DATA ANALYTICS INTERNSHIP REPORT

TOPIC: PREDICTING THE STOCK PRICE MOVEMENT USING THE CROWD DATA ANALYSIS

TEAM MEMBERS:

ISHA SINGH

SAMARTH P SHET

UTKARSH

GEETANJALI S.

AJIT KUMAR

VISHNUPRIYA

SHISHU

INDEX

ABSTRACT

In this project, we aim to predict stock prices by using machine learning techniques on data from Stock-Twits, a social media platform for investors. We demonstrate the results, and compare the prediction error, of several classification and regression techniques using aggregated Stock-Twits messages as a source of input. The 3-day future return, calculated as a percentage change, was used as the prediction target to model the short-term correlation with social media activity, using Sentiment Analysis (for Stock-Twits data) and many other models like Support Vector Regression (SVR), KNN Regression for the Dow Jones Industrial data.

INTRODUCTION

The internet has changed the way traders trade. It gives them instant access to resources, articles, and statistics about whatever company they choose. It allows investors to connect at the speed of light. And now the common investor is not just a consumer of information, but a producer as well. Anyone with internet access can create a social media account and share their ideas at the click of a button. Traders buy, sell, and discuss in real-time leaving a paper trail in the form of Big Data. Our research dives into this dataset to find correlations between investor posts and stock prices.

Since sentiment is abundant in social media, there must be some predictive power in it. We found it can predict things from a consumer taste in music to civil voting habits. This work also discussed Stock-Twits and the public equity market. We believe similar algorithms can predict stock cost. After all, social moods and real-life sentiment are determinants of price movement. It's also proved that worried discussions lead to a downtrend in stock price. Since real-life sentiment plays a role in the market, digital sentiment must as well.

Social media platforms such as Stock-Twits can provide a wealth of information on real-world patterns and behaviours. We offer an analysis on a specific application of social media, pertaining to finance: using aggregated Stock-Twits message data to make statistically significant price predictions. Our underlying assumption is that there exists a correlation between market price action and the metrics that we extract from this aggregate sentiment, indicating that it can provide a meaningful, actionable slice of real market conditions and psychology.

In this project, we aim to focus on making a short-term correlation with the aggregated message data from Stock-Twits, using Sentiment analysis and perform analysis on the component stocks of the Dow Jones Industrial Average. Here model is formed using these two data sets and modelling techniques like Support Vector Regression and K-NN Regression.

DATASET AND FEATURES

We performed data analysis on the component stocks of the Dow Jones Industrial Average.

The Data that was collected ranges between the period of December 2013 and December 2016.

Trading days - 756

Two main datasets used:

- Yahoo finance: Daily split-adjusted price data.
- Stock-Twits: Message data collected and downloaded in raw JSON format, about 540,000 messages

Original format of the dataset: CSV

Columns in the dataset with a brief description:

- Date: Format: DD-MM-YY
- Price: Price of the stock
- Open: Open Price of stock i.e. The price at which stock opened
- High: The highest price the stock touched
- Low: The lowest price the stock touched
- Volume: The number of shares that changed hands during a given day
- Change percent

EXPLORATORY DATA ANALYSIS

In statistics, exploratory data analysis is an approach of analysing data sets to summarize their main characteristics, often using statistical graphics and other data visualization methods. A statistical model can be used or not, but primarily EDA is for seeing what the data can tell us beyond the formal modelling or hypothesis testing task.

EDA refers to exploration (or explanation of the data) of the given data for the future usage. It is a good practice that we need to understand our data and gather as many insights out of the data.

Exploratory Data Analysis on Dow Jones Industrial Average Historical Data

Dow Jones average is an average of stock prices calculated by Dow Jones & Company, Inc.

In the United States, the averages are one of the most widely used gauges of broad trends in stock and bond prices.

In 1896, Dow Jones & Company, a financial news publisher founded by Charles Henry Dow and Edward D. Jones, started generating a daily industrials average by dividing the total price of 12 companies by 12.

Since then, the list of stocks has been expanded, and the divisor has been modified to account for stock splits, stock replacements, and major dividend adjustments. As a result, the averages are not arithmetical means, but rather averages intended to show broad market price patterns.

