ANALYSIS AND VISUALIZATION OF FEEDBACK RECEIVED FROM STOCK NEWS AND STOCK PRICE FORECAST

Submitted in partial fulfillment of the requirements for the degree of

Bachelor of Technology

in

Computer Science and Engineering

by

Francis Alex Kuzhippallil	***18BCE2325***
Adith Kumar Menon	***18BCE2311***
Devoprasad Sunil Nedungade	***18BCE2337***

Under the guidance of

Mrs. Ushus Elizabeth

School of Computer Science and Engineering

VIT, Vellore.

May, 2021

TABLE OF CONTENTS

SNO	CONTENT	PG NO
1	Abstract	3
2	Introduction	3
3	Literature Survey	4
4	Overview of Proposed System	8
5	Result and Discussion	18
6	Conclusion	64
7	References	64

ABSTRACT

In Big Data driven world Stock Price Prediction has been favorite topic for both analyst and researchers. Several factors contribute towards the prediction namely physical factors, political and economic factors. Sentiment analysis of stock data based on historical data or textual information has been less precise. Existing studies in sentiment analysis have depicted high correlation between news articles based on organization and their respective stock prices. Hence it's a daunting tasks to decide the trend of stock price based on news. Hence machine learning based prediction has become pivotal. This project focuses on building a script that analyzes the feedback based on news articles of stocks on Finviz and forecasting the stock price from various sources using ML techniques. Followed by visualizing the results obtained with various tools.

Keywords: Sentiment Analysis, Stock Price Prediction, Stock Price Forecast

1. INTRODUCTION

Stock market commonly known as fortune-maker has been the mantra to successfully predict the stock prices. With the introduction of artificial intelligence and increased computational capabilities, programmed methods of prediction have proved to be more efficient in predicting stock prices. It is extremely difficult for investors to decide trend of stock prices based on the amount of news obtained. The viable options to do stock market predictions is Technical analysis and Fundamental Analysis. Technical analysis makes future predictions based on past prices and volume of stocks. Fundamental analysis involves analyzing financial data for gaining insights. This research opts both analysis techniques. Fundamental analysis is used to discover sentiments of news articles from Finviz stock screener for several organizations. Further news are classified as positive, negative or neutral or combination of two or more sentiments. This research uses technical analysis to predict stock prices and further forecast stock price for next few days using machine learning techniques namely linear regression, KNN, Decision Tree Regressor, FBProphet, Lasso, Elastic Net and LSTM. Visualization of forecast using each technique is done and comparison of stock prediction accuracy of various algorithms is analyzed as well. Yahoo finance is source used for stock price prediction and based on the data provided from yahoo finance, forecast of stock price is being done as well. The data collected from Yahoo finance is already pre-processed which includes the most common terms like 'Date, Open, High, Low, Close, Adj Close, Volume, these features are best to fit into models and make them to train on these features which turns out to be very beneficial. Using the learning curve of 1 year stock market data the model will be able to predict the coming 30 days for a particular scrip. The main solution is that making this one model to learn from the data of any given stock, is in itself one complete new feat achieved. The stocks can be from any index, they can be from NSE, BSE, NYSE, S&P 500, NASDAQ, etc. But the learning curve will be predicted through one single model.

2. LITERATURE SURVEY

2.1 Survey of the Existing Models and Works

[1] Shah discusses about the importance of stock market forecasting in up rise of business activities. Furthermore stock market has become an inter-disciplinary domain across various fields of research. Hence sentiment analysis of stock market prediction is at most sensitive yet pivotal for economy.

[2] Chien-Cheng classifies financial sentiment from US stocks from Stocktwits website followed by using natural language processing to enhance the accuracy of stock prediction. Messages received are classified as positive or negative using BERT based language model. Furthermore using NLP and BERT model for prediction has enabled a rise in accuracy of predictions.

[3] Arul explains how sentiment analysis falls under the discipline of data mining and computational semantics. Similarly he discusses about the importance of sentiment analysis from various sensitive data sources such as news, social media and so on. Furthermore Arul streamlines the focus of study in sentiment analysis prediction of stocks using VADER (Valence Aware Dictionary and Sentiment Reasoner) tool.

[4] Rakhi provides insights about how sentiment analysis have become a viable option of machine learning for extracting opinions from various segments of text from various sources including product, organization, person or an entity. Besides extraction of reviews, sentiment analysis has helped exploit the stock market industry in a tremendous way. Further they implement sentiment analysis of Stock Twits data through SVM model and measure the accuracy.

[5] Ujjwal tells that SVM and Random Forest are the prominent ML based algorithms known for accurately predicting closing prices. To prove that, dataset from India's National Stock Exchange (NSE) had been taken and effectiveness of public opinion of the company was tested. Followed by using Word2Vec model, a company specific hash-tagged posts from twitter have been taken and classified. Lastly they obtain to a conclusion that fusion of technical data indicator with positive/negative tweets doesn't have a tremendous impact on ensemble model.

[6] The main objective of this paper is to study the various methods used for sentiment analysis, which can be performed at three levels, namely at the document level, at the sentiment level and at the aspect level. Sentiment analysis consists of three main methods, identification, classification and aggregation. There are 2 methods of analysis: dictionary-based and supervised learning. In the dictionary-based method, seed words with predefined polarity values are collected manually. An algorithm is then applied that searches dictionaries such as word net to find more words of a similar nature. These new words can then be added to the list and the process can be repeated until new words are found. Supervised learning provides polarity to the new data based on a training dataset. The training data consists of input data and output variables.

[7] This paper provides an overview of how the observed technologies and methodologies are put into practice and their main shortcomings. The paper also talks about the essence of customizing datasets to collect raw, unstructured data from various industries and not stick to just a few mainstream or well known establishments. The regression model used work based on sentiment analysis on news headlines using NLP. In addition, the model is tested on real-time data to assess how well the model identifies irregularities in real industrial data.

[8] In this paper, sentimental analysis was performed by building and analyzing a sentimental dictionary with news articles. Through the sentimental dictionary it is possible to obtain the positive index of news articles for each date. By analyzing the correlation value between the positive value of the index and the value of the stock's return, the usefulness and possibility of sentimental analysis on the stock exchange is confirmed.

[9] The aggregate public opinion gathered by Twitter can be correlated with the Industrial Average Index. This paper aims to observe how changes in a company's stock prices correlate with public views expressed

in tweets about that company. The paper used two different textual representations, Word2vec and N gram, to analyze audience sentiments in tweets. In this paper, sentiment analysis and supervised machine learning principles were applied to tweets pulled from Twitter and to analyze the correlation between a company's stock market movements and sentiments in tweets. It has been observed that positive news and tweets in social media about a company would generally result in a higher investment of that company's stock and consequently that company's stock price would rise. Therefore, it can be concluded that there is a strong correlation between rising and falling stock prices with audience sentiments in tweets.

[10] In this paper, social media generated content about news articles has been used to see its effect on stock prices. The dataset was collected using the Bing API which provides links to news articles about a specific company. Two different ML algorithms were applied to the dataset. After that, a comparative analysis of their accuracies was done. In order to test the results, a general sentiment was attached to each article in the dataset which was compared with the sentiment predicted by the algorithm. In addition, a comparative study of the expected results was performed with the actual variation of the stock prices on the market.

[11] Predicting the stock market remains a difficult task due to the many influencing factors such as investor sentiment, company performance, economics, and social media sentiment. Using web news, financial tweets posted on Twitter, Google trends, and discussion forums, this study examines the association between public sentiment and the predictability of future stock price movement. Using the artificial neural network. We experimented with the proposed predictive framework with stock market data obtained from the Ghana Stock Exchange between January 2010 and September 2019 and predicted the future value of the stock for a window of 1 day, 7 days, 30 days, 60 days, and 90 days. We observed accuracy based on Google trends, on Twitter, based on a forum post, to web news, and based on a combined data set. As a result, we experienced an increase in forecast accuracy as multiple sources of inventory-related data were combined as inputs to our forecast model. Based on the results of the study, we indicated that stock market investors could use the information from online financial news, tweets, and discussion forum.

[12] In this paper, they improve the accuracy of stock price predictions by gathering a large amount of time-series data and analyzing it against related news articles, using deep learning models. The dataset they have put together includes daily stock prices of S & P500 companies for five years, as well as other 265,000 financial news articles related to these companies. Considering the large size of the data set, we use cloud

computing as an invaluable resource to train prediction models and make inferences for a given stock in real-time.

[13] In this paper, we use a 5-year financial news corpus comprising over 50,000 articles collected from the NASDAQ website for the 30 Dow Jones Index ticker symbols to form a system. Directional prediction of stock prices is based on news content and also proves that the information in the articles indicated by the break-in Tweet volumes leads to a statistically significant increase in the hourly directional prediction accuracies for the prices of the DJI shares mentioned in those articles. Second, we show that using document-level sentiment extraction does not give a statistically significant increase in directional predictive precision in the presence of other 1 gram keyword features.

[14] Forecasting stocks through the analysis of market data is an attractive topic of research. Stock prices and news articles were used in forecasting processes. However, how to combine technical stock price indicators and news sentiment from text-based press articles, and how to enable the prediction model to intelligently learn sequential information in time series, remains an unresolved issue. In this article, we build a stock forecasting system and propose an approach that 1) represents numerical price data by technical indicators via technical analysis, and represents textual press articles by sentiment vectors via market analysis. Sentiments, 2) configure a layered deep learning model to learn the sequential information in the series of market snapshots that are built by the technical indicators and sentiment of the news, 3) configure a fully connected neural network to make predictions stock market. Experiments were conducted over five years of data from the Hong Kong Stock Exchange using four dictionaries of different sentiment, and the results show that 1) the proposed approach outperforms benchmarks in validation sets and testing using two different valuation metrics, 2) models incorporating price and timeliness sentiment outperform models that only use technical indicators or timeliness sentiment, both at the stock level individual and industry level, 3) of the four sentiment dictionaries, the McDonald Financial Dictionary (Loughran) better models news sentiment, which improves prediction performance more than the other three dictionaries.
[15] This paper examines the latest machine learning method for financial news article analysis using multiple different textual representations: a bunch of words, nominal sentences, and noun Entities. Using this approach, we investigated 9,211 financial press articles and 10,259,042 Stock quotes covering S&P 500 stocks for a period of five weeks. We applied our analysis to estimate the price of a discrete share twenty minutes after the publication of a press article. Use support Derived from Vector Machine (SVM) specially designed for prediction and discrete numerical models containing different variables specific to the stock, we show that the model containing both the item the conditions and the share price at the time of publication of the article showed the best performance in terms of proximity to the real future stock price,

the same direction of price movement as the future price and highest return using the simulated trading engine.

2.2 Gaps Identified

Following are the few major gaps we have found.

1. In most of the research papers we have traversed, common machine learning algorithms such as SVM, Decision Tree are used. The only novelty that is tried is change in accuracy score.
2. All these researches are completely streamlined only to stock price prediction.
3. The choice of dataset plays a huge factor in providing accurate results. Most of the researchers follow quandl dataset whose last year of updating has been 2019.
4. Research based on sentiment analysis of stock news has been very limited.
5. Lack of comparative analysis of latest machine learning algorithms for stock price prediction and forecast.

3. OVERVIEW OF PROPOSED SYSTEM

3.1 Introduction

Stock market commonly known as fortune-maker has been the mantra to successfully predict the stock prices. With the introduction of artificial intelligence and increased computational capabilities, programmed methods of prediction have proved to be more efficient in predicting stock prices. It is extremely difficult for investors to decide trend of stock prices based on the amount of news obtained. The viable options to do stock market predictions is Technical analysis and Fundamental Analysis. Technical analysis makes future predictions based on past prices and volume of stocks. Fundamental analysis involves analyzing financial data for gaining insights. This research opts both analysis techniques. Fundamental analysis is used to discover sentiments of news articles from Finviz stock screener for several organizations. Further news are classified as positive, negative or neutral or combination of two or more sentiments. This research uses technical analysis to predict stock prices and further forecast stock price for next few days using machine learning techniques namely linear regression, KNN, Decision Tree Regressor, FBProphet, Lasso, Elastic Net and LSTM. Visualization of forecast using each technique is done and comparison of stock prediction accuracy of various algorithms is analyzed as well. Yahoo finance is source used for stock price prediction and based on the data provided from yahoo finance, forecast of stock price is being done as well. The data collected from Yahoo finance is already pre-processed which includes the most common terms like 'Date, Open, High, Low, Close, Adj Close, Volume, these features are best to fit into models and make them to train on these features which turns out to be very beneficial. Using the learning curve of 1 year stock market data the model will be able to predict the coming 30 days for a particular scrip. The main solution is that making this one model to learn from the data of any given stock, is in itself one complete new feat achieved. The stocks can be from any index, they can be from NSE, BSE, NYSE, S&P 500, NASDAQ, etc. But the learning curve will be predicted through one single model.

3.2 Motivation behind choosing this topic

The stock market is known as a place where people can make a fortune if they can crack the mantra to successfully predict stock prices. It is hard for investors to decide the trend of stock prices based on the huge amount of news. Today best prediction models are used by big hedge funds to understand the market and make the best out of every strategical trade but they don't release there models so it evident that there is no open source model which can help traders in predictions of the market. We needed to change that and wanted to introduce this model which can understand the market movement and predict a certainly good outcome.

3.3 Methodology Adopted

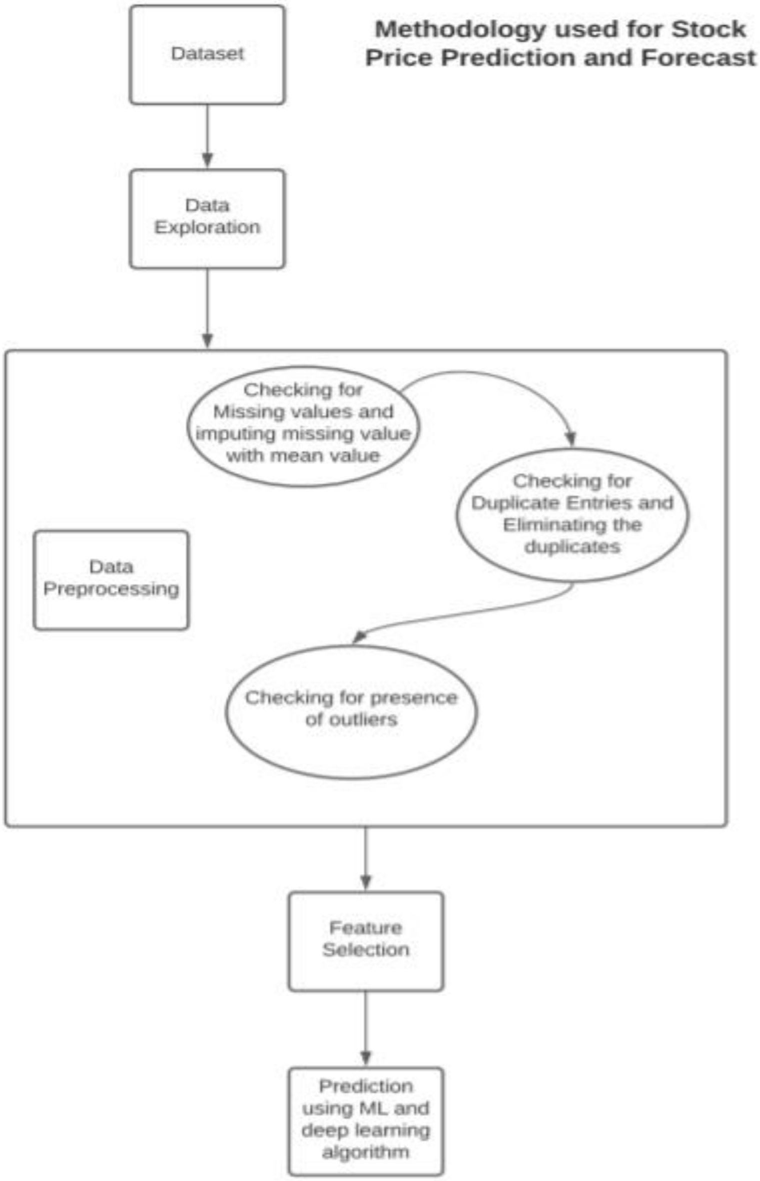
The entire project has been divided into three sections namely Stock Price Prediction, Stock Price Forecast for next 30 days and Sentiment Analysis. The implementation of each of these sections are being executed in Google Colaboratory.

Stock Price Prediction is one of the most difficult and yet the most important factor which determines the popularity and profit of the company. Hence much attention has been given to this aspect as compared to the other two sections. To predict the stock price prediction, we have selected Yahoo finance dataset. The information about the dataset is given below.

Furthermore as a part of data mining technique outliers of data is being checked and we have found none. Then we divided the data into training and testing data in 80:20 ratio. Followed by applying all algorithms namely KNN, Linear Regression, Decision Tree Regressor, Fb Prophet, LSTM, Lasso Regression and Elastic Net. The predicted value are obtained and the accuracy scores and Root Mean Square Error Value are used as comparison parameters of the performance of the algorithms. Lastly visualization of the performance is shown as well.

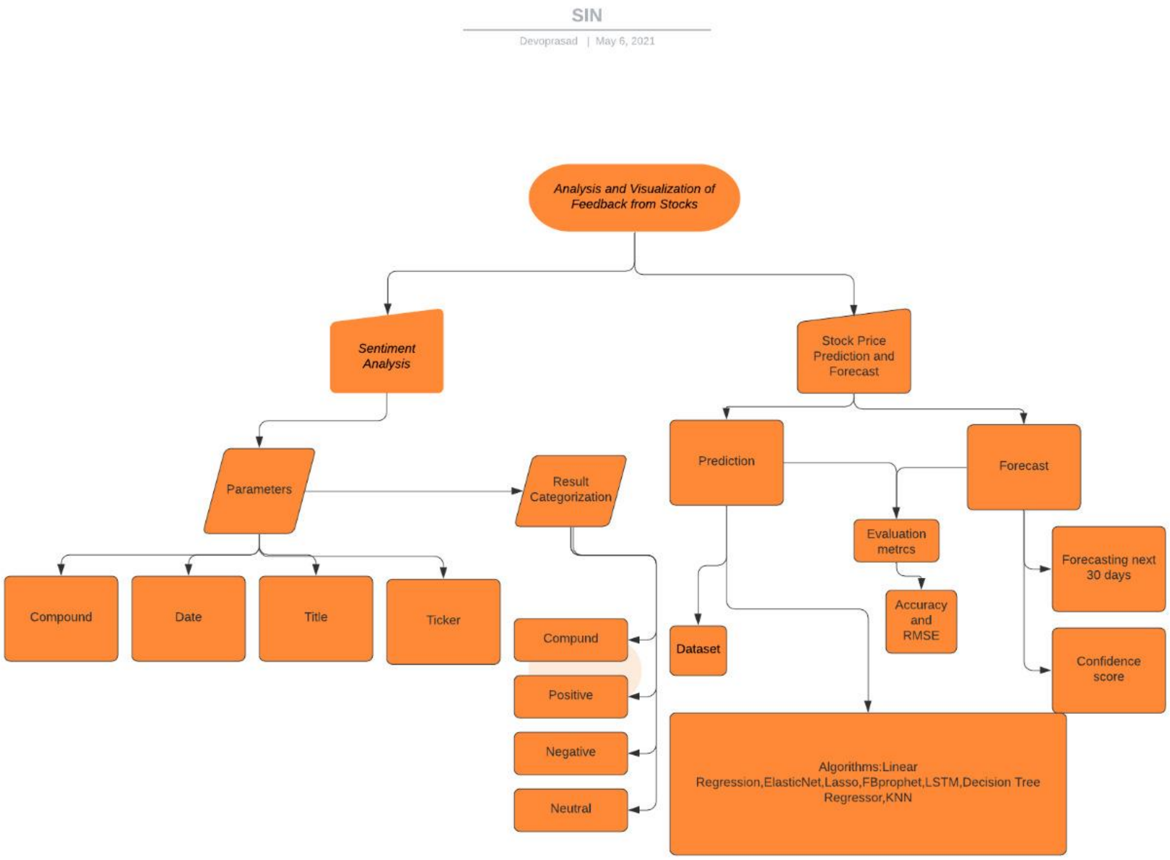
The next section of the project is Stock Price Forecast. For this demonstration, we have planned to forecast stock prices for next 30 days of the current date. For stock price forecast same dataset is being used. The algorithms used forecast is Linear Regression , LSTM , Lasso Regression , Elastic Net , KNN. The confidence score obtained for the forecast is used as a comparison parameter for evaluating the performance of the stock price forecast.

