%d')
    # print(tweet_day)
    # appending to the tweets_data array
    tweets_data.append(tweet)
  except:
    continue
  tweets = pd.DataFrame(tweets_data)
tweets = pd.DataFrame(tweets_data)
```

```
print(tweets.info())
#tweets.to_csv('tweets.cvs')
df = pd.read_csv('tweets.cvs')
chosen_Df = pd.DataFrame(columns=['Date','Tweet','favorite_count'])
print (chosen_Df.info())
# sending data to the new dataframe
chosen_Df[['Date','Tweet','favorite_count']] = df[['created_at','text','favorite_count']]
# here Date is replacing with formated date
chosen_Df['Date'] = tweet_day
chosen_Df.to_csv('ReadyTweet.cvs')
print(chosen_Df)
print(chosen_Df['Tweet'])
print (chosen_Df.info())
```

13. **PCA_Decisiontree_LinearRegression.py**

```
"""
This program will reduce the dimension of yahoo finance 6 columns into two columns as Price and other features, to analyse if the
the behavior of the Linear regression, and svm model
dropped Adj Close because it is the same with Close column
"""
from sklearn import metrics
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import pandas as pd
import pandas_datareader.data as web
from matplotlib import style
from sklearn.tree import DecisionTreeRegressor as DTreeRegree
import matplotlib.pyplot as plt
import math
```

MSVZ

```
import numpy as np
from sklearn.model_selection import cross_val_score
import seaborn as sns; sns.set(font_scale = 1.2)
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

yahoo= pd.read_csv('./Data/yahoo.csv')
# testing for dropping date column, Adj Close column
#df_Forcast.drop(["a"], axis=1, inplace=True)
yahoo.drop(['Date','Adj Close'], axis =1, inplace= True)
yahoo['close']= yahoo['Close']
yahoo.drop(['Close'], axis =1, inplace= True)
yahoo = yahoo.dropna()
print ("Statistic summary of the data")
print (yahoo.describe())

# outlier removal with 2 standard variation

#yahoo = yahoo[np.abs(stats.zscore(yahoo)< 2).all(axis=1)] # this outliter remove increase the
root mean square error.

x = yahoo.iloc[:,: -1]# evey row and every column other than the last column
#print (x.shape)
y = yahoo.iloc[:,-1]# only the last column : this is class data
x_train_data, x_test_data, y_train_data, y_test_data = train_test_split(x,y, random_state=17,
train_size=0.975)# predit about 10 day
PCA = PCA()
TreeModel= DTreeRegree()
LinearModel = LinearRegression()
```

```
X = PCA.fit_transform(x_train_data)
TreeModel.fit(X, y_train_data)
LinearModel.fit(X,y_train_data)
X_testData_PCA_fit = PCA.transform(x_test_data)
#print (X_testData_PCA_fit)
Tree_predit = TreeModel.predict(X_testData_PCA_fit)
Linear_predit = LinearModel.predict((X_testData_PCA_fit))
Tree_score = cross_val_score(TreeModel, x_train_data, y_train_data)
Linear_score = cross_val_score(LinearModel, x_train_data, y_train_data)
print("The prediction score of PCA_Decison Tree Ressor is: ", Tree_score)
print("The prediction score of PCA_Linear Regression is: ", Linear_score)
print ("Root Mean Square Error for Decision Tree is ",
np.sqrt(metrics.mean_squared_error(y_test_data, Tree_predit)))
print ("Root Mean Square Error for linear Regression is ",
np.sqrt(metrics.mean_squared_error(y_test_data, Linear_predit)))
```