STOCK PRICE PREDICTOR

1. IMPORTING LIBRARIES

```
!pip install mplfinance
```

```
Collecting mplfinance
  Downloading mplfinance-0.12.10b0-py3-none-any.whl.metadata (19 kB)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packages (from mplfinance) (3.10.0)
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages (from mplfinance) (2.2.2)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib->mplfinance) (1.3.2)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib->mplfinance) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib->mplfinance) (4.58.1)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib->mplfinance) (1.4.8)
Requirement already satisfied: numpy>=1.23 in /usr/local/lib/python3.11/dist-packages (from matplotlib->mplfinance) (2.0.2)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib->mplfinance) (24.2)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-packages (from matplotlib->mplfinance) (11.2.1)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib->mplfinance) (3.2.3)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.11/dist-packages (from matplotlib->mplfinance) (2.9.0
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas->mplfinance) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas->mplfinance) (2025.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.7->matplotlib->mplfinanc
Downloading mplfinance-0.12.10b0-py3-none-any.whl (75 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 75.0/75.0 kB 2.3 MB/s eta 0:00:00
Installing collected packages: mplfinance
Successfully installed mplfinance-0.12.10b0
```

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score
import mplfinance as mpf
import xgboost as xgb
```

DATA READING AND PREPROCESSING

```
df = pd.read_csv('/content/Apple Stock Price History.csv')
```

```
# df.index = pd.to_datetime(df.Time,format='%d/%m/%Y %H:%M')
# df = df['2023-01-01':]
df = df.set_index('Date')
df.index = pd.to_datetime(df.index)
df = df.rename(columns={'Price': 'Close'})
df.head()
```

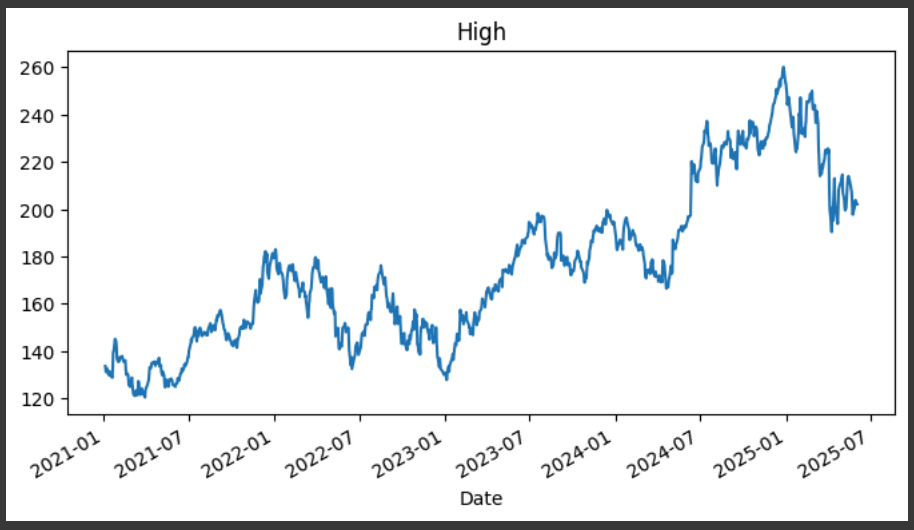
Date	Close	Open	High	Low
2025-06-02	200.53	200.28	202.10	200.13
2025-05-30	200.85	199.37	201.96	196.78
2025-05-29	199.95	203.57	203.81	198.51
2025-05-28	200.42	200.59	202.73	199.90
2025-05-27	200.21	198.30	200.74	197.43

```
# df.isnull().sum()
# df.describe()
# df.info()
# df.columns
df.shape
# df.duplicated().sum()
```