Finally the last section of our project is sentiment analysis. Mainly six modules are being imported for sentiment analysis. The first module is urlopen. This library is mainly used to import the data present in Finviz website (dataset for sentiment analysis). The next module referred is BeautifulSoup which is used for parsing the HTML data that is present in the Finviz website. The next module referred is Sentiment Intensity Analyzer. It is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media. Fourth module is pandas and it is specifically used for data manipulation. Fifth module is matplotlib.pyplot as plt.it helps us in displaying the end results in a more efficient way by using graphs for visualization. Last module is NLTK. The Natural Language Toolkit (NLTK) is a platform used for building Python programs that work with human language data for application in statistical natural language processing (NLP). There are various attributed which are taken from Finviz dataset which includes Tickers, Date, Time, Title. Here tickers act as the company variable which we take to find out the sentiment analysis. Then using the module we analyze the sentiment and categorize them under positive, negative, neutral or compound based on Vader.polarity_scores. Finally, we plot a graph for each parameter that is compound, negative, neutral, and positive to understand the variation in the analysis.



Flow of Work for Stock Price Prediction and Forecast

3.4 Modules of Proposed System



Abstract View of modules of our project

3.5 Major Libraries Imported

Matplotlib

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.

Pandas

pandas is a Python package that provides fast, flexible, and expressive data structures designed to make working with structured (tabular, multidimensional, potentially heterogeneous) and time series data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis / manipulation tool available in any language

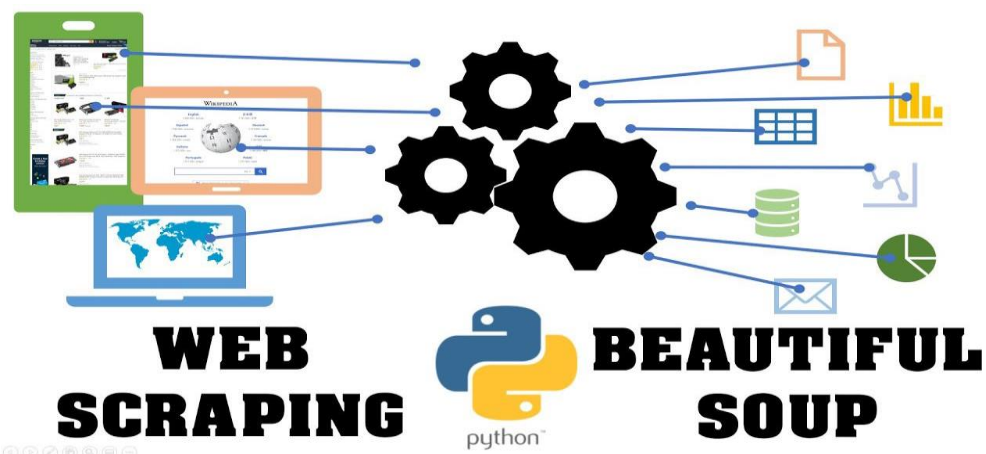
NumPy

NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, Fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely. NumPy stands for Numerical Python.

NLTK

NLTK (Natural Language Toolkit) is a suite that contains libraries and programs for statistical language processing. It is one of the most powerful NLP libraries, which contains packages to make machines understand human language and reply to it with an appropriate response.

BeautifulSoup



Beautiful Soup is a library that makes it easy to scrape information from web pages. It sits atop an HTML or XML parser, providing Python idioms for iterating, searching, and modifying the parse tree.

Sklearn

Library that supports various algorithms like linear regression, Lasso regression, Elastic Net KNN etc. It also necessary to for pre-processing, estimating accuracy score and splitting the data into training and testing data. Scikit learn library makes machine learning in python much more robust and easy.

3.6 Hardware Requirements (Minimum Requirements)

Processor	Intel(R) Core(TM) i5-6500U CPU @ 2.50GHz($ CPU), ~ 2.60GHz
Graphic Card	AMD Radeon Graphics Processor
RAM	16.0 GB
ROM	512GB SSD storage1

3.7 Methods used for Stock Price Prediction and Forecast.

A. Data Selection

Data selection process involves the need for selecting appropriate data for analysis and obtaining effective knowledge by performing diverse data mining techniques. The data used for project is Amazon Stocks Dataset from Yahoo finance.

B. Data Exploration

Data exploration is an initial step of data analysis which inculcates summarizing the data and observing initial patterns in the data and attributes. Various visualization techniques such as line chart and boxplot have been used. Feature correlation of values is assessed in order to identify highly linearly dependent features.

C. Data Pre-processing

1) *Imputation of Missing Values* - It refers to identifying missing values in the data and imputing the empty values with median values. If there is many missing values then imputation can be implemented using KNN imputation. In this scenario we haven't found any missing values, hence imputation using missing value is not required.

2) *Elimination of Duplicate Values* –In order to improve the efficiency and quality of data it's very necessary to eliminate redundant values. But there are no duplicate entries observed as well.

3) *Outlier Detection and Elimination* – Outliers are extreme values that significantly deviates from the rest of the values which may be caused due to inappropriate measurement or experimental error. Here we have checked for outliers and eliminated the outliers as well using box plot visualization.

D. Machine Learning Algorithms used

1) *Linear Regression* – In statistics, linear regression is a linear approach to modelling the relationship between a scalar response and one or more explanatory variables (also known as dependent and independent variables). The case of one explanatory variable is called simple linear regression; for more than one, the process is called multiple linear regression.

2) *Lasso Regression* - Lasso regression is a type of linear regression that uses shrinkage. Shrinkage is where data values are shrunk towards a central point, like the mean. The lasso procedure encourages simple, sparse models (i.e. models with fewer parameters).

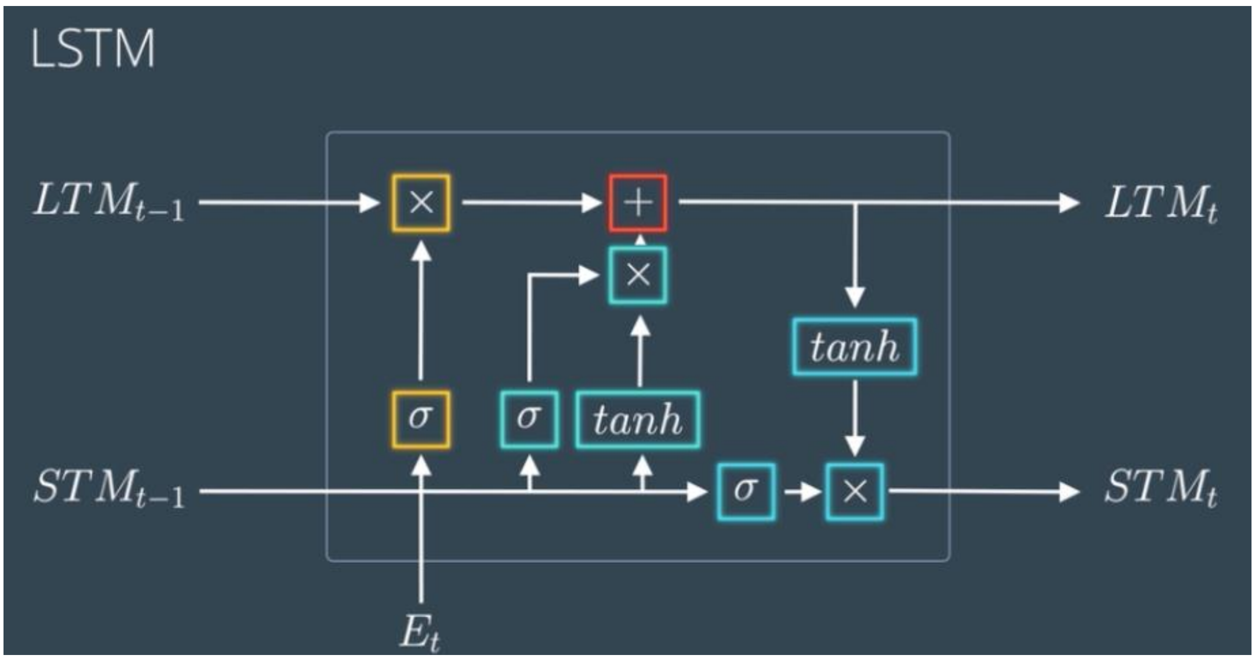
3) *Elastic Net* - Elastic net is a popular type of regularized linear regression that combines two popular penalties, specifically the L1 and L2 penalty functions. Elastic Net is an extension of linear regression that adds regularization penalties to the loss function during training.

4) *Decision Tree Regressor* - Decision tree builds regression or classification models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed.

5) *KNN* - K-Nearest Neighbors is an algorithm that works based on the close proximity of similar data points.

6) *FBProphet* - Prophet is a procedure for forecasting time series data based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects. It works best with time series that have strong seasonal effects and several seasons of historical data. Prophet is robust to missing data and shifts in the trend, and typically handles outliers well.

7) *LSTM*- Long Short-Term Memory (LSTM) networks are a type of recurrent neural network capable of learning order dependence in sequence prediction problems. This is a behavior required in complex problem domains like machine translation, speech recognition, and more. LSTMs are a complex area of deep learning. It can be hard to get your hands around what LSTMs are, and how terms like bidirectional and sequence-to-sequence relate to the field.



Architecture of LSTM

E. Performance Metrics Used

1) *Accuracy* – This performance measure is calculated by performing ratio of correctly predicted observation to the total number of observations.

2) *RMSE Value* - The root-mean-square deviation (RMSD) or root-mean-square error (RMSE) is a frequently used measure of the differences between values (sample or population values) predicted by a model or an estimator and the values observed.

F. Novelty in Our Project

The main points which we have covered in our project work are as follows:

a) Use of new machine learning algorithms namely Lasso Regression, Elastic Net, Decision Tree Regressor and FBProphet.
b) Using fundamental analysis by doing sentiment analysis and then doing technical analysis for prediction and forecast of stock prices.
c) Dataset used is yahoo finance, which is up to date and it makes it very suitable for effective prediction and better forecast of stock prices as compared to quandl dataset.
d) Comparative analysis for evaluating the performance of algorithms for effective prediction and forecast of stock prices of companies are done as well.

4. RESULT AND DISCUSSION

4.1 Dataset Information

Dataset Name	Amazon , Facebook, Intel , Google , Microsoft Stocks Information
Dataset Source	Yahoo Finance Dataset.
Dataset URL	https://finance.yahoo.com/
Dataset Attributes	7 attributes (Date, Open, High, Low, Close, Adj Close and Volume)
Dataset Type	The following data is structured data. Unsupervised Machine Learning Algorithms along with Deep Learning are used for prediction.

Dataset Information for Stock Prediction and Forecast

Dataset Name	Amazon, Google, FB Stocks Information
Dataset Source	Finviz Dataset.
Dataset URL	https://finviz.com/
Dataset Attributes	4 attributes (Date, Time, Ticker , Title)

Dataset Information for Sentiment Analysis

4.2 Screenshots of Code and Output

4.2.1 Stock Price Prediction Code Snippets

IMPORTING LIBRARIES

```
import numpy as np
import pandas as pd
import datetime

%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
plt.style.use('seaborn-darkgrid')
plt.rc('figure', figsize=(16,10))
plt.rc('lines', markersize=4)
import pandas_datareader as web
```

Importing Important Libraries

READING DATA FROM YAHOO FINANCE SOURCE

```
data=web.DataReader('AMZN',data_source='yahoo',start='2012-01-01',end='2021-05-07')
data
```

Date	High	Low	Open	Close	Volume	Adj Close
2012-01-03	179.479996	175.550003	175.889999	179.029999	5110800	179.029999
2012-01-04	180.500000	176.070007	179.210007	177.509995	4205200	177.509995
2012-01-05	178.250000	174.050003	175.940002	177.610001	3809100	177.610001
2012-01-06	184.649994	177.500000	178.070007	182.610001	7008400	182.610001
2012-01-09	184.369995	177.000000	182.759995	178.559998	5056900	178.559998
...	...	...	...	...	...	...
2021-04-30	3554.000000	3462.500000	3525.120117	3467.419922	7001800	3467.419922
2021-05-03	3486.649902	3372.699951	3484.729980	3386.489990	5875500	3386.489990
2021-05-04	3367.979980	3272.129883	3356.189941	3311.870117	5439400	3311.870117
2021-05-05	3354.699951	3264.360107	3338.860107	3270.540039	3711300	3270.540039
2021-05-06	3314.399902	3247.199951	3270.000000	3306.370117	4442600	3306.370117

2351 rows × 6 columns

Reading Amazon Stocks Data from 2012 to 2021

STATISTICAL DESCRIPTION OF DATA

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 2351 entries, 2012-01-03 to 2021-05-06
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   High       2351 non-null   float64
 1   Low        2351 non-null   float64
 2   Open       2351 non-null   float64
 3   Close      2351 non-null   float64
 4   Volume     2351 non-null   int64  
 5   Adj Close  2351 non-null   float64
dtypes: float64(5), int64(1)
memory usage: 128.6 KB
```

Information about the Attributes in the Dataset

```
[ ] data.describe()
```

	High	Low	Open	Close	Volume	Adj Close
count	2351.000000	2351.000000	2351.000000	2351.000000	2.351000e+03	2351.000000
mean	1105.855758	1081.426410	1094.425122	1093.996039	4.109262e+06	1093.996039
std	923.904244	901.122869	913.453570	912.395644	2.233355e+06	912.395644
min	178.250000	172.000000	173.809998	175.929993	8.813000e+05	175.929993
25%	328.994995	321.464996	325.220001	324.964996	2.704400e+06	324.964996
50%	767.000000	757.020020	763.000000	760.590027	3.521100e+06	760.590027
75%	1764.104980	1731.000000	1749.799988	1750.839966	4.780400e+06	1750.839966
max	3554.000000	3486.689941	3547.000000	3531.449951	2.385610e+07	3531.449951

Finding Values for different statistical measurement

```
[ ] data.columns
```

```
Index(['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close'], dtype='object')
```

```
[ ] data.corr()
```

	High	Low	Open	Close	Volume	Adj Close
High	1.000000	0.999788	0.999872	0.999846	0.131380	0.999846
Low	0.999788	1.000000	0.999822	0.999860	0.120820	0.999860
Open	0.999872	0.999822	1.000000	0.999688	0.127084	0.999688
Close	0.999846	0.999860	0.999688	1.000000	0.125889	1.000000
Volume	0.131380	0.120820	0.127084	0.125889	1.000000	0.125889
Adj Close	0.999846	0.999860	0.999688	1.000000	0.125889	1.000000

Overall, the adjusted closing price will give you a better idea of the overall value of the stock and help you make informed decisions about buying and selling, while the closing stock price will tell you the exact cash value of a share of stock at the end of the trading day.

Describing the correlation between attributes

```
[ ] df = pd.DataFrame(data, columns=['Adj Close'])
    df = df.reset_index()
```

```
[ ] df.head()
```

	Date	Adj Close
0	2012-01-03	179.029999
1	2012-01-04	177.509995
2	2012-01-05	177.610001
3	2012-01-06	182.610001
4	2012-01-09	178.559998

Choosing Adjusted Close as our target Variable

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2351 entries, 0 to 2350
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Date       2351 non-null   datetime64[ns]
 1   Adj Close  2351 non-null   float64
dtypes: datetime64[ns](1), float64(1)
memory usage: 36.9 KB
```

```
[ ] df.isna().values.any()
```

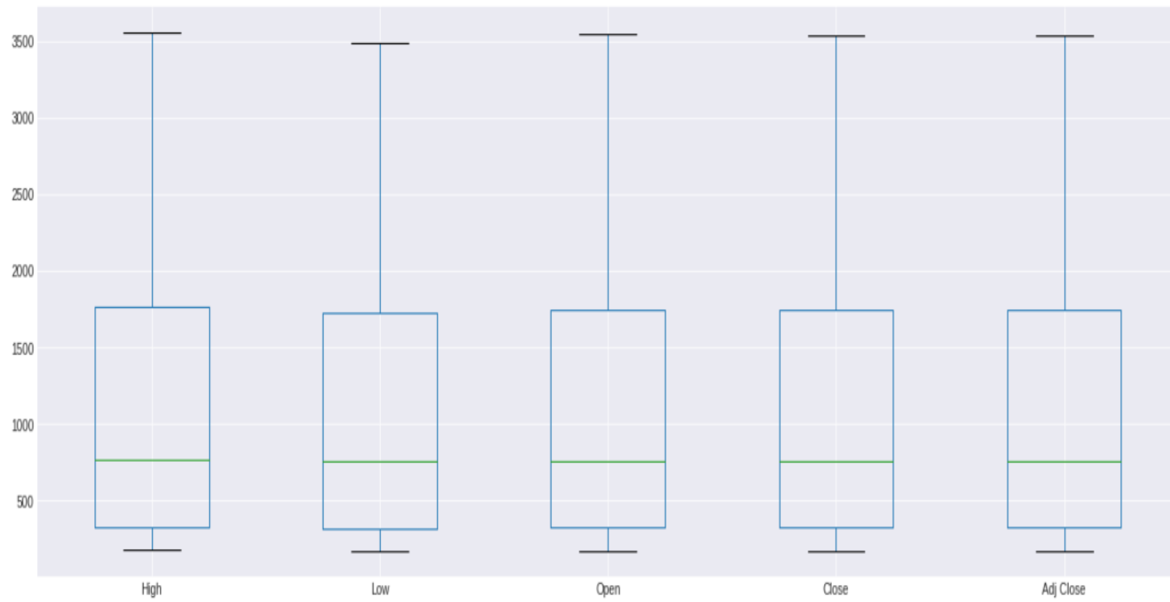
```
False
```

Checking if any missing data points are present

VISUALIZATION OF DATA

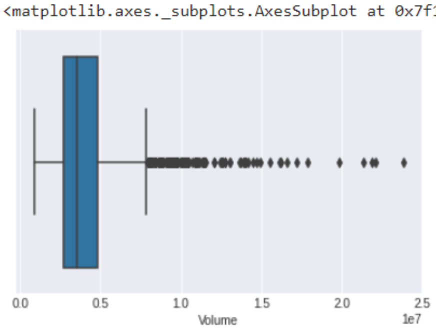
1. Box Plot

```
import seaborn as sns
plt.figure(figsize=(20,7))
boxplot = data.boxplot(column=['High', 'Low', 'Open','Close','Adj Close'])
```




```
[ ] d = pd.DataFrame(data)
    import seaborn as sns
    sns.boxplot(x=d['Volume'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f173d822210>
```



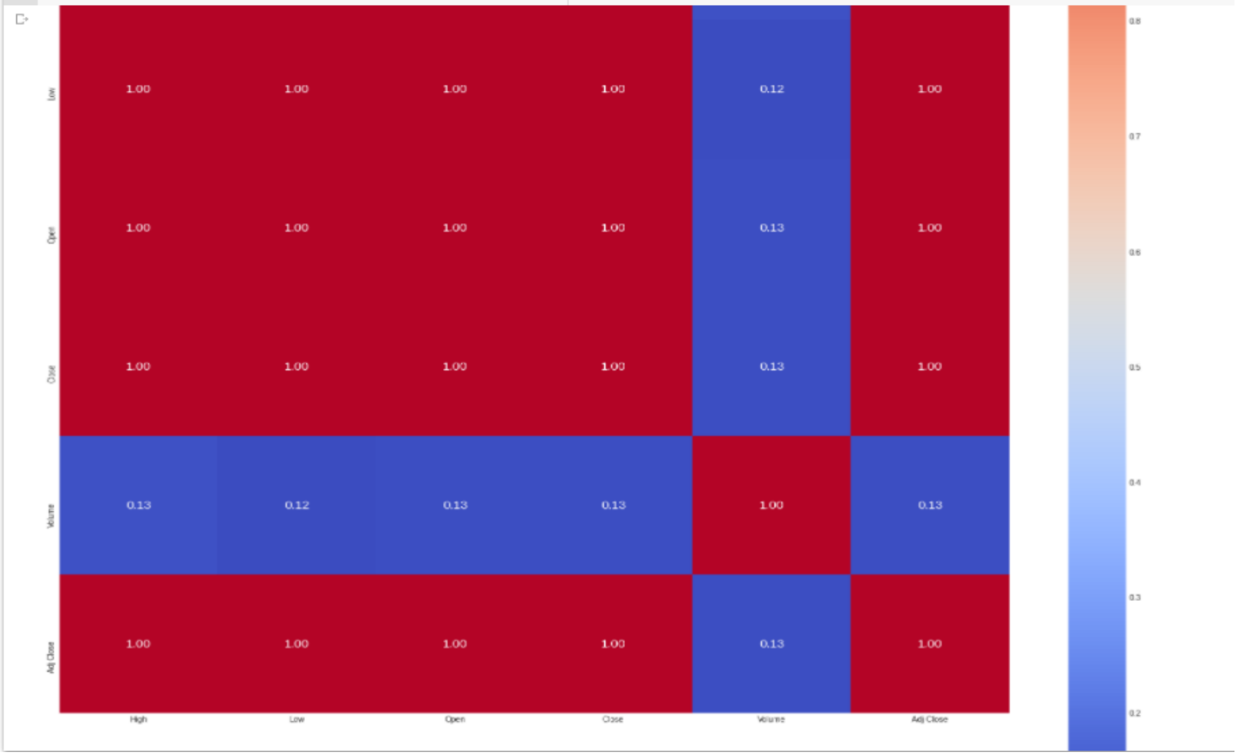
This shows that the volume column contains lot of outlier values which might affect the results of prediction. Hence we remove the volume column itself as part of outlier elimination.

