Week-7 Coding Assignment

Part-1 (50 pts)

For this part of the assignment, you will be using a Linear Regression model on a built-in financial dataset in Python within a Jupyter Notebook.

Scenario: Predicting the stock market index (S&P 500) based on historical financial indicators. You will use Yahoo Finance to fetch financial data, pandas for data handling, and scikit-learn for machine learning.

STEPS:

- fetch S&P 500 and 10-Year Treasury Yield data.
- define Treasury Yield as the independent variable (X) and S&P 500 Close as the dependent variable (y).
- train a Linear Regression Model to predict S&P 500 Close based on Treasury Yield.
- evaluate the model with Mean Absolute Error (MAE) and R² Score.

Reference Dataset: https://pypi.org/project/yfinance/#files

IMPORTANT:

- Firstly, place the file yfinance-0.2.54-py2.py3-none-any.whl in the same directory of your notebook file.
- Then, run the following command in a cell:

pip install yfinance-0.2.54-py2.py3-none-any.whl

In [ ]:

```
pip install yfinance-0.2.54-py2.py3-none-any.whl
```

(2 pts) Import necessary libraries below.

In [1]:

```
import yfinance as yf
```

(5 pts) Step 1: Fetch financial data from Yahoo Finance: sp500 and treasury

In [2]:

```
```

```
YF.download() has changed argument auto_adjust default to True
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
```

In [3]:

```
sp500.columns
```

```
Out[3]: MultiIndex([( 'Close', '^GSPC'),
                    (  'High', '^GSPC'),
                    (   'Low', '^GSPC'),
                    (  'Open', '^GSPC'),
                    ('Volume', '^GSPC')],
                   names=['Price', 'Ticker'])
```

```
In [4]: treasury.columns
```

```
Out[4]: MultiIndex([( 'Close', '^TNX'),
                    (  'High', '^TNX'),
                    (   'Low', '^TNX'),
                    (  'Open', '^TNX'),
                    ('Volume', '^TNX')],
                   names=['Price', 'Ticker'])
```

(5 pts) Step 2: Data Preprocessing using the method .dropna()

```
In [5]:
```

(3 pts) Step 3: Define features (X) and target (y)

```
In [6]:
```

(2 pts) Step 4: Split data into training (80%) and testing (20%) sets

```
In [7]:
```

(5 pts) Step 5: Train the Linear Regression Model

```
In [8]:
```

```
Out[8]: LinearRegression()
```

(3 pts) Step 6: Calculate the Predictions

```
In [9]:
```

(5 pts) Step 7: Evaluate the Model - Calculate the MAE and R-squared.

```
In [10]:
```

