INTERNSHIP TASKS

INTERNSHIP DOMAIN: Machine Learning

Task 1: Student Score Prediction

The following task I run on google collab:

Code:-

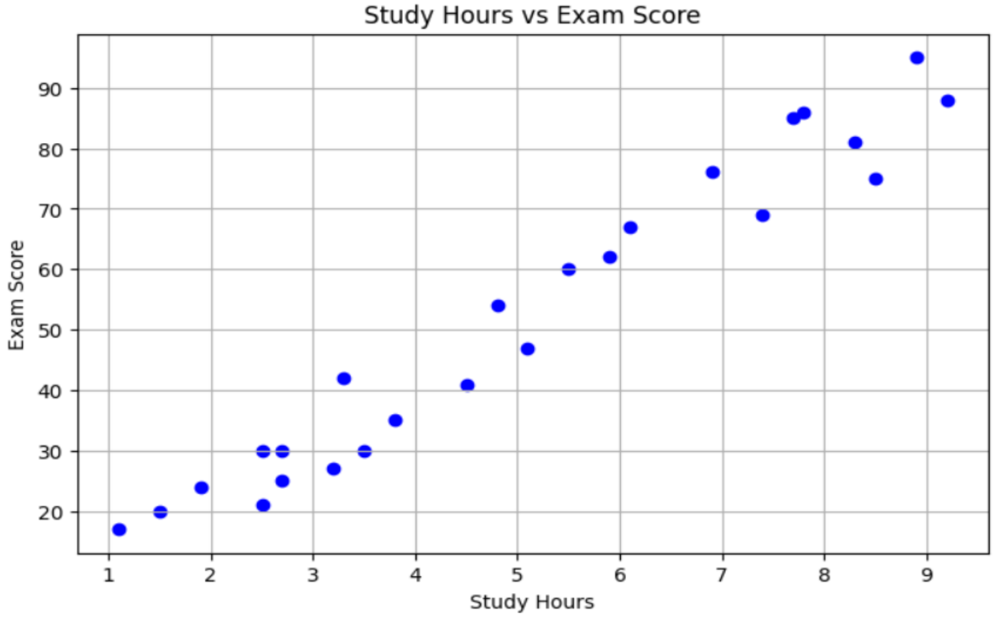
```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, mean_squared_error,
r2_score

# Load dataset (you can upload your own or use this example dataset)
url = "http://bit.ly/w-data"
data = pd.read_csv(url)

# Show first few rows
print("First 5 rows of the dataset:")
print(data.head())

# Basic info
print("\nDataset Info:")
print(data.info())

# Rename columns if necessary (optional)
data.columns = ['StudyHours', 'Scores']

# Basic visualization
plt.figure(figsize=(8, 5))
plt.scatter(data['StudyHours'], data['Scores'], color='blue')
plt.title('Study Hours vs Exam Score')
plt.xlabel('Study Hours')
plt.ylabel('Exam Score')
plt.grid(True)
plt.show()

# Split data into input (X) and output (y)
```

```python
X = data[['StudyHours']]
y = data['Scores']

# Split into training and testing sets (80/20 split)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Train Linear Regression Model
model = LinearRegression()
model.fit(X_train, y_train)

# Predict on test set
y_pred = model.predict(X_test)

# Evaluation
print("\nModel Evaluation Metrics:")
print("Mean Absolute Error (MAE):", mean_absolute_error(y_test, y_pred))
print("Mean Squared Error (MSE):", mean_squared_error(y_test, y_pred))
print("Root Mean Squared Error (RMSE):", np.sqrt(mean_squared_error(y_test,
y_pred)))
print("R2 Score:", r2_score(y_test, y_pred))

# Visualization of predictions
plt.figure(figsize=(8, 5))
plt.scatter(X_test, y_test, color='green', label='Actual')
plt.plot(X_test, y_pred, color='red', linewidth=2, label='Predicted')
plt.title('Actual vs Predicted Exam Scores')
plt.xlabel('Study Hours')
plt.ylabel('Scores')
plt.legend()
plt.grid(True)
plt.show()
```