Boxplot visualization of Attributes to detect the presence of outliers.

```python
import matplotlib.dates as mdates

years = mdates.YearLocator() # Get every year
yearsFmt = mdates.DateFormatter('%Y') # Set year format

# Create subplots to plot graph and control axes
fig, ax = plt.subplots()
ax.plot(df['Date'], df['Adj Close'])

# Format the ticks
ax.xaxis.set_major_locator(years)
ax.xaxis.set_major_formatter(yearsFmt)

# Set figure title
plt.title('Adjusted Close Stock Price History [2009 - 2021] For Amazon Data', fontsize=16)
# Set x label
plt.xlabel('Date', fontsize=14)
# Set y label
plt.ylabel('Adjusted Closing Stock Price in $', fontsize=14)

# Rotate and align the x labels
fig.autofmt_xdate()

# Show plot
plt.show()
```

Line Chart Visualization of Adjusted Stock Price vs Date

```
plt.figure(figsize=(25, 25))
sns.heatmap(correlation, cbar = True,  square = True, annot=True, fmt= '.2f',annot_kws={'size': 15},
           cmap= 'coolwarm')
plt.title('Correlation between features')
```



Correlation

PREDICTING THE ACCURACY OF PREDICTION OF STOCK PRICES USING ML AND DEEP LEARNING

\+ Code   + Text

1. PREDICTION USING LINEAR REGRESSION

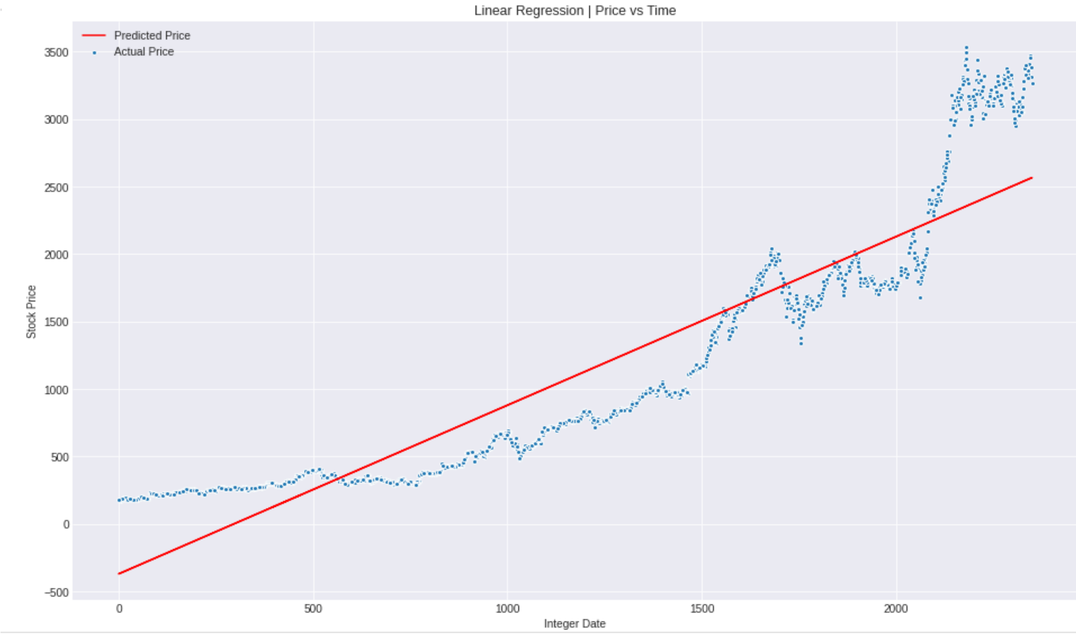
```
from sklearn.model_selection import train_test_split
train, test = train_test_split(df, test_size=0.20)
from sklearn.linear_model import LinearRegression
X_train = np.array(train.index).reshape(-1, 1)
y_train = train['Adj Close']
model = LinearRegression()
# Fit linear model using the train data set
model.fit(X_train, y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
print('Slope: ', np.asscalar(np.squeeze(model.coef_)))
# The Intercept
print('Intercept: ', model.intercept_)
```

```
Slope:  1.249504075867493
Intercept:  -369.3468996188524
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: DeprecationWarning: np.asscalar(a) is deprecated since NumPy v1.16, use a.item() instead
  """Entry point for launching an IPython kernel.
```

```
plt.figure(1, figsize=(16,10))
plt.title('Linear Regression | Price vs Time')
plt.scatter(X_train, y_train, edgecolor='w', label='Actual Price')
plt.plot(X_train, model.predict(X_train), color='r', label='Predicted Price')
plt.xlabel('Integer Date')
plt.ylabel('Stock Price')
plt.legend()
plt.show()
```



Linear Regression Best Fit Line

```
X_test = np.array(test.index).reshape(-1, 1)
y_test = test['Adj Close']
y_pred = model.predict(X_test)
dfr=pd.DataFrame({'Actual Price':y_test,'Predicted Price':y_pred})
print(dfr)
```

```
      Actual Price  Predicted Price
720     311.510010       530.296035
1985   1734.709961      2110.918691
1240    764.719971      1180.038154
1018    575.020020       902.648250
306     256.019989        13.001348
...            ...              ...
2062   1689.150024      2207.130505
1365   1010.070007      1336.226164
2257   3206.179932      2450.783800
2241   3195.340088      2430.791734
1262    796.919983      1207.527244

[471 rows x 2 columns]
```

Comparison of Actual Price Vs Predicted Price

Calculating the accuracy of the prediction

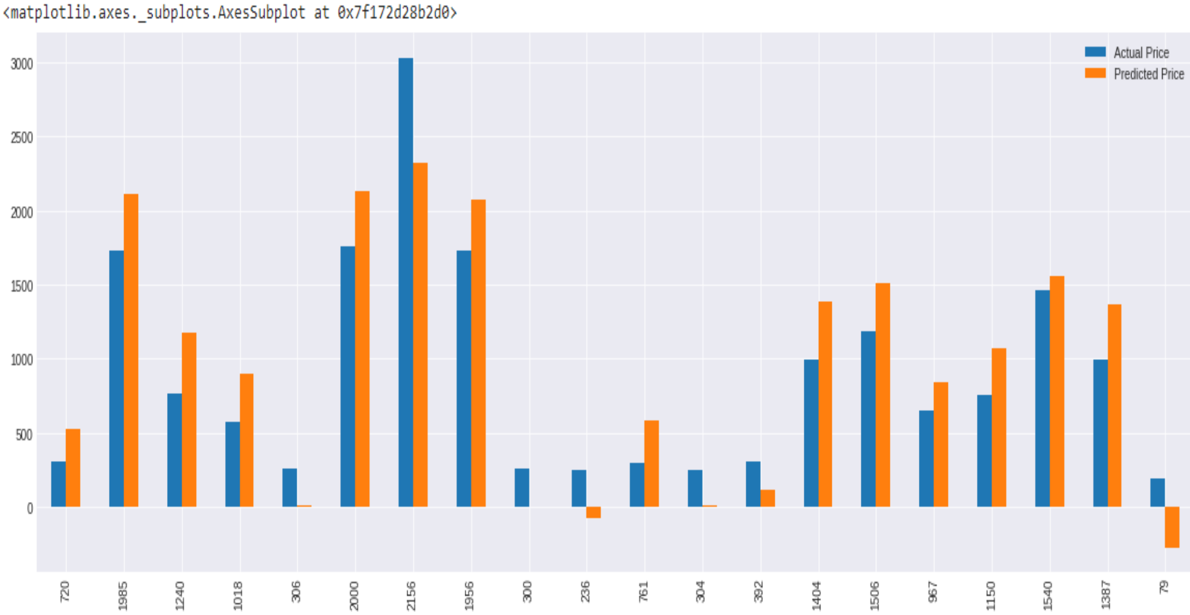
```
from sklearn import metrics
import math
rmse_linear=math.sqrt(metrics.mean_squared_error(y_test,y_pred))
accuracy_linear=model.score(X_test,y_test)

print("Root mean square error= ",rmse_linear)
print("Accurcy Score= ",accuracy_linear*100)
```

```
Root mean square error=  347.8990331607842
Accurcy Score=  83.45656722375315
```

Accuracy of Logistic Regression Prediction

```
graph=dfr.head(20)
graph.plot(kind='bar',figsize=(20,7))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f172d28b2d0>
```



Bar Graph Comparison of Actual vs Predicted Price using Logistic Regression

2.PREDICTION USING KNN

```
[ ]  from sklearn.model_selection import train_test_split
     from sklearn.neighbors import KNeighborsRegressor
     train, test = train_test_split(df, test_size=0.20)
     X_train = np.array(train.index).reshape(-1, 1)
     y_train = train['Adj Close']
     model = KNeighborsRegressor()
     model.fit(X_train, y_train)
```

```
KNeighborsRegressor(algorithm='auto', leaf_size=30, metric='minkowski',
                    metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                    weights='uniform')
```

```
[ ]  X_test = np.array(test.index).reshape(-1, 1)
     y_test = test['Adj Close']
     y_pred = model.predict(X_test)
     dfr=pd.DataFrame({'Actual Price':y_test,'Predicted Price':y_pred})
     print(dfr)
```

```
      Actual Price  Predicted Price
1473   1132.880005      1125.496021
518     386.279999       391.550000
575     324.910004       320.782001
1998   1748.719971      1748.347974
965     625.309998       635.480005
...            ...              ...
1059    553.979980       567.816003
421     295.859985       295.321997
682     331.320007       324.978003
1248    757.770020       768.276001
1772   1696.199951      1659.115991

[471 rows x 2 columns]
```

KNN Prediction

2.1 Calculating the Accuracy

```
[ ]  from sklearn import metrics
     import math
     rmse_knn=math.sqrt(metrics.mean_squared_error(y_test,y_pred))
     accuracy_knn=model.score(X_test,y_test) #Accuracy Score

     print("Root mean square error= ",rmse_knn)
     print("Accurcy Score= ",accuracy_knn*100)
```

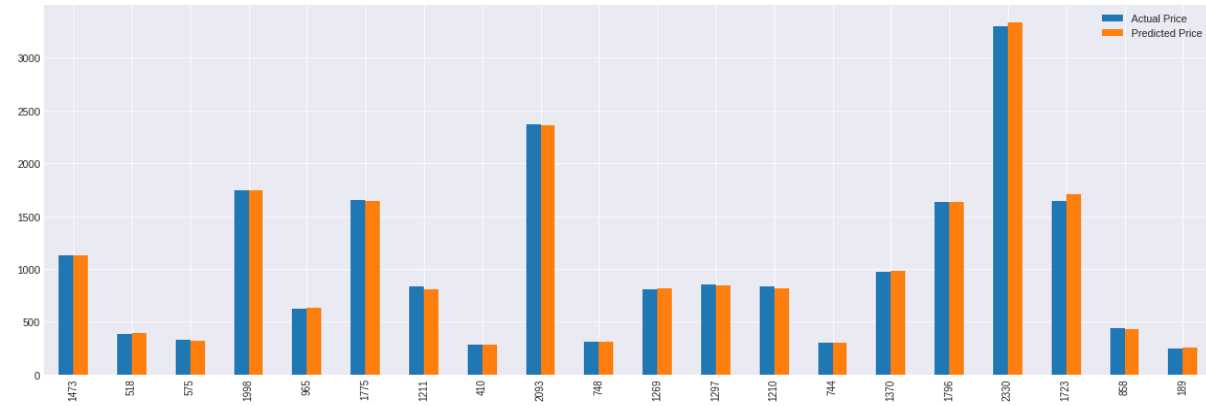
```
Root mean square error=  24.41542817964098
Accurcy Score=  99.92530808271285
```

Accuracy of Prediction of KNN

2.2 Graphical Representation

```
graph=dfr.head(20)
graph.plot(kind='bar',figsize=(20,7))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f172d192dd0>
```



Graphical Representation of Actual Price vs Predicted Price

3.PREDICTION USING DECISION TREE REGRESSOR

```
[ ]  from sklearn.model_selection import train_test_split
     from sklearn.tree import DecisionTreeRegressor
     train, test = train_test_split(df, test_size=0.20)
     X_train = np.array(train.index).reshape(-1, 1)
     y_train = train['Adj Close']
     model =DecisionTreeRegressor ()
     model.fit(X_train, y_train)
```

```
DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=None,
                      max_features=None, max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, presort='deprecated',
                      random_state=None, splitter='best')
```

```
[ ]  X_test = np.array(test.index).reshape(-1, 1)
     y_test = test['Adj Close']
     y_pred = model.predict(X_test)
     dfr=pd.DataFrame({'Actual Price':y_test,'Predicted Price':y_pred})
     print(dfr)
```

```
      Actual Price  Predicted Price
1842   1911.520020      1926.520020
2132   2734.399902      2754.580078
1592   1569.680054      1582.260010
721     316.480011       327.820007
945     537.479980       543.679993
...            ...              ...
904     522.619995       529.460022
656     326.279999       319.320007
1273    836.520020       839.150024
744     298.880005       295.059998
1997   1739.209961      1749.510010

[471 rows x 2 columns]
```

Decision Tree Prediction

3.1 Calculating the Accuracy

```
[ ]  from sklearn import metrics
     import math
     rmse_dt=math.sqrt(metrics.mean_squared_error(y_test,y_pred))
     accuracy_dt=model.score(X_test,y_test) #Accuracy Score

     print("Root mean square error= ",rmse_dt)
     print("Accurcy Score= ",accuracy_dt*100)
```

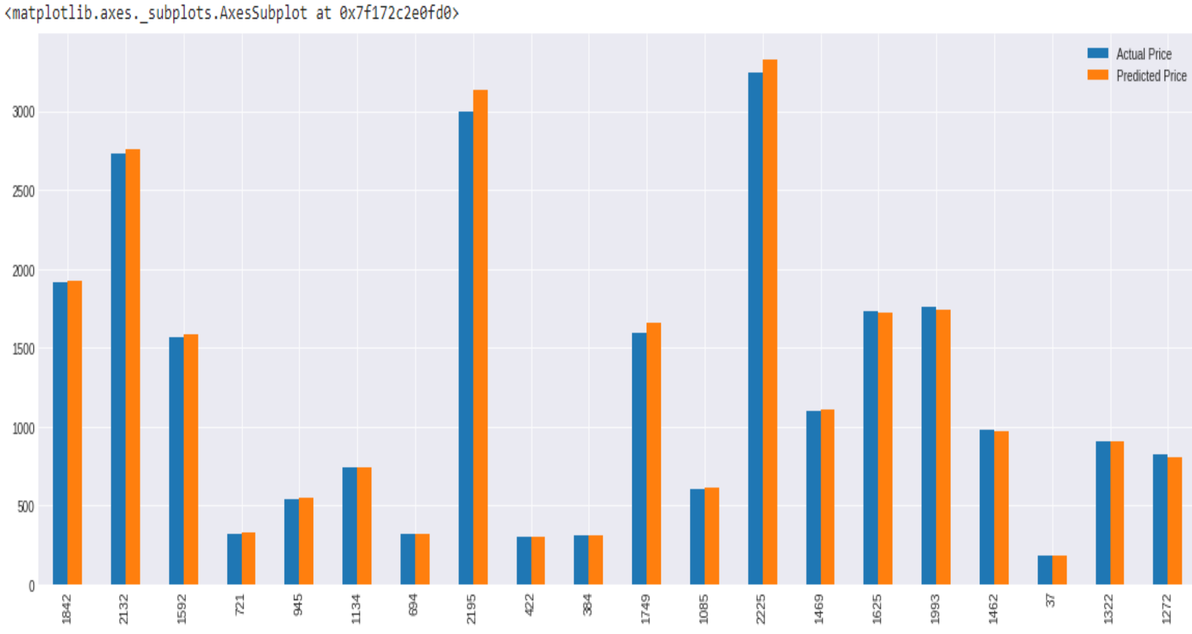
```
Root mean square error=  27.123890855351352
Accurcy Score=  99.9082687270988
```

Accuracy Score

3.2 Graphical Representation

```
[ ]  graph=dfr.head(20)
     graph.plot(kind='bar',figsize=(20,7))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f172c2e0fd0>
```



Graphical Representation of Actual Price vs Predicted Price

4.PREDICTION USING LASSO

```
[ ]  from sklearn.model_selection import train_test_split
     train, test = train_test_split(df, test_size=0.20)
     from sklearn.linear_model import Lasso
     X_train = np.array(train.index).reshape(-1, 1)
     y_train = train['Adj Close']
     model = Lasso()
     # Fit linear model using the train data set
     model.fit(X_train, y_train)
```

```
Lasso(alpha=1.0, copy_X=True, fit_intercept=True, max_iter=1000,
      normalize=False, positive=False, precompute=False, random_state=None,
      selection='cyclic', tol=0.0001, warm_start=False)
```

Lasso Regression

```
[ ]  X_test = np.array(test.index).reshape(-1, 1)
     y_test = test['Adj Close']
     y_pred = model.predict(X_test)
     dfr=pd.DataFrame({'Actual Price':y_test,'Predicted Price':y_pred})
     print(dfr)
```

```
      Actual Price  Predicted Price
2155   3000.330078      2307.461231
1915   1762.959961      2009.923818
508     397.660004       265.610737
1444    956.400024      1426.006646
803     373.350006       631.333806
...            ...              ...
141     217.050003      -189.373557
940     496.070007       801.178079
734     316.500000       545.791800
596     296.760010       374.707788
1169    759.219971      1085.078360

[471 rows x 2 columns]
```

Actual Price vs Predicted Price Values

4.1 Calculating the Accuracy

```
[ ] from sklearn import metrics
    import math
    rmse_lasso=math.sqrt(metrics.mean_squared_error(y_test,y_pred))
    accuracy_lasso=model.score(X_test,y_test) #Accuracy Score

    print("Root mean square error= ",rmse_lasso)
    print("Accurcy Score= ",accuracy_lasso*100)
```