```
(1108, 4)
```

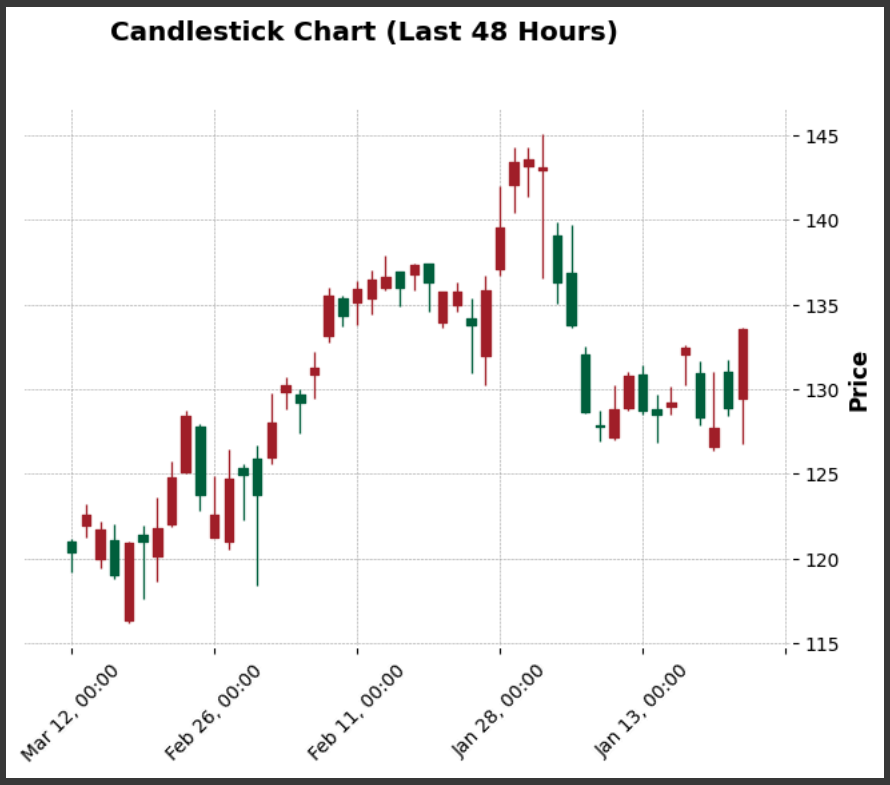
DATA VISUALIZATION

```
df['High'].plot(kind='line', figsize=(8, 4), title='High')
plt.show()
```

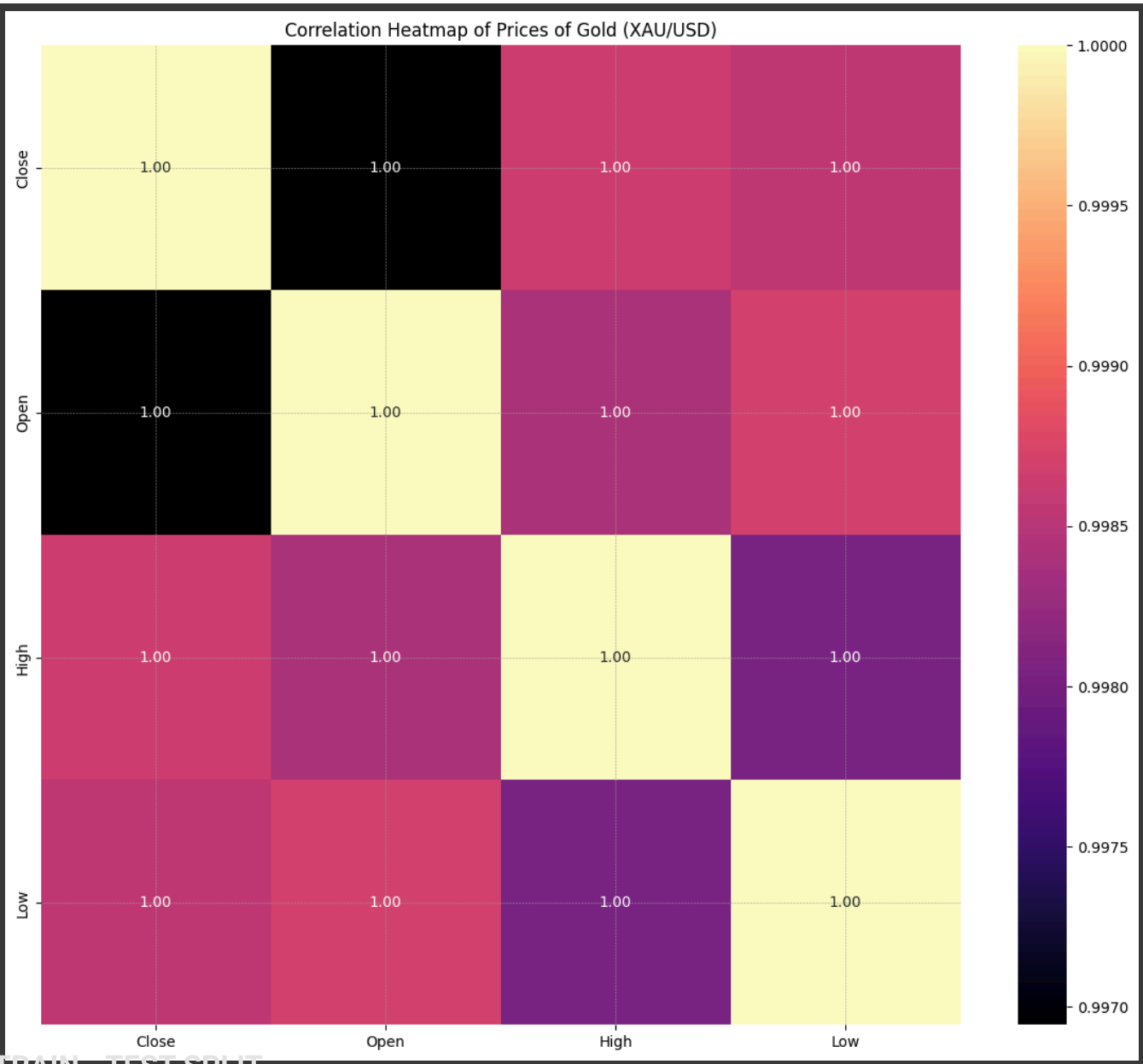