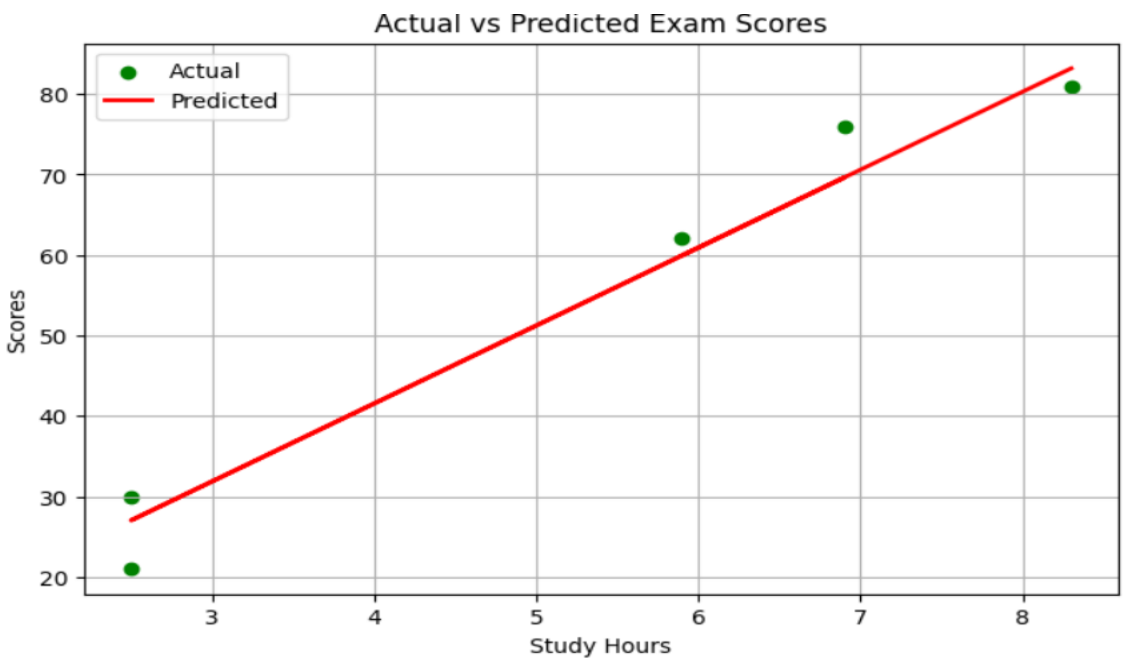
Output:-

```
First 5 rows of the dataset:
   Hours  Scores
0    2.5      21
1    5.1      47
2    3.2      27
3    8.5      75
4    3.5      30

Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25 entries, 0 to 24
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
```

```
---  ------  --------------  -----
 0   Hours   25 non-null     float64
 1   Scores  25 non-null     int64
dtypes: float64(1), int64(1)
memory usage: 532.0 bytes
None
```




```
Model Evaluation Metrics:
Mean Absolute Error (MAE): 3.9207511902099244
Mean Squared Error (MSE): 18.943211722315272
Root Mean Squared Error (RMSE): 4.352380006653288
R2 Score: 0.9678055545167994
```



Task 2: Customer Segmentation

The below code I run on google collab and following output observe:

Code:-

```
# Step 1: Upload the file
from google.colab import files
uploaded = files.upload()

# Step 2: Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

# Step 3: Load dataset
filename = list(uploaded.keys())[0]  # Get uploaded filename
df = pd.read_csv(filename)
print("First 5 rows of data:")
display(df.head())

# Step 4: Basic Info
print("\nDataset Info:")
print(df.info())
print("\nSummary Statistics:")
print(df.describe())

# Step 5: Select relevant features (Annual Income and Spending Score)
X = df[['Annual Income (k$)', 'Spending Score (1-100)']]

# Step 6: Feature scaling
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Step 7: Determine optimal number of clusters using Elbow Method
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, random_state=42)
    kmeans.fit(X_scaled)
    wcss.append(kmeans.inertia_)

plt.figure(figsize=(8, 5))
```

```python
plt.plot(range(1, 11), wcss, marker='o')
plt.title('Elbow Method for Optimal Clusters')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.grid(True)
plt.show()

# Step 8: Apply K-Means (choose optimal k from Elbow method, e.g., k=5)
kmeans = KMeans(n_clusters=5, random_state=42)
df['Cluster'] = kmeans.fit_predict(X_scaled)

# Step 9: Visualize clusters
plt.figure(figsize=(8, 5))
sns.scatterplot(data=df, x='Annual Income (k$)', y='Spending Score (1-
100)', hue='Cluster', palette='Set1')
plt.title('Customer Segmentation')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend(title='Cluster')
plt.grid(True)
plt.show()
```