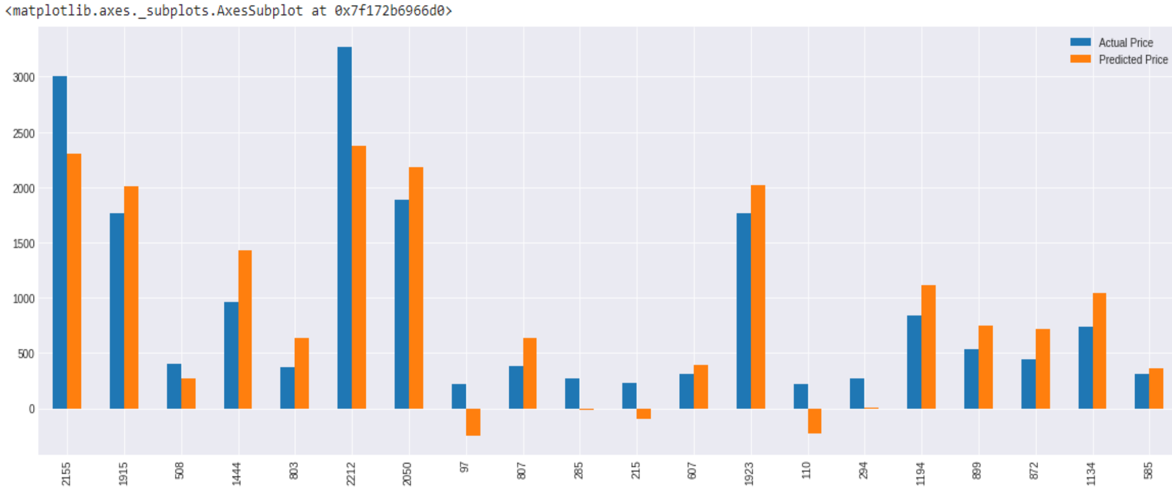
```
Root mean square error=  358.3675536904625
Accurcy Score=  84.50017895049093
```

Accuracy Score

4.2 Graphical Representation

```
[ ] graph=dfr.head(20)
    graph.plot(kind='bar',figsize=(20,7))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f172b6966d0>
```



Graphical Representation of Actual Price vs Predicted Price

5. PREDICTION USING ELASTIC NET

```
[ ] from sklearn.model_selection import train_test_split
    train, test = train_test_split(df, test_size=0.20)
    from sklearn.linear_model import ElasticNet
    X_train = np.array(train.index).reshape(-1, 1)
    y_train = train['Adj Close']
    model = ElasticNet()
    # Fit linear model using the train data set
    model.fit(X_train, y_train)

    ElasticNet(alpha=1.0, copy_X=True, fit_intercept=True, l1_ratio=0.5,
               max_iter=1000, normalize=False, positive=False, precompute=False,
               random_state=None, selection='cyclic', tol=0.0001, warm_start=False)
```

```
[ ] X_test = np.array(test.index).reshape(-1, 1)
    y_test = test['Adj Close']
    y_pred = model.predict(X_test)
    dfr=pd.DataFrame({'Actual Price':y_test,'Predicted Price':y_pred})
    print(dfr)

          Actual Price  Predicted Price
    1879   1913.900024      1961.205061
    306     256.019989        22.959919
    729     335.040009       544.179052
    1505   1176.760010      1500.363559
    133     218.389999      -190.210080
    ...            ...              ...
    1204    822.960022      1129.472403
    449     310.489990       199.164023
    604     312.549988       390.154485
    2289   3322.939941      2466.405639
    1217    765.559998      1145.490958

    [471 rows x 2 columns]
```

Elastic Net

5.1 Calculating the Accuracy

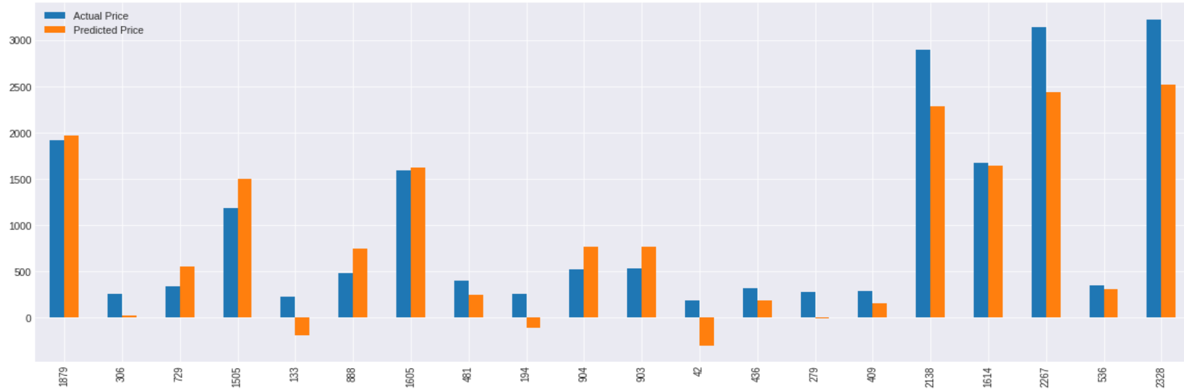
```
from sklearn import metrics
import math
rmse_el=math.sqrt(metrics.mean_squared_error(y_test,y_pred))
accuracy_el=model.score(X_test,y_test) #Accuracy Score

print("Root mean square error= ",rmse_el)
print("Accurcy Score= ",accuracy_el*100)

Root mean square error=  366.5804485968547
Accurcy Score=  84.16144578536725
```

Accuracy Scores

5.2 Graphical Representation

```
[ ] graph=dfr.head(20)
    graph.plot(kind='bar',figsize=(20,7))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f172b595ad0>
```



Graphical Representation of Actual Price vs Predicted Price

6. PREDICTION USING LSTM STACKED ESTIMATOR

```
[ ] df1=data.reset_index()['Adj Close']
```

```
[ ] from sklearn.preprocessing import MinMaxScaler
    scaler=MinMaxScaler(feature_range=(0,1))
    df1=scaler.fit_transform(np.array(df1).reshape(-1,1))
```

```
[ ] training_size=int(len(df1)*0.65)
    test_size=len(df1)-training_size
    train_data,test_data=df1[0:training_size:],df1[training_size:len(df1),:1]
```

```
[ ] import numpy

    def create_dataset(dataset,time_step=1):
      dataX,dataY=[],[]
      for i in range(len(dataset)-time_step-1):
        a=dataset[i:(i+time_step),0]
        dataX.append(a)
        dataY.append(dataset[i+time_step,0])
      return numpy.array(dataX),numpy.array(dataY)
```

```
[ ]  time_step=100
     X_train,y_train=create_dataset(train_data,time_step)
     X_test,ytest=create_dataset(train_data,time_step)
```

LSTM

```
[ ] model.fit(X_train,y_train,validation_data=(X_test,ytest),epochs=100,batch_size=64,verbose=1)
```

```
Epoch 1/100
23/23 [==============================] - 13s 317ms/step - loss: 0.0064 - val_loss: 6.1998e-04
Epoch 2/100
23/23 [==============================] - 5s 231ms/step - loss: 2.7820e-04 - val_loss: 1.0217e-04
Epoch 3/100
23/23 [==============================] - 5s 232ms/step - loss: 8.8104e-05 - val_loss: 7.5009e-05
Epoch 4/100
23/23 [==============================] - 5s 234ms/step - loss: 7.0624e-05 - val_loss: 7.3300e-05
Epoch 5/100
23/23 [==============================] - 5s 234ms/step - loss: 7.2567e-05 - val_loss: 6.7042e-05
Epoch 6/100
23/23 [==============================] - 5s 230ms/step - loss: 7.3446e-05 - val_loss: 8.1407e-05
Epoch 7/100
23/23 [==============================] - 5s 238ms/step - loss: 8.2100e-05 - val_loss: 7.2895e-05
Epoch 8/100
23/23 [==============================] - 5s 239ms/step - loss: 7.8991e-05 - val_loss: 7.2691e-05
Epoch 9/100
23/23 [==============================] - 6s 241ms/step - loss: 7.2705e-05 - val_loss: 6.4573e-05
Epoch 10/100
23/23 [==============================] - 5s 239ms/step - loss: 6.4968e-05 - val_loss: 6.0216e-05
Epoch 11/100
23/23 [==============================] - 5s 238ms/step - loss: 6.4504e-05 - val_loss: 6.2306e-05
Epoch 12/100
23/23 [==============================] - 5s 238ms/step - loss: 7.5307e-05 - val_loss: 6.8699e-05
Epoch 13/100
23/23 [==============================] - 5s 238ms/step - loss: 6.3898e-05 - val_loss: 6.3270e-05
Epoch 14/100
23/23 [==============================] - 5s 241ms/step - loss: 5.5010e-05 - val_loss: 6.8423e-05
Epoch 15/100
23/23 [==============================] - 5s 236ms/step - loss: 6.3654e-05 - val_loss: 6.2295e-05
Epoch 16/100
```

Fitting of data

```
[ ] train_predict=model.predict(X_train)
    test_predict=model.predict(X_test)
```

```
train_predict=scaler.inverse_transform(train_predict)
test_predict=scaler.inverse_transform(test_predict)
```

6.1 Calculating the Accuracy

```
[ ] import math
    from sklearn.metrics import mean_squared_error
    rmse_lstm=math.sqrt(mean_squared_error(ytest,test_predict))
    print("RMSE score of LSTM =",rmse_lstm)
```

```
RMSE score of LSTM = 595.7037364467519
```

Accuracy Score

```
[ ] import datetime as dt
    import matplotlib.pyplot as plt
    import warnings
    warnings.simplefilter(action='ignore',category=FutureWarning)
    from fbprophet import Prophet
```

```
[ ] model=Prophet()
```

```
[ ] dfb=data.reset_index()
```

```
[ ] dfb
```

	Date	High	Low	Open	Close	Volume	Adj Close
0	2012-01-03	179.479996	175.550003	175.889999	179.029999	5110800	179.029999
1	2012-01-04	180.500000	176.070007	179.210007	177.509995	4205200	177.509995
2	2012-01-05	178.250000	174.050003	175.940002	177.610001	3809100	177.610001
3	2012-01-06	184.649994	177.500000	178.070007	182.610001	7008400	182.610001
4	2012-01-09	184.369995	177.000000	182.759995	178.559998	5056900	178.559998
...	...	...	...	...	...	...	...
2346	2021-04-30	3554.000000	3462.500000	3525.120117	3467.419922	7001800	3467.419922
2347	2021-05-03	3486.649902	3372.699951	3484.729980	3386.489990	5875500	3386.489990
2348	2021-05-04	3367.979980	3272.129883	3356.189941	3311.870117	5439400	3311.870117
2349	2021-05-05	3354.699951	3264.360107	3338.860107	3270.540039	3701700	3270.540039
2350	2021-05-06	3290.620117	3247.199951	3270.000000	3266.820068	1829709	3266.820068

2351 rows × 7 columns

```
[ ] dfb[['ds','y']]=dfb[['Date','Close']]
```

Fb Prophet

```
[ ] model.fit(dfb)

    INFO:fbprophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
    <fbprophet.forecaster.Prophet at 0x7f16ecfd3f10>
```

```
    prediction=model.make_future_dataframe(periods=200)
```

```
[ ] pre=model.predict(prediction)
    print(pre)

                 ds        trend  ...  multiplicative_terms_upper        yhat
    0    2012-01-03   197.076164  ...                         0.0   146.124361
    1    2012-01-04   197.195776  ...                         0.0   147.884815
    2    2012-01-05   197.315388  ...                         0.0   146.650561
    3    2012-01-06   197.435000  ...                         0.0   144.761495
    4    2012-01-09   197.793835  ...                         0.0   147.593160
    ...         ...          ...  ...                         ...          ...
    2546 2021-11-18  4066.555149  ...                         0.0  4044.804644
    2547 2021-11-19  4069.358897  ...                         0.0  4045.068237
    2548 2021-11-20  4072.162645  ...                         0.0  4036.939893
    2549 2021-11-21  4074.966393  ...                         0.0  4039.703799
    2550 2021-11-22  4077.770141  ...                         0.0  4053.893470

    [2551 rows x 19 columns]
```

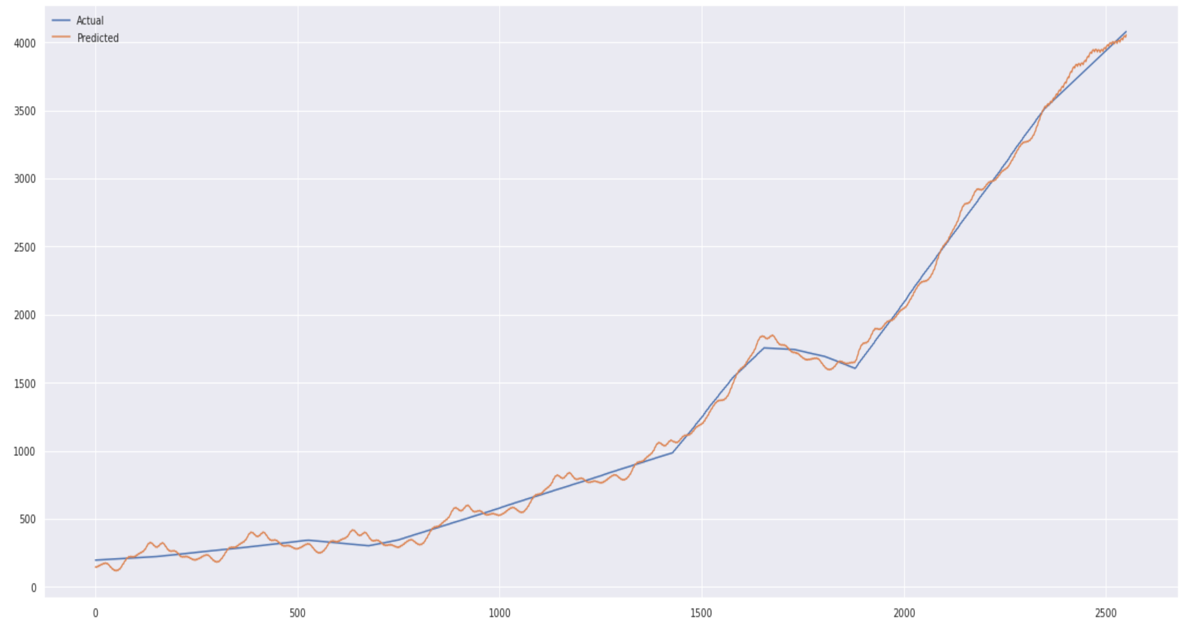
7.1 Calclulating the Accuracy

```
[ ] import math
    from sklearn.metrics import mean_squared_error
    rmse_fb=math.sqrt(mean_squared_error(pre['trend'],pre['yhat']))
    print("RMSE error of FBProphet =",rmse_fb)

    RMSE error of FBProphet = 53.90936450539521
```

Accuracy Score

7.2 Graphical Visualization

```
from matplotlib import pyplot
pyplot.plot(pre['trend'], label='Actual')
pyplot.plot(pre['yhat'], label='Predicted')
pyplot.legend()
pyplot.show()
```



Graphical Visualization of Actual vs Predicted Price

Evaluation of the Performance of Algorithms

1. BASED ON THE ACCURACY SCORE

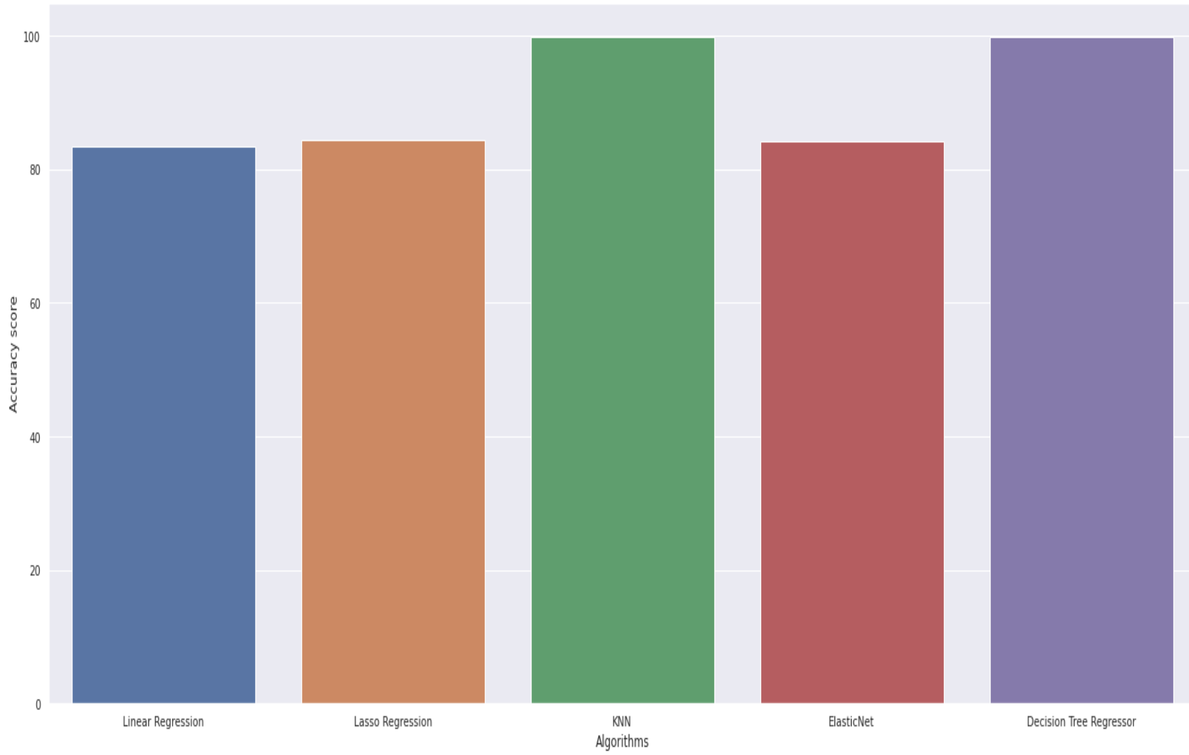
```
scores = [accuracy_linear*100,accuracy_lasso*100,accuracy_knn*100,accuracy_el*100,accuracy_dt*100]
algorithms = ["Linear Regression","Lasso Regression","KNN","ElasticNet","Decision Tree Regressor"]
for i in range(len(algorithms)):
    print("The accuracy score achieved using "+algorithms[i]+" is: "+str(scores[i])+" %")
```

```
The accuracy score achieved using Linear Regression is: 83.45656722375315 %
The accuracy score achieved using Lasso Regression is: 84.50017895049093 %
The accuracy score achieved using KNN is: 99.92530808271285 %
The accuracy score achieved using ElasticNet is: 84.16144578536725 %
The accuracy score achieved using Decision Tree Regressor is: 99.9082687270988 %
```