1. Finding the null values present in the given series of objects.

```
In [12]: # finding the null values present in the given series of object.
         Dow_Jones.isnull().sum()

Out[12]: Date        0
         Price       0
         Open        0
         High        0
         Low         0
         Vol.        0
         Change %    0
         dtype: int64
```

There are no null or missing values present in the dataset

As we can see the dataset is not in the proper format, hence further the format of few attributes will be changed, as shown

2. Converting the Date Column to the datetime format, for further reference

```
In [14]: # Converting the Date Column to the datetime format, for further reference
         Dow_Jones['Date'] =Dow_Jones['Date'].apply(lambda x: x.replace(',',''))
         Dow_Jones['Date'] = pd.to_datetime(Dow_Jones['Date'])
         Dow_Jones['Date']
```

```
Out[14]: 0      2019-12-31
         1      2019-12-30
         2      2019-12-27
         3      2019-12-26
         4      2019-12-24
                   ...
         2761   2009-01-09
         2762   2009-01-08
         2763   2009-01-07
         2764   2009-01-06
         2765   2009-01-05
         Name: Date, Length: 2766, dtype: datetime64[ns]
```

3. Replacing the Price Column of the dataset

```
In [15]: # Replacing the Price Column of the dataset
         Dow_Jones['Price'] = Dow_Jones['Price'].apply(lambda x: x.replace(',',''))

         # Replacing the Open Column of the dataset
         Dow_Jones['Open'] = Dow_Jones['Open'].apply(lambda x: x.replace(',',''))

         # Replacing the High Column of the dataset
         Dow_Jones['High'] = Dow_Jones['High'].apply(lambda x: x.replace(',',''))

         # Replacing the Low Column of the dataset
         Dow_Jones['Low'] =Dow_Jones['Low'].apply(lambda x: x.replace(',',''))
```

```
In [16]: Dow_Jones
```

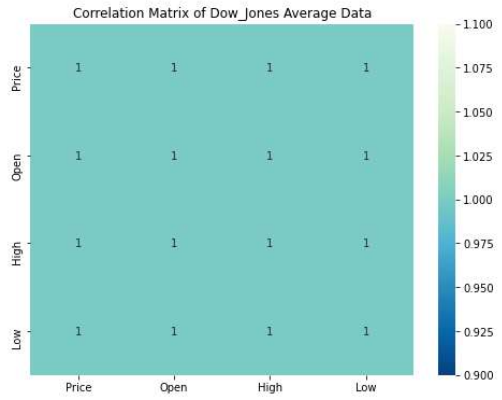
Out[16]:

	Date	Price	Open	High	Low	Vol.	Change %
0	2019-12-31	28538.44	28414.64	28547.35	28376.49	193.34M	0.27%
1	2019-12-30	28462.14	28654.76	28664.69	28428.98	185.07M	-0.64%
2	2019-12-27	28645.26	28675.34	28701.66	28608.98	184.93M	0.08%
3	2019-12-26	28621.39	28539.46	28624.10	28535.15	155.97M	0.37%
4	2019-12-24	28515.45	28572.57	28576.80	28503.21	95.29M	-0.13%
...	...	...	...	...	...	...	...
2761	2009-01-09	8599.18	8738.80	8800.45	8541.75	-	-1.64%
2762	2009-01-08	8742.46	8769.94	8807.14	8593.52	-	-0.31%
2763	2009-01-07	8769.70	8996.94	8996.94	8690.45	-	-2.72%
2764	2009-01-06	9015.10	8954.57	9175.19	8868.07	-	0.69%
2765	2009-01-05	8952.89	9027.13	9093.47	8841.70	-	-0.91%

4. Converting the format of the Price Column from object <--> Float
5. Correlation Matrix of Dow-Jones Average Data

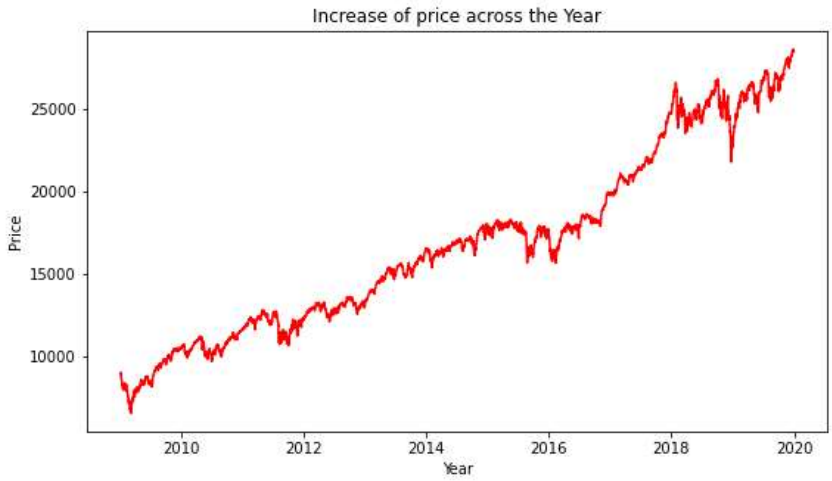
Out[23]:

	Price	Open	High	Low
Price	1.000000	0.999680	0.999840	0.999865
Open	0.999680	1.000000	0.999870	0.999793
High	0.999840	0.999870	1.000000	0.999762
Low	0.999865	0.999793	0.999762	1.000000

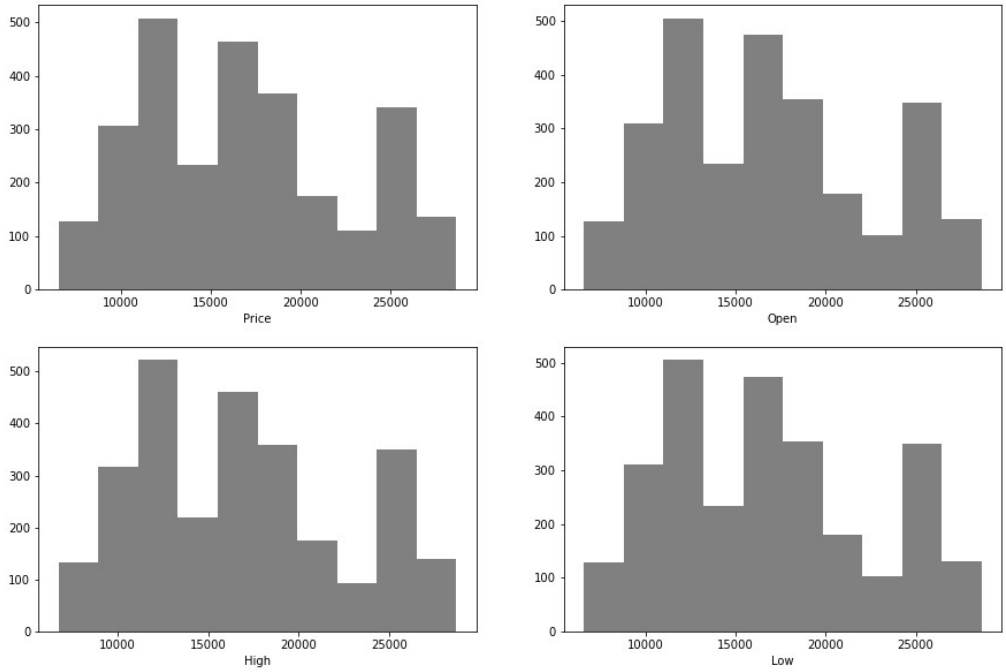


6. Increase of price across the Year

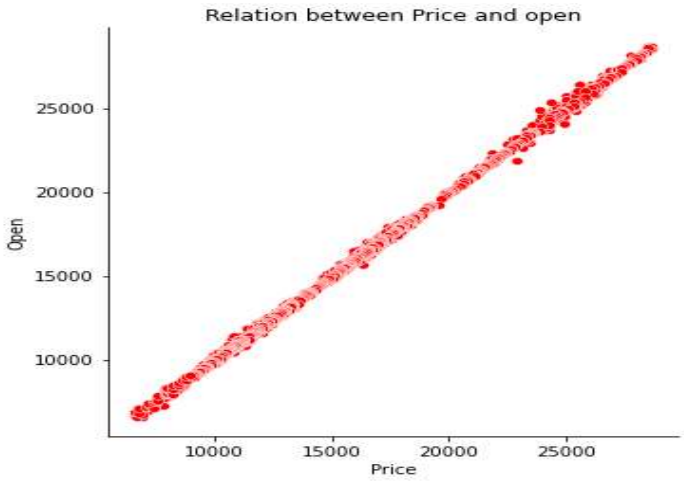
```
Text(0.5, 0, 'Year')
```



7. Price, Open, High and Low



8. Relation between price and open



Sentimental Analysis on Combined FAANG Dataset:

The aim of this work was to perform the Text processing for the given data that is the Combined FAANG binary dataset.

	symbol	message	datetime	user	message_id	Date	Time	label
0	AAPL	qq next 60min confirm start rally aapl coming ...	2015-12-21 18:37:24	191996.0	47148173.0	2015-12-21	18:37:24	1
1	AAPL	aapl watching gap fill 169 20	2018-11-24 07:02:32	1665234.0	146068732.0	2018-11-24	07:02:32	1
2	AAPL	aapl weekly options gamblers lose	2014-07-22 21:48:13	71738.0	24904954.0	2014-07-22	21:48:13	1
3	AAPL	aapl	2020-01-27 07:07:03	1229493.0	191978042.0	2020-01-27	07:07:03	0
4	AAPL	key levels watch aapl	2014-06-27 15:19:47	106412.0	24190263.0	2014-06-27	15:19:47	1

With Text-Blob, we get a polarity and a subjectivity metric. The polarity is the sentiment itself, ranging from a -1 to a +1.

In [11]:
```
FAANG_binary['polarity'] = FAANG_binary['message'].apply(lambda x: TextBlob(x).sentiment.polarity)
```

In [12]:
```
FAANG_binary.head()
```

Out[12]:

	message	Date	label	polarity
0	qq next 60min confirm start rally aapl coming ...	2015-12-21	1	0.0
1	aapl watching gap fill 169 20	2018-11-24	1	0.0
2	aapl weekly options gamblers lose	2014-07-22	1	0.0
3	aapl	2020-01-27	0	0.0
4	key levels watch aapl	2014-06-27	1	0.0

Separating the data frame into positive, negative and neutral values

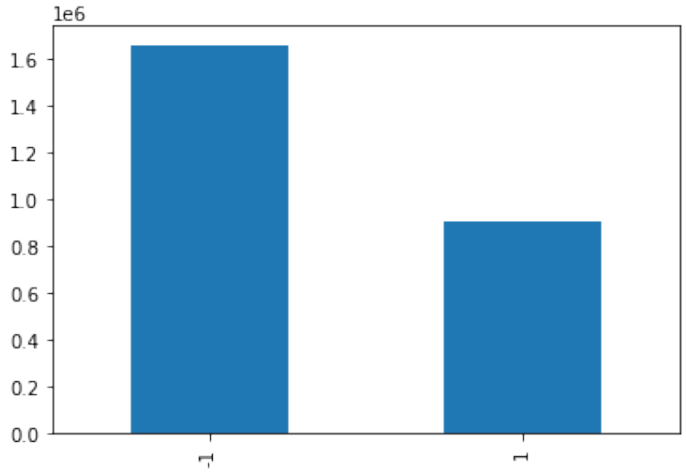
In [17]:
```
# Positive value
Postive_value = FAANG_binary[FAANG_binary['Cat_Polarity'] == 1]
Postive_value = Postive_value.reset_index(drop = True)

# Negative value
Negative_value = FAANG_binary[FAANG_binary['Cat_Polarity'] == -1]
Negative_value = Negative_value.reset_index(drop = True)

# Neutral value
Neutral_value = FAANG_binary[FAANG_binary['Cat_Polarity'] == 0]
Neutral_value = Neutral_value.reset_index(drop = True)
```

In [18]:
```
# Printing the first 5 rows of Positive value
Postive_value.head()
```

Out[18]:

	message	Date	label	polarity	Cat_Polarity
0	aapl 39 done 180 easy div payout	2018-05-15	0	0.433333	1
1	aapl 912 dow jones points week nice risk manag...	2016-01-07	0	0.442857	1
2	aapl amzn apple wil start pay taxes europe ful...	2015-05-25	0	0.350000	1
3	jim cramer simply many ways win amazon aapl	2018-08-13	1	0.650000	1
4	nflx biggest headline tomorrow beat expectatio...	2019-01-18	0	0.329924	1

The following plot represents the number of count of both positive values and the negative values:



As we can see from the above bar plot, the count for the negative values and positive values

Total Count - 2566858

1. Negative value - 1662303 [-1]
2. Positive value - 904555 [+1]

```
In [39]: FAANG_binary['stop_comments'] = FAANG_binary['message'].apply(lambda x : remove_stopwords(x))
```

```
In [40]: FAANG_binary.head()
```