Output:-

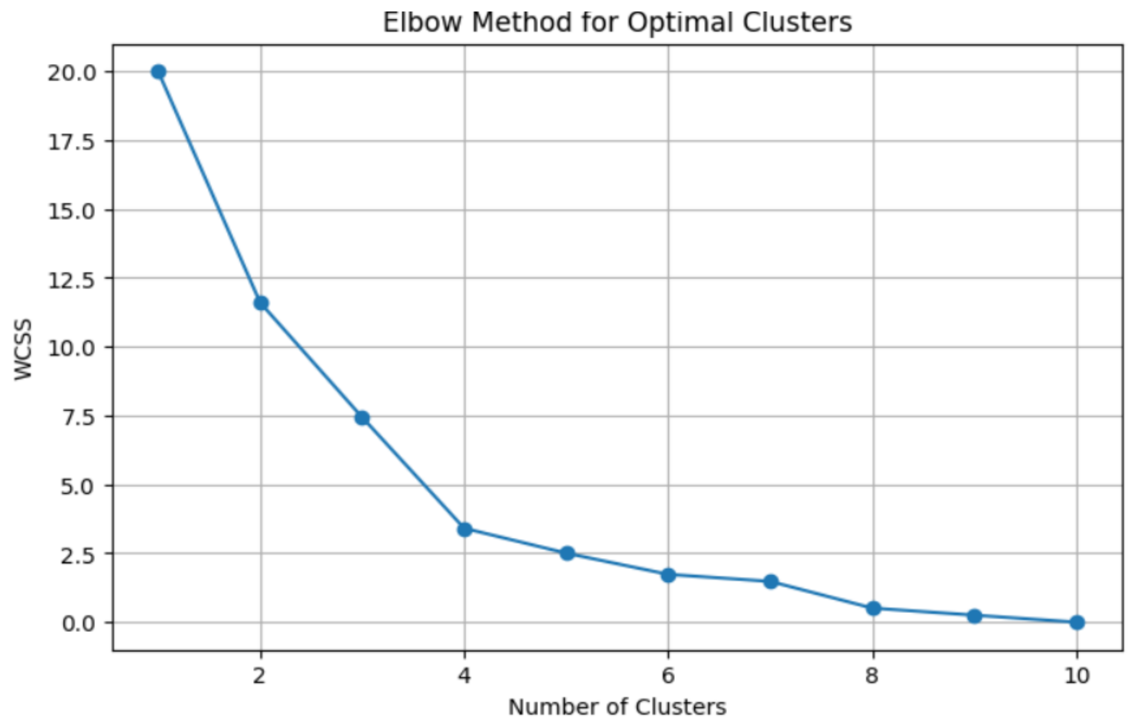
- **Mall_Customers (1).csv**(text/csv) - 232 bytes, last modified: 8/2/2025 - 100% done

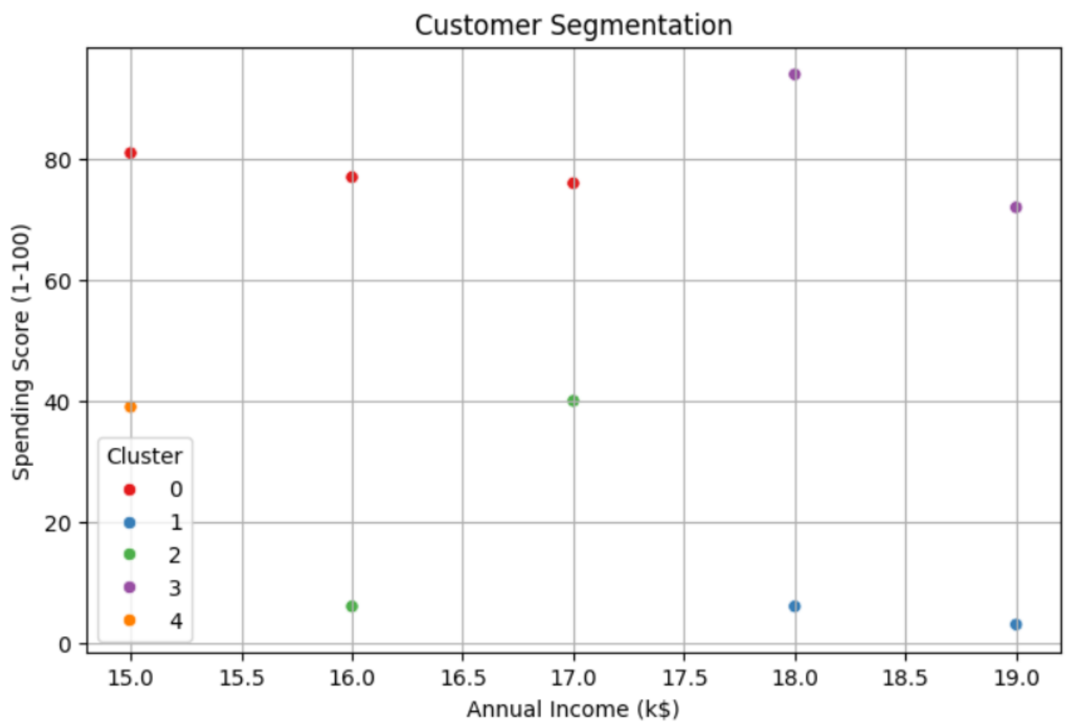
```
Saving Mall_Customers (1).csv to Mall_Customers (1) (2).csv
First 5 rows of data:
```