Stock Price Prediction Evaluation Based on Accuracy Scores

```
[ ] sns.set(rc={'figure.figsize':(25,10)})
    plt.xlabel("Algorithms")
    plt.ylabel("Accuracy score")

    sns.barplot(algorithms,scores)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f16eac43550>



Graphical Comparison of Accuracy Scores

2. BASED ON RMSE VALUES

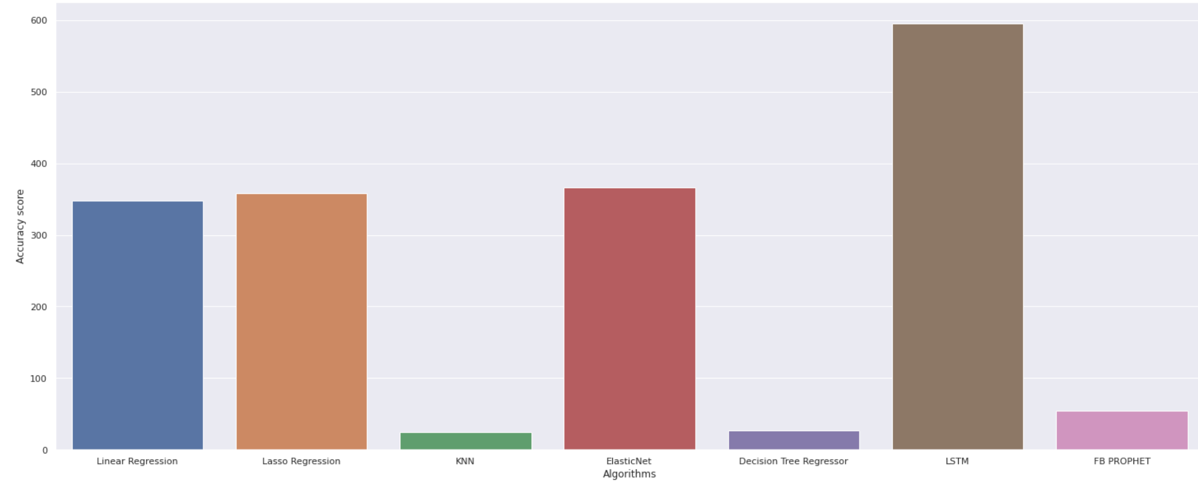
```
scores = [rmse_linear,rmse_lasso,rmse_knn,rmse_el,rmse_dt,rmse_lstm,rmse_fb]
algorithms = ["Linear Regression","Lasso Regression","KNN","ElasticNet","Decision Tree Regressor","LSTM","FB PROPHET"]
for i in range(len(algorithms)):
    print("The accuracy score achieved using "+algorithms[i]+" is: "+str(scores[i])+" ")
```

```
The accuracy score achieved using Linear Regression is: 347.8990331607842
The accuracy score achieved using Lasso Regression is: 358.3675536904625
The accuracy score achieved using KNN is: 24.41542817964098
The accuracy score achieved using ElasticNet is: 366.5804485968547
The accuracy score achieved using Decision Tree Regressor is: 27.123890855351352
The accuracy score achieved using LSTM is: 595.7037364467519
The accuracy score achieved using FB PROPHET is: 53.90936450539521
```

Stock Price Prediction Evaluation Based on RMSE Value

```
sns.set(rc={'figure.figsize':(25,10)})
plt.xlabel("Algorithms")
plt.ylabel("Accuracy score")

sns.barplot(algorithms,scores)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f16eaa64410>
```



Graphical Comparison of RMSE Scores

Note: The above code snippets were for Prediction on Amazon Data. Similarly prediction was done on Intel and Facebook Stock dataset as well. In order to reduce the number of pages we are including the analysis part here.

1. BASED ON THE ACCURACY SCORE

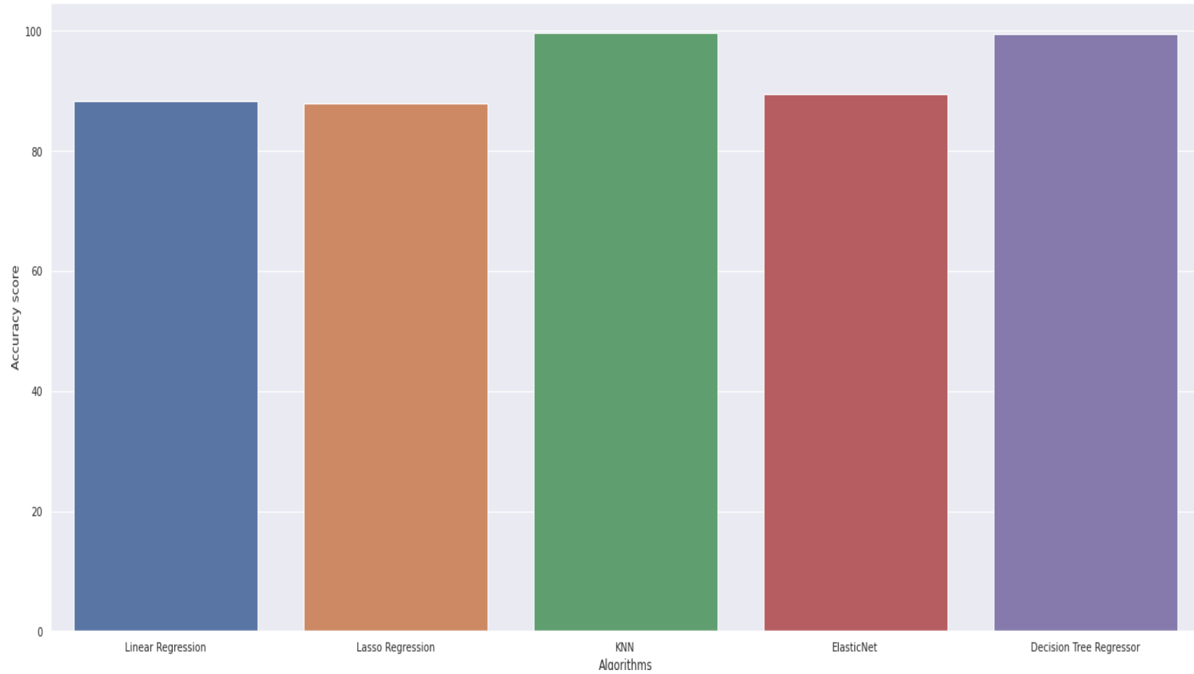
```
[ ] scores = [accuracy_linear*100,accuracy_lasso*100,accuracy_knn*100,accuracy_el*100,accuracy_dt*100]
    algorithms = ["Linear Regression","Lasso Regression","KNN","ElasticNet","Decision Tree Regressor"]
    for i in range(len(algorithms)):
        print("The accuracy score achieved using "+algorithms[i]+" is: "+str(scores[i])+" %")
```

```
The accuracy score achieved using Linear Regression is: 88.32310797132934 %
The accuracy score achieved using Lasso Regression is: 87.88689663926792 %
The accuracy score achieved using KNN is: 99.76600661903066 %
The accuracy score achieved using ElasticNet is: 89.48196822797844 %
The accuracy score achieved using Decision Tree Regressor is: 99.54990082527561 %
```

Accuracy Scores for Intel Dataset

```
sns.set(rc={'figure.figsize':(25,10)})
plt.xlabel("Algorithms")
plt.ylabel("Accuracy score")

sns.barplot(algorithms,scores)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fa683073810>



Graphical Visualization of Accuracy Scores for Intel Stock Price

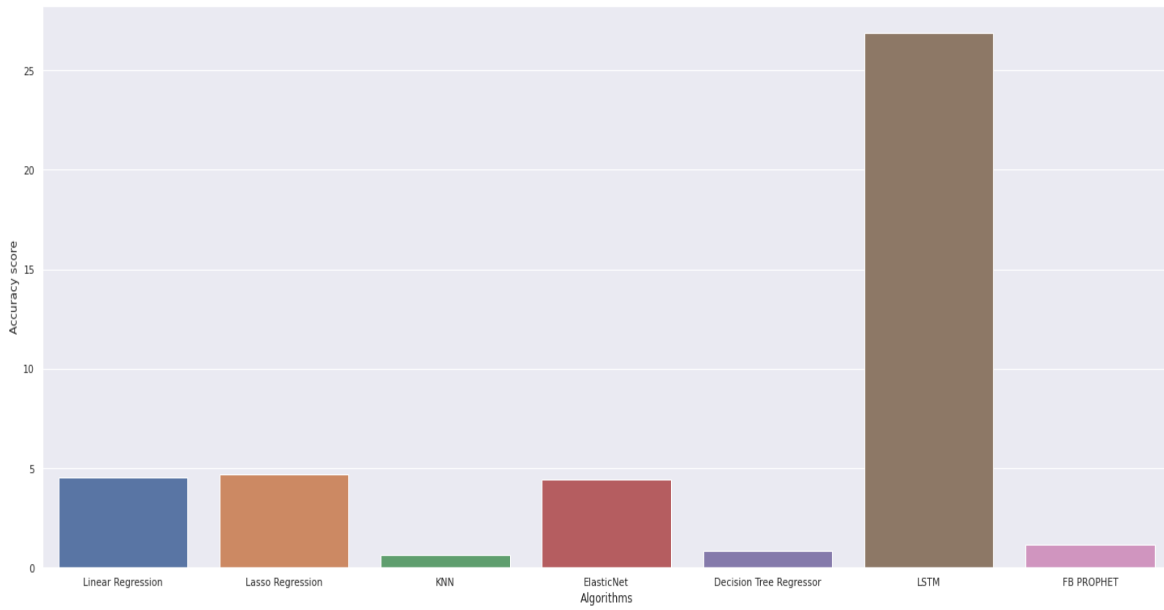
2. BASED ON RMSE VALUES

```
[ ] scores = [rmse_linear,rmse_lasso,rmse_knn,rmse_el,rmse_dt,rmse_lstm,rmse_fb]
    algorithms = ["Linear Regression","Lasso Regression","KNN","ElasticNet","Decision Tree Regressor","LSTM","FB PROPHET"]
    for i in range(len(algorithms)):
        print("The accuracy score achieved using "+algorithms[i]+" is: "+str(scores[i])+" ")
```

```
The accuracy score achieved using Linear Regression is: 4.5139969566163245
The accuracy score achieved using Lasso Regression is: 4.671341996471193
The accuracy score achieved using KNN is: 0.6280963151380109
The accuracy score achieved using ElasticNet is: 4.444493578834416
The accuracy score achieved using Decision Tree Regressor is: 0.8436449783524984
The accuracy score achieved using LSTM is: 26.892166801634513
The accuracy score achieved using FB PROPHET is: 1.1663313308995444
```

RMSE Scores for Intel Dataset

```
[ ] sns.set(rc={'figure.figsize':(25,10)})
    plt.xlabel("Algorithms")
    plt.ylabel("Accuracy score")

    sns.barplot(algorithms,scores)

<matplotlib.axes._subplots.AxesSubplot at 0x7fa682f6c910>
```



Graphical Representation for RMSE Scores for Intel Data

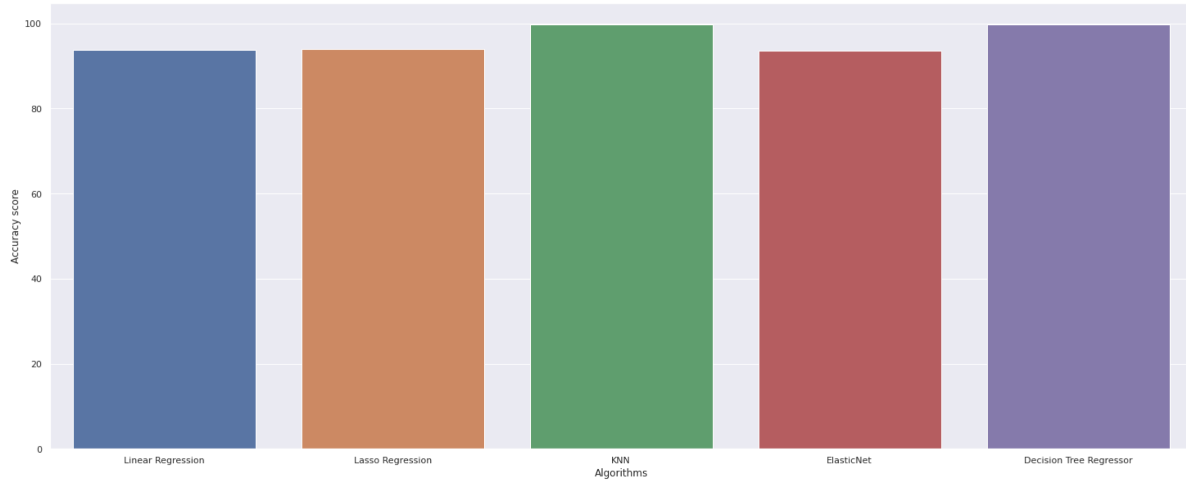
1. BASED ON THE ACCURACY SCORE

```
[ ] scores = [accuracy_linear*100,accuracy_lasso*100,accuracy_knn*100,accuracy_el*100,accuracy_dt*100]
    algorithms = ["Linear Regression","Lasso Regression","KNN","ElasticNet","Decision Tree Regressor"]
    for i in range(len(algorithms)):
        print("The accuracy score achieved using "+algorithms[i]+" is: "+str(scores[i])+" %")

    The accuracy score achieved using Linear Regression is: 93.82343589670482 %
    The accuracy score achieved using Lasso Regression is: 93.91015377554682 %
    The accuracy score achieved using KNN is: 99.83836544106552 %
    The accuracy score achieved using ElasticNet is: 93.61823136954321 %
    The accuracy score achieved using Decision Tree Regressor is: 99.80104149322598 %
```

Accuracy Score for Facebook Data

```
sns.set(rc={'figure.figsize':(25,10)})
plt.xlabel("Algorithms")
plt.ylabel("Accuracy score")

sns.barplot(algorithms,scores)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f40023f9ad0>
```



Graphical Visualization of Accuracy Scores for Facebook Stock Price

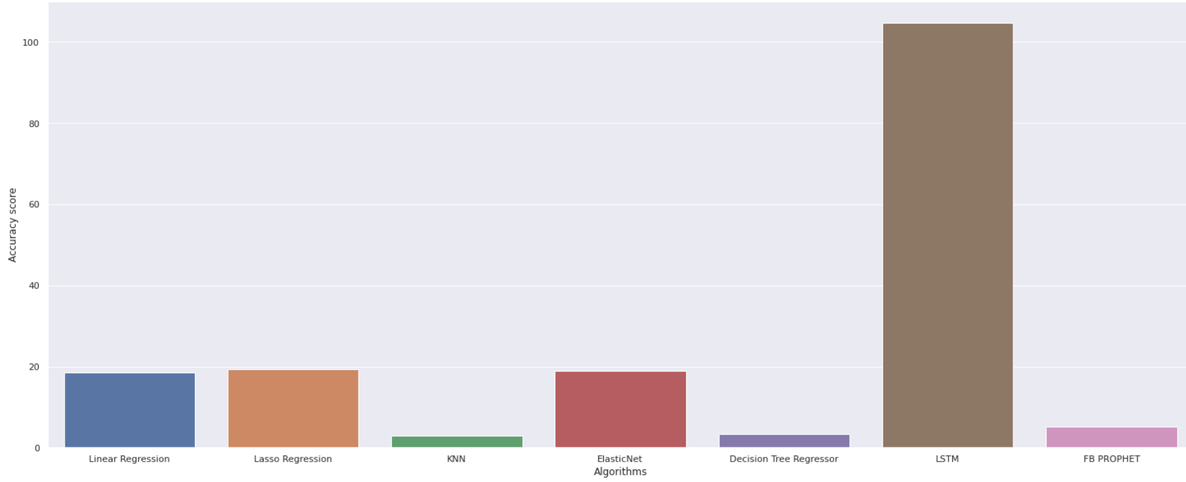
2. BASED ON RMSE VALUES

```
scores = [rmse_linear,rmse_lasso,rmse_knn,rmse_el,rmse_dt,rmse_lstm,rmse_fb]
algorithms = ["Linear Regression","Lasso Regression","KNN","ElasticNet","Decision Tree Regressor","LSTM","FB PROPHET"]
for i in range(len(algorithms)):
    print("The accuracy score achieved using "+algorithms[i]+" is: "+str(scores[i])+" ")
```

```
The accuracy score achieved using Linear Regression is: 18.495384690750118
The accuracy score achieved using Lasso Regression is: 19.27941619349092
The accuracy score achieved using KNN is: 2.930899236276912
The accuracy score achieved using ElasticNet is: 18.950476455935522
The accuracy score achieved using Decision Tree Regressor is: 3.310082366740375
The accuracy score achieved using LSTM is: 104.81994455237668
The accuracy score achieved using FB PROPHET is: 5.153324587087542
```

RMSE Scores for Facebook Data

```
[ ] sns.set(rc={'figure.figsize':(25,10)})
    plt.xlabel("Algorithms")
    plt.ylabel("Accuracy score")

    sns.barplot(algorithms,scores)

<matplotlib.axes._subplots.AxesSubplot at 0x7f4042b33590>
```



Graphical Visualization of RMSE value for Facebook Stock Price

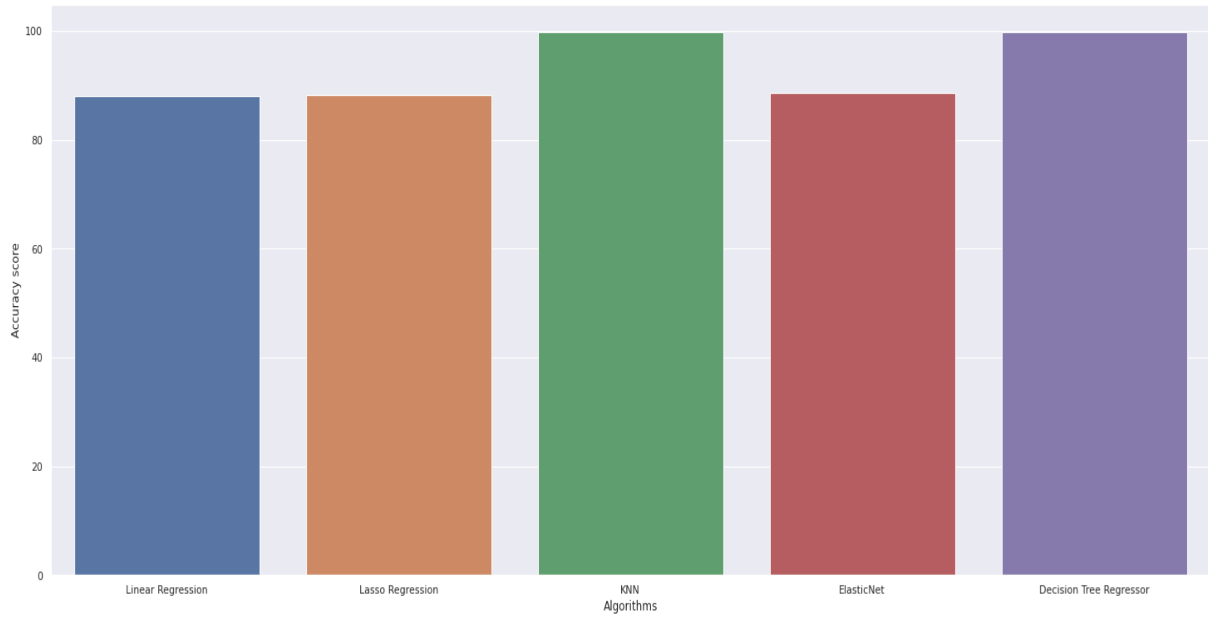
1. BASED ON THE ACCURACY SCORE

```
[60] scores = [accuracy_linear*100,accuracy_lasso*100,accuracy_knn*100,accuracy_el*100,accuracy_dt*100]
     algorithms = ["Linear Regression","Lasso Regression","KNN","ElasticNet","Decision Tree Regressor"]
     for i in range(len(algorithms)):
         print("The accuracy score achieved using "+algorithms[i]+" is: "+str(scores[i])+" %")

The accuracy score achieved using Linear Regression is: 87.96219669390433 %
The accuracy score achieved using Lasso Regression is: 88.23475100807417 %
The accuracy score achieved using KNN is: 99.87253597638369 %
The accuracy score achieved using ElasticNet is: 88.52968764587808 %
The accuracy score achieved using Decision Tree Regressor is: 99.8563034258301 %
```

Accuracy Score for Google Data

```
[61] sns.set(rc={'figure.figsize':(25,10)})
     plt.xlabel("Algorithms")
     plt.ylabel("Accuracy score")

     sns.barplot(algorithms,scores)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4c54dbc990>
```



