Smart Internz

Stock Market Price Prediction

Using Machine learning Algorithms

Team 211

Team Members:

Bhawani Priya Boddi	20BCD7008
Vedire Chandu Reddy	20BCD7023
Jami Joshika	20BCD7050
Ravi Harini	20BCD7064

Campus: VIT-AP

1. INTRODUCTION

1.1 Overview

Market cap estimation using machine learning (ML) algorithms has received a lot of attention in recent years. Machine learning models analyze historical market data and identify patterns and trends to make predictions about future stock prices. Python is a popular programming language for machine learning algorithms because of its rich libraries and frameworks. An overview of the business value estimation process using machine learning and Python:

- Data collection: The first step is to collect business history data, which is usually calculated with attributes such as opening price, closing price, highest and lowest price, trading. volume and other financial indicators. Some online sites like Alpha Vantage or Yahoo Finance provide APIs to get market data. Alternatively, you can import data from financial records or CSV files.
- Data Preprocessing: Once data is received, it must be preprocessed before being fed into the ML model. This step includes tasks such as processing missing values, dealing with outliers, normalizing or standardizing data, and splitting it into training and testing. Python libraries such as pandas and scikit-learn are often used for preprocessing.
- Model Selection: There are many machine learning algorithms suitable for job prediction, including linear regression, decision tree, random forest. Model selection depends on the complexity of the problem and the availability of data. Libraries such as scikit-learn provide implementations of these algorithms.
- Model Training: The ML model is trained using the data prepared in this step. The training process includes feeding historical data into the model and adjusting its internal parameters to minimize prediction errors. Training can be done using algorithms such as gradient descent or back propagation, depending on the ML algorithm chosen.
- Model evaluation: After the model is trained, it should be evaluated to evaluate its performance. Evaluation can be made using various metrics such as square of mean error (MSE), root of mean error (RMSE), mean error (MAE), or true. Cross-validation methods such as K-fold cross-validation can provide better evaluation.
- Model deployment and prediction: After evaluating the model, it can be used to make predictions on new, unseen data. The model includes

important features of unobserved data and enables prediction of future market prices. Python libraries such as Scikitlearn or TensorFlow provide functionality to save and load training models for export.

1.2 Purpose

There are many benefits to using machine learning (ML) techniques, specifically Python, to predict business value. Some of the things these projects can do are:

- Stock Price Prediction: Machine learning models can analyze historical price data and identify patterns, trends, and trends that will help predict future price. These forecasts are useful for investors and traders to make decisions about buying, selling or holding stocks.
- Risk Management: Machine learning models can help assess and manage risk by analyzing past price changes and identifying potential or unusual business trends. This information can help investors organize their portfolios, cancel orders or implement risk reduction strategies.
- Portfolio optimization: Machine learning models can optimize portfolios by considering various factors such as risk, diversification, and expected return. This model can recommend the best asset allocation based on historical data and business models to maximize returns while minimizing risk.
- Trading Strategies: Machine learning algorithms can be trained to recognize specific patterns or signals that indicate potential trades in the market.

For example, models can be built to analyze signals, opinions, or relationships with other industries on social media or social media. These strategies can be used to automate trading or to generate signals for trading decisions.

- Sentiment Analysis: Machine learning can be used to analyze newspaper articles, opinion polls, or other sources of information to measure market sentiment and its impact on stock prices. The sentiment analysis model helps identify the positive or negative aspects of a company, business or industry, providing more insight into decision making.

2. LITERATURE SURVEY

2.1 Existing problem

There are many ways and methods that can be used to solve stock market problems with Machine Learning (ML) in Python. Here are some ideas:

- Linear Regression: This is a simple and widely used machine learning algorithm for regression problems. It assumes a relationship between strategic inputs and the target variable (stock price). You can use the regression line in Python using libraries like scikit-learn or statsmodels.
- Support Vector Machine (SVM): SVM is a supervised learning algorithm that can also be used for regression work. It tries to find the hyperplane that best separates the
  data points. Python provides libraries such as scikit-learn that provide SVM implementations.
- Random Forest: Random Forest is a learning algorithm that combines multiple decision trees to make predictions. Perfect for processing high-dimensional data and capturing relationships. You can implement the random forest in Python using the scikit-learn library.
- Long-Term Memory (LSTM): LSTM is a type of recurrent neural network (RNN) that can learn long-term dependencies in temporal data, making it suitable for time series analysis such as stock market forecasting. Python libraries such as TensorFlow or Keras provide LSTM implementations.
- Gradient Boosting: Gradient boosting algorithms such as XGBoost or LightGBM create a group of weak models (usually decision trees) that are sequentially trained to correct errors from previous models. These algorithms have been successful in many machine learning challenges and can be used effectively for business forecasting.
- Deep Neural Networks (DNN): You can use deep neural networks to build deep learning models for business forecasting. This includes building multi-layer neural network architectures such as feedforward neural networks, convolutional neural networks (CNNs), or hybrid models. Python libraries such as TensorFlow or PyTorch can be used to implement DNNs.

2.2 Proposed solution

Estimating business value using machine learning (ML) includes predictive models that can analyse historical data and make predictions about future prices. Here are some tips or solutions using machine learning and Python:

First, make sure you have the necessary Python libraries installed: scikit-learn,

pandas, and matplotlib. You can install them using pip:

Once you have the libraries installed, you can follow the steps below:

Import the required libraries

Load the stock market data into a pandas Data Frame. For this example, let's assume you have a CSV file with two columns: 'Date' and 'Price'.

Preprocess the data by splitting it into features (X) and the target variable (y)

Split the data into training and testing sets

Create an instance of each model

Train the models on the training data

Make predictions using the trained models

Evaluate the models using mean squared error (MSE)

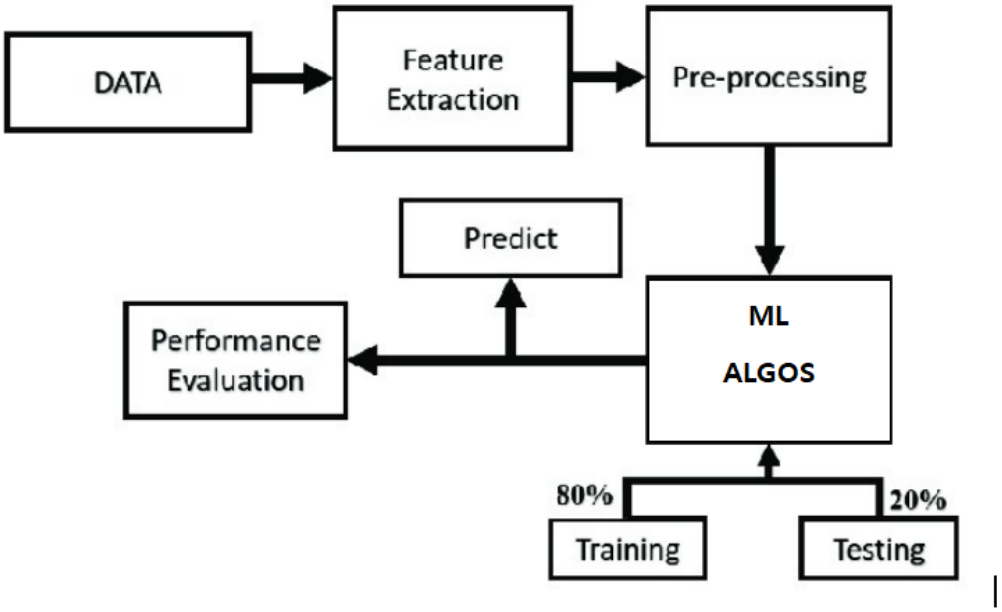
Visualize the predictions against the actual prices

Using linear regression, decision trees, and random forests for stock market price prediction.

- Linear regression is a simple and widely used algorithm for predicting continuous values, such as stock prices. In this case, you can consider historical stock price data and corresponding features (e.g., volume, news sentiment, technical indicators) as inputs.
- Decision trees are powerful algorithms for regression and classification tasks. They partition the input space based on different features and make predictions based on the majority class in each partition.
- Random forests are an ensemble of decision trees that make predictions by averaging the outputs of individual trees. They reduce overfitting and can provide more accurate predictions compared to a single decision tree.

3. THEORICTICAL ANALYSIS

3.1 Block Diagram



3.2 Hardware / Software designing

For completing the project,

Hardware components: we used our laptops of Intel core or Ryzen 64-bit operating system with x64-based processor.

Software Components: Anaconda played a major role, as it contains Jupyter notebook which was used to develop source code . And we used VS code for deploying the model using Flask application.

4. EXPERIMENTAL INVESTIGATIONS

The different steps in the process of investigation include:

1. Data set Collection: For this project, we have extracted our dataset from kaggle NSE_GLOBAL.csv and having a target attribute ‘close’.
2. Data Preprocessing: The data was explored and exploratory data analysis was performed and various plots were generated. The presence of null

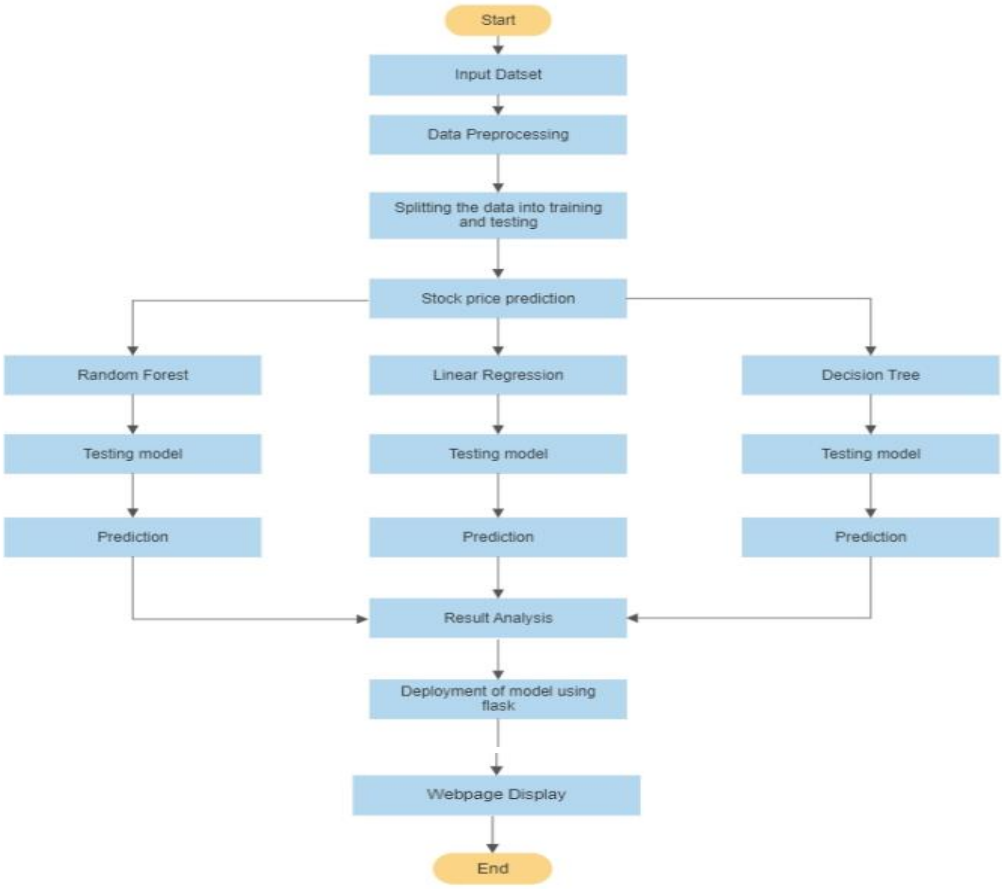
values was checked. We did not perform any encoding techniques as we don't have any string attributes in our dataset but hence we performed scaling techniques like MinMaxScaler.

A correlation coefficient is also generated to know the dependency between the attributes present in our dataset.

3. The data was split into independent and dependent variables which were later split into training and testing data in the ratio 80:20
4. Various Machine Learning algorithms were applied and the best one was selected for deployment
5. The best model was deployed using Flask and the webpage was built which takes values from users and predicts the genre as the output.

5.FLOWCHART

The proposed approach uses machine learning concepts. The flow chart for this approach is as follows:



6.RESULT

After implementing 3 different algorithms, namely Linear Regression, Decision Tree and Random Forest , we conclude that Random Forest is best for deployment.

The screenshots of the output are as follows:

RANDOM FOREST

In [36]:
```
from sklearn.ensemble import RandomForestRegressor
models = RandomForestRegressor()
RF = models.fit(x_train,y_train)
RF
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_20904\4162453472.py:3: DataConversionWarning: A column-vector y was pa
rray was expected. Please change the shape of y to (n_samples,), for example using ravel().
  RF = models.fit(x_train,y_train)
```

Out[36]:

RandomForestRegressor

RandomForestRegressor()

In [37]:
```
pred = RF.predict(x_test)
pred_y = pd.DataFrame(pred)
pred_y
```

Out[37]:

	0
0	1215.143010
1	658.436507
2	1466.006523
3	590.403245
4	604.163250
...	...
894	2250.166018

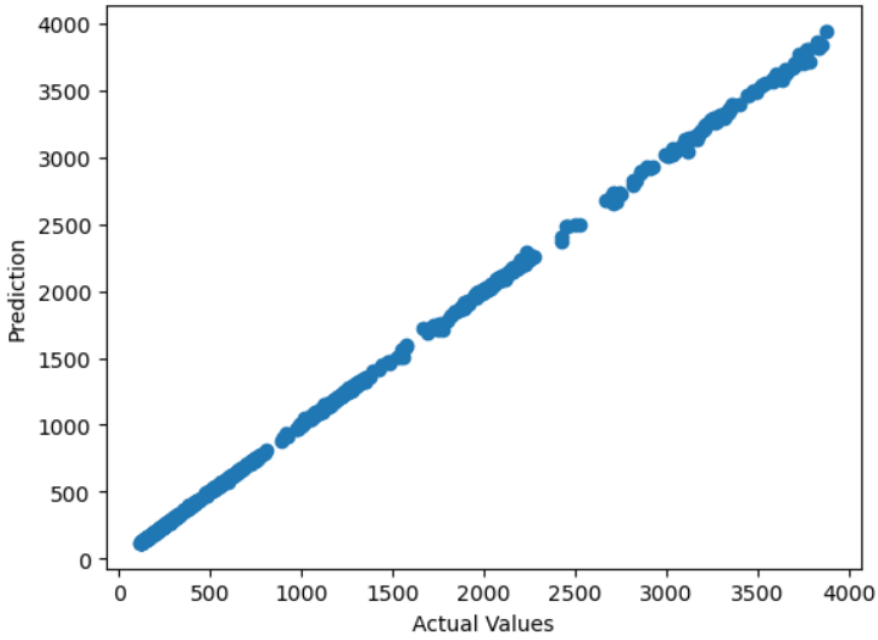
In [50]:
```
#R2 score
from sklearn.metrics import r2_score
acc=r2_score(y_test,pred_y)
acc
```

Out[50]: 0.9998630294720846

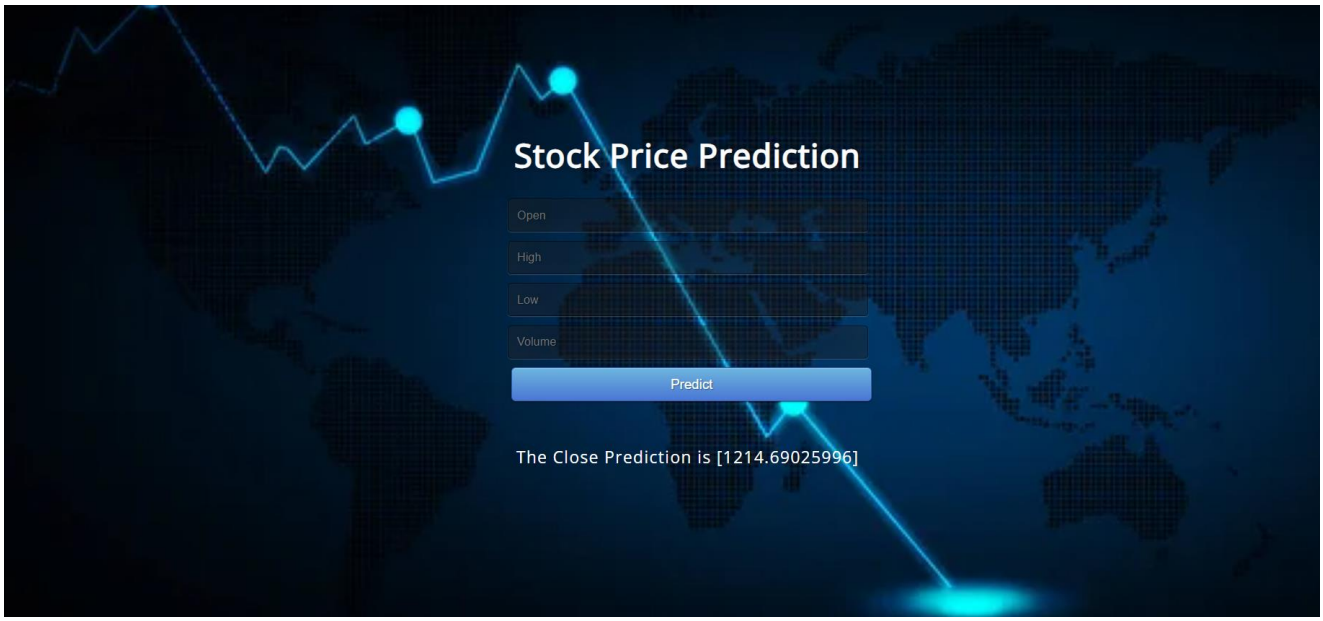
In [41]:
```
print("Mean Absolute Error:", round(metrics.mean_absolute_error(y_test, pred_y), 4))
print("Mean Squared Error:", round(metrics.mean_squared_error(y_test, pred_y), 4))
print("Root Mean Squared Error:", round(np.sqrt(metrics.mean_squared_error(y_test, pred_y)), 4))
print("(R^2) Score:", round(metrics.r2_score(y_test, pred_y), 4))
errors = abs(pred - y_test)
mape = 100 * (errors / y_test)
accuracy = 100 - np.mean(mape)
print('Accuracy:', round(accuracy, 2), '%.')
```

```
Mean Absolute Error: 6.6011
Mean Squared Error: 127.7196
Root Mean Squared Error: 11.3013
(R^2) Score: 0.9999
Accuracy: Close    99.29
dtype: float64 %.
```

In [42]:
```
plt.scatter(y_test,pred_y)
plt.xlabel("Actual Values")
plt.ylabel("Prediction")
plt.show()
```



After deployment of model using flask, the webpage looks are as follows:



7. ADVANTAGES & DISADVANTAGES

Advantages of Stock Price Prediction:

- Informed Decision Making: Accurate stock price predictions can provide investors and traders with valuable insights to make informed decisions about buying, selling, or holding stocks. It helps them identify potential opportunities and risks in the market.
- Risk Management: Stock price predictions can assist in managing risks associated with stock investments. By understanding the potential price movements, investors can implement risk mitigation strategies such as setting stop-loss orders or diversifying their portfolio.
- Timing of Trades: Predicting stock prices can aid in determining the optimal timing for buying or selling stocks. It enables investors to capitalize on favorable market conditions and potentially maximize their returns.
- Algorithmic Trading: Stock price prediction models can be utilized in algorithmic trading systems. These systems use predefined rules based on predictive models to automatically execute trades, eliminating human emotions and biases, and potentially improving trading efficiency.
- Market Analysis: Stock price predictions often involve analyzing various factors such as company financials, market trends, news, and economic

indicators. This analysis can provide a deeper understanding of the market and help identify underlying factors influencing stock prices.

Disadvantages of Stock Price Prediction:

- Inherent Uncertainty: The stock market is highly complex and influenced by numerous unpredictable factors such as geopolitical events, regulatory changes, natural disasters, and market sentiment. Predicting stock prices accurately is challenging due to this inherent uncertainty.
- Data Limitations: Stock price predictions heavily rely on historical data, financial indicators, and market trends. However, past performance may not always accurately reflect future outcomes, and the models may not capture unforeseen events or market shifts that can impact stock prices.
- Market Manipulation: Stock price predictions can be vulnerable to market manipulation. Unscrupulous traders or organizations may spread false information or artificially inflate or deflate stock prices to benefit from inaccurate predictions, leading to potential losses for investors.
- Over-Reliance on Models: Excessive reliance on stock price prediction models can lead to overconfidence and complacency. Investors may neglect other important factors such as fundamental analysis, company news, or qualitative assessments, which can impact stock performance.
- Ethical Concerns: The use of predictive models and algorithmic trading systems may raise ethical concerns, as it can contribute to market volatility and create unfair advantages for high-frequency traders. It can also exacerbate wealth inequality and reduce market efficiency.

It's important to note that stock price predictions should be used as tools for decision support, rather than as guarantees of future performance. Investors should consider these advantages and disadvantages while using stock price predictions and employ a comprehensive approach to investment decision-making.

8. APPLICATIONS

Machine learning (ML) can be used to predict business value in many areas. Some of the applications and techniques used in this field are:

- Time Forecasting: ML algorithms can be used to analyze historical market data and make predictions about future prices. Techniques such as

autoregressive integral moving average (ARIMA), recurrent neural network (RNN), short-term trend (LSTM), and gated repetitive unit (GRU) are often used.

- Technical Indicators: Machine learning can be used to create and analyze technical indicators that are calculated based on historical price and volume data. These indicators can help identify patterns and trends in stocks such as moving averages, relative strength index (RSI), and Bollinger Bands.
- Fundamental analysis: Machine learning can help analyze key data such as financial statements, income statements, and business indicators to estimate the costs of products. Algorithms help investors make informed decisions by learning patterns and relationships between factors and stock prices.
- Microstructural Market Analysis: Machine learning can be used to analyze ledger data, trading volumes and other microstructural data to predict short-term price movements and business liquidity. Techniques such as support vector machine (SVM), random forest and deep learning models can be used.
- Event-Driven Forecasting: Machine learning can be used to predict stock prices in response to specific events, such as earnings announcements, stock announcements, or macroeconomic news. Machine learning models can identify patterns and make predictions by analyzing historical event data and their corresponding price interactions.

You can use popular machine learning libraries like scikit-learn, TensorFlow, Keras, PyTorch or XGBoost to implement these applications in Python. These libraries provide a wide variety of algorithms and tools for generating preliminary data, architecture, model training, and evaluation. You'll also need to gather and prepare relevant documents, including products such as business prices, financial statements, news, information, and financial indicators, depending on the application you're working on.

9.CONCLUSION

In summary, this project focuses on using machine learning techniques to predict market prices using Python. Throughout the study, we explore many aspects of preliminary data, sample selection and evaluation.

First, we emphasize the importance of creating a preliminary database, which includes tasks such as processing missing values, evaluating features, and classifying data according to training and exams. This step ensures that the data is in a format suitable for the training model.

In terms of model selection, we explore several popular machine learning algorithms, including linear regression, decision tree, random forests.

These models are trained and evaluated using performance measures such as square mean error (MSE) or accuracy.

We also emphasize the importance of regular modeling to avoid overloading and improve generalization. Techniques such as L1 and L2 editing, convolution, and hyperparameter transformation are discussed as ways to achieve model performance.

Throughout this study, we show that machine learning models can provide reasonable predictive power for market prices. However, it is worth noting that forecasting the stock market is a difficult task, as there are many factors and uncertainties that affect the prices of products.

Therefore, the results of this study show that although machine learning models can provide good insights and aid decision making, they should be used as one of many tools in the implementation of Good investment. It's important to consider other avenues of value and analysis and stay up to date with business news and events.

Overall, this project lays the foundation for understanding and using product price prediction using Python machine learning. It emphasizes the importance of preliminary data, sample selection and evaluation. However, caution should be exercised in economic forecasting and using these methods as part of a larger investment.

10. FUTURE SCOPE

As the stock market is bound tightly to a country's economic growth and brings in huge investments by the investors and issues equities in the public interest, forecasting the movement of the stock prices and the market becomes essential in order to prevent huge losses and make relevant decisions. In this paper, we proposed a comparative study of various algorithms for forecasting the prices of different stocks. The study was extended from the traditional ML algorithms such as RF, KNN, SVM, Naïve Bayes, etc. to Deep Learning and Neural Network models such as Convolutional Neural Networks, Artificial Neural Networks, Long Short Term Memory, etc.The study also includes various other approaches such as Sentiment analysis, Time series analysis and GraphBased algorithms and compares results of these algorithms to predict the stock prices of various companies. In the future , researchers can focus on combining the sentiment analysis of stocks related information and the numeric value associated with historical value of stocks in predicting stock prices. Further effective stock recommendation systems can also be built by leveraging both information. Deep learning based approaches can be further exploited for better and efficient feature extraction techniques. Graphknowledge approaches are promising solutions which can be used for building stock prediction engines, however research works should address the complexity and gradient of graphs with large number of nodes. Our survey gives insights on future research directions in this area.

1. Our dataset and analysis method can improve potentially.
2. If more accurate algorithm and refined data with precise research is taken then future scope can be done with possible improvement.
3. Introduction of twitter feeds.
4. Advanced predictions form news feed and different websites can be taken for better results.

5.Refining key phase extraction and doing more work will definitely produce better results

11. BIBILOGRAPHY

[1] Stock market prediction by Radu Iacomin (2015)

[2] Stock Market Prediction Using Machine Learning Techniques by Mehak Usmani, Syed Hasan Adil, Kamran Raza, Syed Saad Azhar Ali(2016)

[3] Stock Price Prediction using Machine Learning by Sanya Ahuja

[4] Stock Market Prediction using Machine Learning by Ishita Parmar, Navanshu Agarwal, Sheirsh Saxena, Ridam Arora, Shikin Gupta, Himanshu Dhiman, Lokesh Chouhan (2018)

[5] Share Price Predication using machine learning Technique by Jeevan B, Naresh E, Vijaya kumar B P, Prashanth Kambli(2018)

APPENDIX

Source code

```
# # IMPORTING LIBRARIES
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
data=pd.read_csv('NSE_GLOBAL_DS')
data.head()
data.info()
data.shape
data.describe()
data.corr()

# # DATA PREPROCESSING
data.isnull().sum()

# replacing null values with the mean of the open column
data['Open'] = data['Open'].fillna(data['Open'].mean())
```

```
# replacing null values with the mean of the High column
data['High'] = data['High'].fillna(data['High'].mean())

# replacing null values with the mean of the Low column
data['Low'] = data['Low'].fillna(data['Low'].mean())

# replacing null values with the mean of the Close column
data['Close'] = data['Close'].fillna(data['Close'].mean())

# replacing null values with the mean of the AdjClose column
data['Adj Close'] = data['Adj Close'].fillna(data['Adj Close'].mean())

# replacing null values with the mean of the Volume column
data['Volume'] = data['Volume'].fillna(data['Volume'].mean())

data.isnull().sum()
# there is no missing values present in this dataset

# checking whtether it contains duplicates
data.duplicated().sum()

# # DATA VISUALIZATION
data['Open'].plot(figsize=(16,6))

features = ['Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume']

plt.subplots(figsize=(20,10))

for i, col in enumerate(features):
  plt.subplot(2,3,i+1)
  sns.distplot(data[col])
plt.show()
```

```
#Boxplot
plt.subplots(figsize=(20,10))
for i, col in enumerate(features):
  plt.subplot(2,3,i+1)
  sns.boxplot(data[col])
plt.show()

#HeatMap
plt.figure(figsize=(10, 10))
sns.heatmap(data.corr() > 0.9, annot=True, cbar=False)
plt.show()

# this gives the correlation between the variables
corr_matrix = data.corr()
corr_matrix

# Scatter Plot Between Open and Close
plt.scatter(data['Open'],data['Close'])
plt.xlabel('Open')
plt.ylabel('Close')
plt.title('open and Close')
# open and close are highly correlated

# Scatter Plot Between Volume and Low
plt.scatter(data['Volume'],data['Close'])
plt.xlabel('Volume')
plt.ylabel('Close')
plt.title('Volume and Close')
# Volume and close are not much correlated

# # SPLITTING DATA
```

```
x = data.iloc[:,[1,2,3,6]]
y = data.iloc[:,[4]]

from sklearn.preprocessing import StandardScaler
name = x.columns
scale=StandardScaler()
x_old=scale.fit_transform(x)
x = pd.DataFrame(x_old , columns=name)
x
y

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
x_train.shape
x_train.shape
x_test.shape
y_test.shape
x_test

# # LINEAR REGRESSION
from sklearn.linear_model import LinearRegression
LR=LinearRegression()
model=LR.fit(x_train,y_train)
model
pred=model.predict(x_test)
pred
y_test

#Accuracy
from sklearn.metrics import r2_score
acc=r2_score(y_test,pred)
acc
```

```
from sklearn import metrics
print("Mean Absolute Error:", round(metrics.mean_absolute_error(y_test, pred), 4))
print("Mean Squared Error:", round(metrics.mean_squared_error(y_test, pred), 4))
print("Root Mean Squared Error:", round(np.sqrt(metrics.mean_squared_error(y_test, pred)), 4))
print("(R^2) Score:", round(metrics.r2_score(y_test, pred), 4))
errors = abs(pred - y_test)
mape = 100 * (errors / y_test)
accuracy = 100 - np.mean(mape)
print('Accuracy:', round(accuracy, 2), '%.')

prediction_df = x_test.copy()
prediction_df['Close'] = y_test
prediction_df['Predicted Price'] = pred
prediction_df.head()

plt.scatter(y_test,pred)
plt.xlabel("Actual Values")
plt.ylabel("Prediction")
plt.show()

# # RANDOM FOREST
from sklearn.ensemble import RandomForestRegressor
models = RandomForestRegressor()
RF = models.fit(x_train,y_train)
RF
pred = RF.predict(x_test)
pred_y = pd.DataFrame(pred)
pred_y

#Accuracy
```

```
from sklearn.metrics import r2_score
acc=r2_score(y_test,pred_y)
acc
pred = pred.reshape(899,1)
pred
prediction_df = x_test.copy()
prediction_df['Close'] = y_test
prediction_df['Predicted Price'] = pred
prediction_df.head()

print("Mean Absolute Error:", round(metrics.mean_absolute_error(y_test, pred_y), 4))
print("Mean Squared Error:", round(metrics.mean_squared_error(y_test, pred_y), 4))
print("Root Mean Squared Error:", round(np.sqrt(metrics.mean_squared_error(y_test,
pred_y)), 4))
print("(R^2) Score:", round(metrics.r2_score(y_test, pred_y), 4))
errors = abs(pred - y_test)
mape = 100 * (errors / y_test)
accuracy = 100 - np.mean(mape)
print('Accuracy:', round(accuracy, 2), '%.')

plt.scatter(y_test,pred_y)
plt.xlabel("Actual Values")
plt.ylabel("Prediction")
plt.show()

# # DECISION TREE
from sklearn.tree import DecisionTreeRegressor
algo = DecisionTreeRegressor()
DT = algo.fit(x_train, y_train)
DT

ypred = DT.predict(x_test)
```

```
ypreds = pd.DataFrame(ypred)
ypreds
ypred = ypred.reshape(899,1)
ypred

#Accuracy
from sklearn.metrics import r2_score
acc=r2_score(y_test,ypreds)
acc

print("Mean Absolute Error:", round(metrics.mean_absolute_error(y_test, ypreds), 4))
print("Mean Squared Error:", round(metrics.mean_squared_error(y_test, ypreds), 4))
print("Root Mean Squared Error:", round(np.sqrt(metrics.mean_squared_error(y_test,
ypreds)), 4))
print("(R^2) Score:", round(metrics.r2_score(y_test, ypreds), 4))
errors = abs(ypred - y_test)
mape = 100 * (errors / y_test)
accuracy = 100 - np.mean(mape)
print('Accuracy:', round(accuracy, 2), '%.')

plt.scatter(y_test,ypred)
plt.xlabel("Actual Values")
plt.ylabel("Prediction")
plt.show()

import pickle
pickle.dump(models,open('model.pkl','wb'))
model = pickle.load(open('model.pkl','rb'))
print(model.predict([[0.053689, 0.063834, 0.064263, -0.662774]]))
```