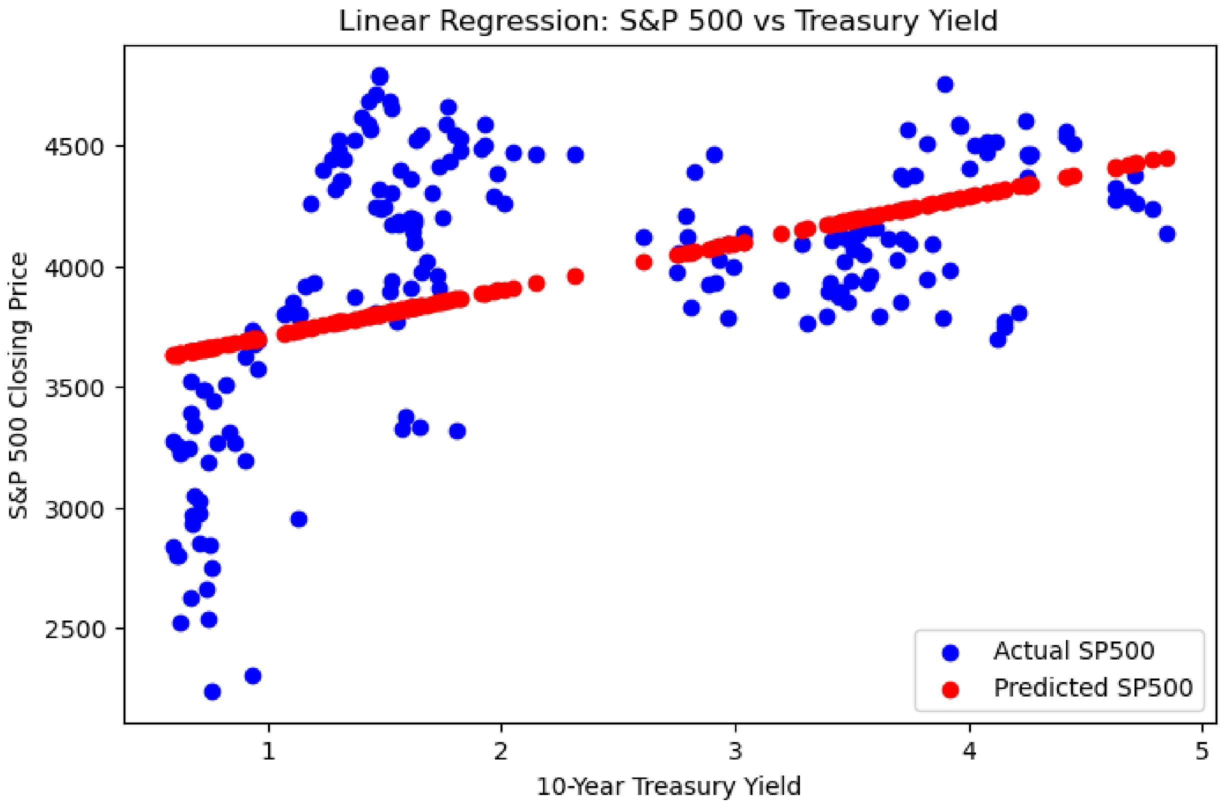
(5 pts) Step 8: Print Model Performance values - MAE, r2, slope and intercept.

```
In [11]:
```

```
Mean Absolute Error: 371.40
R-squared Score: 0.22
Model Coefficient: 192.96
Model Intercept: 3516.63
```

(5 pts) Step 9: Plot results

```
In [12]:
```



(10 pts) What is your interpretation based on the R-squared and MAE values?

Part.2 (50 pts)

Logistic Regression in Finance: Predicting Stock Market Movement

Scenario: You will use Logistic Regression to predict whether the S&P 500 index will go up or down based on financial indicators.

Target Variable: Market direction (1 = Up, 0 = Down)

Predictors: Treasury Yield, Moving Averages, and S&P 500 Volatility

In [13]:
```
# (2 pts) Import necessary libraries
```

In [14]:
```
# (5 pts) Step 1: Download financial data (S&P 500 and Treasury Yield)
```

```
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
```

In [15]:
```
sp500.columns
```

Out[15]:
```
MultiIndex([( 'Close', '^GSPC'),
            (  'High', '^GSPC'),
            (   'Low', '^GSPC'),
            (  'Open', '^GSPC'),
            ('Volume', '^GSPC')],
           names=['Price', 'Ticker'])
```

In [16]:

```
treasury.columns
```

Out[16]:

```
MultiIndex([( 'Close', '^TNX'),
            (  'High', '^TNX'),
            (   'Low', '^TNX'),
            (  'Open', '^TNX'),
            ('Volume', '^TNX')],
           names=['Price', 'Ticker'])
```

In [17]:

```
# (5 pts) Step 2: Feature Engineering
# Find out the following for SP500:
# Daily_Return
# Market_Direction: 1 is up, 0 is down.
# 50_MA: 50-day moving average
# Volatility: 20-day rolling volatility
```