Graphical Visualization of Accuracy Score for Google Stock Price

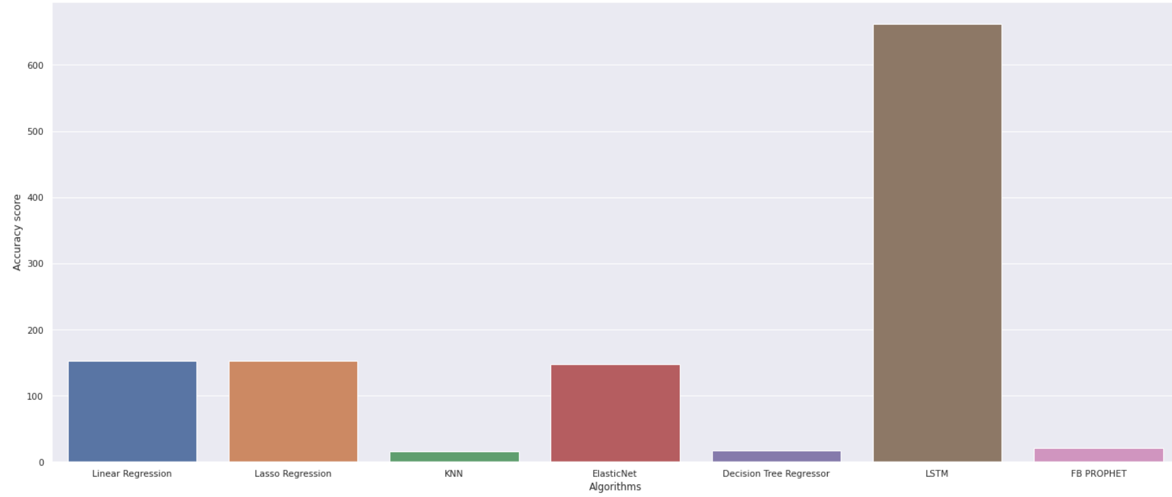
2. BASED ON RMSE VALUES

```
[62] scores = [rmse_linear,rmse_lasso,rmse_knn,rmse_el,rmse_dt,rmse_lstm,rmse_fb]
     algorithms = ["Linear Regression","Lasso Regression","KNN","ElasticNet","Decision Tree Regressor","LSTM","FB PROPHET"]
     for i in range(len(algorithms)):
         print("The accuracy score achieved using "+algorithms[i]+" is: "+str(scores[i])+" ")
```

```
The accuracy score achieved using Linear Regression is: 152.45320843141187
The accuracy score achieved using Lasso Regression is: 153.27891707019504
The accuracy score achieved using KNN is: 15.823945485615955
The accuracy score achieved using ElasticNet is: 147.0098506717504
The accuracy score achieved using Decision Tree Regressor is: 16.79007094237889
The accuracy score achieved using LSTM is: 662.479418230532
The accuracy score achieved using FB PROPHET is: 20.23540029739034
```

RMSE Value for Google Data

```
[63] sns.set(rc={'figure.figsize':(25,10)})
     plt.xlabel("Algorithms")
     plt.ylabel("Accuracy score")

     sns.barplot(algorithms,scores)

<matplotlib.axes._subplots.AxesSubplot at 0x7f4c54beb890>
```



Graphical Visualization of RMSE Value for Google Stock Price

Evaluation of the Performance of Algorithms

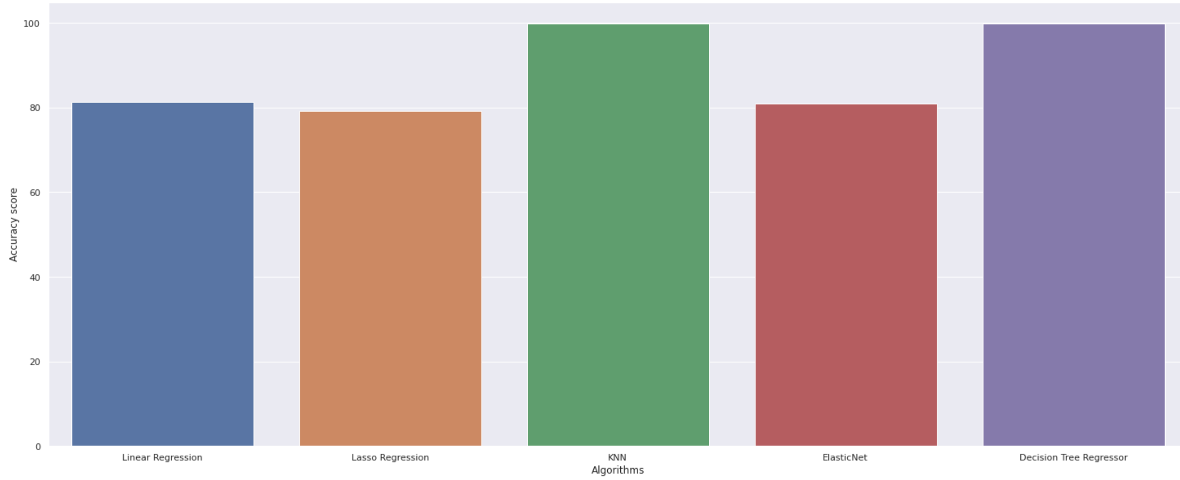
1. BASED ON THE ACCURACY SCORE

```
[85] scores = [accuracy_linear*100,accuracy_lasso*100,accuracy_knn*100,accuracy_el*100,accuracy_dt*100]
     algorithms = ["Linear Regression","Lasso Regression","KNN","ElasticNet","Decision Tree Regressor"]
     for i in range(len(algorithms)):
         print("The accuracy score achieved using "+algorithms[i]+" is: "+str(scores[i])+" %")

     The accuracy score achieved using Linear Regression is: 81.34730363661639 %
     The accuracy score achieved using Lasso Regression is: 79.26448975785706 %
     The accuracy score achieved using KNN is: 99.91362489291325 %
     The accuracy score achieved using ElasticNet is: 80.83927862295064 %
     The accuracy score achieved using Decision Tree Regressor is: 99.90783312799248 %
```

Accuracy Score for Microsoft Data

```
[86] sns.set(rc={'figure.figsize':(25,10)})
     plt.xlabel("Algorithms")
     plt.ylabel("Accuracy score")

     sns.barplot(algorithms,scores)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4f74a92210>
```



Graphical Visualization of Accuracy Score Value for Microsoft Stock Price

2. BASED ON RMSE VALUES

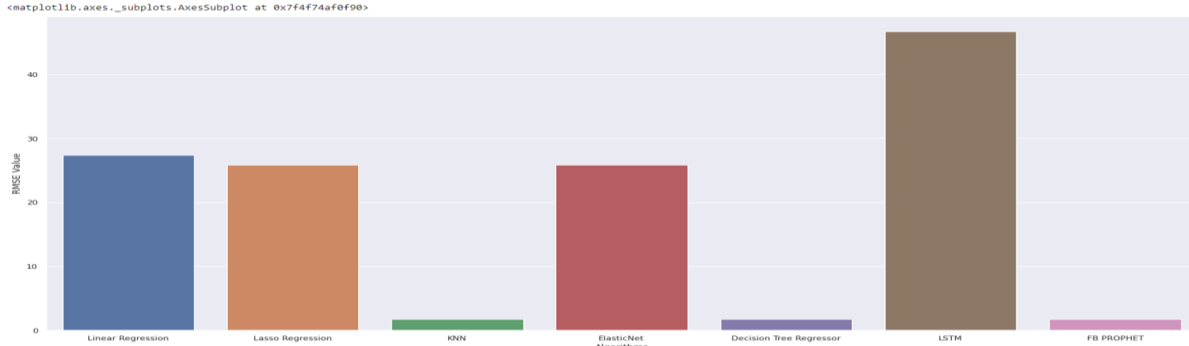
```
[88] scores = [rmse_linear,rmse_lasso,rmse_knn,rmse_el,rmse_dt,rmse_lstm,rmse_fb]
     algorithms = ["Linear Regression","Lasso Regression","KNN","ElasticNet","Decision Tree Regressor","LSTM","FB PROPHET"]
     for i in range(len(algorithms)):
         print("The RMSE Value achieved using "+algorithms[i]+" is: "+str(scores[i])+" ")
```

```
The RMSE Value achieved using Linear Regression is: 27.427243291799382
The RMSE Value achieved using Lasso Regression is: 25.8734416225507
The RMSE Value achieved using KNN is: 1.7512739821987657
The RMSE Value achieved using ElasticNet is: 25.86212337326192
The RMSE Value achieved using Decision Tree Regressor is: 1.7522608977057978
The RMSE Value achieved using LSTM is: 46.749862824985215
The RMSE Value achieved using FB PROPHET is: 1.7721375811220887
```

RMSE scores for Microsoft Data

```
sns.set(rc={'figure.figsize':(25,10)})
plt.xlabel("Algorithms")
plt.ylabel("RMSE Value")

sns.barplot(algorithms,scores)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4f74af0f90>
```



Graphical Visualization of RMSE value pf Microsoft Data

4.2.2 Stock Price Forecast Code Snippets

FORECASTING THE FUTURE STOCK PRICES USING ML AND DEEP LEARNING

```
df=web.DataReader('AMZN',data_source='yahoo',start='2012-01-01',end='2021-05-15')
df
```

Date	High	Low	Open	Close	Volume	Adj Close
2012-01-03	179.479996	175.550003	175.889999	179.029999	5110800	179.029999
2012-01-04	180.500000	176.070007	179.210007	177.509995	4205200	177.509995
2012-01-05	178.250000	174.050003	175.940002	177.610001	3809100	177.610001
2012-01-06	184.649994	177.500000	178.070007	182.610001	7008400	182.610001
2012-01-09	184.369995	177.000000	182.759995	178.559998	5056900	178.559998
...	...	...	...	...	...	...
2021-05-10	3283.000000	3190.000000	3282.320068	3190.489990	5838600	3190.489990
2021-05-11	3238.000000	3127.370117	3136.280029	3223.909912	4619800	3223.909912
2021-05-12	3207.939941	3133.100098	3185.000000	3151.939941	4936400	3151.939941
2021-05-13	3203.840088	3133.000000	3185.469971	3161.469971	3350900	3161.469971
2021-05-14	3221.399902	3183.409912	3185.560059	3216.000000	1621450	3216.000000

2357 rows × 6 columns

Dataset Info

1.FORECASTING USING LINEAR REGRESSION

```
[16] from sklearn import preprocessing
     forecast = 30
     df['Prediction'] = df[['Adj Close']].shift(-forecast)

     X = np.array(df.drop(['Prediction'], 1))
     X = preprocessing.scale(X)

     X_forecast = X[-forecast:]
     X = X[:-forecast]

     y = np.array(df['Prediction'])
     y = y[:-forecast]
```

```
[20] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

     clf = LinearRegression()
     clf.fit(X_train, y_train)

     confidence_linear = clf.score(X_test, y_test)

     forecast_predicted_linear = clf.predict(X_forecast)
     print(forecast_predicted_linear)
```

```
[3311.57189501 3268.49823369 3367.91523267 3348.61472233 3462.5657016
 3433.03116814 3446.34379153 3440.03407236 3438.51108328 3466.77555487
 3463.65429946 3427.74817934 3436.64735046 3409.81036637 3417.22038601
 3510.14788051 3497.69744398 3527.66862332 3561.11138453 3567.06290707
 3526.09166894 3429.27883569 3385.3624696  3383.50391488 3345.59250773
 3305.40797601 3336.15264967 3242.11199949 3254.37530655 3270.36723338]
```

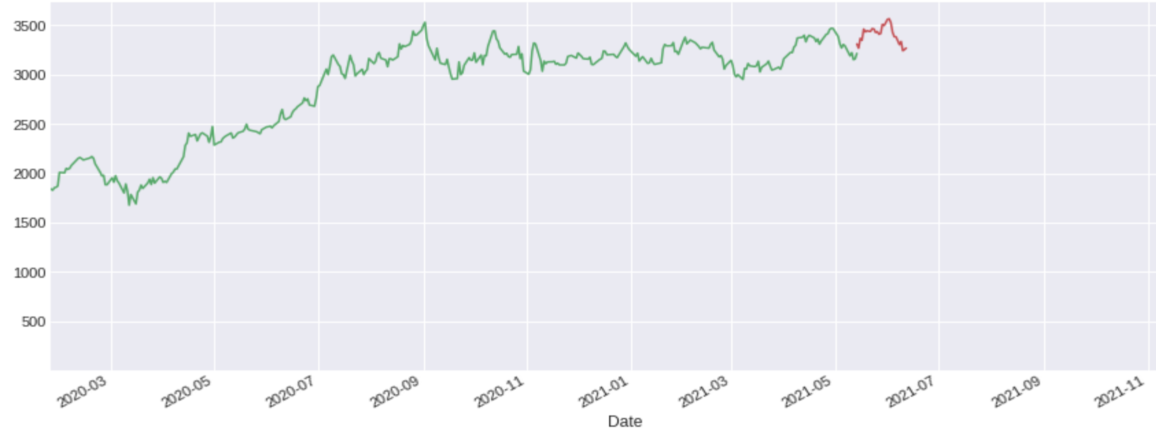
```
[21] print(confidence_linear*100)
```

```
98.34385011662154
```

Forecasting with Linear Regression

```
dates = pd.date_range(start="2021-05-14", end="2021-06-12")
plt.plot(dates, forecast_predicted_linear,color='r')
df['Adj Close'].plot(figsize=(15,6),color='g')
plt.xlim(xmin=datetime.date(2020,1,26))
```

```
(737450.0, 738125.4)
```



Graphical trend using Linear Regression (for next 30 days)

2. FORECASTING USING LASSO REGRESSION

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

clf = Lasso()
clf.fit(X_train, y_train)

confidence_lasso = clf.score(X_test, y_test)

forecast_predicted_lasso= clf.predict(X_forecast)
print(forecast_predicted_lasso)
```

```
[3290.03047669 3299.07665668 3355.73570209 3376.81015016 3429.36114392
 3449.83464316 3484.23140688 3448.90065027 3451.41445234 3463.27401754
 3482.48992881 3431.80464372 3420.2177972  3418.72943852 3426.19316557
 3482.59790294 3510.51676741 3542.62608219 3564.67129443 3596.88237482
 3526.49019363 3414.57427205 3396.37444851 3369.37968393 3380.28553969
 3321.83189613 3290.92547686 3252.76377682 3251.45603716 3276.37126423]
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_coordinate_descent.py:476: ConvergenceWarning:
  positive)
```

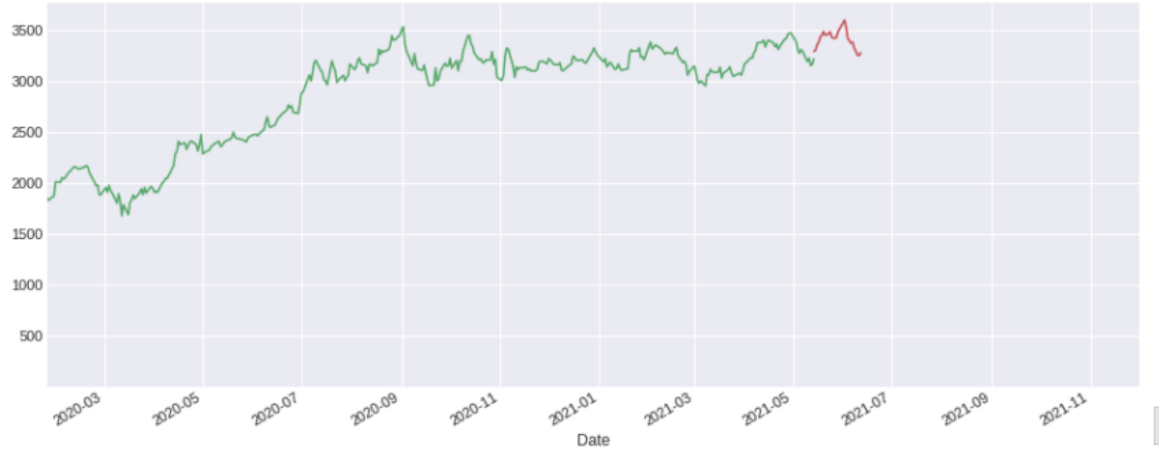
```
[26] print(confidence_lasso*100)
```

```
97.51530727585505
```

Forecasting using Lasso Regression

```
dates = pd.date_range(start="2021-05-14", end="2021-06-12")
plt.plot(dates, forecast_predicted_lasso,color='r')
df['Adj Close'].plot(figsize=(15,6),color='g')
plt.xlim(xmin=datetime.date(2020,1,26))
```

```
(737450.0, 738125.4)
```



Graphical trend using Lasso Regression (for next 30 days)

3. FORECASTING USING ELASTIC NET

```
[29] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

     clf = ElasticNet()
     clf.fit(X_train, y_train)

     confidence_elastic = clf.score(X_test, y_test)

     forecast_predicted_elastic= clf.predict(X_forecast)
     print(forecast_predicted_elastic)
```

```
[3073.53214681 3090.58839896 3128.13680684 3163.73715722 3204.59211546
 3227.4558176  3258.61892241 3215.65306013 3230.54133152 3241.93072218
 3237.76407469 3202.58016719 3194.26336726 3188.01257313 3194.6073542
 3246.70625921 3280.76975795 3308.48932271 3346.43919305 3357.8898191
 3286.45159163 3192.95034028 3162.73175914 3156.14054851 3171.69612465
 3105.62718252 3065.69319142 3045.11749103 3040.69792143 3065.50892638]
```

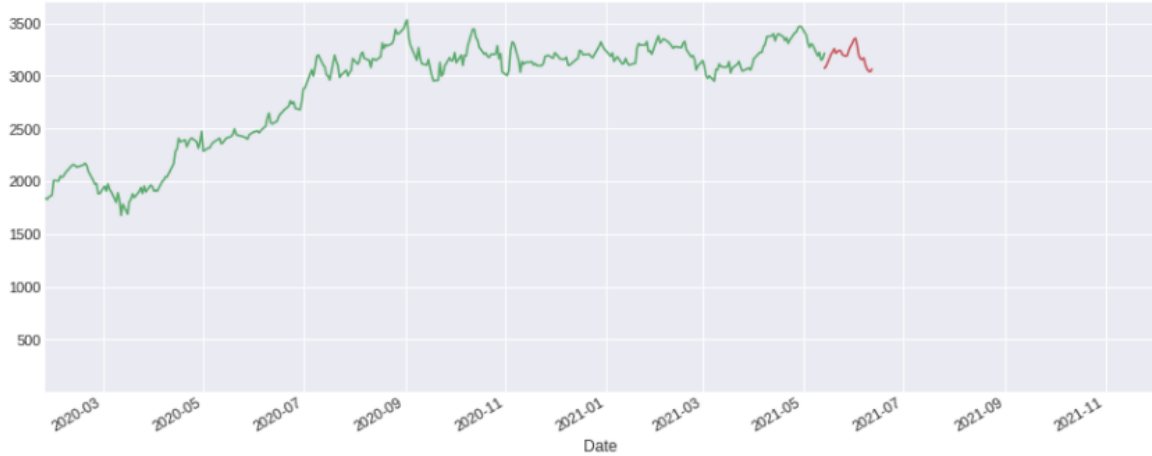
```
[30] print(confidence_elastic*100)
```

```
97.06341642531152
```

Forecasting using Elastic Net

```
[ ] dates = pd.date_range(start="2021-05-14", end="2021-06-12")
    plt.plot(dates, forecast_predicted_elastic,color='r')
    df['Adj Close'].plot(figsize=(15,6),color='g')
    plt.xlim(xmin=datetime.date(2020,1,26))
```

```
(737450.0, 738125.4)
```



Graphical trend using Elastic Net (for next 30 days)

4.FORECASTING USING KNN

```
[32] from sklearn.neighbors import KNeighborsRegressor

     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

     clf = KNeighborsRegressor()
     clf.fit(X_train, y_train)

     confidence_knn = clf.score(X_test, y_test)

     forecast_predicted_knn= clf.predict(X_forecast)
     print(forecast_predicted_knn)
```