```
ohlc_df = df[['Open', 'High', 'Low', 'Close']].copy()
ohlc_df.index.name = 'Date'

mpf.plot(ohlc_df[-48:], type='candle', style='charles', title='Candlestick Chart (Last 48 Hours)', ylabel='Price')
```




```
plt.figure(figsize=(14, 12))
sns.heatmap(df.corr(numeric_only=True), annot=True, fmt=".2f", cmap='magma')
plt.title('Correlation Heatmap of Prices of Gold (XAU/USD)')
plt.show()
```

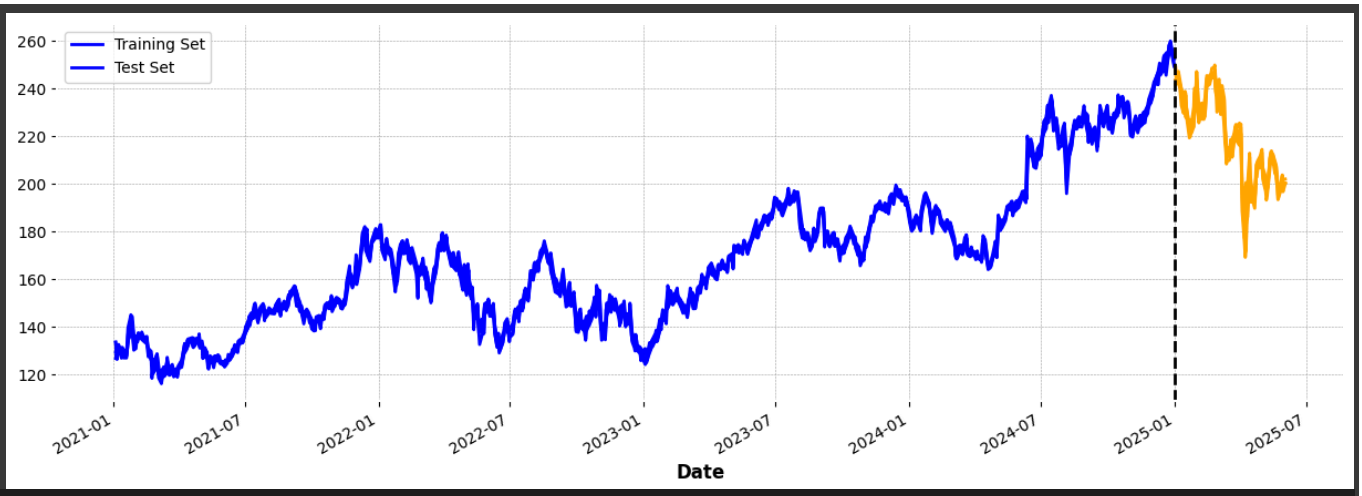


TRAIN - TEST SPLIT

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

