APPLIED DATA SCIENCE - PAHSE 3

STOCK PRICE PREDICTION

DATASET

A contain various types of information, such as text, numbers, images, or any other data. Datasets are commonly used in various fields, including research, machine learning, statistics, and data dataset is a structured collection of data that is organized and formatted for a specific purpose. It can analysis. They serve as the foundation for tasks like training machine learning models, conducting experiments, and drawing conclusions based on data. Datasets can vary in size and complexity, from small and simple ones to large and intricate collections of information.

This dataset has been taken from kaggle.com

Link: https://www.kaggle.com/datasets/prasoonkottarathil/microsoft-lifetime-stocks-dataset

DETAILS OF COLUMN USED:

Date - in format: yy-mm-dd.

Open - price of the stock at market open (this is NYSE data so all in USD)

High - Highest price reached in the day

Low Close - Lowest price reached in the day

Volume - Number of shares traded

In the model, we generally make use of the date column to mark it correctly in the corresponding row of the graph and we use the close value of the stocks to predict its future price.

We can even make use of open high volume and close to predict even more accurate prices as the no. of parameter increases hence making the model more precise.

LOADING THE DATASET:

Dataset is the most integral part of the data science process as It is the base on which the model feeds on and through this only the ML/AI algorithms interpret the pattern of the data and eventually help us finding the insights from them.

The loading of a dataset is generally done by the read_csv() method in python which is available in the pandas library of python.

Syntax-

variable = pandas.read_csv('FileName.csv')

PRE-PROCESSING THE DATA:

Preprocessing the data is a crucial step in building a stock price prediction model. It involves cleaning, transforming, and scaling the data to make it suitable for the machine learning algorithm. Here are some key preprocessing steps that could be used in the model:

1. Handling Missing Data: Check for missing values in the dataset and handle them appropriately. You can choose to remove rows with missing values or use techniques like imputation to fill in the missing values.

2. Feature Scaling: Since stock prices may have different scales, it's essential to scale the features to a standard range. Common scaling techniques include Min-Max scaling or standardization.

3. Handling Time Series Data: If your dataset includes time series data, consider applying time-based transformations or generating lag features to capture temporal patterns effectively.

4. Feature Engineering: Create new features that can provide additional information for the model. This can include technical indicators, moving averages, or other derived metrics that can capture important market trends.

5. Splitting the Data: Split the dataset into training and testing sets to evaluate the model's performance on unseen data. You can use techniques like time-based splitting or random shuffling, depending on the characteristics of the data.

6. Handling Outliers: Identify and handle outliers in the dataset to prevent them from adversely affecting the model's performance. You can choose to remove outliers or apply transformations to mitigate their impact.

7. Normalization: Normalize the data to a common scale to ensure that features with different ranges do not dominate the model's training process. Normalization can help improve the model's convergence and performance.

8. Handling Categorical Data: If your dataset includes categorical data, you may need to encode it using techniques like one-hot encoding or label encoding to convert it into a format that can be processed by the machine learning algorithm.

By following these preprocessing steps, you can ensure improving the model's accuracy and generalization to new data.

The pre-process approach used in our file is:

The 'date' column is converted to datetime format using the pd.to_datetime() method to ensure it is treated as a time series index.

Rows with any missing values are dropped from the dataset using the dropna() method.

These methods are applied as they are the most common issues that happen/occur in any dataset.

PERFORMING DIFFERENT ANALYSIS:

When predicting stock prices, several types of analyses can be helpful in understanding and forecasting market trends. Here are some key analyses commonly used in stock price prediction:

1. Time Series Analysis: Analyze historical stock price data to identify patterns, trends, and seasonality in the data. Time series analysis techniques like Autoregressive Integrated Moving Average (ARIMA) and Seasonal Decomposition of Time Series (STL) can help in understanding the time-dependent behavior of stock prices.

2. Technical Analysis: Utilize technical indicators such as moving averages, Relative Strength Index (RSI), and Bollinger Bands to identify trends, momentum, and potential reversal points in the stock price data.

3. Fundamental Analysis: Evaluate a company's financial statements, earnings, dividends, and other key indicators to assess the intrinsic value of the stock and make predictions based on the company's financial health and market position.

4. Sentiment Analysis: Analyze news, social media, and other textual data to gauge market sentiment and investor opinions. This analysis can provide insights into how public perception and news sentiment can impact stock prices.

5. Correlation Analysis: Explore the correlations between different stocks, market indices, or macroeconomic indicators to identify dependencies and understand how external factors influence stock prices.