```
[3179.06401367 3200.27597656 3193.25600586 3170.48798828 3162.98999023
 3156.23203125 3208.79399414 3222.64399414 3226.00200195 3226.00200195
 3214.88598633 3203.04799805 3211.65400391 3203.04799805 3159.05800781
 3152.81396484 3250.95195312 3286.20390625 3181.83398438 3177.01801758
 3178.1340332  3153.25200195 3164.87402344 3169.19799805 3136.46396484
 3298.1340332  3227.99995117 3133.1659668  3171.5699707  3237.59799805]
```

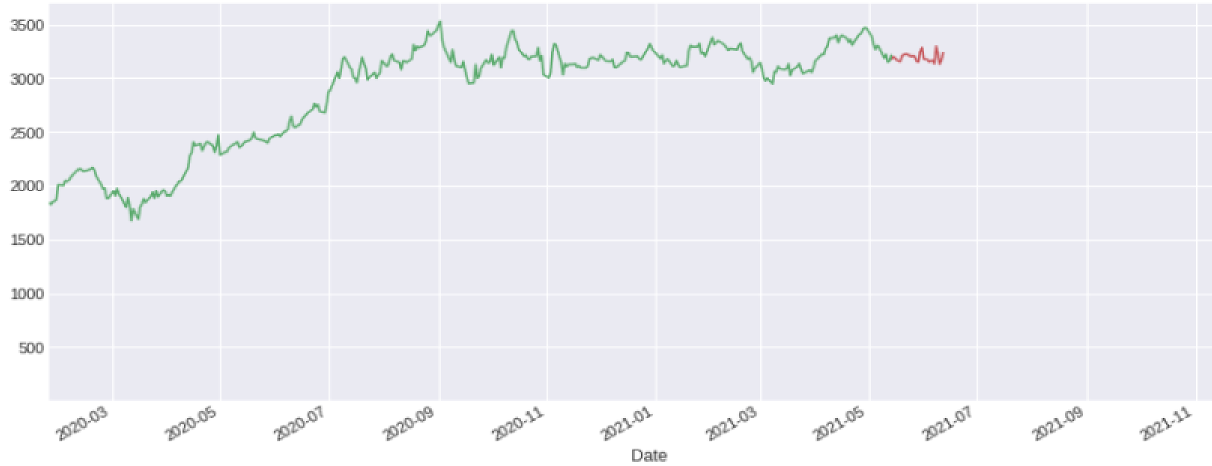
```
[33] print(confidence_knn*100)
```

```
98.08296234271074
```

Forecasting using KNN

```
dates = pd.date_range(start="2021-05-14", end="2021-06-12")
plt.plot(dates, forecast_predicted_knn,color='r')
df['Adj Close'].plot(figsize=(15,6),color='g')
plt.xlim(xmin=datetime.date(2020,1,26))
```

```
(737450.0, 738125.4)
```



Graphical trend using KNN (for next 30 days)

5.FORECASTING USING DECISION TREE

```
[35] from sklearn.tree import DecisionTreeRegressor
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

     clf = DecisionTreeRegressor()
     clf.fit(X_train, y_train)

     confidence_dt = clf.score(X_test, y_test)

     forecast_predicted_dt = clf.predict(X_forecast)
     print(forecast_predicted_dt)
```

```
[3328.22998047 3075.72998047 3268.94995117 2951.94995117 3262.12988281
 3094.08007812 3442.92993164 3137.5        3137.5        3137.5
 2951.94995117 2951.94995117 2951.94995117 2951.94995117 3137.5
 3190.55004883 3443.62988281 3443.62988281 3118.06005859 3118.06005859
 3118.06005859 3099.95996094 3199.19995117 3203.5300293  3201.64990234
 3312.5300293  2977.57006836 3180.73999023 3180.73999023 3075.72998047]
```

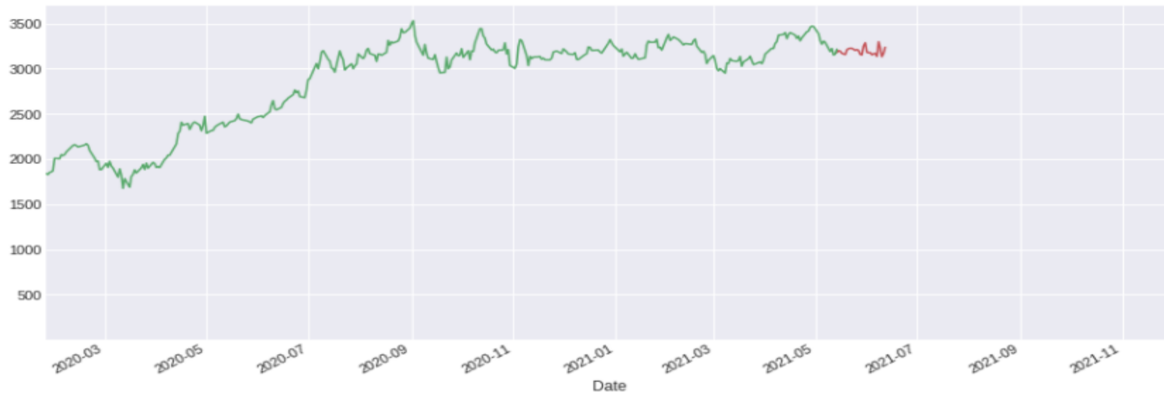
```
[36] print(confidence_dt*100)
```

```
97.87171529948864
```

Forecasting using Decision Tree

```
[37] dates = pd.date_range(start="2021-05-14", end="2021-06-12")
     plt.plot(dates, forecast_predicted_knn,color='r')
     df['Adj Close'].plot(figsize=(15,6),color='g')
     plt.xlim(xmin=datetime.date(2020,1,26))
```

```
(737450.0, 738125.4)
```



Graphical trend using Decision Tree Regressor (for next 30 days)

```
[38] compare=pd.DataFrame({'Linear Regression':forecast_predicted_linear,'Lasso':forecast_predicted_lasso,
```

```
compare.head(20)
```

	Linear Regression	Lasso	Decision tree Regressor	Elastic Net	K Neighbhors Regressor
0	3311.571895	3290.030477	3328.229980	3073.532147	3179.064014
1	3268.498234	3299.076657	3075.729980	3090.588399	3200.275977
2	3367.915233	3355.735702	3268.949951	3128.136807	3193.256006
3	3348.614722	3376.810150	2951.949951	3163.737157	3170.487988
4	3462.565702	3429.361144	3262.129883	3204.592115	3162.989990
5	3433.031168	3449.834643	3094.080078	3227.455818	3156.232031
6	3446.343792	3484.231407	3442.929932	3258.618922	3208.793994
7	3440.034072	3448.900650	3137.500000	3215.653060	3222.643994
8	3438.511083	3451.414452	3137.500000	3230.541332	3226.002002
9	3466.775555	3463.274018	3137.500000	3241.930722	3226.002002
10	3463.654299	3482.489929	2951.949951	3237.764075	3214.885986
11	3427.748179	3431.804644	2951.949951	3202.580167	3203.047998
12	3436.647350	3420.217797	2951.949951	3194.263367	3211.654004
13	3409.810366	3418.729439	2951.949951	3188.012573	3203.047998
14	3417.220386	3426.193166	3137.500000	3194.607354	3159.058008
15	3510.147881	3482.597903	3190.550049	3246.706259	3152.813965
16	3497.697444	3510.516767	3443.629883	3280.769758	3250.951953
17	3527.668623	3542.626082	3443.629883	3308.489323	3286.203906
18	3561.111385	3564.671294	3118.060059	3346.439193	3181.833984
19	3567.062907	3596.882375	3118.060059	3357.889819	3177.018018

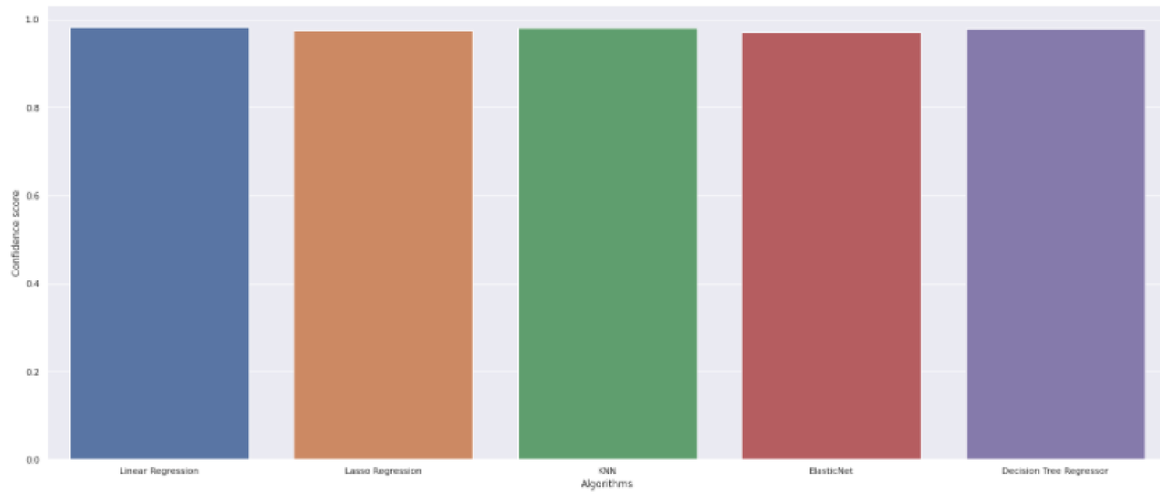
Comparison by value of forecasted Adjusted Close Price by Various Algorithms

EVALUATION OF PERFORMANCE IN FORECASTING

```
[40] scores = [confidence_linear,confidence_lasso,confidence_knn,confidence_elastic,confidence_dt]
     algorithms = ["Linear Regression","Lasso Regression","KNN","ElasticNet","Decision Tree Regressor"]
     for i in range(len(algorithms)):
         print("The confidence scores achieved using "+algorithms[i]+" is: "+str(scores[i]*100)+"%")

     The confidence scores achieved using Linear Regression is: 98.34385011662154%
     The confidence scores achieved using Lasso Regression is: 97.51530727585505%
     The confidence scores achieved using KNN is: 98.08296234271074%
     The confidence scores achieved using ElasticNet is: 97.06341642531152%
     The confidence scores achieved using Decision Tree Regressor is: 97.87171529948864%
```

Confidence Scores Comparison

```
sns.set(rc={'figure.figsize':(25,10)})
plt.xlabel("Algorithms")
plt.ylabel("Confidence score")

sns.barplot(algorithms,scores)

/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variables
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f9048411210>
```



Graphical Comparison

Note: The above screenshots is demonstrating the stock price forecasting for next 30 days for Amazon Data using five machine learning algorithms namely Linear Regression, KNN, Lasso Regression, Decision Tree Regressor and Elastic Net.

Similarly the comparison of forecast for Intel and Facebook Data is given below.

EVALUATION OF PERFORMANCE IN FORECASTING

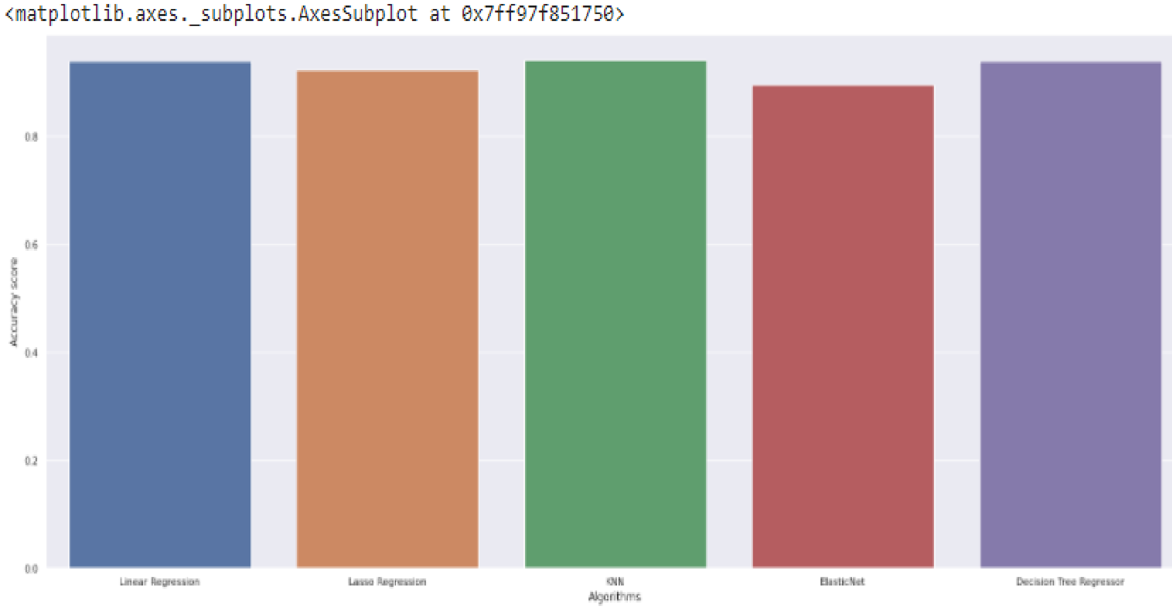
```
[93] scores = [confidence_linear,confidence_lasso,confidence_knn,confidence_elastic,confidence_dt]
     algorithms = ["Linear Regression","Lasso Regression","KNN","ElasticNet","Decision Tree Regressor"]
     for i in range(len(algorithms)):
         print("The confidence scores achieved using "+algorithms[i]+" is: "+str(scores[i]*100)+"%")
```

```
The confidence scores achieved using Linear Regression is: 93.88529907156517%
The confidence scores achieved using Lasso Regression is: 92.24971419304799%
The confidence scores achieved using KNN is: 94.15253427528%
The confidence scores achieved using ElasticNet is: 89.52577200798471%
The confidence scores achieved using Decision Tree Regressor is: 93.88593616609826%
```

Comparison of confidence scores for Intel Data

```
[94] sns.set(rc={'figure.figsize':(25,10)})
     plt.xlabel("Algorithms")
     plt.ylabel("Accuracy score")

     sns.barplot(algorithms,scores)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff97f851750>
```



Graphical comparison of algorithms in forecasting for Intel data

EVALUATION OF PERFORMANCE IN FORECASTING

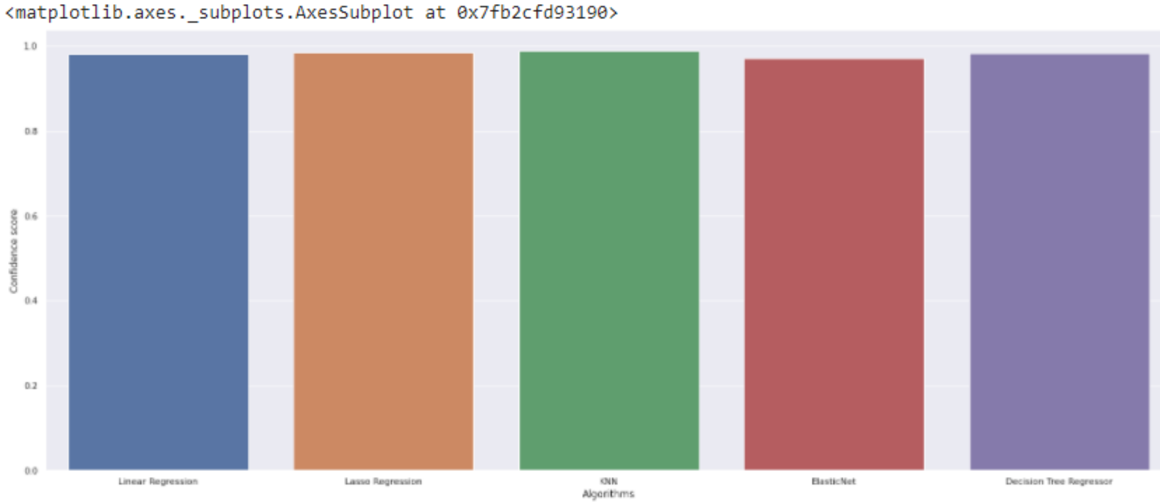
```
[88] scores = [confidence_linear,confidence_lasso,confidence_knn,confidence_elastic,confidence_dt]
     algorithms = ["Linear Regression","Lasso Regression","KNN","ElasticNet","Decision Tree Regressor"]
     for i in range(len(algorithms)):
         print("The confidence scores achieved using "+algorithms[i]+" is: "+str(scores[i]*100)+"%")
```

```
The confidence scores achieved using Linear Regression is: 98.06622147828195%
The confidence scores achieved using Lasso Regression is: 98.40662627025033%
The confidence scores achieved using KNN is: 98.9106734785713%
The confidence scores achieved using ElasticNet is: 97.05432829929515%
The confidence scores achieved using Decision Tree Regressor is: 98.22659684345659%
```

Comparison of Forecasting using Various Algorithms based on Confidence Scores for Facebook Data

```
sns.set(rc={'figure.figsize':(25,10)})
plt.xlabel("Algorithms")
plt.ylabel("Confidence score")

sns.barplot(algorithms,scores)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb2cfd93190>
```



Graphical Visualization of Comparison for Facebook Data

EVALUATION OF PERFORMANCE IN FORECASTING

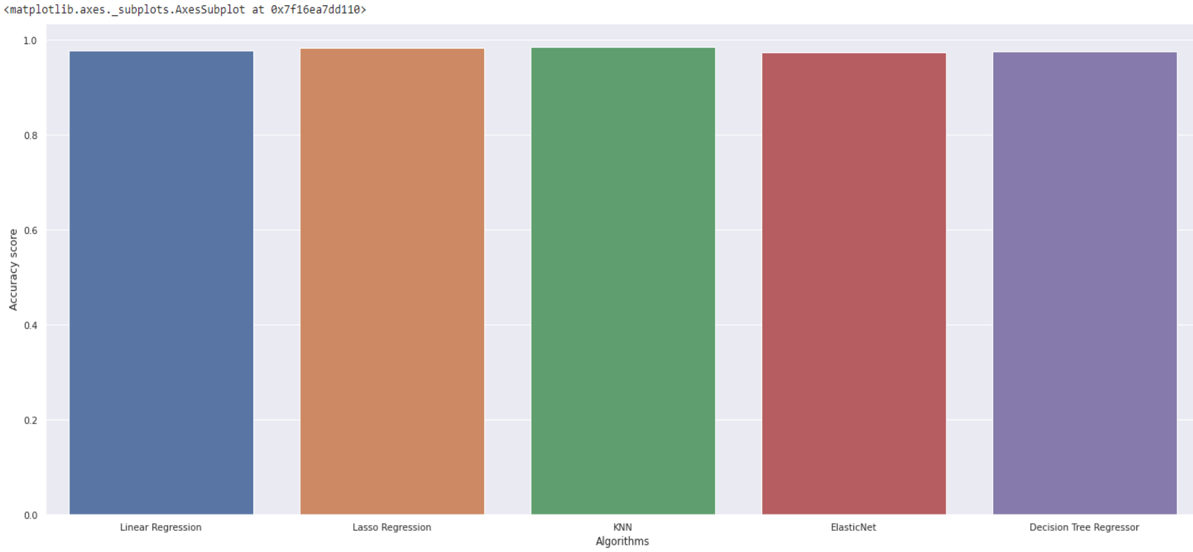
```
scores = [confidence_linear,confidence_lasso,confidence_knn,confidence_elastic,confidence_dt]
algorithms = ["Linear Regression","Lasso Regression","KNN","ElasticNet","Decision Tree Regressor"]
for i in range(len(algorithms)):
    print("The confidence scores achieved using "+algorithms[i]+" is: "+str(scores[i]*100)+"%")
```

```
The confidence scores achieved using Linear Regression is: 98.25559175418526%
The confidence scores achieved using Lasso Regression is: 98.32876134580675%
The confidence scores achieved using KNN is: 98.53161611057811%
The confidence scores achieved using ElasticNet is: 97.3831062985231%
The confidence scores achieved using Decision Tree Regressor is: 97.65346615737917%
```

Comparison of Forecasting using Various Algorithms based on Confidence Scores for Google Data