Out[40]:

	message	Date	label	polarity	Cat_Polarity	stop_comments
0	aapl huh 39 happened	2015-11-12	0	0.000000	-1	aapl huh 39 happened
1	google yahoo search partnership may add shareh...	2015-07-10	1	0.000000	-1	google yahoo search partnership may add shareh...
2	aapl 39 done 180 easy div payout	2018-05-15	0	0.433333	1	aapl 39 done 180 easy div payout
3	twtr cramer likes twitter takeover thinks prob...	2015-06-16	1	0.000000	-1	twtr cramer likes twitter takeover thinks prob...
4	aapl 912 dow jones points week nice risk manag...	2016-01-07	0	0.442857	1	aapl 912 dow jones points week nice risk manag...

Implementing the Logistic regression for the dataset:

Splitting up the dataset into dependent and the independent variable:

X = FAANG_binary ['stop_comments'],

Y = FAANG_binary ['Cat_Polarity']

```
In [46]: from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
         vect = CountVectorizer()
         tf_train = vect.fit_transform(X_train)
         tf_test = vect.transform(X_test)
```

```
In [47]: tf_train.shape
```

```
Out[47]: (2053486, 162815)
```

In [49]:
```
#import Logistic Regression classifer and fit on the Training dataset
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(tf_train,y_train)
```

Out[49]:
```
LogisticRegression()
```

Calculating the accuracy of the model implemented:

In [50]:
```
# Accuracy score on training dataset
lr.score(tf_train,y_train)
```

Out[50]:
```
0.9906446890799353
```

In [51]:
```
# Accuracy on Test FAANG_binaryset
lr.score(tf_test,y_test)
```

Out[51]:
```
0.9892826254645753
```

We have obtained an accuracy of 98% using the logistic regression

In [56]:
```
print(metrics.classification_report(expected, predicted))
print("Confusion Matrix: ",metrics.confusion_matrix(expected,predicted))
```

```
              precision    recall  f1-score   support

          -1       0.99      0.99      0.99    332327
           1       0.98      0.99      0.98    181045

    accuracy                           0.99    513372
   macro avg       0.99      0.99      0.99    513372
weighted avg       0.99      0.99      0.99    513372

Confusion Matrix:  [[329360   2967]
 [  2535 178510]]
```

In [57]:
```
# Calculating F1 score
from sklearn.metrics import f1_score
f1_score(expected, predicted, average='macro')
```

Out[57]:
```
0.9882698074581957
```

MODELLING

1. SUPPORT VECTOR MACHINE:

The "Support Vector Machine" (SVM) is a supervised machine learning method that may be used for classification and regression tasks. It is, however, primarily utilised in categorization issues. Each data item is plotted as a point in n-dimensional space (where n is the number of features you have), with the value of each feature being the value of a certain coordinate in the SVM algorithm. Then, we do classification by locating the hyper-plane that best distinguishes the two classes.

The advantages of support vector machines are:

- Effective in high dimensional spaces.
- Still effective in cases where number of dimensions is greater than the number of samples.
- Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
- Versatile: different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

2. NAVIE BAYES

A classifier is a machine learning model that is used to distinguish between distinct objects based on specific attributes.

The Naive Bayes Classifier Principle: A Naive Bayes classifier is a probabilistic machine learning model used for classification tasks. The classifier's crux is based on the Bayes theorem.

Using the Bayes theorem, we may calculate the likelihood of A occurring given that B has occurred. In this case, B represents the evidence and A represents the hypothesis. The predictors/features are assumed to be independent in this case. That is, the existence of one characteristic has no effect on the presence of another. As a result, it is said to as naive.

$$\text{Bayes theorem, } P(A|B) = \frac{P(B/A)*P(A)}{P(B)}$$

Naive Bayes algorithms are commonly used in sentiment analysis, spam filtering, recommendation systems, and other applications. They are quick and simple to deploy, but their main limitation is that predictors must be independent. In most real-world instances, the predictors are dependent, which reduces the classifier's performance.

3. K-NEAREST NEIGHBOR

The k-nearest neighbours (KNN) technique is a straightforward supervised machine learning approach that may be used to tackle classification and regression issues. A supervised machine learning algorithm is one that uses labelled input data to train a function to create an acceptable output when presented with additional unlabelled data.