```
train = df.loc[df.index < '2025-01-01']
test = df.loc[df.index >= '2025-01-01']
X_train = train[features]
y_train = train[target]
X_test = test[features]
y_test = test[target]
```

```
fig , ax = plt.subplots(figsize=(15,5))
train.plot(ax=ax, label = 'Training Set',color='blue')
test.plot(ax=ax, label = 'Test Set',color='orange')
ax.axvline('2025-01-01',color = 'black',ls = '--')
ax.legend(['Training Set','Test Set'])
plt.show()
```




```
# X_train
# y_train
# X_test
# y_test
X_train.shape, y_train.shape, X_test.shape, y_test.shape
```

```
((1005, 3), (1005, 1), (103, 3), (103, 1))
```

MODEL TRAINING (Xtreme Gradient Boost Regressor - XGBoost Regressor)

```
xgbr = xgb.XGBRegressor(learning_rate = 0.1,max_depth = 1, n_estimators = 150)
```

```
xgbr.fit(X_train,y_train)
xgbr.score(X_train,y_train)
```

```
0.9984488487243652
```

```
predictions = xgbr.predict(X_test)
```

```
model_accuracy = xgbr.score(X_test,y_test)*100
```

EVALUATION METRICS

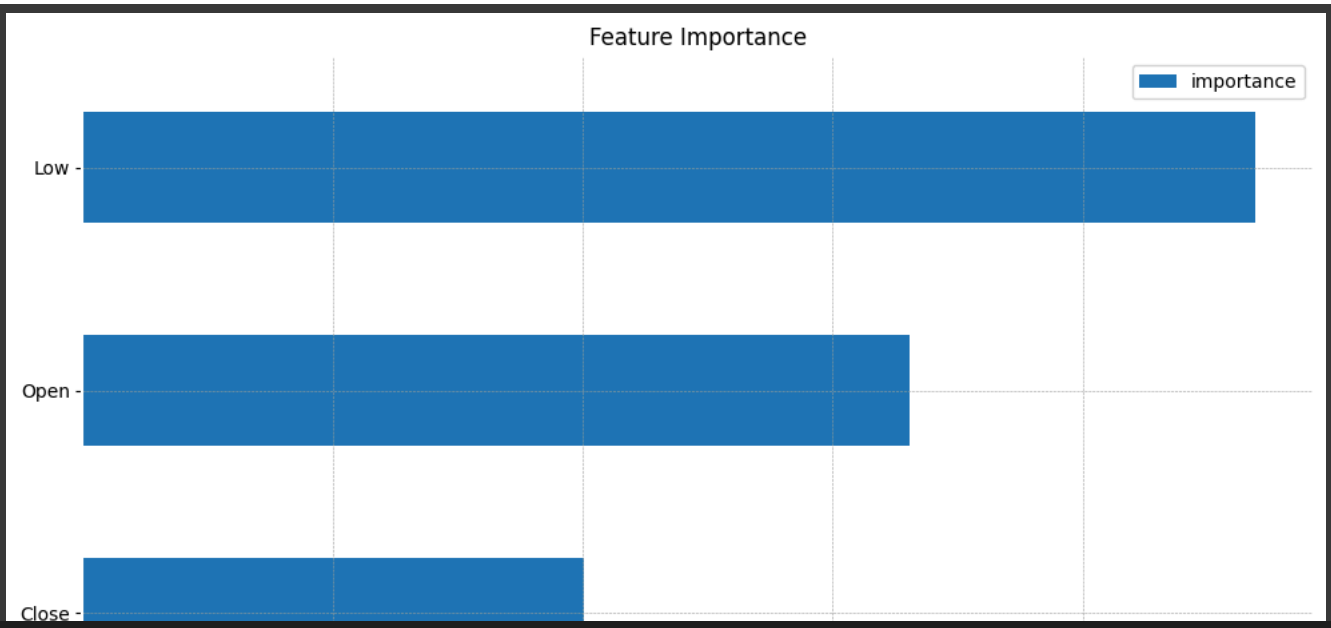
```
mae = mean_absolute_error(y_test, predictions)
mse = mean_squared_error(y_test, predictions)
rmse = np.sqrt(mean_squared_error(y_test, predictions))
r2 = r2_score(y_test, predictions)

print("Mean Absolute Error:",round(mae,2))
print("Mean Squared Error:",round(mse,2))
print("Root Mean Squared Error:",round(rmse,2))
print("R2 Score",round(r2,4))
print('Model Accuracy on test data:',round(model_accuracy,2))
```