```
sns.set(rc={'figure.figsize':(25,10)})
plt.xlabel("Algorithms")
plt.ylabel("Confidence Score")

sns.barplot(algorithms,scores)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f16ea7dd110>
```



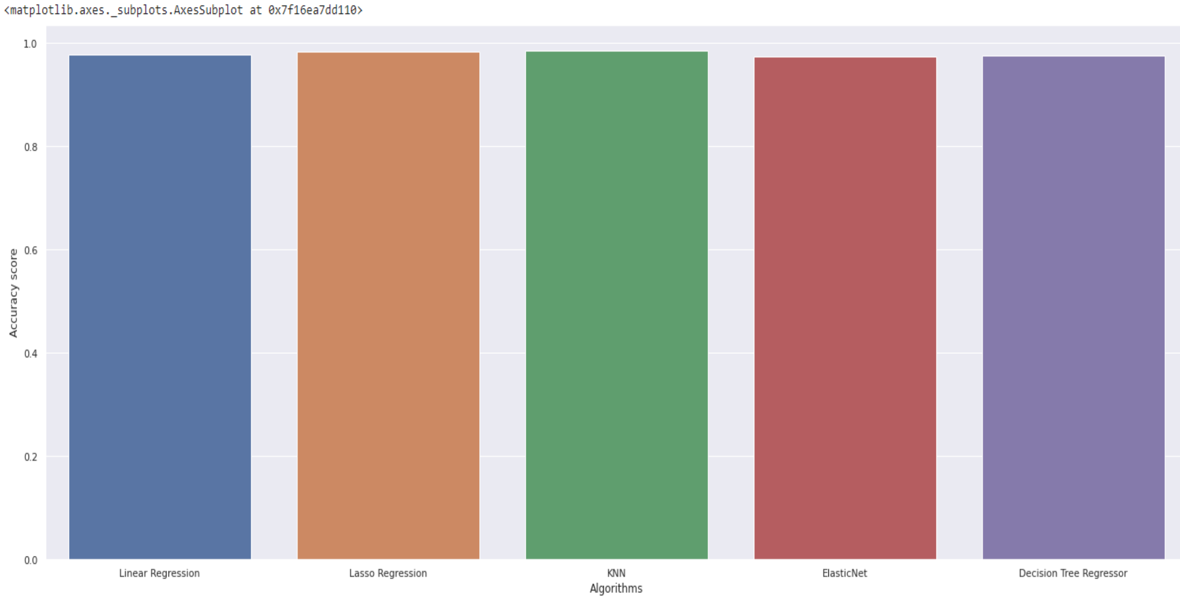
Graphical Visualization of Comparison for Google Data

EVALUATION OF PERFORMANCE IN FORECASTING

```
[ ] scores = [confidence_linear,confidence_lasso,confidence_knn,confidence_elastic,confidence_dt]
    algorithms = ["Linear Regression","Lasso Regression","KNN","ElasticNet","Decision Tree Regressor"]
    for i in range(len(algorithms)):
        print("The confidence scores achieved using "+algorithms[i]+" is: "+str(scores[i]*100)+"%")

    The confidence scores achieved using Linear Regression is: 98.25559175418526%
    The confidence scores achieved using Lasso Regression is: 98.32876134580675%
    The confidence scores achieved using KNN is: 98.53161611057811%
    The confidence scores achieved using ElasticNet is: 97.3831062985231%
    The confidence scores achieved using Decision Tree Regressor is: 97.65346615737917%
```

Comparison of Forecasting using Various Algorithms based on Confidence Scores for Microsoft Data

```
sns.set(rc={'figure.figsize':(25,10)})
plt.xlabel("Algorithms")
plt.ylabel("Confidence Score")

sns.barplot(algorithms,scores)

<matplotlib.axes._subplots.AxesSubplot at 0x7f16ea7dd110>
```



Graphical Visualization of Comparison for Microsoft Data

4.2.3 Sentiment Analysis Code Snippets

```
from urllib.request import urlopen, Request
from bs4 import BeautifulSoup
from nltk.sentiment.vader import SentimentIntensityAnalyzer
import pandas as pd
import matplotlib.pyplot as plt
import nltk
nltk.download('vader_lexicon')
finviz_url = 'https://finviz.com/quote.ashx?t='
tickers = ['AMZN', 'GOOG', 'FB']

news_tables = {}
for ticker in tickers:
    url = finviz_url + ticker

    req = Request(url=url, headers={'user-agent': 'my-app'})
    response = urlopen(req)

    html = BeautifulSoup(response, features='html.parser')
    news_table = html.find(id='news-table')
    news_tables[ticker] = news_table

parsed_data = []

for ticker, news_table in news_tables.items():

    for row in news_table.findAll('tr'):
```

```
for ticker, news_table in news_tables.items():

    for row in news_table.findAll('tr'):

        title = row.a.text
        date_data = row.td.text.split(' ')

        if len(date_data) == 1:
            time = date_data[0]
        else:
            date = date_data[0]
            time = date_data[1]

        parsed_data.append([ticker, date, time, title])

df = pd.DataFrame(parsed_data, columns=['ticker', 'date', 'time', 'title'])
df1 = pd.DataFrame(parsed_data, columns=['ticker', 'date', 'time', 'title'])
print(df)

vader = SentimentIntensityAnalyzer()

print (df['title'])

f = lambda title: vader.polarity_scores(title)['compound']
df['compound'] = df['title'].apply(f2)
f2 = lambda title: vader.polarity_scores(title)['pos']
df['pos'] = df['title'].apply(f2)
```

```python
f = lambda title: vader.polarity_scores(title)['compound']
df['compound'] = df['title'].apply(f2)
f2 = lambda title: vader.polarity_scores(title)['pos']
df['pos'] = df['title'].apply(f2)
f3 = lambda title: vader.polarity_scores(title)['neg']
df['neg'] = df['title'].apply(f2)
f4 = lambda title: vader.polarity_scores(title)['neu']
df['neu'] = df['title'].apply(f2)
print(df.head())
print(df)



f = lambda title: vader.polarity_scores(title)['compound']

df['compund'] = df['title'].apply(f)
df['date'] = pd.to_datetime(df.date).dt.date

f1 = lambda title: vader.polarity_scores(title)['pos']
df1['pos'] = df1['title'].apply(f1)
df1['date'] = pd.to_datetime(df.date).dt.date
```

```python
plt.figure(figsize=(10,8))
mean_df = df.groupby(['ticker', 'date']).mean().unstack()
mean_df = mean_df.xs('compound', axis="columns")
mean_df.plot(kind='bar')
plt.show()


plt.figure(figsize=(20,8))
mean_df1 = df1.groupby(['ticker', 'date']).mean().unstack()
mean_df1 = mean_df1.xs('pos', axis="columns")
mean_df1.plot(kind='bar')
plt.show()
```

Sentiment Analysis Code

```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
    ticker  ...                                              title
0     AMZN  ...                       Investing in Amazon Stock (AMZN)
1     AMZN  ...        3 Must-See Quotes From Amazon's Earnings Call
2     AMZN  ...  Amazon Doubled its Warehouse Space in Philadel...
3     AMZN  ...  Lyft co-founder: Drivers have made it clear th...
4     AMZN  ...  Zacks Earnings Trends Highlights: Apple, Micro...
..     ...  ...                                                ...
295     FB  ...  Corporations sent $226M to government causes o...
296     FB  ...  Why Facebook and Apple Couldn't Lift the Nasda...
297     FB  ...         How To Play This Incredible Earnings Season
298     FB  ...  Dow Rallies, Tech Stocks Slide As Apple Breako...
299     FB  ...  US STOCKS-S&P 500 near record high on Facebook...
```

Displaying the information from Finviz data

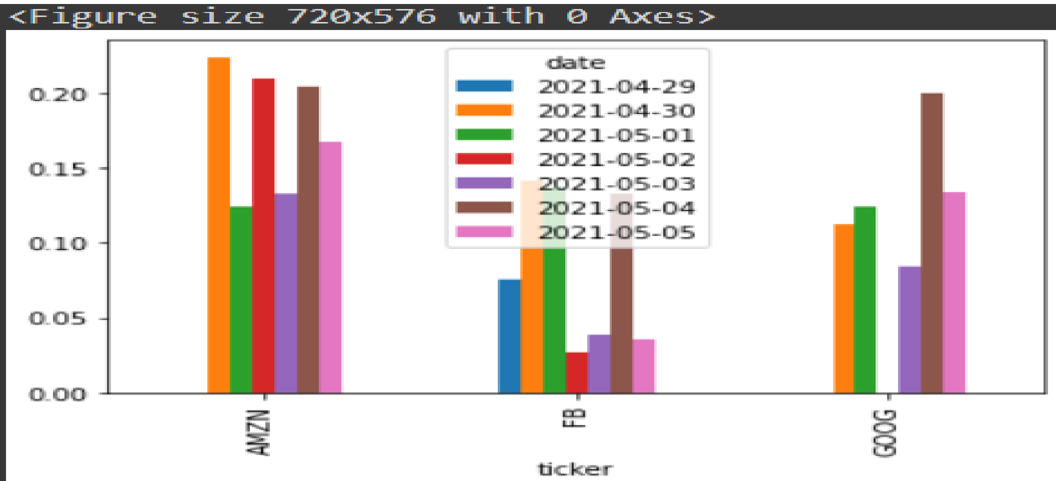
```
[300 rows x 4 columns]
0                     Investing in Amazon Stock (AMZN)
1          3 Must-See Quotes From Amazon's Earnings Call
2      Amazon Doubled its Warehouse Space in Philadel...
3      Lyft co-founder: Drivers have made it clear th...
4      Zacks Earnings Trends Highlights: Apple, Micro...
                             ...
295    Corporations sent $226M to government causes o...
296    Why Facebook and Apple Couldn't Lift the Nasda...
297          How To Play This Incredible Earnings Season
298    Dow Rallies, Tech Stocks Slide As Apple Breako...
299    US STOCKS-S&P 500 near record high on Facebook...
Name: title, Length: 300, dtype: object
  ticker       date       time  ...    pos    neg    neu
0   AMZN  May-05-21  09:57AM    ...  0.298  0.298  0.298
1   AMZN  May-05-21  09:45AM    ...  0.000  0.000  0.000
2   AMZN  May-05-21  09:00AM    ...  0.116  0.116  0.116
3   AMZN  May-05-21  08:54AM    ...  0.175  0.175  0.175
4   AMZN  May-05-21  08:48AM    ...  0.159  0.159  0.159

[5 rows x 8 columns]
    ticker       date       time  ...    pos    neg    neu
0     AMZN  May-05-21  09:57AM    ...  0.298  0.298  0.298
1     AMZN  May-05-21  09:45AM    ...  0.000  0.000  0.000
2     AMZN  May-05-21  09:00AM    ...  0.116  0.116  0.116
3     AMZN  May-05-21  08:54AM    ...  0.175  0.175  0.175
4     AMZN  May-05-21  08:48AM    ...  0.159  0.159  0.159
```

Classifying the stock news under positive, negative, neutral and compound.

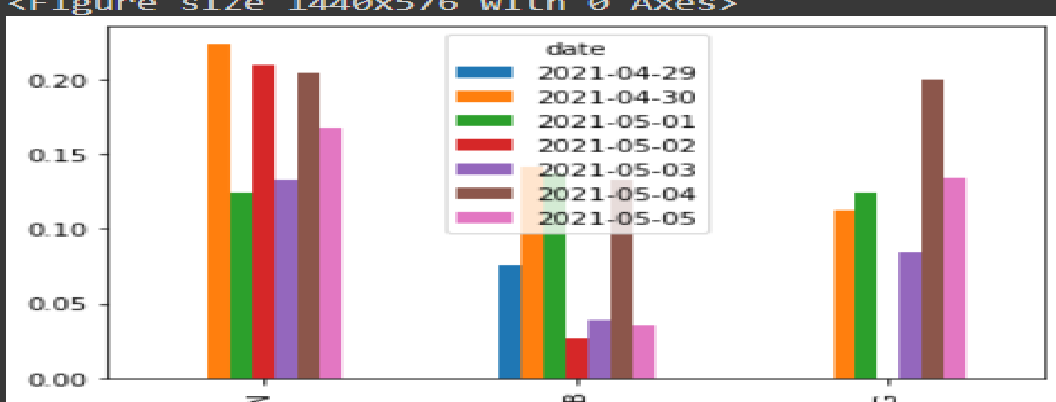
```
[5 rows x 8 columns]
    ticker       date      time  ...    pos    neg    neu
0     AMZN  May-05-21  09:57AM   ...  0.298  0.298  0.298
1     AMZN  May-05-21  09:45AM   ...  0.000  0.000  0.000
2     AMZN  May-05-21  09:00AM   ...  0.116  0.116  0.116
3     AMZN  May-05-21  08:54AM   ...  0.175  0.175  0.175
4     AMZN  May-05-21  08:48AM   ...  0.159  0.159  0.159
..     ...        ...       ...  ...    ...    ...    ...
295     FB  Apr-29-21  02:20PM   ...  0.000  0.000  0.000
296     FB  Apr-29-21  01:49PM   ...  0.000  0.000  0.000
297     FB  Apr-29-21  01:21PM   ...  0.286  0.286  0.286
298     FB  Apr-29-21  01:20PM   ...  0.000  0.000  0.000
299     FB  Apr-29-21  12:58PM   ...  0.357  0.357  0.357
```

Classifying the stock news under positive, negative, neutral and compound.

<Figure size 720x576 with 0 Axes>



<Figure size 1440x576 with 0 Axes>



Visualization of Sentiment Analysis

4.3 Evaluation Results

Data	Algorithms ->	Linear Regression	Lasso Regression	KNN	Elastic Net	Decision Tree Regressor
Amzn	**Accuracy Score**	84.30%	85.09%	99.87%	85.01%	99.90%
Intc	**Accuracy Score**	88.97%	88.16%	99.68%	88.94%	99.58%
Fb	**Accuracy Score**	93.29%	94.28%	99.80%	93.58%	99.80%
Google	**Accuracy Score**	87.96%	88.23%	99.87%	88.52%	99.85%
Microsoft	**Accuracy Score**	81.34%	79.26%	99.91%	80.83%	99.90%

Accuracy Score Comparison for Predictions

Data	Algorithms ->	Linear Regression	Lasso Regression	KNN	Elastic Net	Decision Tree Regressor	LSTM	FbProphet
Amzn	**RMSE Value**	356.62	356.78	32.40	364.33	29.611	595.18	53.37

Intc	**RMSE Value**	4.34	4.61	0.75	4.54	0.85	26.40	1.07
Fb	**RMSE Value**	20.38	18.49	3.12	17.91	3.14	104.86	5.04
Google	**RMSE Value**	152.45	153.27	15.82	147.00	16.79	662.47	20.23
Microsoft	**RMSE Value**	27.42	25.87	1.75	25.86	1.75	46.74	1.77

RMSE Value Comparison for Predictions

Data	**Algorithms ->**	**Linear Regression**	**Lasso Regression**	**KNN**	**Elastic Net**	**Decision Tree Regressor**
Amzn	**Confidence Score**	98.34%	97.51%	98.08%	97.06%	97.87%
Intc	**Confidence Score**	93.88%	92.24%	94.15%	89.52%	93.88%
Fb	**Confidence Score**	98.06%	98.40%	98.91%	97.05%	98.22%
Google	**Confidence Score**	98.25%	98.32%	98.53%	97.38%	97.65%
Microsoft	**Confidence Score**	98.25%	98.32%	98.53%	97.38%	97.65%

Confidence Score Comparison for Forecast

5. CONCLUSION

The main aim of this project was to analyze both sentiments from stock news along with stock price prediction and forecast. Our objective of the project was fulfilled as we were able to perform a comparative analysis of performance of algorithms in predicting stock price prediction and forecast. Overall it was found that three algorithms predicted highly accurate results. The algorithms are KNN, Decision Tree Regressor and Prophet. While Elastic Net, Lasso and Linear performed fairly with more than 83% accuracy. LSTM was the algorithm which performed least out of all. In addition we performed sentiment analysis and classified each of stock news of various companies into positive, negative, neutral and compound. Overall the project has been able to fulfill all the requirements and provide accurate results.

REFERENCES

1. D. Shah, H. Isah and F. Zulkernine, "Predicting the Effects of News Sentiments on the Stock Market," 2018 IEEE International Conference on Big Data (Big Data), Seattle, WA, USA, 2018, pp. 4705-4708, doi: 10.1109/BigData.2018.8621884.

2. C. -C. Lee, Z. Gao and C. -L. Tsai, "BERT-Based Stock Market Sentiment Analysis," 2020 IEEE International Conference on Consumer Electronics - Taiwan (ICCE-Taiwan), Taoyuan, Taiwan, 2020, pp. 1-2, doi: 10.1109/ICCE-Taiwan49838.2020.9258102.

3. A. Agarwal, "Sentiment Analysis of Financial News," 2020 12th International Conference on Computational Intelligence and Communication Networks (CICN), Bhimtal, India, 2020, pp. 312-315, doi: 10.1109/CICN49253.2020.9242579.

4. R. Batra and S. M. Daudpota, "Integrating StockTwits with sentiment analysis for better prediction of stock price movement," 2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET), Sukkur, Pakistan, 2018, pp. 1-5, doi: 10.1109/ICOMET.2018.8346382.

5. U. Pasupulety, A. Abdullah Anees, S. Anmol and B. R. Mohan, "Predicting Stock Prices using Ensemble Learning and Sentiment Analysis," 2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE), Sardinia, Italy, 2019, pp. 215-222, doi: 10.1109/AIKE.2019.00045.

6. Nausheen S, Anil Kumar M, and Amrutha K K. (2017). “SURVEY ON SENTIMENT ANALYSIS OF STOCK MARKET.” International Journal of Research - Granthaalayah, 5(4) RACSIT, 69-75

7. Ankit Sinha, Yash Agrawal, Vidhan Kumar, Chandan Kumar. (2020). “]SURVEY OF STOCK PRICE PREDICTION USING SENTIMENT ANALYSIS.” International Research Journal of Engineering and Technology

8. J. Kim, J. Seo, M. Lee and J. Seok, "Stock Price Prediction Through the Sentimental Analysis of News Articles," 2019 Eleventh International Conference on Ubiquitous and Future Networks (ICUFN), Zagreb, Croatia, 2019, pp. 700-702, doi: 10.1109/ICUFN.2019.8806182.

9. Pagolu, Sasank & Reddy, Kamal & Panda, Ganapati & Majhi, Babita. (2016). Sentiment analysis of Twitter data for predicting stock market movements. 1345-1350. 10.1109/SCOPES.2016.7955659.

10. Kalyanaraman, Vaanchitha & Kazi, Sarah & Tondulkar, Rohan & Oswal, Sangeeta. (2014). Sentiment Analysis on News Articles for Stocks. 10-15. 10.1109/AMS.2014.14.

11. Nti, Isaac kofi & Adekoya, Adebayo & Weyori, Benjamin. (2020). Predicting Stock Market Price Movement Using Sentiment Analysis: Evidence From Ghana. Applied Computer Systems. 25. 33-42. 10.2478/acss-2020-0004.

12. S. Mohan, S. Mullapudi, S. Sammeta, P. Vijayvergia and D. C. Anastasiu, "Stock Price Prediction Using News Sentiment Analysis," 2019 IEEE Fifth International Conference on Big Data

Computing Service and Applications (BigDataService), Newark, CA, USA, 2019, pp. 205-208, doi: 10.1109/BigDataService.2019.00035.

13. Alostad and H. Davulcu, "Directional Prediction of Stock Prices Using Breaking News on Twitter," 2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT), Singapore, 2015, pp. 523-530, doi: 10.1109/WI-IAT.2015.82.

14. Li, Xiaodong & Wu, Pangjing & Wang, Wenpeng. (2020). Incorporating stock prices and news sentiments for stock market prediction: A case of Hong Kong. Information Processing & Management. 57. 102212. 10.1016/j.ipm.2020.102212.

15. Schumaker, Rob & Chen, Hsiu-chin. (2009). Textual analysis of stock market prediction using breaking financial news: The AZFin text system. ACM Trans. Inf. Syst.. 27. 10.1145/1462198.1462204.

APPENDIX

1. **https://drive.google.com/drive/folders/1Wux4AXp0YeTEmGyO6C8eBKCVG5R-YqFA?usp=sharing**

The following file contains the code of stock price prediction and forecast along with sentiment analysis. All the files are .ipynb files.