# Experiment 3 : Linear regression model for forecasting time series data

## 1. Importing Necessary Libraries

**Explanation:**
This section imports all the necessary libraries required for data manipulation, machine learning, and visualization.
**Corresponding Code:**

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error,
mean_squared_error, r2_score
```

---

## 2. Loading the Dataset

**Explanation:**
Here, the dataset is loaded from a CSV file into a pandas DataFrame. The dataset file (Microsoft_Stock.csv) needs to be replaced with the actual file path.

**Corresponding Code:**

```python
df = pd.read_csv("Microsoft_Stock.csv") # Replace with actual
file path
```

---

## 3. Data Preprocessing

**Explanation:**

- The 'Date' column is converted to datetime format to facilitate time-based operations.
- The dataset is then sorted by the 'Date' column to ensure the data is ordered chronologically.
- A new feature Prev_Close is created, which contains the previous day's close price. This helps in predicting the next day's stock price.

**Corresponding Code:**

```python
df['Date'] = pd.to_datetime(df['Date'])
df = df.sort_values(by='Date')
df['Prev_Close'] = df['Close'].shift(1) # Adding previous day's close as a feature
df.dropna(inplace=True) # Drop rows with missing values
```

---

## 4. Feature Selection and Target Variable

**Explanation:**

- The features used for prediction are Open, High, Low, Volume, and `Prev_Close`.
- The target variable is Close, which is the stock price we want to predict.

**Corresponding Code:**

```python
features = ['Open', 'High', 'Low', 'Volume', 'Prev_Close']
target = 'Close'
```

---

## 5. Splitting the Data into Training and Testing Sets

**Explanation:**
The dataset is split into training and testing sets. 80% of the data is used for training, and 20% for testing. The shuffle=False ensures that the data is not shuffled, maintaining the chronological order.

**Corresponding Code:**

```python
X = df[features]
y = df[target]
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, shuffle=False)
```

---

## 6. Training the Linear Regression Model

**Explanation:**
A Linear Regression model is instantiated and trained on the training data (X_train and y_train).
**Corresponding Code:**

```python
model = LinearRegression()
model.fit(X_train, y_train)
```

---

## 7. Making Predictions

**Explanation:**
The trained model is used to predict stock prices on the test set.
**Corresponding Code:**

```python
y_pred = model.predict(X_test)
```

---

## 8. Model Evaluation

**Explanation:**

The performance of the model is evaluated using various metrics:

- **Mean Absolute Error (MAE)**: Measures the average magnitude of errors.

- **Mean Squared Error (MSE)** : Measures the average of the squares of the errors.
- **Root Mean Squared Error (RMSE)** : The square root of the MSE, providing error in the same unit as the target variable.
- **$R^2$ Score**: Measures the proportion of variance explained by the model.

**Corresponding Code:**

```python
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

print(f"MAE:       {mae}")
print(f"MSE:       {mse}")
print(f"RMSE:     {rmse}")
print(f"R2 Score: {r2}")
```

---

## 9. Visualization of Actual vs Predicted Prices

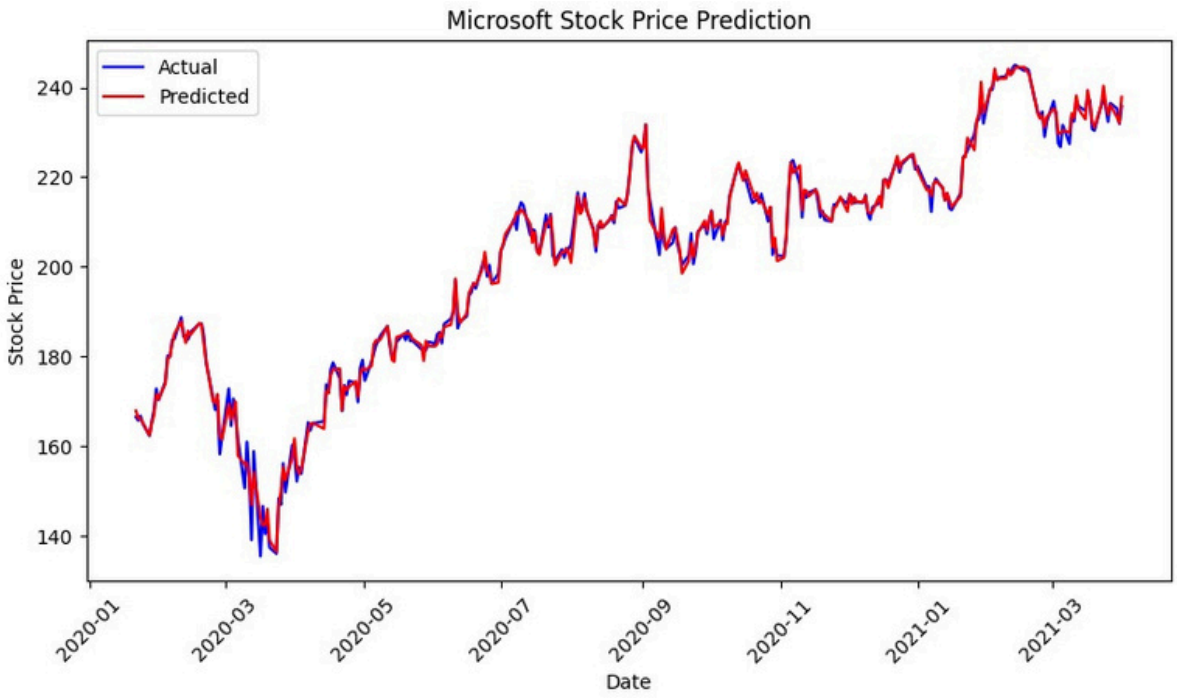
**Explanation:**

This section visualizes the predicted stock prices against the actual stock prices over time. The x-axis represents the dates, and the y-axis represents the stock price. The blue line indicates actual prices, and the red line shows the predicted prices.
**Corresponding Code:**

```python
plt.figure(figsize=(10,5))
plt.plot(df['Date'].iloc[-len(y_test):], y_test, label='Actual',
color='blue')
plt.plot(df['Date'].iloc[-len(y_test):], y_pred,
label='Predicted', color='red')
plt.xlabel('Date')
plt.ylabel('Stock Price')
plt.legend()
plt.title('Microsoft Stock Price Prediction')
plt.xticks(rotation=45)
```

```
plt.show()
```

**Output**




**Result:**

Thus , the linear regression model for forecasting time series data has been studied and executed successfully.