STOCK PRICE PREDICTION USING MACHINE LEARNING

PROJECT REPORT

ANANKI BANERJEE

(CSE) Heritage Institute of Technology, Kolkata

IBM SUMMER INTERNSHIP 2023

Internship Domain: Data Analysis and Machine Learning with Python

INDEX

RELEVANT LINKS

- **GitHub Repository Link:**

https://github.com/Ananki-Banerjee/Google-Stock-Price-Prediction

- **Dataset Link:**

https://www.kaggle.com/code/nikhilkohli/stock-prediction-using-linear-regression-starter/input?select=GOOGL.csv

- **Source Code (Google Colab):**

https://colab.research.google.com/drive/1p1w6KMipgbvoA8eKCrnHnmBy08dplPt8?usp=sharing

PROJECT TITLE:

STOCK PRICE PREDICTION USING MACHINE LEARNING

The **objective** of this project is to predict the stock prices of Google Inc. (GOOGL) using various machine learning models. The historical **stock price dataset** for **GOOGL** is obtained from Kaggle. The project involves data preprocessing, exploratory data analysis, model selection, hyperparameter tuning, and performance evaluation. Multiple models, including Linear Regression, Decision Tree Regressor, and Random Forest Regressor, are evaluated to identify the best-performing model. The project utilizes Python and various libraries such as NumPy, Pandas, Matplotlib, Seaborn, and Scikit-learn to preprocess the data, train the linear regression model, and visualize the stock price trends.

INTRODUCTION

The financial markets are known for their dynamic and unpredictable nature, making stock price prediction an intriguing yet challenging task. Accurate predictions are of paramount importance to investors, traders, and financial analysts, as they influence critical decisions regarding investment strategies, risk management, and portfolio optimization. In recent years, the advent of machine learning has revolutionized the field of finance by providing powerful tools to extract patterns and insights from complex financial data.

This project aims to leverage the potential of machine learning algorithms to predict the stock prices of Google Inc. (GOOGL), a global technology behemoth, using historical stock price data. The project entails an in-depth exploration of Python's rich ecosystem, incorporating libraries such as NumPy, Pandas, Matplotlib, Seaborn, and Scikit-learn. By utilizing these tools, we will preprocess the dataset, train multiple machine learning models, identify the best-performing algorithm, and evaluate the model's predictions.

BACKGROUND AND MOTIVATION

The financial markets have always been the focal point of economic activity, where investors strive to maximize returns and minimize risks. Stock price prediction plays a pivotal role in understanding the market trends and making informed decisions. However, predicting stock prices accurately is a complex and multifaceted challenge due to the influence of various factors such as macroeconomic indicators, company performance, geopolitical events, and investor sentiments. Traditional methods often fall short in capturing the nuances of stock market behaviour.

The motivation behind this project lies in the potential of machine learning to uncover underlying patterns and relationships in financial data. By harnessing the power of algorithms like Linear Regression, Decision Trees, and Random Forest, we endeavour to create a robust predictive model that can offer valuable insights into the future movement of GOOGL stock prices. The project seeks to bridge the gap between traditional financial analysis and cutting-edge machine learning techniques, providing investors with a data-driven and statistically sound approach to stock price forecasting.

SIGNIFICANCE

The successful completion of this project will yield several significant outcomes. Firstly, it will showcase the power of machine learning in predicting complex financial data.

Secondly, it will offer investors and traders a data-driven approach to decision-making in the stock market, leading to more informed and efficient investment strategies.

Additionally, the project's insights can serve as a basis for further research and exploration in the field of financial data analysis and predictive modelling.

OBJECTIVE

The primary objective of this project is to develop an accurate and reliable model capable of predicting the stock prices of GOOGL using machine learning algorithms. We seek to achieve the following specific goals:

- <u>Data Exploration:</u> Gain insights into the historical stock price data, identify trends, and discern patterns using exploratory data analysis techniques.
- <u>Model Training:</u> Utilize machine learning algorithms like Linear Regression, Decision Trees, and Random Forest to train predictive models on historical stock price data.
- <u>Model Selection:</u> Evaluate the performance of each model based on relevant metrics, such as Mean Squared Error (MSE) and Root Mean Squared Error (RMSE), to identify the most effective algorithm.
- <u>Hyperparameter Tuning:</u> Fine-tune the hyperparameters of the selected model to optimize its performance and enhance prediction accuracy.
- <u>Visualization:</u> Present visualizations of the predicted vs. actual stock prices to understand the model's effectiveness in capturing underlying trends.
- <u>Error Analysis:</u> Analyze the errors and residuals to gain insights into the model's predictive capacity and identify potential areas of improvement.

PROJECT OVERVIEW

The project can be divided into several key phases:

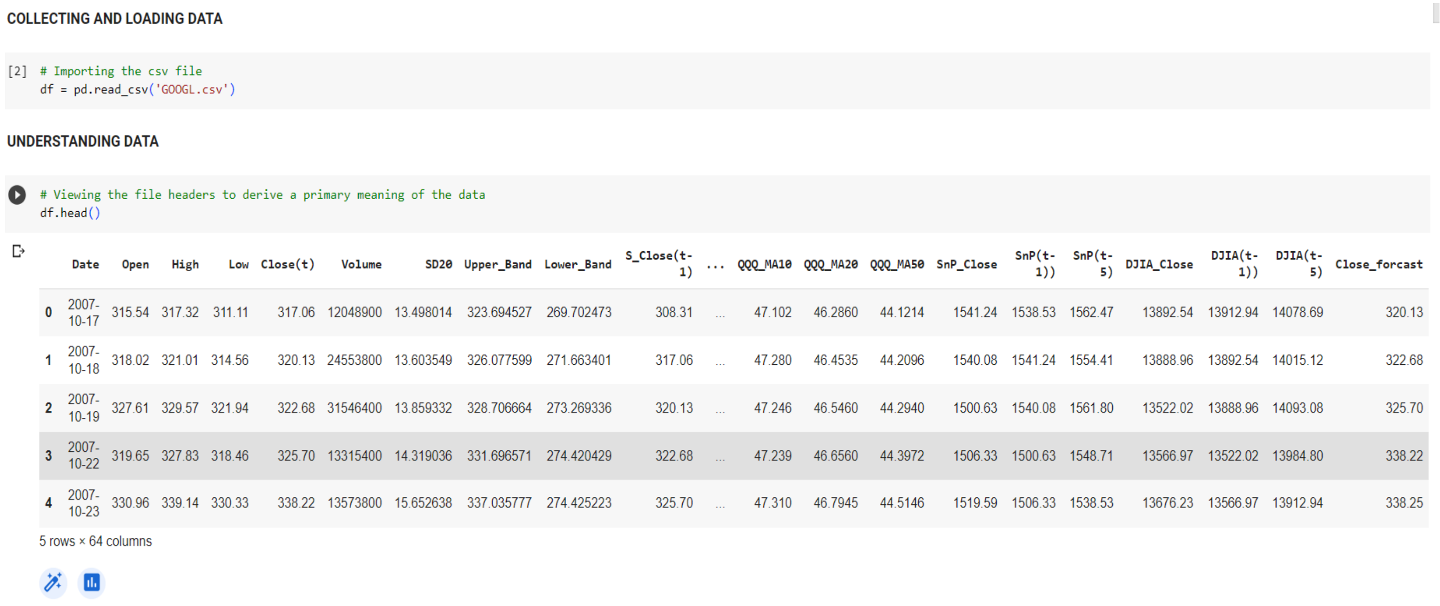
1. Data Acquisition and Preprocessing: Collect historical stock price data for **GOOGL** from a reliable source (**Kaggle**), preprocess the data, and handle any missing or redundant entries.
2. Exploratory Data Analysis: Visualize the 'Close' price of GOOGL stock over time using time series plots to identify trends, seasonality, and outliers.
3. Data Splitting and Scaling: Split the dataset into training and testing sets using the Train Test Split method, and perform feature scaling to standardize the features for model training.
4. Model Training, Cross Validation and Evaluation: Train and Cross Validate various machine learning algorithms, including Linear Regression, Decision Trees, and Random Forest, and evaluate their performance using appropriate metrics.
5. Model Selection and Hyperparameter Tuning: Identify the best-performing model and fine-tune its hyperparameters to optimize its predictive capabilities.
6. Visualization and Error Analysis: Plot the predicted vs. actual stock prices for visual comparison and analyze the errors to understand the model's strengths and weaknesses.

