

**COLLEGE CODE: 5113**

# APPLIED DATA SCIENCE

**Project No.6 - STOCK PRICE PREDICTION**

Batch Members:

1. ABHINANDAN A - (au511321104001) - abhiarjun0610@gmail.com
2. ABISHEK V P - (au511321104002) – abishekreddy7112003@gmail.com
3. MAHEDHAR V – (au511321104051) - vmahedhar7@gmail.com
4. BHAKTHULA CHANDU – (au511321104009)- chanchandu353@gmail.com

**PHASE 3:DATA SET PREPROCESSING**

**Dataset Explanation:**

**ABOUT DATASET:**

Where did we get the dataset?

**Kaggle:**

The dataset provided on Kaggle, titled "Microsoft Lifetime Stocks Dataset" (accessible at <https://www.kaggle.com/datasets/prasoonkottarathil/microsoft-lifetime-stocks-dataset>), offers a valuable resource for our project aimed at forecasting stock prices. This dataset primarily focuses on Microsoft's stock market performance over a substantial period, making it an excellent choice for our predictive modelling task.

**Dataset Details:**

This dataset comprises a comprehensive set of attributes that are essential for analysing and forecasting stock prices:

1. **Date:** A crucial component of time series data, the date allows us to track stock price changes over time.
2. **Open Price:** This represents the opening price of Microsoft's stock on a given trading day. It is one of the primary indicators of daily market dynamics.
3. **High Price:** The highest price reached during the trading day, providing insights into intraday fluctuations.
4. **Low Price:** The lowest price recorded on the same trading day, indicating the day's lowest level of market activity.
5. **Close Price:** The closing price of Microsoft's stock, which holds significance as it often reflects investors' sentiment and can influence trading decisions.
6. **Volume:** This attribute records the trading volume for the stock on a specific date, helping to identify days of high market activity.

This dataset spans a significant time frame, which is vital for training and testing our predictive model. The availability of additional features, such as high and low prices, further enriches the dataset, enabling us to capture a wide range of market dynamics. Overall, the "Microsoft Lifetime Stocks Dataset" on Kaggle is an ideal resource for our stock price forecasting project

**DATASET LOADING:**

To load the dataset, you'll need to have the dataset file downloaded and in your working directory.

import pandas as pd

path="C:/Users/91861/Desktop/MSFT.csv"

data=pd.read\_csv(path)

print(data.head())

**OUTPUT:**

Date Open High Low Close Adj Close Volume

0 1986-03-13 0.088542 0.101563 0.088542 0.097222 0.062549 1031788800

1 1986-03-14 0.097222 0.102431 0.097222 0.100694 0.064783 308160000

2 1986-03-17 0.100694 0.103299 0.100694 0.102431 0.065899 133171200

3 1986-03-18 0.102431 0.103299 0.098958 0.099826 0.064224 67766400

4 1986-03-19 0.099826 0.100694 0.097222 0.098090 0.063107 47894400

**DATASET PREPROCESSING:**

Dataset preprocessing is a critical step in preparing your data for stock price prediction. Here, I'll provide you with a basic outline of preprocessing tasks. Depending on your specific project requirements, you may need to perform more extensive preprocessing. Make sure you've loaded the dataset as described in the previous response before starting with preprocessing.

**Importing the required Libraries:**

To perform data preprocessing, you will need several libraries in Python. Here's how you can import the required libraries for data preprocessing:

**import pandas as pd** #For data manipulation and analysis

**import numpy as np** #For numerical operations

**from sklearn.preprocessing import StandardScaler, MinMaxScaler**  #For feature scaling

**from sklearn.impute import SimpleImputer** #For handling missing data

**from sklearn.model\_selection import train\_test\_split** #For splitting data into training and testing sets

**Importing the data set**

To import and read a dataset and create a matrix from it, you can use the pandas library in Python. In this code, we first load the dataset using pd.read\_csv into a Pandas DataFrame. Then, we use the .values attribute to extract the data from the DataFrame and create a NumPy array (matrix). This matrix can be used for various data analysis or machine learning tasks.

**import pandas as pd**

**dataset\_path = "microsoft\_data.csv" # Replace with the actual file path**

**data = pd.read\_csv(dataset\_path)**

**matrix = data.values**

**output:**

[['1986-03-13' 0.088542 0.101563 ... 0.097222 0.062549 1031788800]

['1986-03-14' 0.097222 0.102431 ... 0.100694 0.064783 308160000]

['1986-03-17' 0.100694 0.103299 ... 0.102431 0.065899 133171200]

...

['2020-01-03' 158.320007 159.949997 ... 158.619995 158.619995 21116200]

['2020-01-06' 157.080002 159.100006 ... 159.029999 159.029999 20813700]

['2020-01-07' 159.320007 159.669998 ... 157.580002 157.580002 18017762]]

**Handling the Missing Data**

To handle missing data using the sklearn.preprocessing library, you can utilize the SimpleImputer class. Here's how you can do it:

**import pandas as pd**

**from sklearn.impute import SimpleImputer**

**dataset\_path = "C:/Users/91861/Desktop/MSFT.csv”**

**data = pd.read\_csv(dataset\_path)**

**missing\_values = data.isnull().sum()**

**print("Missing Values:\n", missing\_values)**

**imputer = SimpleImputer(strategy='mean')**

**data\_imputed = pd.DataFrame(imputer.fit\_transform(data), columns=data.columns)**

**missing\_values\_after\_imputation = data\_imputed.isnull().sum()**

**print("Missing Values After Imputation:\n", missing\_values\_after\_imputation)**

**output:**

Missing Values

Date 0

Open 0

High 0

Low 0

Close 0

Adj Close 0

Volume 0

**Encoding Categorical Data:**

To encode categorical data, particularly for features with multiple categories, you can use one-hot encoding. This process converts categorical variables into binary columns for each category. Here's how to perform one-hot encoding using the pandas library in Python:

**import pandas as pd**

**dataset\_path = "C:/Users/91861/Desktop/MSFT.csv”**

**data = pd.read\_csv(dataset\_path)**

**categorical\_column = "YourCategoricalColumn"**

**data\_encoded=pd.get\_dummies(data, columns=[categorical\_column])**

**Splitting the data set into test set and training set:**

To split your dataset into training and testing sets, you can use the train\_test\_split function from the sklearn.model\_selection library. This function randomly divides your data into two subsets: one for training your model and another for testing its performance. Here's how to do it:

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

**X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(Open, Close, test\_size=0.2, random\_state=42)**

**Feature Scaling:**

Feature scaling is an important preprocessing step when working with machine learning models. The StandardScaler from the sklearn.preprocessing library can be used to scale your feature variables (X) so that they have a mean of 0 and a standard deviation of 1. This helps to standardize the range and units of your features, making them suitable for many machine learning algorithms. Here's how to use the StandardScaler:

**scaler = StandardScaler()**

**scaler.fit(X\_train)**

**X\_train\_scaled = scaler.transform(X\_train)**

**X\_test\_scaled = scaler.transform(X\_test)**

**PERFORMING DIFFERENT ANALYSIS:**

Performing different types of analysis on a dataset depends on the goals of your analysis and the nature of the data. Here are some common types of analysis that you might performon a dataset:

**Descriptive Analysis:**

Summarize and describe the main characteristics of the

dataset, including measures of central tendency, dispersion,

and visualizations such as histograms, box plots, and bar

charts.

**Exploratory Data Analysis (EDA):**

Explore the dataset to uncover patterns, relationships, and

anomalies.

Visualize data using scatter plots, heatmaps, and correlation

matrices.

Identify potential outliers and trends.

**Statistical Analysis:**

Conduct hypothesis testing and statistical inference to make

inferences about the data.

Perform t-tests, ANOVA, chi-squared tests, and other statistical

tests as appropriate.

**Regression Analysis:**

Build regression models to predict a continuous target variable

based on one or more predictor variables.

Evaluate model performance using metrics like R-squared,

Mean Squared Error (MSE), and Root Mean Squared Error

(RMSE).

**Classification Analysis:**

Develop classification models to predict categorical outcomes

or classes.

Evaluate model performance using metrics such as accuracy,

precision, recall, F1-score, and ROC curves.

**Clustering Analysis:**

Apply clustering algorithms to group similar data points

together.

Use techniques like K-means, hierarchical clustering, or

DBSCAN.

**Anomaly Detection:**

Identify outliers or anomalies in the dataset using methods like

isolation forests or one-class SVM.

**CODE:**

**import pandas as pd**

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

**from sklearn.linear\_model import LinearRegression**

**from sklearn.metrics import mean\_squared\_error, r2\_score**

**import matplotlib.pyplot as plt**

**dataset\_path = "microsoft\_data.csv"**

**data = pd.read\_csv(dataset\_path)**

**X = data[['Open Price']]**

**Y = data['Close Price']**

**X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.2, random\_state=42)**

**model = LinearRegression()**

**model.fit(X\_train, Y\_train)**

**Y\_pred = model.predict(X\_test)**

**mse = mean\_squared\_error(Y\_test, Y\_pred)**

**r2 = r2\_score(Y\_test, Y\_pred)**

**print("Mean Squared Error:", mse)**

**print("R-squared (R2) Score:", r2)**

**plt.scatter(X\_test, Y\_test, color='blue', label='Actual')**

**plt.plot(X\_test, Y\_pred, color='red', linewidth=2, label='Predicted')**

**plt.xlabel('Open Price')**

**plt.ylabel('Close Price')**

**plt.legend()**

**plt.show()**

**CONCLUSION:**

In conclusion, data preprocessing sets the foundation for successful stock price prediction and financial data analysis. It ensures that your data is clean, properly formatted, and ready for machine learning. A well-pre processed dataset can lead to more accurate models and, ultimately, better investment decisions in the context of stock price prediction.