KNN can be done using several metrics like -

- Manhattan distance
- Euclidean distance
- Minkowski distance

Manhattan distance - $d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$

Euclidean distance - $d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$

Minkowski distance - $d(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i|^c \right)^{\frac{1}{c}}$

The KNN Algorithm

1. Load the data
2. Initialize K to your chosen number of neighbors
3. For each example in the data

3.1 Calculate the distance between the query example and the current example from the data.

3.2 Add the distance and the index of the example to an ordered collection

4. Sort the ordered collection of distances and indices from smallest to largest (in ascending order) by the distances
5. Pick the first K entries from the sorted collection
6. Get the labels of the selected K entries
7. If regression, return the mean of the K labels
8. If classification, return the mode of the K labels

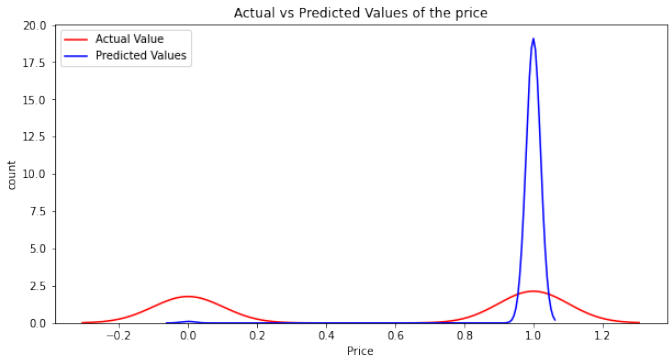
MODEL EVALUATION

1. Naïve Bayes Algorithm:

Considering the dataset "Dow Jones Industrial data" which contains different features like the date, open price, low price, high price and the percentage change in the volume feature. The 'Change %' has the values in terms of percentage, so will be converting it and create a new feature called label:

If the Price feature is greater than the Open feature, then will be labelling it as 1.

If the Price feature is smaller than the Open feature, then will be labelling it as 0.



Naïve Bayes Algorithm results

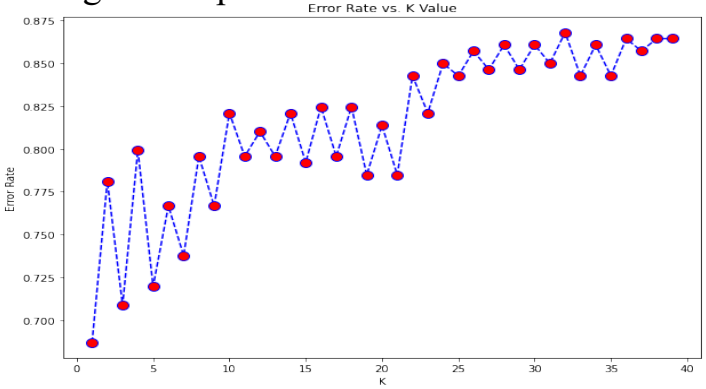
Parameter	Value
Mean Squared Error1	0.45207956
Root Mean Squared Error	0.45207956
Mean Absolute Error	0.67236864
Accuracy of the model	0.54792084
Accuracy in %	54%

Conclusion:

This complete notebook represents an Exploratory Data Analysis (EDA) process to show broad market price patterns. We have dealt with the data Dow Jones Industrial Average Historical Data. The purpose of this study is to analyse the dataset and obtain important insights form it. As visual representations are flexible and easy to understand, the results or outputs produced in the form of graphs can help people comprehend the current situation insights easily. The dataset used may not be an updated version; hence the inferences may vary from time to time as graphs can be generated as the data increases. As the amount of data increases, the trends may change and lead to different inferences and solutions. As the data was insufficient, the accuracy of the model was not upto the mark. We can use Support Vector Machine Algorithm, K-Nearest Neighbor algorithm as the accuracy of the Naive Bayes model was not Satisfactory. So we further need to decide to choose any other model which provides better accuracy.