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

```
Exception ignored on calling ctypes callback function: <function
ThreadpoolController._find_libraries_with_dl_iterate_phdr.<locals>.match_librar
y_callback at 0x7f247bea2200>
Traceback (most recent call last):
```

```
  File "/usr/local/lib/python3.11/dist-packages/threadpoolctl.py", line 1005,
in match_library_callback
    self._make_controller_from_path(filepath)
  File "/usr/local/lib/python3.11/dist-packages/threadpoolctl.py", line 1187,
in _make_controller_from_path
    lib_controller = controller_class(
                     ^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/threadpoolctl.py", line 114, in
__init__
    self.dynlib = ctypes.CDLL(filepath, mode=_RTLD_NOLOAD)
                  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/lib/python3.11/ctypes/__init__.py", line 376, in __init__
    self._handle = _dlopen(self._name, mode)
                   ^^^^^^^^^^^^^^^^^^^^^^^^^
OSError: /usr/local/lib/python3.11/dist-
packages/numpy.libs/libscipy_openblas64_-99b71e71.so: cannot open shared object
file: No such file or directory

Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   CustomerID              10 non-null     int64
 1   Gender                  10 non-null     object
 2   Age                     10 non-null     int64
 3   Annual Income (k$)      10 non-null     int64
 4   Spending Score (1-100)  10 non-null     int64
dtypes: int64(4), object(1)
memory usage: 532.0+ bytes
None

Summary Statistics:
       CustomerID        Age  Annual Income (k$)  Spending Score (1-100)
count    10.00000  10.000000           10.000000               10.000000
mean      5.50000  28.800000           17.000000               49.400000
std       3.02765  13.464356            1.490712               35.094159
min       1.00000  19.000000           15.000000                3.000000
25%       3.25000  21.250000           16.000000               14.250000
50%       5.50000  23.000000           17.000000               56.000000
75%       7.75000  30.750000           18.000000               76.750000
max      10.00000  64.000000           19.000000               94.000000
```





Task:3 Forest Cover Type Classification

The below code I run on google collab amd observe the following output:

Code:-

```
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 RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.preprocessing import StandardScaler

import gzip

# Step 1: Load dataset
with gzip.open('covtype.data.gz', 'rt') as f:
    df = pd.read_csv(f, header=None)

# Step 2: Rename columns
column_names = [f'feature_{i}' for i in range(df.shape[1] - 1)] +
['Cover_Type']
df.columns = column_names

# Step 3: Print shape and basic info
print("Shape:", df.shape)
print("\nTarget variable value counts:\n", df['Cover_Type'].value_counts())

# Step 4: Feature and target split
X = df.drop('Cover_Type', axis=1)
y = df['Cover_Type']

# Step 5: Scale features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Step 6: Train/test split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y,
test_size=0.2, random_state=42)

# Step 7: Train Random Forest
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
y_pred = rf.predict(X_test)
```

```python
# Step 8: Evaluation
print("\nClassification Report:\n", classification_report(y_test, y_pred))

# Step 9: Confusion Matrix
plt.figure(figsize=(10, 6))
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d',
cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

# Step 10: Feature Importance
importances = rf.feature_importances_
indices = np.argsort(importances)[-10:][::-1]  # Top 10 features

plt.figure(figsize=(10, 6))
sns.barplot(x=importances[indices], y=np.array(X.columns)[indices])
plt.title('Top 10 Feature Importances')
plt.xlabel('Importance')
plt.ylabel('Feature')
plt.tight_layout()
plt.show()
```

Output:-

```
Shape: (581012, 55)

Target variable value counts:
 Cover_Type
2    283301
1    211840
3     35754
7     20510
6     17367
5      9493
4      2747
Name: count, dtype: int64

Classification Report:
               precision    recall  f1-score   support

           1       0.97      0.94      0.95     42557
           2       0.95      0.97      0.96     56500
           3       0.94      0.96      0.95      7121
           4       0.91      0.85      0.88       526
           5       0.94      0.77      0.85      1995
           6       0.93      0.90      0.92      3489
```

7	0.97	0.96	0.97	4015
accuracy			0.96	116203
macro avg	0.95	0.91	0.93	116203
weighted avg	0.96	0.96	0.95	116203