Double-click (or enter) to edit

WEEK 3 -- TASK 3

Task 3: Model Building & Training

Goal: Train a Machine Learning model on the dataset.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score



df = pd.read_csv('stock_data_july_2025.csv')
print(df.head())
```

```
         Date Ticker  Open Price  Close Price  High Price  Low Price  \
0  2025-06-01   AAPL      185.96       187.79      188.85     183.74
1  2025-06-01   ABBV      189.63       193.64      194.31     188.03
2  2025-06-01    ABT      118.79       120.08      120.53     118.01
3  2025-06-01   ADBE      584.51       567.78      587.97     561.25
4  2025-06-01    ADP      253.46       255.60      258.59     249.91

   Volume Traded    Market Cap  PE Ratio  Dividend Yield    EPS  52 Week High  \
0       42257183  2.931356e+12     28.37            0.49   6.62        248.20
1       33074970  3.289227e+11     16.74            3.51  11.57        210.83
2       17130934  2.001333e+11     19.39            1.84   6.19        140.79
3       11080360  2.460380e+11     43.29            0.00  13.12        727.11
4       10319559  1.052471e+11     29.92            2.04   8.54        303.75

   52 Week Low      Sector
0       155.68  Technology
1       127.91  Healthcare
2        99.26  Healthcare
3       407.82  Technology
4       234.28  Financials
```

```
print(df.info())
print(df.describe())
print(df.isnull().sum())

# Drop rows with missing target (if any)
df = df.dropna(subset=['Close Price'])

# Optional: Drop rows with any NA values, just for demo
# df = df.dropna()
```

```
 3   Close Price     4346 non-null   float64
 4   High Price      4346 non-null   float64
 5   Low Price       4346 non-null   float64
 6   Volume Traded   4346 non-null   int64
 7   Market Cap      4346 non-null   float64
 8   PE Ratio        4346 non-null   float64
 9   Dividend Yield  4346 non-null   float64
 10  EPS             4346 non-null   float64
 11  52 Week High    4346 non-null   float64
 12  52 Week Low     4346 non-null   float64
 13  Sector          4346 non-null   object
dtypes: float64(10), int64(1), object(3)
memory usage: 475.5+ KB
None
        Open Price  Close Price   High Price    Low Price  Volume Traded  \
count  4346.000000  4346.000000  4346.000000  4346.000000   4.346000e+03
```

```
              Market Cap     PE Ratio  Dividend Yield          EPS  52 Week High  \
count  4.346000e+03  4346.000000     4346.000000  4346.000000   4346.000000
mean   4.027126e+11    25.581173        1.807835    13.693049    410.097975
std    6.202114e+11    12.923249        1.557671    20.268110    575.709969
min    6.469973e+10     6.850000        0.000000     1.560000     21.130000
25%    1.313524e+11    16.820000        0.470000     5.140000    130.547500
50%    1.738019e+11    22.900000        1.800000     8.470000    235.125000
75%    3.714750e+11    30.307500        2.767500    14.265000    531.735000
max    3.589151e+12    97.700000        7.760000   197.370000   5069.750000

       52 Week Low
count  4346.000000
mean    266.144823
std     377.470429
min      13.670000
25%      83.862500
50%     152.095000
75%     338.062500
max    3529.240000
Date              0
Ticker            0
Open Price        0
Close Price       0
High Price        0
Low Price         0
Volume Traded     0
Market Cap        0
PE Ratio          0
Dividend Yield    0
EPS               0
52 Week High      0
52 Week Low       0
Sector            0
dtype: int64
```

```
features = ['Open Price', 'High Price', 'Low Price', 'Volume Traded', 'Market Cap', 'PE Ratio', 'Dividend Yield', 'EPS']
X = df[features]
y = df['Close Price']
```

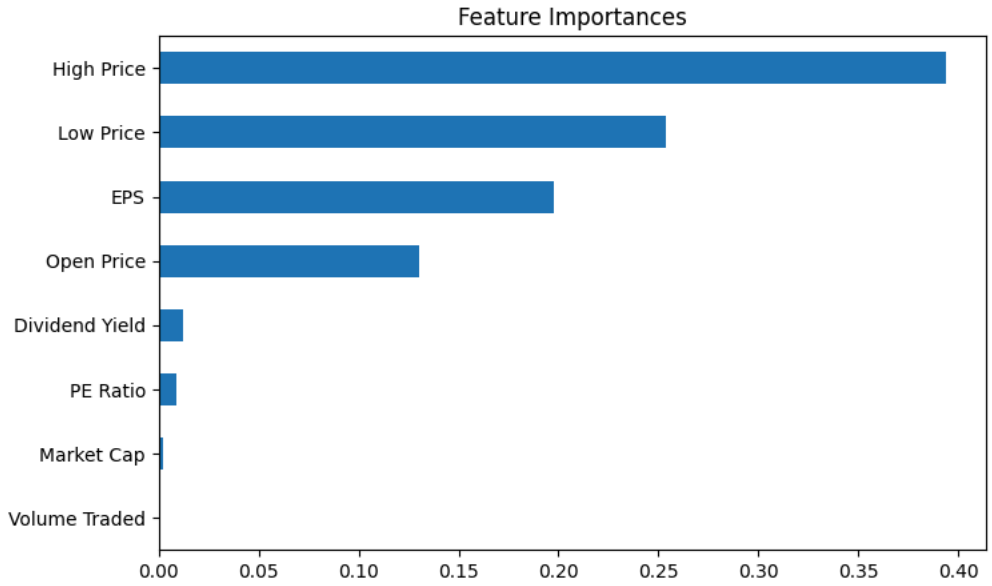
```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
```

```
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

```
print("Mean Squared Error:", mean_squared_error(y_test, y_pred))
print("R^2 Score:", r2_score(y_test, y_pred))
```

```
Mean Squared Error: 11.254113341345189
R^2 Score: 0.9999546643729622
```

```
importances = pd.Series(model.feature_importances_, index=features)
importances.sort_values().plot(kind='barh', figsize=(8,5))
plt.title("Feature Importances")
plt.show()
```




```
plt.figure(figsize=(8,5))
plt.scatter(y_test, y_pred, alpha=0.6)
plt.xlabel('Actual Close Price')
plt.ylabel('Predicted Close Price')
plt.title('Actual vs Predicted Close Price')
plt.show()
```