6. Volatility Analysis: Assess the volatility of stock prices using measures such as standard deviation or the Average True Range (ATR) to understand the risk associated with the stock and potential price fluctuations.

7. Machine Learning Models: Utilize various machine learning algorithms such as linear regression, decision trees, random forests, and neural networks to build predictive models based on historical stock price data and relevant features.

8. Econometric Analysis: Incorporate macroeconomic variables like interest rates, inflation, and GDP growth to understand how broader economic conditions can influence stock prices.

By combining these different analyses, investors and analysts can develop a more comprehensive understanding of the factors driving stock prices and make more informed predictions about future price movements. It's crucial to note that no single analysis method can provide a complete picture, and a combination of these techniques can lead to more robust and accurate predictions.

1. Time Series Analysis:

- Visualization of historical stock prices: The code includes a line plot that visualizes the historical closing prices of the stock over time.

- Calculation of moving averages: The code calculates and visualizes the 20-day and 50-day moving averages of the stock prices.

2. Feature Scaling:

- The `MinMaxScaler` from the `sklearn.preprocessing` module is used to scale the features ('open', 'low', 'high', 'volume') to a standard range.

3. Model Training and Evaluation:

- A linear regression model is trained and evaluated using mean squared error (MSE) and mean absolute error (MAE) to assess the model's performance in predicting stock prices.

These analyses are performed for the following reasons:

1. Time Series Analysis: Understanding historical price trends and patterns is crucial in predicting future stock prices. Visualizing historical stock prices and calculating moving averages helps in identifying long-term trends, patterns, and potential points of support and resistance in the stock price data.

2. Feature Scaling: Scaling the features ensures that they are on the same scale, preventing certain features from dominating the model simply because of their larger scale. This is particularly important in algorithms that use distance-based calculations, gradient descent, or regularization.

3. Model Training and Evaluation: Training the linear regression model allows us to learn the relationships between the input features and the target variable. Evaluating the model's performance using mean squared error (MSE) and mean absolute error (MAE) helps to assess the model's accuracy and understand how well it performs in predicting stock prices.

By performing these analyses, we gain valuable insights into the historical behavior of the stock prices, ensure that the features are appropriately scaled for the model, and assess the model's predictive capability in forecasting future stock prices.

CODE:

```
import pandas as pd
from sklearn.model_selection import train_test_split
```

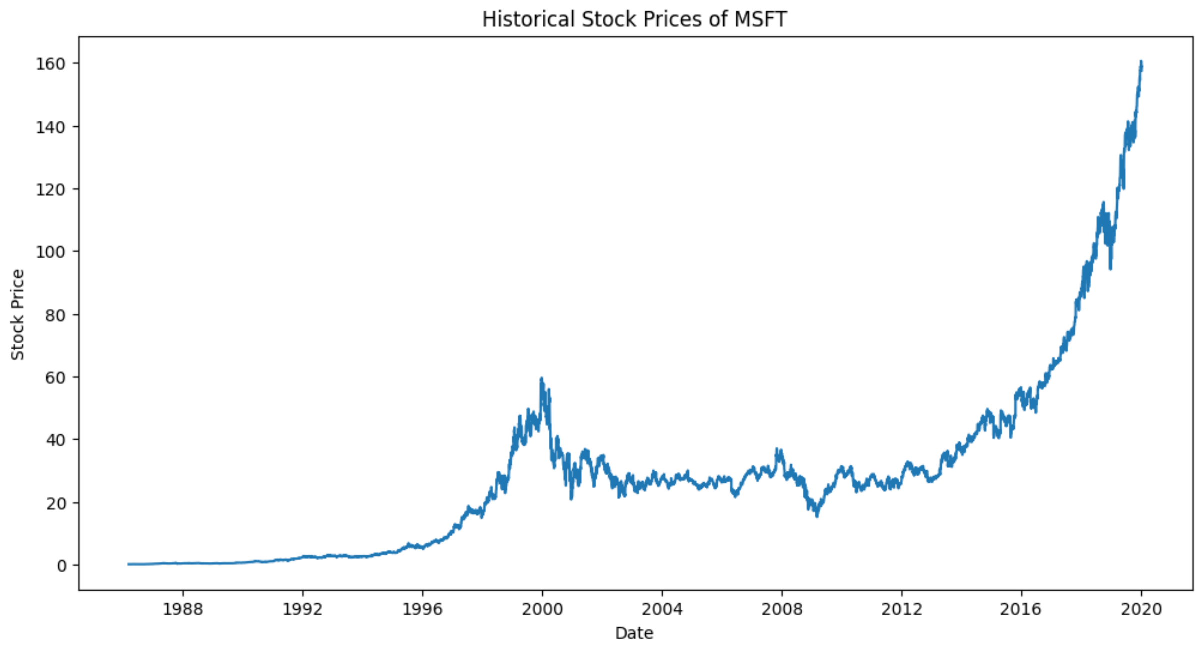
```python
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error
import matplotlib.pyplot as plt

# Load the dataset
df = pd.read_csv('MSFT.csv')  # Replace 'path_to_your_dataset.csv' with
the actual path to your dataset
print(df.describe())
# Preprocess the data
df['Date'] = pd.to_datetime(df['Date'])
df.set_index('Date', inplace=True)
df.dropna(inplace=True)

# Visualize the closing prices
plt.figure(figsize=(12, 6))
plt.plot(df.index, df['Close'])
plt.title('Historical Stock Prices of MSFT')
plt.xlabel('Date')
plt.ylabel('Stock Price')
plt.show()

# Create a new column for the target variable (e.g., next day's closing
price)
df['target'] = df['Close'].shift(-1)

# Drop any remaining rows with missing values
df.dropna(inplace=True)

df['20MA'] = df['Close'].rolling(window=20).mean()
df['50MA'] = df['Close'].rolling(window=50).mean()

# Plotting the moving averages
plt.figure(figsize=(12,6))
plt.plot(df.index, df['Close'], label='Closing Price')
plt.plot(df.index, df['20MA'], label='20-day Moving Average')
plt.plot(df.index, df['50MA'], label='50-day Moving Average')
plt.legend()
plt.title('Moving Averages for Stock Prices')
plt.xlabel('Date')
plt.ylabel('Price')
plt.show()

# Split the data into features and target
X = df[['Open', 'Low', 'High', 'Volume']]
y = df['target']

# Feature scaling
```