```
Mean Absolute Error: 2.55
Mean Squared Error: 14.0
Root Mean Squared Error: 3.74
R2 Score 0.9511
Model Accuracy on test data: 95.11
```

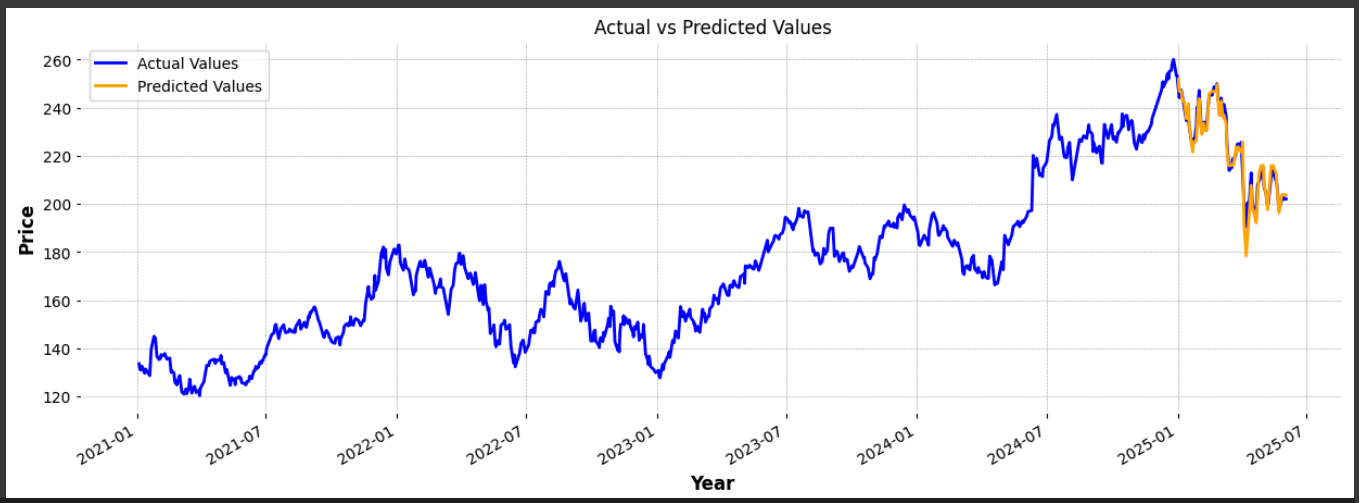
```
fi = pd.DataFrame(data=xgbr.feature_importances_,
                  index=X_train.columns,
                  columns=['importance'])

fi.sort_values('importance').plot(kind='barh', title='Feature Importance', figsize=(10, 6))
plt.xlabel("Feature Importance")
plt.tight_layout()
plt.show()
```




```
extended_df = df
preds = xgbr.predict(X_test)
pred_df = pd.DataFrame(preds, index=X_test.index, columns=['Predictions'])
extended_df = extended_df.merge(pred_df, how='left', left_index=True, right_index=True)
```

```
ax = extended_df['High'].plot(figsize=(15, 5), color='blue')
extended_df['Predictions'].plot(ax=ax, color='orange')
plt.legend(['Actual Values', 'Predicted Values'])
plt.xlabel('Year')
plt.ylabel('Price')
ax.set_title("Actual vs Predicted Values")
plt.show()
```




```
#Viewing results according to latest dates
extended_df[extended_df.index > '2025-05-23']
```

Date	Close	Open	High	Low	Predictions
2025-06-02	200.53	200.28	202.10	200.13	203.727661
2025-05-30	200.85	199.37	201.96	196.78	203.727661
2025-05-29	199.95	203.57	203.81	198.51	203.727661
2025-05-28	200.42	200.59	202.73	199.90	203.727661
2025-05-27	200.21	198.30	200.74	197.43	203.727661

```
import pickle
```