Students

stock price pediction

TEAM MEMBER

# 

AU732521104024 :   KAVIN.P

**PHASE 3: SUBMISSION DOCUMENT**

Team member

Project title: stock price prediction

A graph with arrows pointing up

Description automatically generated with medium confidence

**INTRODUCTION:**

Greetings from the Kaggle bot! This is an automatically-generated kernel with starter code demonstrating how to read in the data and begin exploring. If you're inspired to dig deeper, click the blue "Fork Notebook" button at the top of this kernel to begin editing.

1. **Data Collection:** Stock price prediction relies heavily on historical data. This data typically includes historical stock prices, trading volumes, financial reports, and other relevant information. Data can be collected from various sources such as financial news websites, stock exchanges, and financial databases.
2. **Features:** In machine learning and data analysis, features are variables that help in making predictions. In stock price prediction, features can include historical stock prices, trading volumes, company financial indicators, economic indicators, sentiment analysis of news articles, and more

.

1. **Time Series Analysis:** Stock price data is typically analyzed as a time series, which means that the data points are ordered chronologically. Time series analysis techniques are often used to identify patterns, trends, and seasonality in the data.
2. **Model Selection:** To predict stock prices, various machine learning models can be used, such as linear regression, decision trees, random forests, support vector machines, and deep learning models like recurrent neural networks (RNNs) and long short-term memory networks (LSTMs). The choice of model depends on the complexity of the problem and the quality of the data.
3. **Evaluation Metrics:** To assess the performance of a stock price prediction model, several evaluation metrics can be used. Common metrics include Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and accuracy scores.
4. **Feature Engineering:** This is the process of selecting, transforming, or creating new features from the raw data to improve the model's predictive performance. It may involve techniques like smoothing, differencing, or lagging.
5. **Overfitting and Generalization:** Preventing overfitting is crucial in stock price prediction. Overfitting occurs when a model learns to perform well on the training data but doesn't generalize well to new, unseen data. Regularization techniques and careful model selection help mitigate overfitting.
6. **Market Sentiment Analysis:** Public sentiment, as expressed in news articles, social media, and analyst reports, can influence stock prices. Sentiment analysis techniques are often integrated into models to capture this aspect.
7. **Risk Management:** Stock price prediction is inherently risky. It's important to understand that no model can predict stock prices with absolute certainty. Risk management strategies, such as stop-loss orders and portfolio diversification, are essential for minimizing potential losses.
8. **Backtesting:** To assess the effectiveness of a stock price prediction model, it's common to conduct backtesting. This involves applying the model to historical data to see how it would have performed in the past.

**EXPLORATORY ANALYSIS**

To begin this exploratory analysis, first import libraries and define functions for plotting the data using matplotlib. Depending on the data, not all plots will be made. (Hey, I'm just a simple kerneling bot, not a Kaggle Competitions Grandmaster!)

**DATA SOURCE:**

**DATASET LINK**: <https://www.kaggle.com/datasets/prasoonkottarathil/microsoft-lifetime-stocks-dataset/>

**Data Collection and Preprocessing:**

* Importing the dataset: Obtain a comprehensive dataset containing relevant features

such as square footage, number of bedrooms, location, amenities, etc.

* Data preprocessing: Clean the data by handling missing values, outliers, and

categorical variables. Standardize or normalize numerical features.

**PROGRAM:**

**STOCK PRICE PREDICTION**

**Importing Libraries**

Python libraries make it very easy for us to handle the data and perform typical and complex tasks with a single line of code.

[**Pandas**](https://www.geeksforgeeks.org/python-pandas-dataframe/)– This library helps to load the data frame in a 2D array format and has multiple functions to perform analysis tasks in one go.

[**Numpy**](https://www.geeksforgeeks.org/python-numpy/)– Numpy arrays are very fast and can perform large computations in a very short time. [**Matplotlib**](https://www.geeksforgeeks.org/matplotlib-tutorial/)/[**Seaborn**](https://www.geeksforgeeks.org/introduction-to-seaborn-python/)– This library is used to draw visualizations.

Sklearn – This module contains multiple libraries having pre-implemented functions to perform tasks from data preprocessing to model development and evaluation.

[**XGBoost**](https://www.geeksforgeeks.org/xgboost/)– This contains the eXtreme Gradient Boosting machine learning algorithm which is one of the algorithms which helps us to achieve high accuracy on predictions.

**PROGRAM:**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sb

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LogisticRegression

from sklearn.svm import SVC

from xgboost import XGBClassifier

from sklearn import metrics

import warnings

warnings.filterwarnings('ignore')

**Importing Dataset**

The dataset we will use here to perform the analysis and build a predictive model is Tesla Stock Price data. We will use OHLC(‘Open’, ‘High’, ‘Low’, ‘Close’) data from 1st January 2010 to 31st December 2017 which is for 8 years for the Tesla stocks.

You can download the CSV file from <https://www.kaggle.com/datasets/prasoonkottarathil/microsoft-lifetime-stocks-dataset/code>

**DATA SOURCE FILE IMPORTED:**

df = pd.read\_csv('/content/MSFT.csv')

df.head()

**Output:**



From the first five rows, we can see that data for some of the dates is missing the reason for that is on weekends and holidays Stock Market remains closed hence no trading happens on these days.

* Python3

|  |
| --- |
| df.shape |

**Output**:

(1692, 7)

From this, we got to know that there are 1692 rows of data available and for each row, we have 7 different features or columns.

* Python3

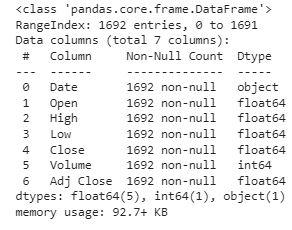
|  |
| --- |
| df.describe() |

**Output:**



|  |
| --- |
| df.info() |

**Output:**



**Exploratory Data Analysis**

[EDA](https://www.geeksforgeeks.org/what-is-exploratory-data-analysis/) is an approach to analyzing the data using visual techniques. It is used to discover trends, and patterns, or to check assumptions with the help of statistical summaries and graphical representations.

While performing the EDA of the Tesla Stock Price data we will analyze how prices of the stock have moved over the period of time and how the end of the quarters affects the prices of the stock.

|  |
| --- |
| plt.figure(figsize=(15,5))  plt.plot(df['Close'])  plt.title('Tesla Close price.', fontsize=15)  plt.ylabel('Price in dollars.')  plt.show() |

**Output:**



The prices of tesla stocks are showing an upward trend as depicted by the plot of the closing price of the stocks.

|  |
| --- |
| df.head() |

**Output:**



If we observe carefully we can see that the data in the ‘Close’ column and that available in the ‘Adj Close’ column is the same let’s check whether this is the case with each row or not.

* Python3

|  |
| --- |
| df[df['Close'] == df['Adj Close']].shape |

**Output:**

(1692, 7)

From here we can conclude that all the rows of columns ‘Close’ and ‘Adj Close’ have the same data. So, having redundant data in the dataset is not going to help so, we’ll drop this column before further analysis.

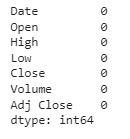
|  |
| --- |
| df = df.drop(['Adj Close'], axis=1) |

Now let’s draw the distribution plot for the continuous features given in the dataset.

Before moving further let’s check for the null values if any are present in the data frame.

|  |
| --- |
| df.isnull().sum() |

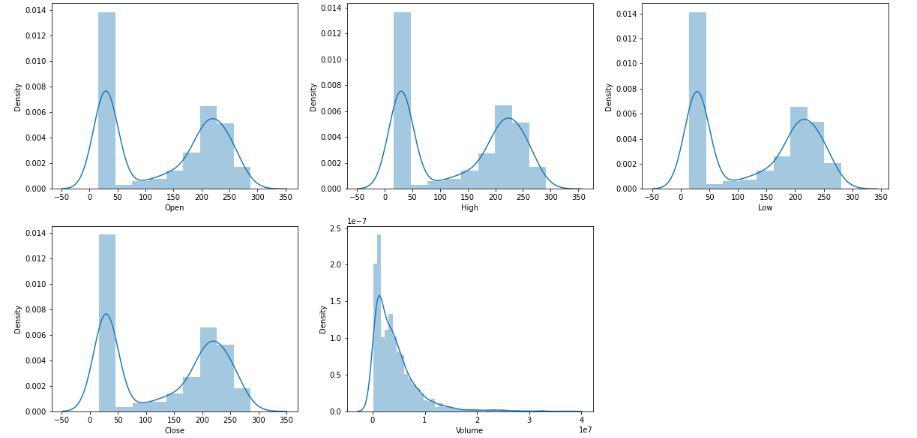
**Output:**



This implies that there are no null values in the data set provided.

|  |
| --- |
| features = ['Open', 'High', 'Low', 'Close', 'Volume']    plt.subplots(figsize=(20,10))    for i, col in enumerate(features):    plt.subplot(2,3,i+1)    sb.distplot(df[col])  plt.show() |

**Output:**

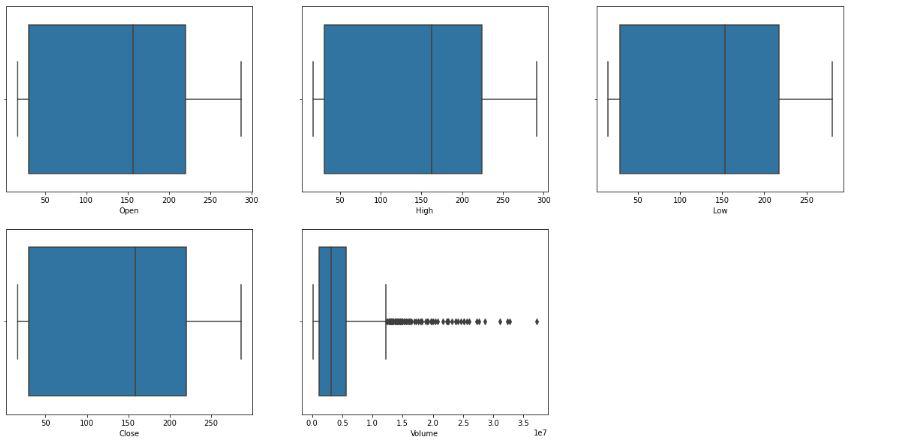


*Distribution Plot of the Continuous Variable*

In the [distribution plot](https://www.geeksforgeeks.org/seaborn-distribution-plots/) of OHLC data, we can see two peaks which means the data has varied significantly in two regions. And the Volume data is left-skewed.

|  |
| --- |
| plt.subplots(figsize=(20,10))  for i, col in enumerate(features):    plt.subplot(2,3,i+1)    sb.boxplot(df[col])  plt.show() |

**Output:**



*Box Plot of the Continuous Variable*

From the above [boxplots](https://www.geeksforgeeks.org/horizontal-boxplots-with-seaborn-in-python/), we can conclude that only volume data contains outliers in it but the data in the rest of the columns are free from any outlier.

**Feature Engineering**

Feature Engineeringhelps to derive some valuable features from the existing ones. These extra features sometimes help in increasing the performance of the model significantly and certainly help to gain deeper insights into the data.

|  |
| --- |
| splitted = df['Date'].str.split('/', expand=True)    df['day'] = splitted[1].astype('int')  df['month'] = splitted[0].astype('int')  df['year'] = splitted[2].astype('int')    df.head() |

**Output:**

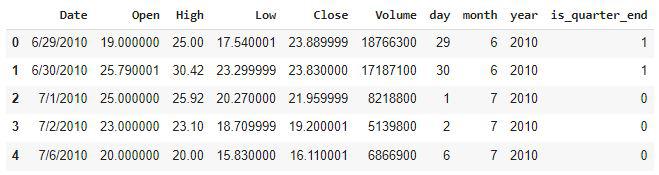


Now we have three more columns namely ‘day’, ‘month’ and ‘year’ all these three have been derived from the ‘Date’ column which was initially provided in the data.

* Python3

|  |
| --- |
| df['is\_quarter\_end'] = np.where(df['month']%3==0,1,0)  df.head() |

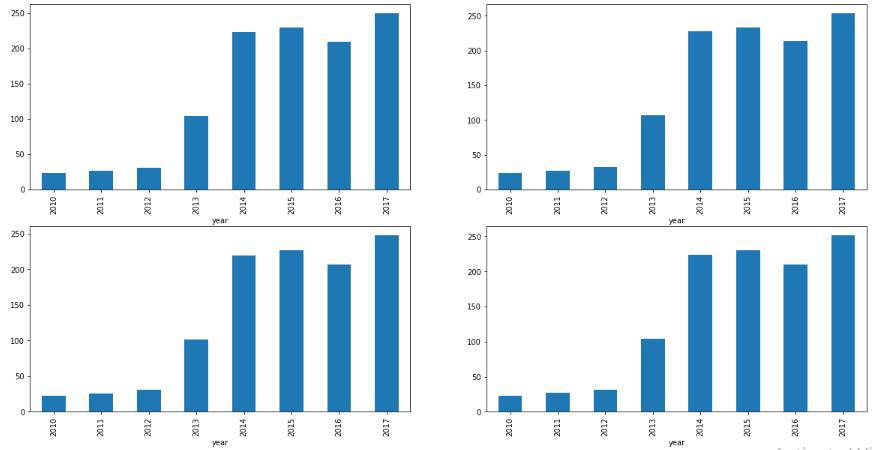
**Output:**



A quarter is defined as a group of three months. Every company prepares its quarterly results and publishes them publicly so, that people can analyze the company’s performance. These quarterly results affect the stock prices heavily which is why we have added this feature because this can be a helpful feature for the learning model.

|  |
| --- |
| data\_grouped = df.groupby('year').mean()  plt.subplots(figsize=(20,10))    for i, col in enumerate(['Open', 'High', 'Low', 'Close']):    plt.subplot(2,2,i+1)    data\_grouped[col].plot.bar()  plt.show() |

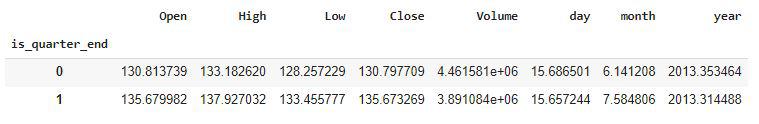
**Output:**



From the above [bar graph](https://www.geeksforgeeks.org/bar-plot-in-matplotlib/), we can conclude that the stock prices have doubled from the year 2013 to that in 2014.

|  |
| --- |
| df.groupby('is\_quarter\_end').mean() |

**Output:**



Here are some of the important observations of the above-grouped data:

* Prices are higher in the months which are quarter end as compared to that of the non-quarter end months.
* The volume of trades is lower in the months which are quarter end.
* Python3

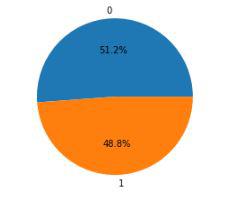
|  |
| --- |
| df['open-close']  = df['Open'] - df['Close']  df['low-high']  = df['Low'] - df['High']  df['target'] = np.where(df['Close'].shift(-1) > df['Close'], 1, 0) |

Above we have added some more columns which will help in the training of our model. We have added the target feature which is a signal whether to buy or not we will train our model to predict this only. But before proceeding let’s check whether the target is balanced or not using a [pie chart](https://www.geeksforgeeks.org/plot-a-pie-chart-in-python-using-matplotlib/).

* Python3

|  |
| --- |
| plt.pie(df['target'].value\_counts().values,          labels=[0, 1], autopct='%1.1f%%')  plt.show() |

**Output:**

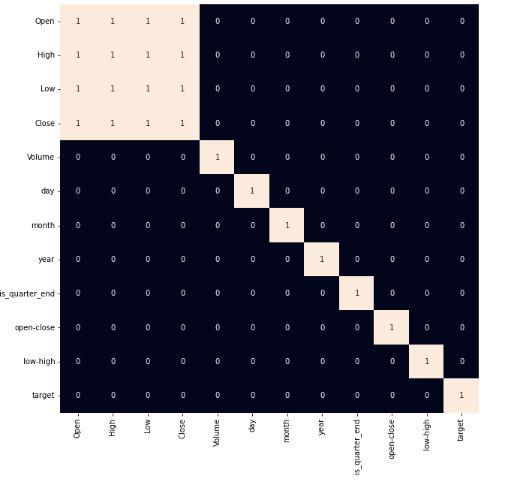


When we add features to our dataset we have to ensure that there are no highly correlated features as they do not help in the learning process of the algorithm.

* Python3

|  |
| --- |
| plt.figure(figsize=(10, 10))    # As our concern is with the highly  # correlated features only so, we will visualize  # our heatmap as per that criteria only.  sb.heatmap(df.corr() > 0.9, annot=True, cbar=False)  plt.show() |

**Output:**



*Heatmap of the correlation between the features*

From the above [heatmap](https://www.geeksforgeeks.org/seaborn-heatmap-a-comprehensive-guide/), we can say that there is a high correlation between OHLC that is pretty obvious, and the added features are not highly correlated with each other or previously provided features which means that we are good to go and build our model.

**Data Splitting and Normalization**

* Python3

|  |
| --- |
| features = df[['open-close', 'low-high', 'is\_quarter\_end']]  target = df['target']    scaler = StandardScaler()  features = scaler.fit\_transform(features)    X\_train, X\_valid, Y\_train, Y\_valid = train\_test\_split(      features, target, test\_size=0.1, random\_state=2022)  print(X\_train.shape, X\_valid.shape) |

**Output:**

(1522, 3) (170, 3)

After selecting the features to train the model on we should [**normalize**](https://www.geeksforgeeks.org/what-is-data-normalization-and-why-is-it-important/)the data because normalized data leads to stable and fast training of the model. After that whole data has been [**split**](https://www.geeksforgeeks.org/how-to-split-a-dataset-into-train-and-test-sets-using-python/)into two parts with a 90/10 ratio so, that we can evaluate the performance of our model on unseen data.

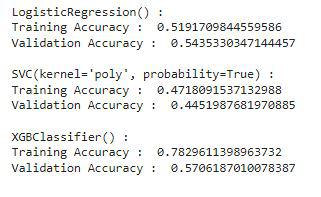
**Model Development and Evaluation**

Now is the time to train some state-of-the-art machine learning models([Logistic Regression](https://www.geeksforgeeks.org/understanding-logistic-regression/), [Support Vector Machine](https://www.geeksforgeeks.org/support-vector-machine-algorithm/), [XGBClassifier](https://www.geeksforgeeks.org/ml-xgboost-extreme-gradient-boosting/)), and then based on their performance on the training and validation data we will choose which ML model is serving the purpose at hand better.

For the evaluation metric, we will use the [ROC-AUC curve](https://www.geeksforgeeks.org/auc-roc-curve/) but why this is because instead of predicting the hard probability that is 0 or 1 we would like it to predict soft probabilities that are continuous values between 0 to 1. And with soft probabilities, the ROC-AUC curve is generally used to measure the

|  |
| --- |
| models = [LogisticRegression(), SVC(    kernel='poly', probability=True), XGBClassifier()]    for i in range(3):    models[i].fit(X\_train, Y\_train)      print(f'{models[i]} : ')    print('Training Accuracy : ', metrics.roc\_auc\_score(      Y\_train, models[i].predict\_proba(X\_train)[:,1]))    print('Validation Accuracy : ', metrics.roc\_auc\_score(      Y\_valid, models[i].predict\_proba(X\_valid)[:,1]))    print() |

**Output:**



*Evaluation of the model on training and the testing data.*

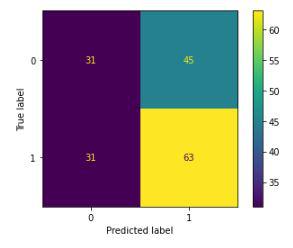
Among the three models, we have trained XGBClassifier has the highest performance but it is pruned to [overfitting](https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/) as the difference between the training and the validation accuracy is too high. But in the case of the Logistic Regression, this is not the case.

Now let’s plot a [**confusion matrix**](https://www.geeksforgeeks.org/confusion-matrix-machine-learning/) for the validation data.

* Python3

|  |
| --- |
| metrics.plot\_confusion\_matrix(models[0], X\_valid, Y\_valid)  plt.show() |

**Output:**



*Confusion matrix for the validation data*

**Conclusion:**

We can observe that the accuracy achieved by the state-of-the-art ML model is no better than simply guessing with a probability of 50%. Possible reasons for this may be the lack of data or using a very simple model to perform such a complex task as Stock Market prediction.