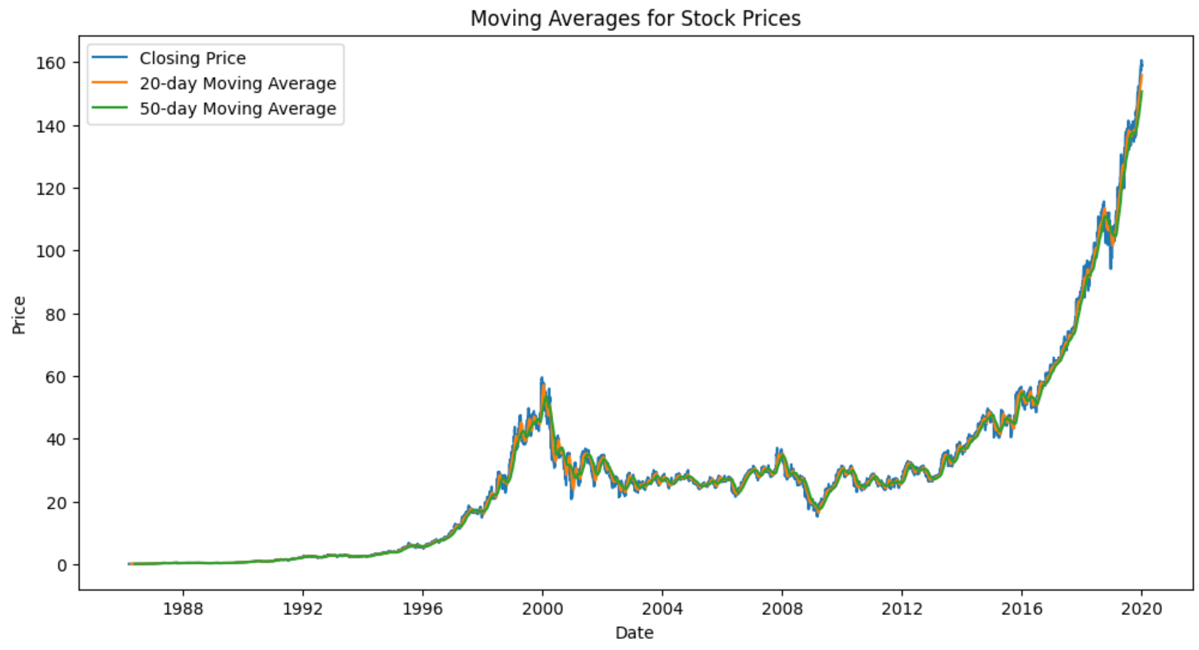
```python
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y,
test_size=0.2, random_state=0)

# Train a linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Model evaluation
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
print(f"Mean Absolute Error: {mae}")

# Visualize the predicted values
plt.figure(figsize=(12, 6))
plt.scatter(y_test.index, y_test.values, label='Actual',color='red')
plt.scatter(y_test.index, y_pred,
label='Predicted',alpha=0.5,color='black')
plt.title('Actual vs Predicted Stock Prices')
plt.xlabel('Date')
plt.ylabel('Stock Price')
plt.legend()
plt.show()
```

OUTPUT:

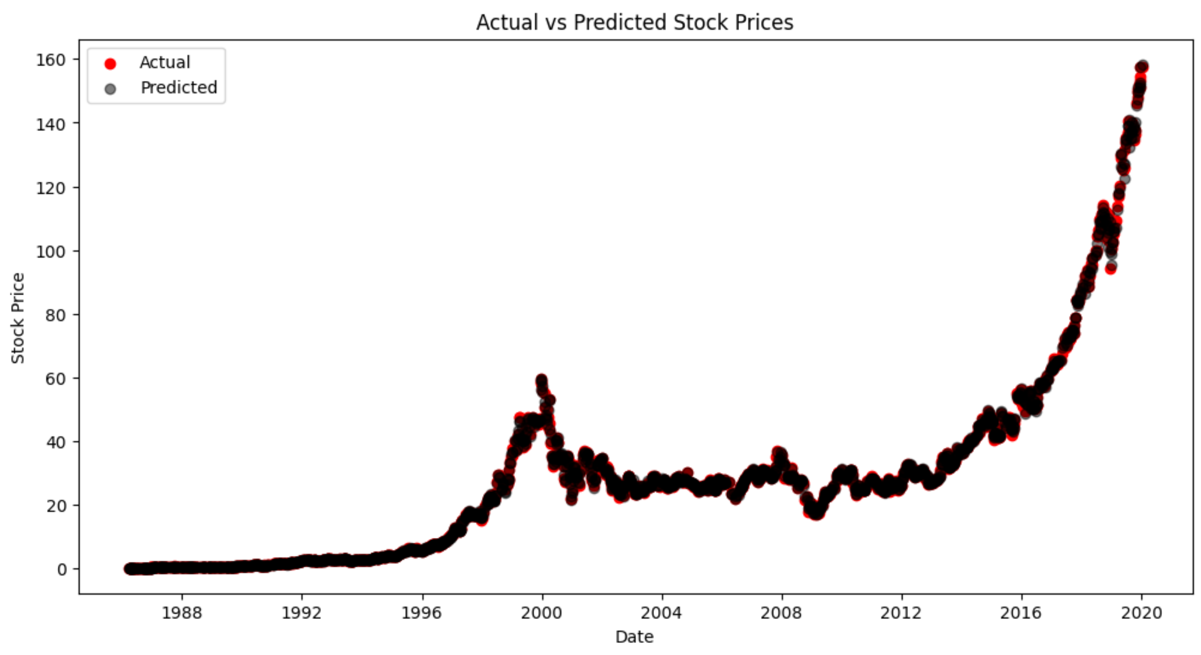
```
              Open         High          Low        Close    Adj Close  \
count  8525.000000  8525.000000  8525.000000  8525.000000  8525.000000
mean     28.220247    28.514473    27.918967    28.224480    23.417934
std      28.626752    28.848988    28.370344    28.626571    28.195330
min       0.088542     0.092014     0.088542     0.090278     0.058081
25%       3.414063     3.460938     3.382813     3.414063     2.196463
50%      26.174999    26.500000    25.889999    26.160000    18.441576
75%      34.230000    34.669998    33.750000    34.230000    25.392508
max     159.449997   160.729996   158.330002   160.619995   160.619995

             Volume
count  8.525000e+03
mean   6.045692e+07
std    3.891225e+07
min    2.304000e+06
25%    3.667960e+07
50%    5.370240e+07
```

```
75%     7.412350e+07
max     1.031789e+09
```





```
Mean Squared Error: 0.45474141895263165
Mean Absolute Error: 0.35983741296866834
```


Actual vs Predicted Stock Prices
Actual
Predicted
Stock Price
Date
0
20
40
60
80
100
120
140
160
1988
1992
1996
2000
2004
2008
2012
2016
2020