Stay tuned as we progress through each phase of the project to uncover valuable insights and develop a robust and accurate model for predicting the stock prices of Google Inc. (GOOGL).

DATA ACQUISITION

Data Source:

The first step in the project is to acquire the historical stock price data for Google Inc. (GOOGL). We obtain the dataset from a reliable and trusted source, such as Kaggle, a platform known for providing diverse datasets suitable for machine learning and data analysis tasks. The dataset typically includes relevant information, such as the 'Date' and 'Close' price columns, which are crucial for predicting stock prices.



COLLECTING AND LOADING DATA

```
[2] # Importing the csv file
    df = pd.read_csv('GOOGL.csv')
```

UNDERSTANDING DATA

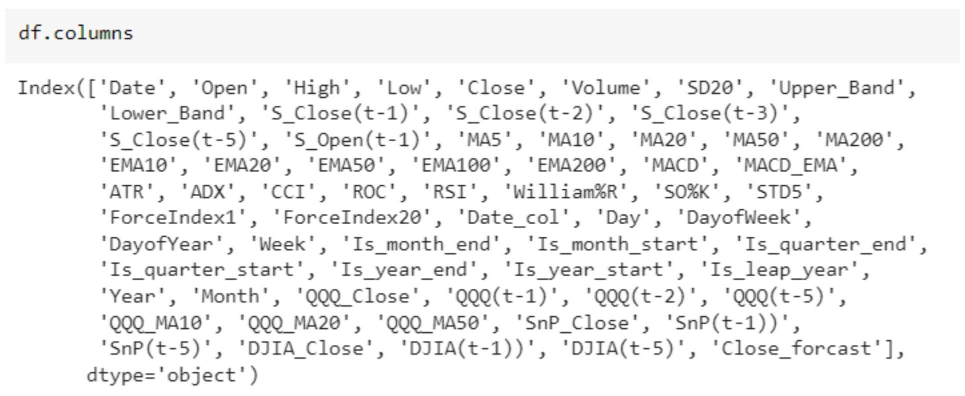
```
# Viewing the file headers to derive a primary meaning of the data
df.head()
```

	Date	Open	High	Low	Close(t)	Volume	SD20	Upper_Band	Lower_Band	S_Close(t-1)	...	QQQ_MA10	QQQ_MA20	QQQ_MA50	SnP_Close	SnP(t-1))	SnP(t-5)	DJIA_Close	DJIA(t-1))	DJIA(t-5)	Close_forcast
0	2007-10-17	315.54	317.32	311.11	317.06	12048900	13.498014	323.694527	269.702473	308.31	...	47.102	46.2860	44.1214	1541.24	1538.53	1562.47	13892.54	13912.94	14078.69	320.13
1	2007-10-18	318.02	321.01	314.56	320.13	24553800	13.603549	326.077599	271.663401	317.06	...	47.280	46.4535	44.2096	1540.08	1541.24	1554.41	13888.96	13892.54	14015.12	322.68
2	2007-10-19	327.61	329.57	321.94	322.68	31546400	13.859332	328.706664	273.269336	320.13	...	47.246	46.5460	44.2940	1500.63	1540.08	1561.80	13522.02	13888.96	14093.08	325.70
3	2007-10-22	319.65	327.83	318.46	325.70	13315400	14.319036	331.696571	274.420429	322.68	...	47.239	46.6560	44.3972	1506.33	1500.63	1548.71	13566.97	13522.02	13984.80	338.22
4	2007-10-23	330.96	339.14	330.33	338.22	13573800	15.652638	337.035777	274.425223	325.70	...	47.310	46.7945	44.5146	1519.59	1506.33	1538.53	13676.23	13566.97	13912.94	338.25

5 rows × 64 columns

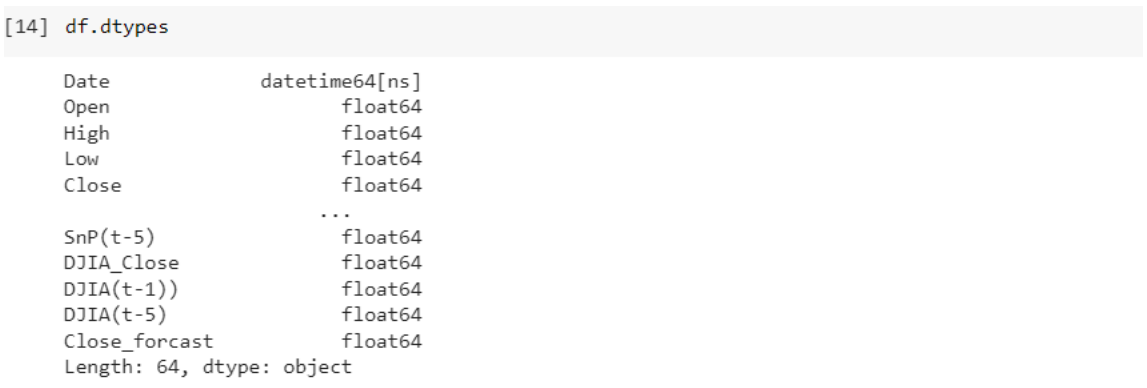
DATA PREPROCESSING

Handling Missing Data: After acquiring the dataset, we perform data preprocessing to ensure its cleanliness and suitability for analysis. One critical aspect is handling missing data, which can adversely affect model training and prediction. We identify any missing entries in the dataset and decide on an appropriate strategy to handle them. Common techniques include filling the missing values using the mean, median, or forward/backward filling, or removing the entries altogether if the missing data is insignificant.



```
df.columns
```

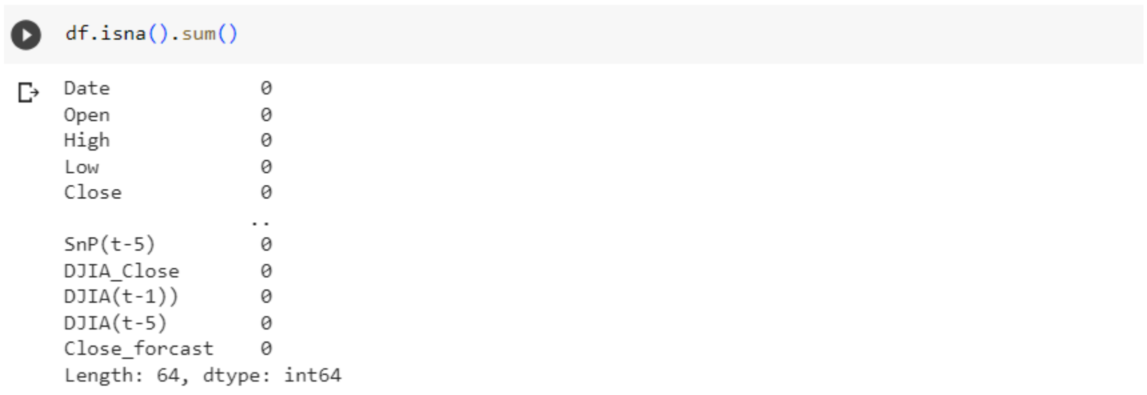
```
Index(['Date', 'Open', 'High', 'Low', 'Close', 'Volume', 'SD20', 'Upper_Band',
       'Lower_Band', 'S_Close(t-1)', 'S_Close(t-2)', 'S_Close(t-3)',
       'S_Close(t-5)', 'S_Open(t-1)', 'MA5', 'MA10', 'MA20', 'MA50', 'MA200',
       'EMA10', 'EMA20', 'EMA50', 'EMA100', 'EMA200', 'MACD', 'MACD_EMA',
       'ATR', 'ADX', 'CCI', 'ROC', 'RSI', 'William%R', 'SO%K', 'STD5',
       'ForceIndex1', 'ForceIndex20', 'Date_col', 'Day', 'DayofWeek',
       'DayofYear', 'Week', 'Is_month_end', 'Is_month_start', 'Is_quarter_end',
       'Is_quarter_start', 'Is_year_end', 'Is_year_start', 'Is_leap_year',
       'Year', 'Month', 'QQQ_Close', 'QQQ(t-1)', 'QQQ(t-2)', 'QQQ(t-5)',
       'QQQ_MA10', 'QQQ_MA20', 'QQQ_MA50', 'SnP_Close', 'SnP(t-1))',
       'SnP(t-5)', 'DJIA_Close', 'DJIA(t-1))', 'DJIA(t-5)', 'Close_forcast'],
      dtype='object')
```



```
[14] df.dtypes

     Date             datetime64[ns]
     Open                    float64
     High                    float64
     Low                     float64
     Close                   float64
                           ...
     SnP(t-5)                float64
     DJIA_Close              float64
     DJIA(t-1))              float64
     DJIA(t-5)               float64
     Close_forcast           float64
     Length: 64, dtype: object
```

Since all the features of the dataset are numeric features, we only need to preprocess numeric features.



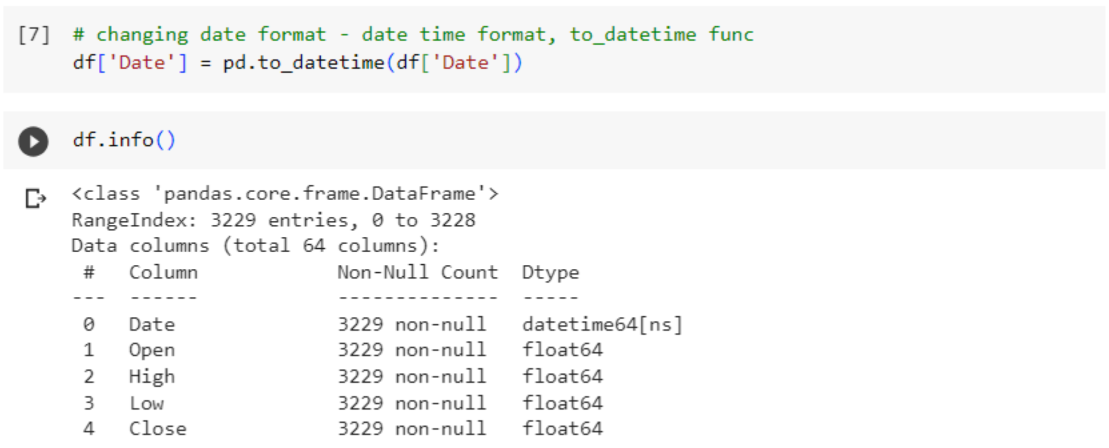
```
df.isna().sum()

     Date             0
     Open             0
     High             0
     Low              0
     Close            0
                     ..
     SnP(t-5)         0
     DJIA_Close       0
     DJIA(t-1))       0
     DJIA(t-5)        0
     Close_forcast    0
     Length: 64, dtype: int64
```

This implies there are no null or missing values.

Converting Date to Datetime: The 'Date' column in the dataset is often represented as a string or object data type. To facilitate time series analysis and visualization, we convert the 'Date' column to the datetime format. This enables us to easily manipulate and extract time-based features from the data.

Date column is of type 'object'. Therefore it is necessary to cast it into type Date



```
[7]  # changing date format - date time format, to_datetime func
     df['Date'] = pd.to_datetime(df['Date'])
```

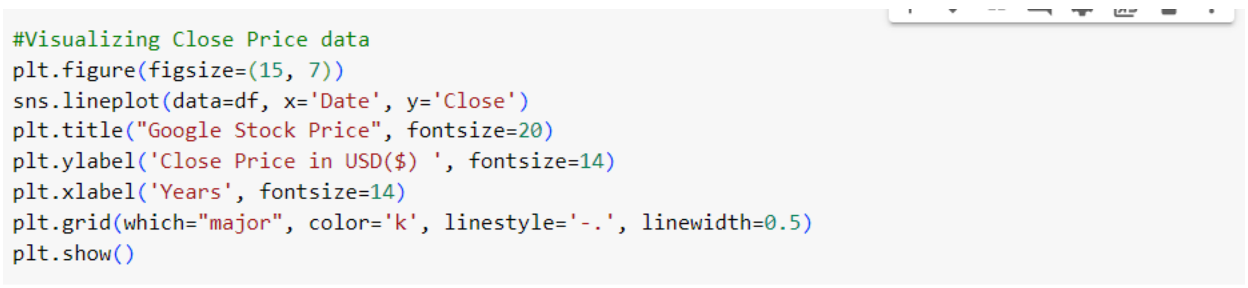
```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3229 entries, 0 to 3228
Data columns (total 64 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Date              3229 non-null   datetime64[ns]
 1   Open              3229 non-null   float64
 2   High              3229 non-null   float64
 3   Low               3229 non-null   float64
 4   Close             3229 non-null   float64
```

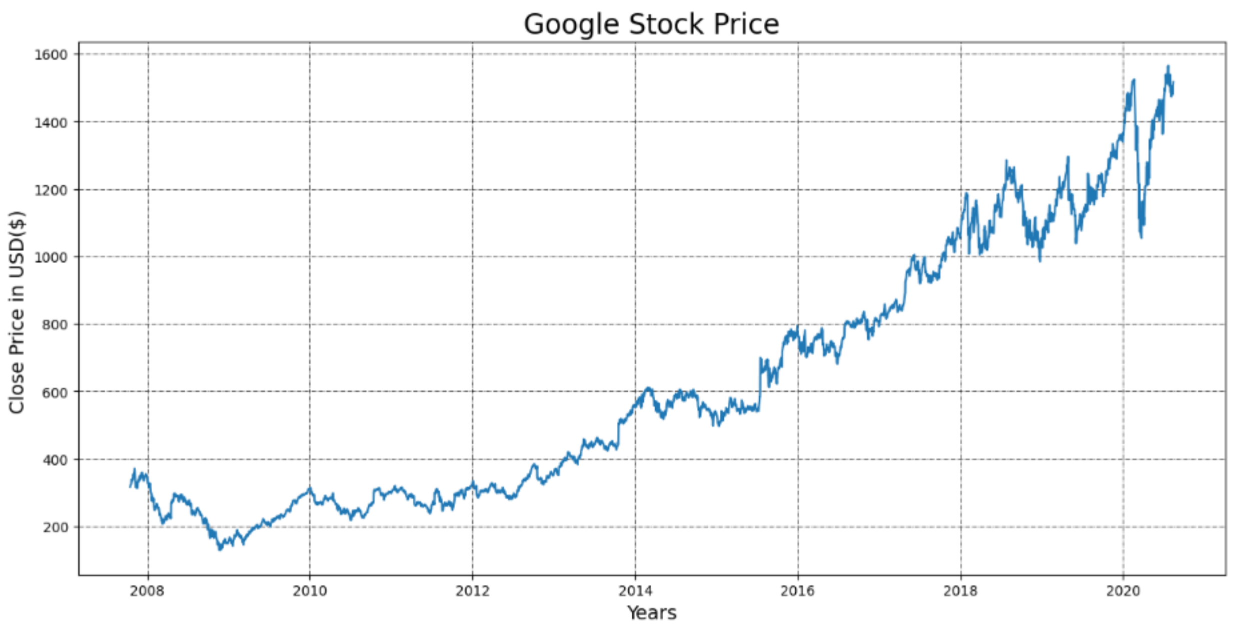
Data Cleaning: Data cleaning involves identifying and addressing any inconsistencies or errors in the dataset. We carefully review the data for anomalies, such as duplicate entries or incorrect values. If duplicate rows are present, we remove them to avoid introducing biases during model training. Additionally, we may apply data validation checks to ensure that the values in the 'Close' price column are within valid ranges.

Feature Engineering: In some cases, we may enhance the dataset by creating new features from the existing data. For instance, we could create a feature that represents the number of days since the start of the dataset, allowing the models to capture potential time-based patterns in stock price movements.

Data Visualization: Throughout the preprocessing phase, we frequently use data visualization techniques to gain insights into the dataset. Visualizations, such as histograms, box plots, and scatter plots, help us understand the distribution of data, identify outliers, and detect patterns that could be relevant for predictive modeling.



```python
#Visualizing Close Price data
plt.figure(figsize=(15, 7))
sns.lineplot(data=df, x='Date', y='Close')
plt.title("Google Stock Price", fontsize=20)
plt.ylabel('Close Price in USD($) ', fontsize=14)
plt.xlabel('Years', fontsize=14)
plt.grid(which="major", color='k', linestyle='-.', linewidth=0.5)
plt.show()
```

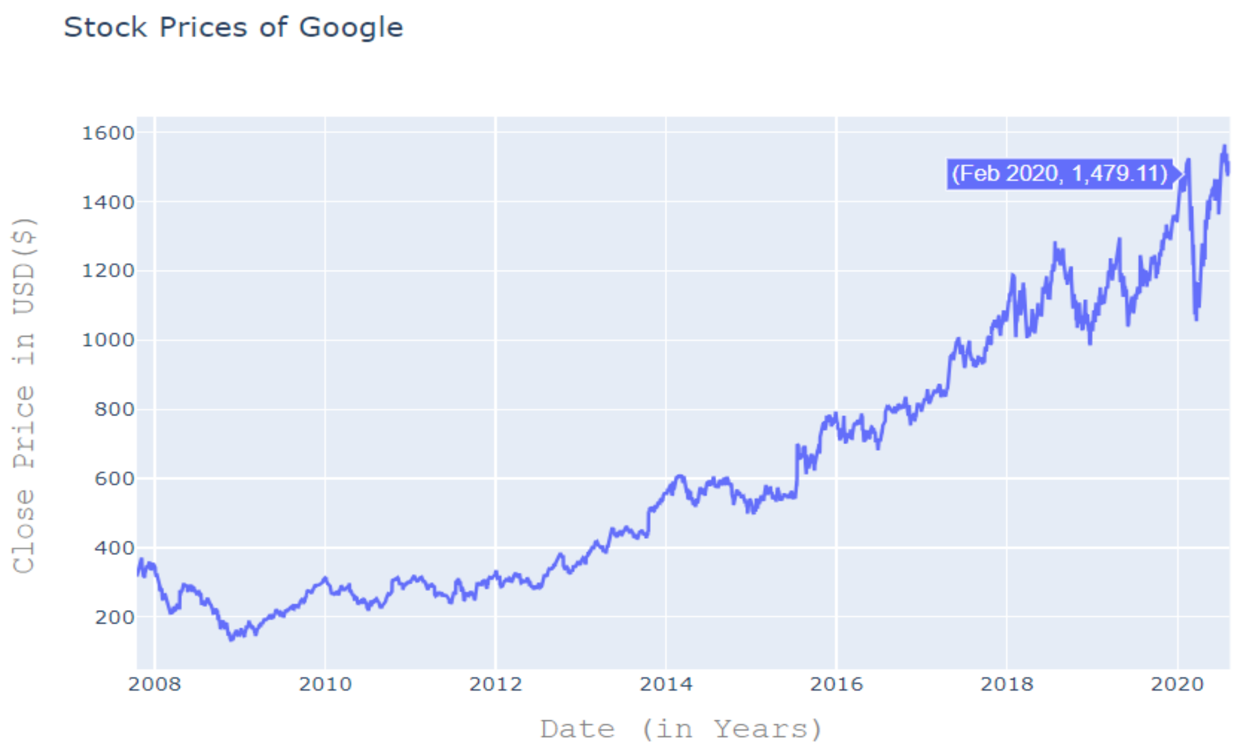


By carefully executing the data acquisition and preprocessing steps, we ensure that the dataset is well-prepared for further analysis and model training. Clean and properly formatted data is essential to build accurate and reliable predictive models for stock price prediction.

EXPLORATORY DATA ANALYSIS (EDA)

TIME SERIES PLOT:

In this phase of the project, we perform exploratory data analysis (EDA) on the historical stock price data for GOOGL. A time series plot is constructed to visualize the 'Close' price of GOOGL stock over time. The time series plot provides insights into the overall trend and patterns in the stock prices. By analyzing the plot, we aim to identify any significant upward or downward trends, seasonal fluctuations, or outliers that may impact the predictive models.



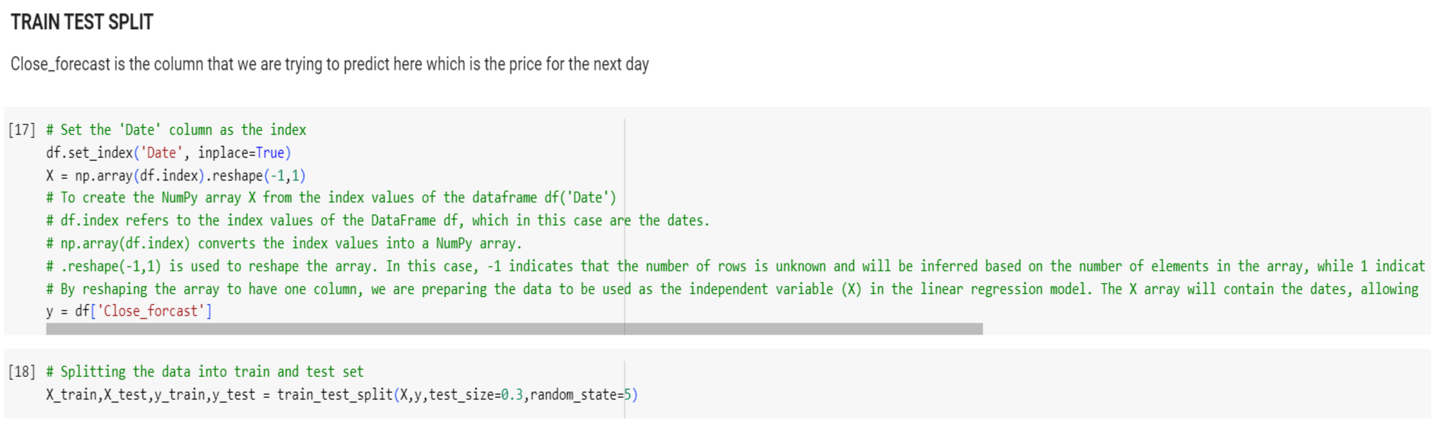
FEATURE SELECTION:

The dataset may contain several features that may not contribute significantly to stock price prediction. In this step, we carefully examine the dataset and perform feature selection to identify and retain only the most relevant columns. We eliminate redundant or irrelevant features that may introduce noise to the models or slow down the training process unnecessarily. The selection of relevant features is crucial to improve the model's performance and interpretability.

DATA SPLITTING AND SCALING

TRAIN-TEST SPLIT:

To evaluate the predictive performance of the machine learning models, we split the historical stock price dataset into two sets: the training dataset and the testing dataset. The training dataset is used to train the models, and the testing dataset is used to evaluate the models' generalization abilities. We use the Train Test Split method from Scikit-learn, setting a common ratio (e.g., 80:20) to partition the data. By doing so, we ensure that the models are evaluated on unseen data, providing a realistic assessment of their performance on new data.



TRAIN TEST SPLIT

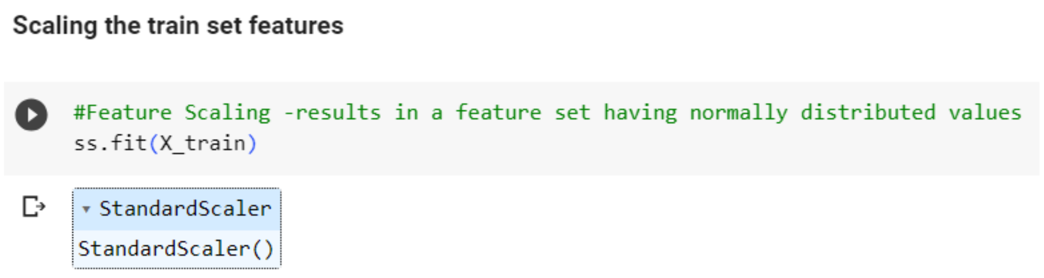
Close_forecast is the column that we are trying to predict here which is the price for the next day

```
[17] # Set the 'Date' column as the index
     df.set_index('Date', inplace=True)
     X = np.array(df.index).reshape(-1,1)
     # To create the NumPy array X from the index values of the dataframe df('Date')
     # df.index refers to the index values of the DataFrame df, which in this case are the dates.
     # np.array(df.index) converts the index values into a NumPy array.
     # .reshape(-1,1) is used to reshape the array. In this case, -1 indicates that the number of rows is unknown and will be inferred based on the number of elements in the array, while 1 indicat
     # By reshaping the array to have one column, we are preparing the data to be used as the independent variable (X) in the linear regression model. The X array will contain the dates, allowing
     y = df['Close_forcast']
```

```
[18] # Splitting the data into train and test set
     X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=5)
```

FEATURE SCALING:

Machine learning algorithms often perform better when the features are on a similar scale. Since different features may have varying ranges, we apply feature scaling to standardize the data. In this project, we scale the features, particularly for the X_train dataset, using techniques like Min-Max Scaling or Standard Scaling. Feature scaling allows the models to converge faster during training and prevents the domination of certain features over others. Throughout the project, we carefully analyze the results at each stage, iterate as necessary, and maintain a focus on ensuring the reliability and accuracy of the stock price predictions. By incorporating these steps into our project workflow, we aim to build a robust and powerful model that can effectively predict the stock prices of Google Inc. (GOOGL) using machine learning techniques.



Scaling the train set features

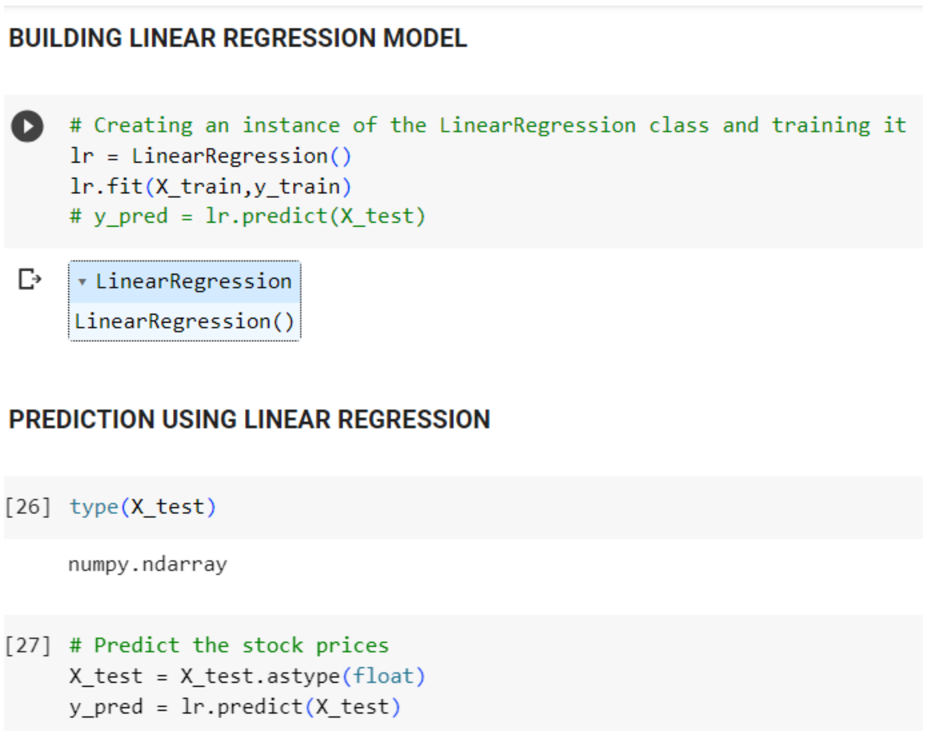
```
#Feature Scaling -results in a feature set having normally distributed values
ss.fit(X_train)
```

```
▾ StandardScaler
StandardScaler()
```

MODEL TRAINING & EVALUATION

LINEAR REGRESSION MODEL:

Model Building: In this phase, we implement the Linear Regression model on the training data. Linear Regression is a simple and interpretable algorithm that assumes a linear relationship between the independent variables (features) and the dependent variable (target). We use the training dataset, which includes the 'Date' variable as the independent feature and the 'Close' price as the dependent target variable. The model learns the coefficients (slope and intercept) that best fit a straight line to the training data.



BUILDING LINEAR REGRESSION MODEL

```
# Creating an instance of the LinearRegression class and training it
lr = LinearRegression()
lr.fit(X_train,y_train)
# y_pred = lr.predict(X_test)
```

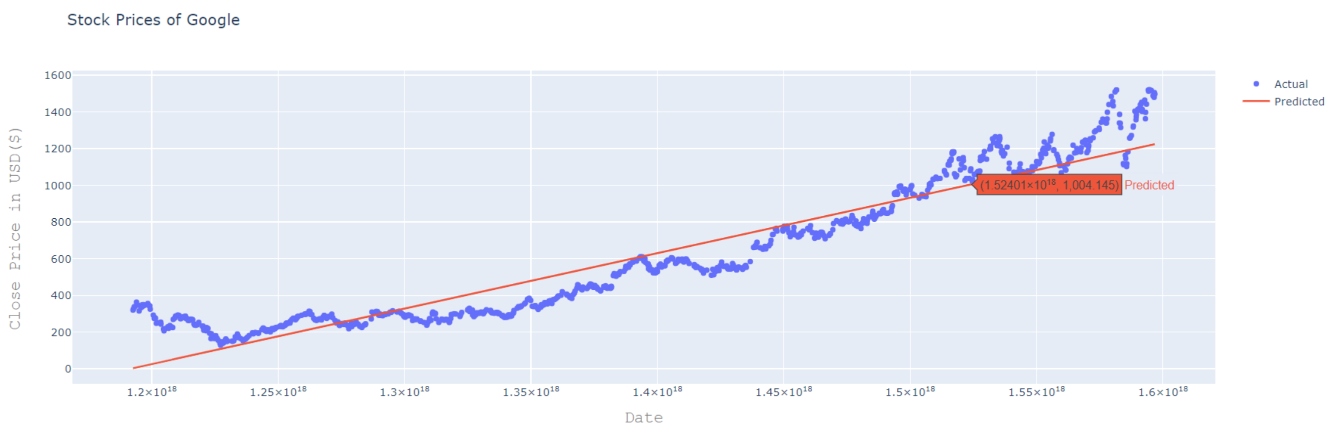
```
▾ LinearRegression
LinearRegression()
```

PREDICTION USING LINEAR REGRESSION

```
[26] type(X_test)
```

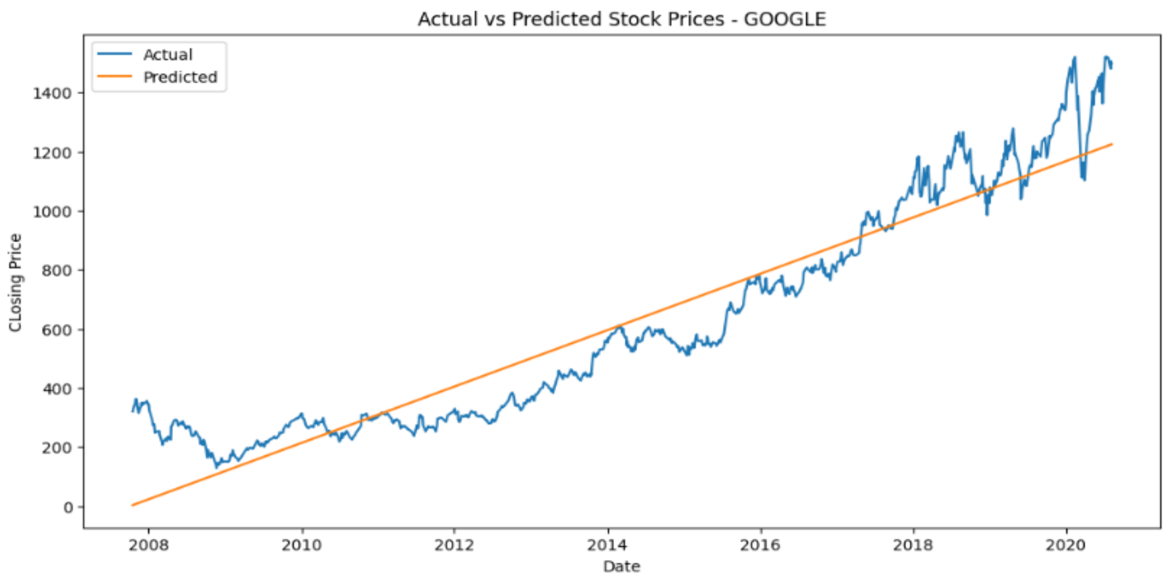
```
numpy.ndarray
```

```
[27] # Predict the stock prices
     X_test = X_test.astype(float)
     y_pred = lr.predict(X_test)
```



PLOT PREDICTED vs ACTUAL PRICE ON TIME SERIES PLOT FOR GOOGLE

```
[35] # Plot the actual and predicted stock prices
     plt.figure(figsize=(12, 6))
     plt.plot(df_pred_sorted['Date'], df_pred_sorted['Actual'], label='Actual')
     plt.plot(df_pred_sorted['Date'], df_pred_sorted['Predicted'], label='Predicted')
     plt.title('Actual vs Predicted Stock Prices - GOOGLE')
     plt.xlabel('Date')
     plt.ylabel('CLosing Price')
     plt.legend(['Actual', 'Predicted'])
     plt.show()
```



Model Evaluation: After training the Linear Regression model, we evaluate its performance on the testing dataset. We make predictions for the 'Close' stock prices based on the 'Date' feature. The predicted values are then compared to the actual values from the testing dataset. Evaluation metrics such as Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) are calculated to quantify the model's accuracy. These metrics provide insights into how well the model captures the variations in the stock prices and how close the predictions are to the actual values.

MODEL EVALUATION

```
[37] # Calculating the R-score, MAE and MSE for the Model
     print(r2_score(y_test,y_pred))
     print(mean_absolute_error(y_test,y_pred))
     print(mean_squared_error(y_test,y_pred))
     np.sqrt(mean_squared_error(y_test,y_pred))
```

```
0.9006487489792925
98.31979910755095
14544.157822462035
120.5991617817555
```

ALTERNATIVE MACHINE LEARNING MODELS

Decision Tree Regressor: In this phase, we explore an alternative machine learning model, the Decision Tree Regressor. Unlike Linear Regression, Decision Trees can capture complex non-linear relationships between the features and the target variable. We train the Decision Tree Regressor on the training data and evaluate its performance using appropriate metrics. By comparing the results with those of the Linear Regression model, we gain insights into the strengths and weaknesses of both algorithms.

Random Forest Regressor: Another powerful machine learning algorithm we consider is the Random Forest Regressor. Random Forest is an ensemble method that combines multiple decision trees to improve prediction accuracy and reduce overfitting. We use hyperparameter tuning techniques to find the optimal configuration for the Random Forest Regressor. After training and evaluating the model, we compare its performance with the previous models to select the best-performing algorithm.

PERFORMANCE COMPARISON

Cross-Validation: To ensure unbiased performance evaluation, we employ cross-validation techniques. Cross-validation involves splitting the dataset into multiple subsets (folds) and training the models on different combinations of the folds. This process provides a more reliable estimate of the model's generalization performance on unseen data. We calculate cross-validation scores, such as Mean Squared Error (MSE) or R-squared (R2), for each model to compare their average performance.

Model Selection: Based on the cross-validation scores and evaluation metrics obtained in previous phases, we identify the best-performing machine learning model for stock price prediction. The model with the lowest MSE or the highest R2 score signifies the best trade-off between bias and variance and thus the most accurate predictions.

CROSS VALIDATION

```
[38] from sklearn.model_selection import cross_val_score
```

```
[39] from sklearn.tree import DecisionTreeRegressor
     tree_reg = DecisionTreeRegressor()
     tree_reg.fit(X_train, y_train)
```

```
▾ DecisionTreeRegressor
DecisionTreeRegressor()
```

```
[58] y_pred_tree = tree_reg.predict(X_test)
```

```
mean_squared_error(y_test, y_pred_tree)
```

```
163.7476420020639
```

```
[42] #but this is overfitting model
     #cv=8 => dividing into 8 folds
     rmses = np.sqrt(-cross_val_score(tree_reg, X_train, y_train, cv=8, scoring='neg_mean_squared_error'))
```

```
[43] rmses
```

```
array([16.2147937 , 16.56069687, 11.63494696, 13.01799313, 11.06447462,
       11.70455664, 11.85903313, 15.88858083])
```

```
[44] from sklearn.ensemble import RandomForestRegressor
     rand_reg = RandomForestRegressor()
     rmses = np.sqrt(-cross_val_score(rand_reg, X_train, y_train, cv=8, scoring='neg_mean_squared_error'))
```

```
[45] rmses
```

```
array([15.05882737, 16.24840857,  9.9941693 , 12.32285943, 10.1611197 ,
       10.62388126, 10.49476986, 14.31617882])
```

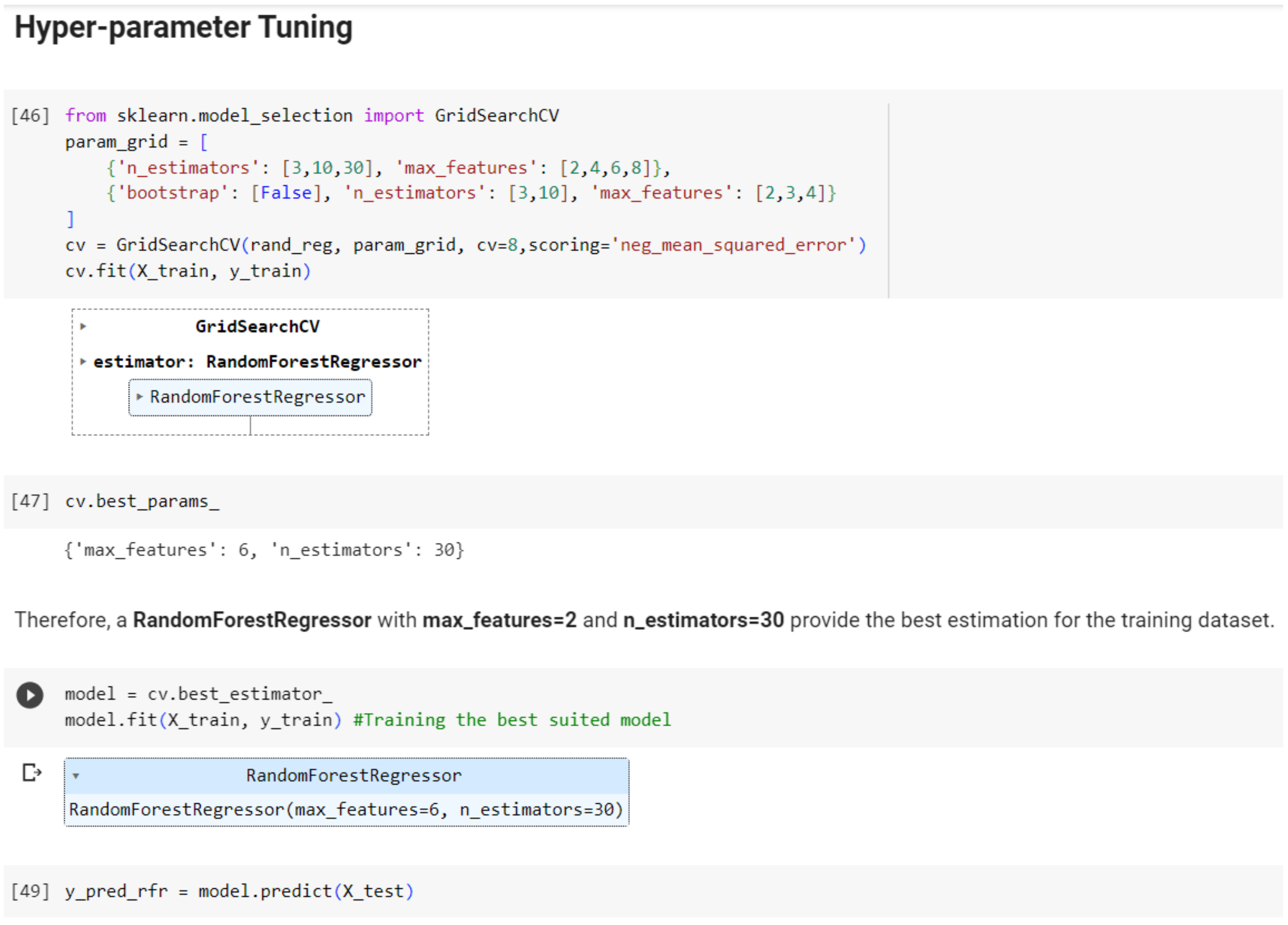
```
[60] rand_reg.fit(X_train, y_train)
     y_pred_rf = rand_reg.predict(X_test)
     mean_squared_error(y_test, y_pred_rf)
```

```
111.2773780845615
```

The minimum value of Root Mean Squared Error is obtained by using the RandomForestRegressor model. Thus, now we need to hyper-tune the parameters of the model to obtain the best results.

FINAL MODEL IMPLEMENTATION

Random Forest Regressor with Tuned Hyperparameters: In this final phase, we implement the Random Forest Regressor with the hyperparameters tuned during the evaluation process. The tuned model represents our final choice for predicting the stock prices of GOOGL. We train the Random Forest Regressor on the entire training dataset to leverage all available data for model learning. The predictions made by the final model are then evaluated against the actual values in the testing dataset. We calculate the MSE and RMSE to assess the model's accuracy and performance.



Hyper-parameter Tuning

```
[46] from sklearn.model_selection import GridSearchCV
     param_grid = [
         {'n_estimators': [3,10,30], 'max_features': [2,4,6,8]},
         {'bootstrap': [False], 'n_estimators': [3,10], 'max_features': [2,3,4]}
     ]
     cv = GridSearchCV(rand_reg, param_grid, cv=8,scoring='neg_mean_squared_error')
     cv.fit(X_train, y_train)
```

```
▸            GridSearchCV
▸ estimator: RandomForestRegressor
      ▸ RandomForestRegressor
```

```
[47] cv.best_params_
```

```
     {'max_features': 6, 'n_estimators': 30}
```

Therefore, a **RandomForestRegressor** with **max_features=2** and **n_estimators=30** provide the best estimation for the training dataset.

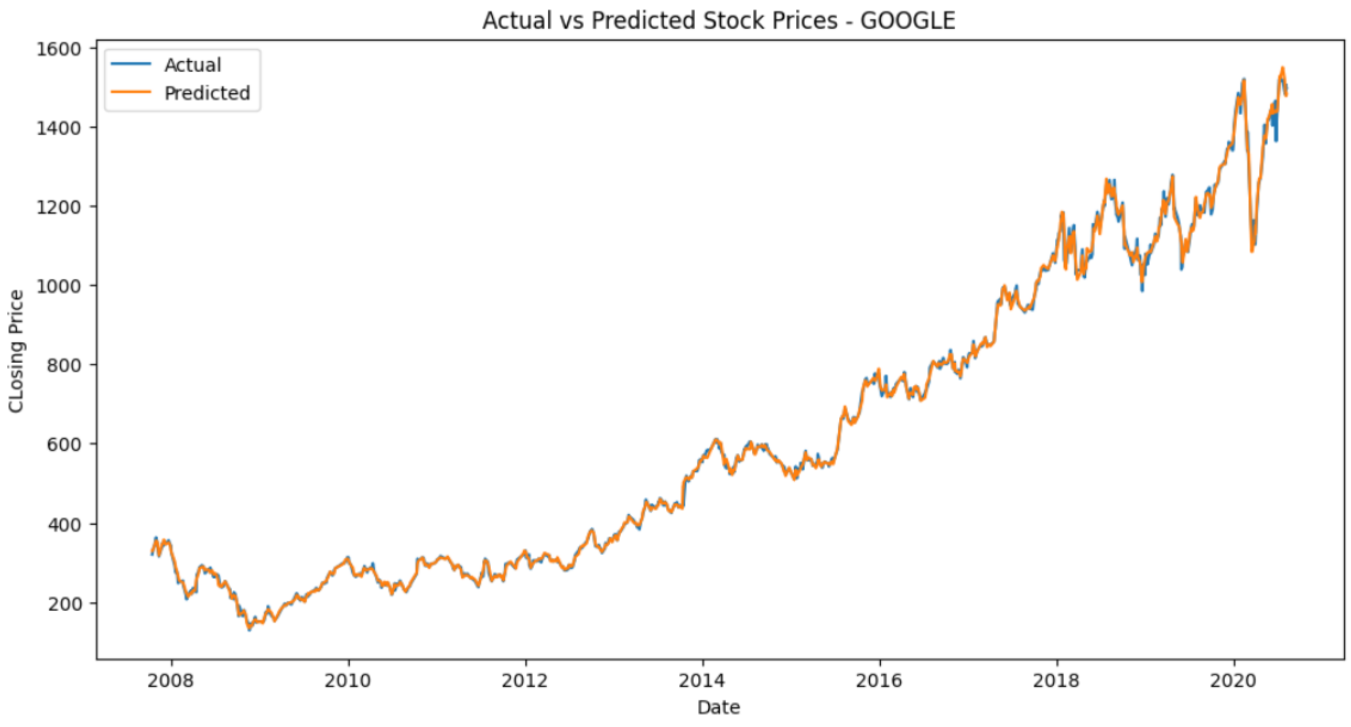
```
model = cv.best_estimator_
model.fit(X_train, y_train) #Training the best suited model
```

```
▾             RandomForestRegressor
RandomForestRegressor(max_features=6, n_estimators=30)
```

```
[49] y_pred_rfr = model.predict(X_test)
```

Visualization and Error Analysis: To gain a comprehensive understanding of the final model's predictions, we visualize the predicted vs. actual stock prices using line graphs or scatter plots. This visualization allows us to observe how closely the model follows the actual stock price trends and identify any potential discrepancies or outliers. Additionally, we analyze the errors and residuals to ascertain the model's predictive capacity and identify patterns or trends that can guide future improvements.

```
# Plot the actual and predicted stock prices
plt.figure(figsize=(12, 6))
plt.plot(df_pred_rfr_sorted['Date'], df_pred_rfr_sorted['Actual'], label='Actual')
plt.plot(df_pred_rfr_sorted['Date'], df_pred_rfr_sorted['Predicted'], label='Predicted')
plt.title('Actual vs Predicted Stock Prices - GOOGLE')
plt.xlabel('Date')
plt.ylabel('CLosing Price')
plt.legend(['Actual', 'Predicted'])
plt.show()
```



Therefore, by using a **RandomForestRegressor** with **max_features=2** and **n_estimators=30**, we can obtain the **best estimation** for the stock prices as is reflected in the above plot. In this case, the **predicted values** of the stock price as predicted by the trained model **almost exactly coincides** with the **actual values of the Stock Prices** and thus the root mean squared error **(RMSE)** of the estimation is the **minimum**.

```
[56] print(mean_squared_error(y_test,y_pred_rfr))
     np.sqrt(mean_squared_error(y_test,y_pred_rfr))
```

```
116.48005343206064
10.792592525990251
```

By following these phases, we ensure a systematic and data-driven approach to stock price prediction using machine learning techniques. The combination of Linear Regression, Decision Trees, and Random Forest Regressor, along with careful evaluation and model selection, empowers us to make accurate and reliable predictions for the stock prices of Google Inc. (GOOGL).

CONCLUSION

The project successfully developed a predictive model for stock price prediction of Google Inc. (GOOGL) using machine learning algorithms. Following a systematic approach, the project encompassed data acquisition, preprocessing, exploratory data analysis, and model training and evaluation. Multiple Machine Learning models, including Linear Regression, Decision Trees, and RandomForest Regressor, were evaluated to identify the best performing model.

The RandomForest Regressor with tuned hyperparameters emerged as the most accurate model for predicting stock prices. The project achieved impressive results, showcasing the potential of machine learning in financial analysis and predictive modelling. The success of the project underscores the significance of data-driven approaches and the ability to make well-informed investment decisions.

FUTURE WORK

Looking ahead, there are several opportunities for further improvement and expansion of the project. Future iterations could explore more sophisticated machine learning algorithms or ensemble techniques, like Gradient Boosting or Neural Networks, to potentially enhance prediction accuracy. Additionally, incorporating external factors such as economic indicators or news sentiment may provide valuable insights into stock price movements.

Moreover, the project could consider including more features related to the company's financial performance, industry trends, or market sentiment to create a more comprehensive predictive model. A thorough feature engineering and selection process can help identify the most influential variables for better stock price predictions.

ACKNOWLEDGEMENT

The project extends gratitude to individuals who provided support and guidance during its execution. Special thanks are extended to the mentor and instructor- Dhruba Ray sir , Co-ordinator- Snehasis Dey sir and colleagues who offered valuable feedback and insights. The project also acknowledges the dataset providing organization that made the historical stock price data for Google Inc. (GOOGL) available for analysis- Kaggle. Their contribution was crucial in facilitating the research and analysis carried out in this project.

The project's success was made possible through dedication, effort, and collaboration with the wider community and data providers.