2. K-Nearest Neighbour Algorithm:

The K-Nearest Neighbours (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problem



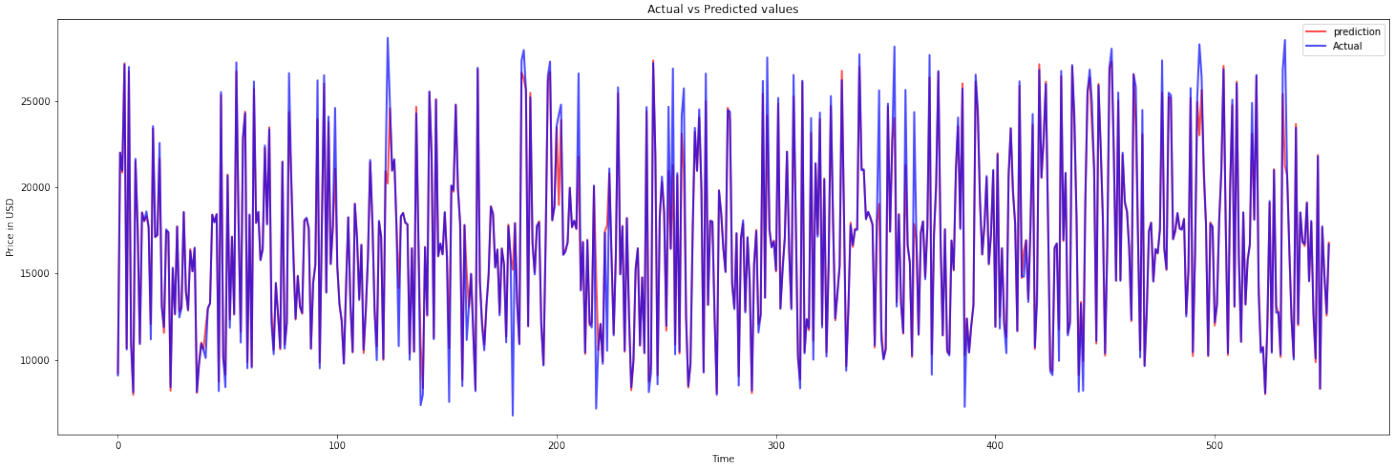
KNN Algorithm results

Parameter	Value
Mean Squared Error	0.86437956
Root Mean Squared Error	1.72207956
Mean Absolute Error	1.72368604
Accuracy of the model	0.56372084
Accuracy in %	57%

As we can see the previously used algorithms that is the Naïve Bayes algorithm and the K-Nearest Neighbour Algorithm accuracy was not satisfactory; hence we further decided to implement support vector machine algorithm and check for the accuracy.

3. Support Vector Machine Algorithm:

Support vector regression is a form of support vector machine that can handle both linear and non-linear regression. SVR necessitates the following training data: X, Y, which covers the area of interest and is supplemented with domain-specific solutions.



SVM Algorithm results

Parameter	Value
Mean Squared Error	385.8643796
Root Mean Squared Error	1347220.08
Mean Absolute Error	1152.368604
Accuracy of the model	0.956372084
Accuracy in %	96%

Comparison of all the algorithms:

Comparison of the Algorithm results

Parameter	SVM	KNN	NB
Mean Squared Error	385.8643796	0.86457566	0.45207956
Root Mean Squared Error	1347220.08	1.72245787	0.45204106
Mean Absolute Error	1152.368604	1.72368604	0.67230685
Accuracy of the model	0.956372084	0.563017208	0.548372084
Accuracy in %	96%	56%	55%

Hence, the accuracy of the SVM algorithm is better as compared to the other two algorithms (KNN & NB).

MODEL DEPLOYMENT

Model Deployment is our final phase of our project. In this phase we deploy our close price prediction model using flask, before deploying ML model we have convert ML model into model.pkl file using pickle library. In nutshell our ML model take four different input i.e ***open, high, low and volume*** and on these input value model predicts close price. Model deployment file contain five major files and these are ***svr_model.py, svr_model.pkl, app.py, dataset*** and ***template folder*** that hold ***index.html*** file.

Once we deploy our model using flask we have to start development server by app.py, and app.py provides local address and by clicking on this address index.html will be launch in default web browser where we enter input and get desired result.

ML MODEL DEPLOYMENT USING FLASK

1. Build Machine Learning Model
2. Deploy it using flask.

Files to be Created:

1. Model.py (Machine learning model)
2. Model.pkl (pickle file of the Machine Learning model)
3. App.py (Flask application file)
4. Index.html (inside the template folder)
5. The Dataset; Dow Jones Industrial Dataset

The Machine Learning model deployed result: