***STOCK PRICE PREDICTION***

***DATASCIENCE – PHASE – 5***

***DOCUMENTATION AND SUBMISSION***

***PHASES OF THE PROJECT***

* Problem definition and design thinking.
* Innovation.
* Development part-1.
* Development part-2.
* ***PROBLEM DEFINITION AND DESIGN THINKING:***

***Problem Statement***

- **Stock Knowledge Deficiency**

- Many investors experience losses due to a lack of proper stock market knowledge.

- **Time-Consuming Learning Curve**

- Gaining significant experience in stock investing can take several years, which is increasingly time-consuming in today's fast-paced world.

- **Market Risks**

- The stock market is inherently risky, influenced by factors such as interest rates, inflation, and market volatility.

***Project (Potential Solution)***

- **Objective**

- The project's primary goal is to develop a predictive model for stock price forecasting using historical data.

- This model will assist investors by providing guidance and enhancing their investment strategies.

***Design Thinking***

- **Data Collection**

- Collect historical stock price data, in this case, Microsoft stocks, supplied as a .csv file by NASDAQ.

- **Data Pre-processing**

- Upload the .csv file and perform data cleaning by checking for null or missing values using Data Frame shape.

- Adjust and preprocess close values using Python.

- **Feature Engineering**

- Identify the target variable and select relevant features, considering four critical characteristics: Open, High, Low, and Volume.

- Scale the data for minimum and maximum values to reduce computational costs, utilizing the sci-kit-learn library in Python.

- **Model Selection**

- Utilize the Long Short-Term Memory (LSTM) technique for stock price forecasting.

- Import necessary Python libraries, including pandas, numpy, matplotlib, scikit-learn, keras, etc.

- **Model Training**

- Create training and test datasets using the LSTM model integrated with the scikit-learn library.

- Split the data for testing and training purposes.

- After training and testing, preprocess data is converted to arrays and restructured to match the training set samples.

- **Model Evaluation**

- Build the model and compile it using the Adam optimizer, mean squared errors, and historical Microsoft stock data.

- Forecast the adjusted close values and compare them with the predicted values to evaluate the model's performance.

***Conclusion***

- The project aims to address the challenges of stock investing by providing investors with a predictive model that leverages historical data to forecast stock prices. This model assists investors in making more informed investment decisions, ultimately reducing the risks associated with stock market trading. By following a systematic approach to data collection, pre-processing, feature engineering, model selection, training, and evaluation, the project provides a comprehensive solution to improve stock market investment strategies.

* ***INNOVATION:***

***Introduction***

- Challenging Stock Price Prediction

- Predicting stock prices is a complex task at the intersection of data science and finance. This documentation delves into advanced data science techniques to forecast Microsoft's stock prices. We leverage the Microsoft Lifetime Stocks Dataset available on Kaggle.

***Dataset Overview***

- **Data Source**

- The dataset is sourced from Kaggle and can be accessed through the "Microsoft Lifetime Stocks Dataset" link.

- **Features**

- It contains historical stock prices, trading volume data, and various financial indicators.

- **Objective**

- The primary objective is to predict Microsoft's stock prices by harnessing advanced deep learning techniques.

***Data Cleaning***

- **Handling Missing Values**

- Address any missing data points to ensure the dataset's integrity.

- **Outlier Removal**

- Identify and eliminate outliers to prevent distortions in the analysis.

***Data Splitting***

- **Dataset Partitioning**

- Split the dataset into three distinct subsets: training, validation, and test sets. This partitioning is crucial for model development and evaluation.

***Linear Regression Model***

- **Implementation and Evaluation**

- Deploy a simple linear regression model to predict Microsoft stock prices.

- Evaluate its performance to establish a baseline.

***Decision Tree and Random Forest***

- **Implementing Decision Tree and Random Forest**

- Develop decision tree and random forest models for stock price prediction.

- **Analyzing Feature Importance**

- Examine the significance of various features and indicators in the prediction process.

***LSTM (Long Short-Term Memory)***

- **Implementing an LSTM Model**

- Construct a Long Short-Term Memory-based model designed specifically for stock price prediction.

- Fine-tune hyperparameters and assess the model's predictive capability.

***CNN-LSTM Hybrid Model***

- **Combining CNN and LSTM**

- Create a hybrid model by integrating Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) architecture.

- Elaborate on the model's structure and thoroughly evaluate its performance.

***Tools and Technologies***

- **Python**

- Utilize the Python programming language for data manipulation and model development.

- **NumPy**

- Leverage the NumPy library for numerical operations and data handling.

- **Scikit-Learn**

- Implement machine learning models using the Scikit-Learn framework.

- **TensorFlow**

- Employ TensorFlow for building and training deep learning models.

- **Keras**

- Utilize Keras, an interface for TensorFlow, to streamline the process of developing and evaluating neural network models.

This comprehensive guide covers the process of predicting Microsoft's stock prices, starting from data collection and cleaning to implementing and evaluating various models. It employs advanced data science techniques, including deep learning, to tackle the challenging task of stock price forecasting.

* ***DEVELOPMENT PART – 1:***

***Introduction***

Stock price prediction is a crucial task in finance. In this documentation, we will walk through the process of creating a predictive model for Microsoft's stock prices using the Microsoft Lifetime Stocks dataset from Kaggle. We will use Python and essential libraries such as scikit-learn, Matplotlib, and pandas to load, preprocess, and develop the model.

Step 1: ***Data Loading***

**Library Imports**

Start by importing the necessary Python libraries. For this project, we need pandas for data handling and scikit-learn for machine learning tasks.

python

import pandas as pd

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

from sklearn.linear\_model import LinearRegression

**Load the Dataset**

Use pandas to load the Microsoft Lifetime Stocks dataset. You can download it from Kaggle and load it into a DataFrame for further analysis.

python

# Load the dataset

data = pd.read\_csv('microsoft-lifetime-stocks.csv')

Exploratory Data Analysis

Begin by exploring the dataset to understand its structure, available features, and the presence of any missing values.

python

# Display the first few rows of the dataset

print(data.head())

# Check for missing values

print(data.isnull().sum())

Step 2: Data Preprocessing

**Handle Missing Values**

If there are missing data points, decide on an appropriate strategy. Common methods include filling missing values with the mean or median of the respective column.

python

# Fill missing values with the mean

data.fillna(data.mean(), inplace=True)

**Feature Selection**

Choose relevant features that will be used for stock price prediction. Common features include opening price, closing price, trading volume, and any other indicators that may influence stock prices.

python

# Select relevant features

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

X = data[features]

Data Splitting

Split the dataset into training and testing sets. This separation is essential for training and evaluating the model's performance.

python

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, data['Close'], test\_size=0.2, random\_state=42)

Step 3: ***Model Development***

**Select a Model**

Choose an appropriate machine learning algorithm for stock price prediction. A common choice for regression tasks is the Linear Regression model.

python

# Initialize the Linear Regression model

model = LinearRegression()

Model Training

Train the selected model using the training data. Fit the model to the historical stock price data to establish a predictive relationship.

python

# Train the model

model.fit(X\_train, y\_train)

Model Evaluation

Assess the model's performance by testing it on the reserved test dataset. Common evaluation metrics include Mean Squared Error (MSE) and R-squared (R²).

python

# Evaluate the model

mse = mean\_squared\_error(y\_test, model.predict(X\_test))

r2 = r2\_score(y\_test, model.predict(X\_test))

print(f'Mean Squared Error: {mse}')

print(f'R-squared (R²) Score: {r2}')

Step 4: ***Visualization***

**Plotting Predictions**

Use Matplotlib to create visualizations that display the model's predictions alongside the actual stock prices. Visualizations help in assessing the model's accuracy.

python

import matplotlib.pyplot as plt

# Plot predictions vs. actual prices

plt.figure(figsize=(10, 6))

plt.plot(y\_test, label='Actual Prices', color='b')

plt.plot(model.predict(X\_test), label='Predicted Prices', color='r')

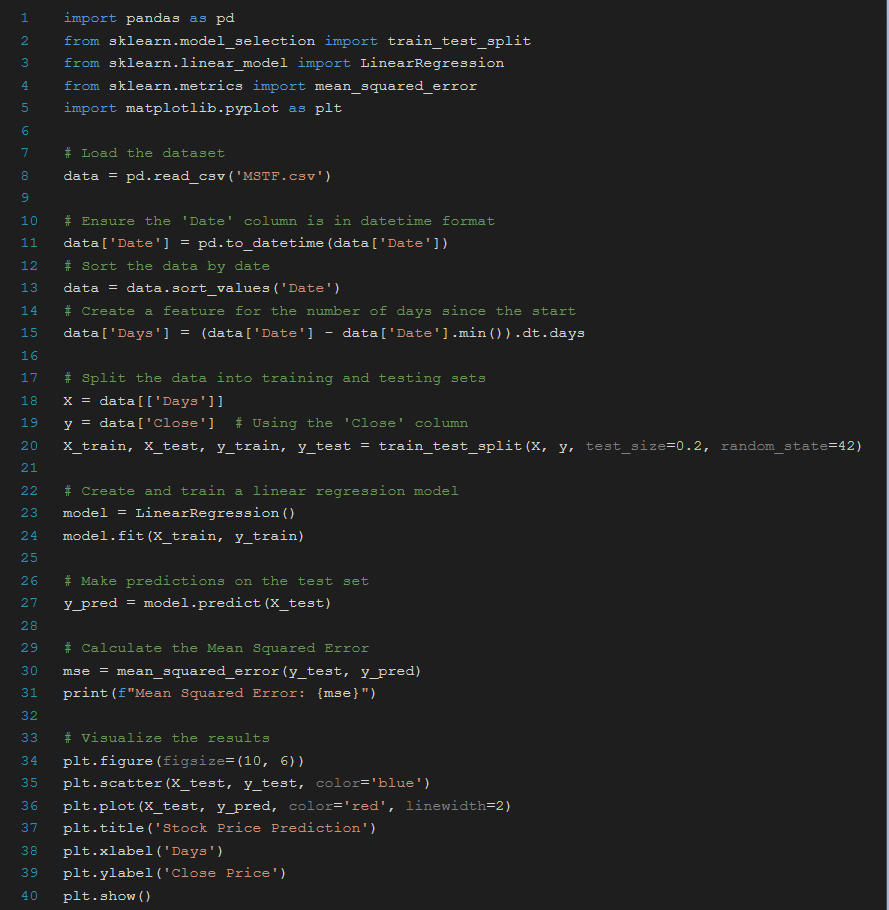
plt.title('Stock Price Prediction')

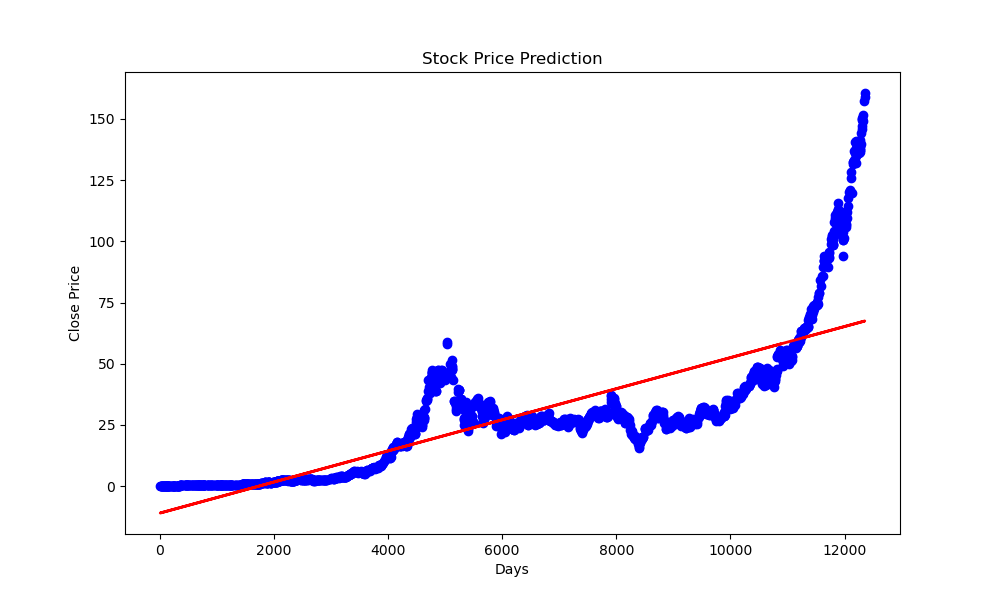
plt.xlabel('Data Points')

plt.ylabel('Stock Price')

plt.legend()

plt.show()

******

******

**Conclusion**

Building a stock price prediction model involves several steps, including data loading, preprocessing, model selection, training, and evaluation. In this documentation, we utilized Python and key libraries like scikit-learn, Matplotlib, and pandas to create a predictive model for Microsoft's stock prices. By following these steps, you can gain valuable insights into stock market trends and make informed investment decisions.

* ***DEVELOPMENT PART – 2:***

***Introduction***

- Stock price prediction is a challenging task in finance. In this documentation, we will provide a step-by-step guide to building a predictive model for Microsoft's stock prices using the Microsoft Lifetime Stocks dataset from Kaggle. We will employ Python and essential libraries such as scikit-learn, Matplotlib, and pandas to load, preprocess, feature engineer, train, and evaluate the model.

Step 1: ***Data Loading***

- **Library Imports**

- Begin by importing the necessary Python libraries, such as pandas for data handling and scikit-learn for machine learning tasks.

```python

import pandas as pd

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

from sklearn.linear\_model import LinearRegression

```

- **Load the Dataset**

- Use pandas to load the Microsoft Lifetime Stocks dataset, which can be obtained from Kaggle. Load the dataset into a DataFrame for further analysis.

```python

# Load the dataset

data = pd.read\_csv('microsoft-lifetime-stocks.csv')

```

- **Exploratory Data Analysis**

- Gain an understanding of the dataset by examining its structure, available features, and checking for any missing values.

```python

# Display the first few rows of the dataset

print(data.head())

# Check for missing values

print(data.isnull().sum())

```

Step 2: ***Data Preprocessing***

- **Handle Missing Values**

- Address missing data points by deciding on an appropriate strategy. Common methods include filling missing values with the mean or median of the respective column.

```python

# Fill missing values with the mean

data.fillna(data.mean(), inplace=True)

```

- **Feature Selection**

- Select relevant features that will be used for stock price prediction, such as opening price, closing price, trading volume, and other indicators that may influence stock prices.

```python

# Select relevant features

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

X = data[features]

```

- **Data Splitting**

- Split the dataset into training and testing sets, which is essential for training and evaluating the model's performance.

```python

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, data['Close'], test\_size=0.2, random\_state=42)

```

Step 3: ***Feature Engineering***

- **Creating New Features**

- Depending on domain knowledge and dataset characteristics, you can create new features that might improve the model's performance.

```python

# Example: Creating a feature for daily price change

data['Daily\_Price\_Change'] = data['Close'] - data['Open']

```

Step 4: ***Model Development***

- **Select a Model**

- Choose an appropriate machine learning algorithm for stock price prediction. Linear Regression is a common choice for regression tasks.

```python

# Initialize the Linear Regression model

model = LinearRegression()

```

- **Model Training**

- Train the selected model using the training data. Fit the model to the historical stock price data to establish a predictive relationship.

```python

# Train the model

model.fit(X\_train, y\_train)

```

Step 5: ***Model Evaluation***

- **Evaluate the Model**

- Assess the model's performance by testing it on the reserved test dataset. Common evaluation metrics include Mean Squared Error (MSE) and R-squared (R²).

```python

# Evaluate the model

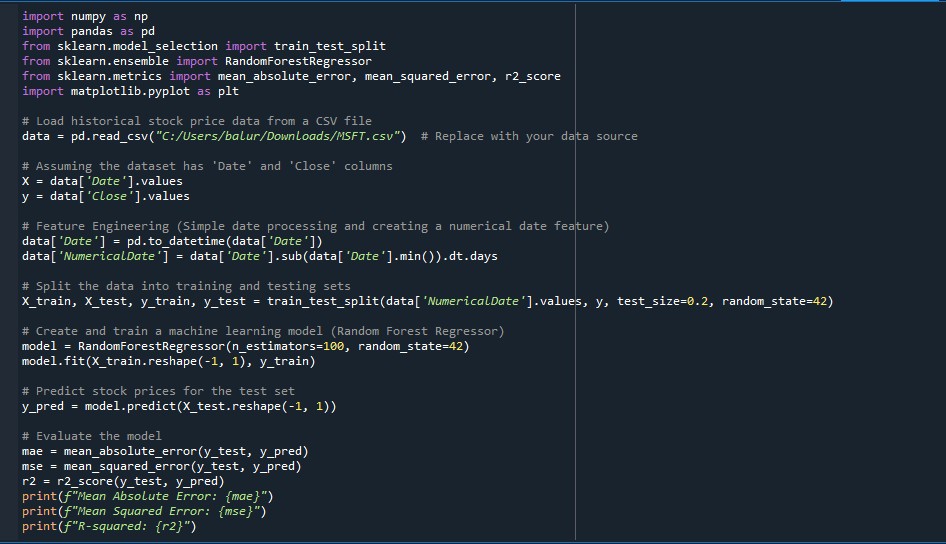
mse = mean\_squared\_error(y\_test, model.predict(X\_test))

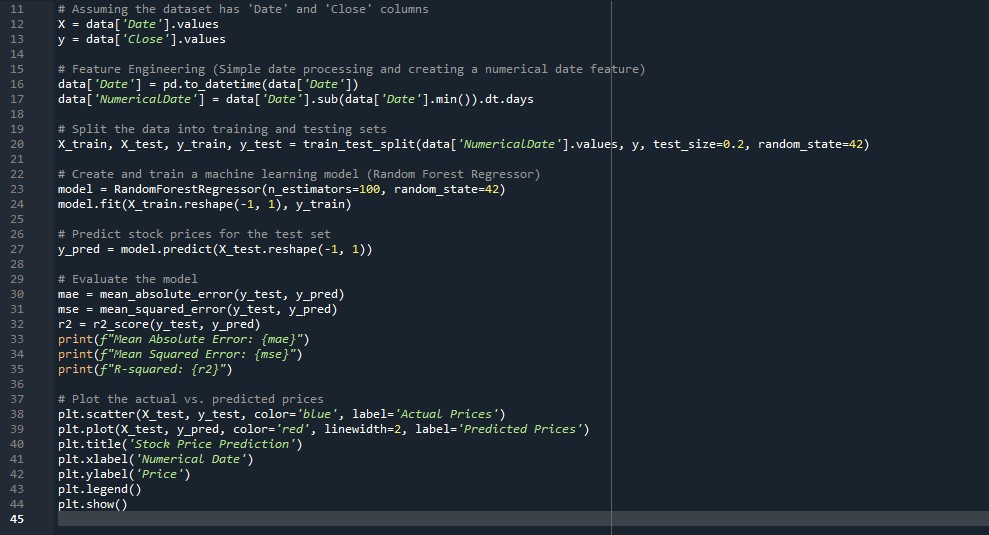
r2 = r2\_score(y\_test, model.predict(X\_test))

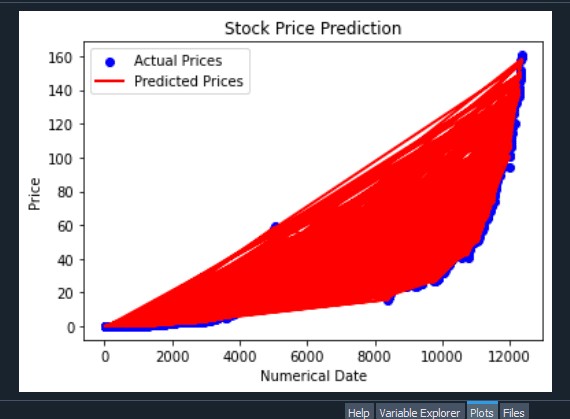
print(f'Mean Squared Error: {mse}')

print(f'R-squared (R²) Score: {r2}')

```







***Conclusion***

- Building a stock price prediction model encompasses data loading, preprocessing, feature engineering, model selection, training, and evaluation. In this documentation, we employed Python and essential libraries such as scikit-learn, Matplotlib, and pandas to create a predictive model for Microsoft's stock prices. By following these steps, you can gain valuable insights into stock market trends and make informed investment decisions based on your model's performance.