In [18]:

```
# (2 pts) Drop NaN values created by rolling calculations
```

In [19]:

```
# (2 pts) Merge Treasury Yield data
```

In [20]:

```
# (3 pts) Define Features (X) and Target (y)
```

In [21]:

```
# (3 pts) Step 3: Train-Test Split (80%-20%)
```

In [22]:

```
# Step 4: Standardize Features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
/Users/mesutozdag/opt/anaconda3/lib/python3.9/site-packages/sklearn/utils/vali
dation.py:1688: FutureWarning: Feature names only support names that are all s
trings. Got feature names with dtypes: ['tuple']. An error will be raised in
1.2.
  warnings.warn(
/Users/mesutozdag/opt/anaconda3/lib/python3.9/site-packages/sklearn/utils/vali
dation.py:1688: FutureWarning: Feature names only support names that are all s
trings. Got feature names with dtypes: ['tuple']. An error will be raised in
1.2.
  warnings.warn(
/Users/mesutozdag/opt/anaconda3/lib/python3.9/site-packages/sklearn/utils/vali
dation.py:1688: FutureWarning: Feature names only support names that are all s
trings. Got feature names with dtypes: ['tuple']. An error will be raised in
1.2.
  warnings.warn(
```

In [23]:

```
# (3 pts) Step 5: Train the Logistic Regression Model
```

Out[23]:

```
LogisticRegression()
```

In [24]:

```
# (5 pts) Step 6: Calculate the Predictions
```

In [25]:

```
# (5 pts) Step 7: Evaluate Model Performance - accuracy, confusion matrix, cla
```

In [26]:

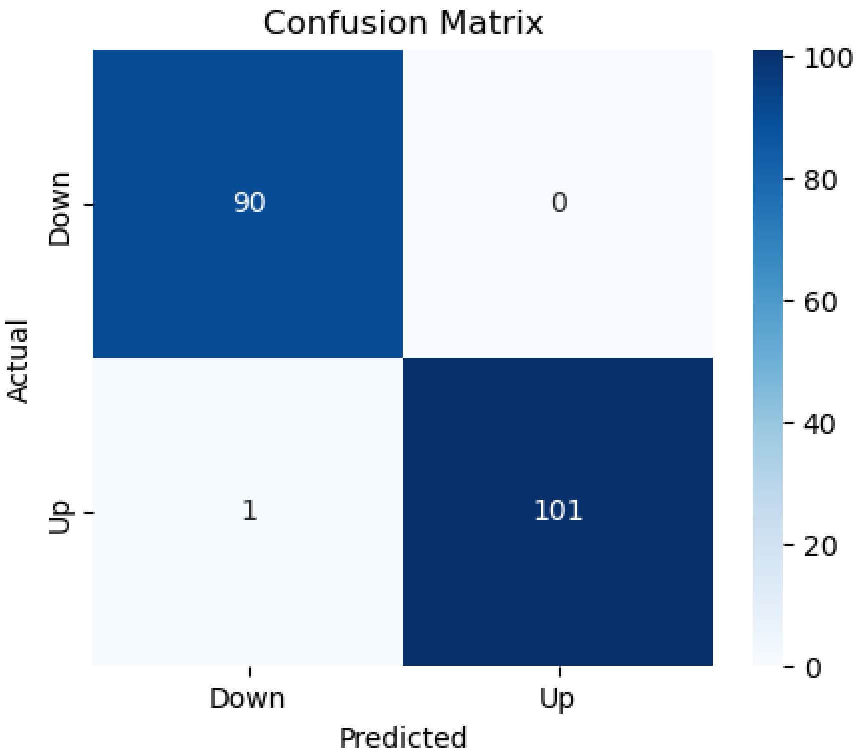
```
# Print Results
```

```
Model Accuracy: 0.99
Confusion Matrix:
 [[ 90   0]
 [  1 101]]
Classification Report:
               precision    recall  f1-score   support

           0       0.99      1.00      0.99        90
           1       1.00      0.99      1.00       102

    accuracy                           0.99       192
   macro avg       0.99      1.00      0.99       192
weighted avg       0.99      0.99      0.99       192
```

```
In [27]: # (10 pts) Step 8: Plot Confusion Matrix
```



(5 pts) What is your interpretation based on the Confusion Matrix?

```
In [ ]:
```