TARUN SURESH

ML LAB ASSIGNMENT 1

Exploring Python Libraries (Numpy, Pandas, Scipy, Scikit-learn, Matplotlib)

Objective

Apply Linear Regression to predict the loan amount sanctioned to users using the dataset provided.

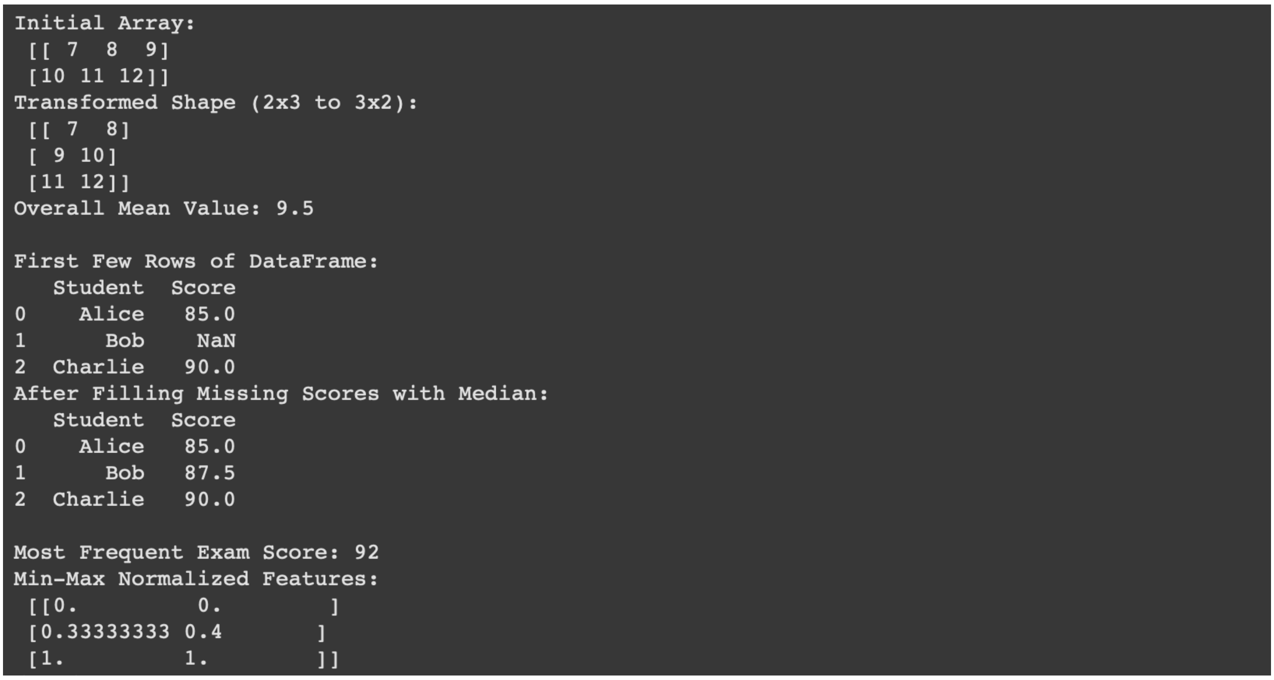
Libraries Used

Numpy, Pandas, Scikit learn, seaborn, matplotlib

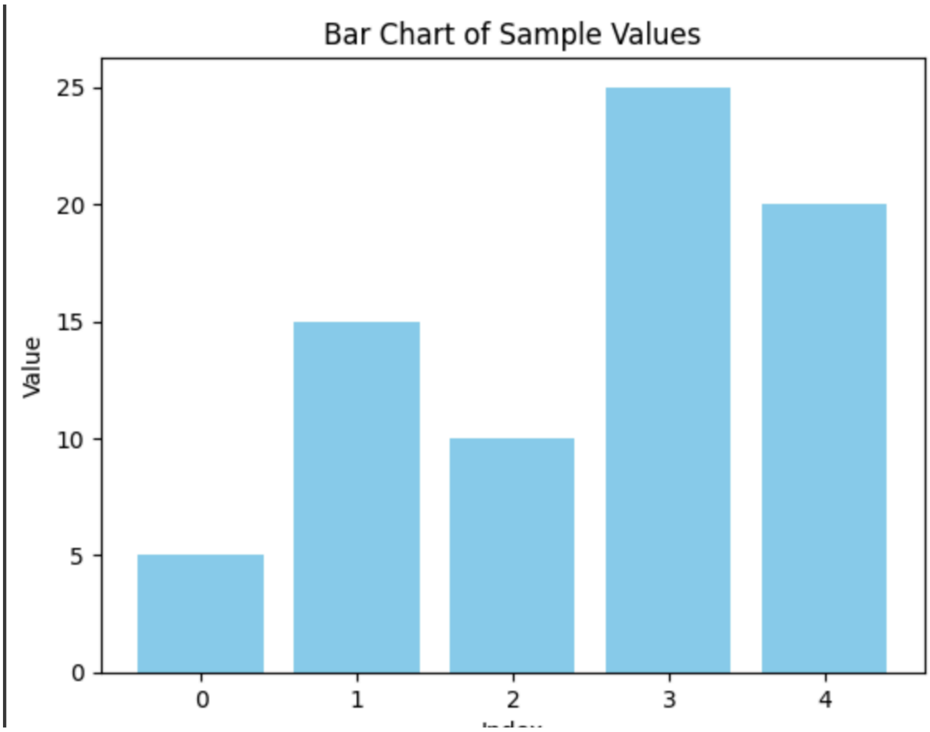
Code

```
# NumPy - Array Manipulations
import numpy as np
arr = np.array([[7, 8, 9], [10, 11, 12]])
print("Initial Array:\n", arr)
print("Transformed Shape (2x3 to 3x2):\n", arr.reshape(3, 2))
print("Overall Mean Value:", np.mean(arr))

# Pandas - Data Preprocessing
import pandas as pd
data = {'Student': ['Alice', 'Bob', 'Charlie'], 'Score': [85, np.nan, 90]}
df = pd.DataFrame(data)
print("\nFirst Few Rows of DataFrame:")
print(df.head())
df['Score'].fillna(df['Score'].median(), inplace=True)
print("After Filling Missing Scores with Median:")
print(df)

# Scipy - Mathematical Computing
from scipy import stats
exam_scores = [88, 92, 92, 75, 83, 92, 70]
mode_result = stats.mode(exam_scores, keepdims=True)
print("\nMost Frequent Exam Score:", mode_result.mode[0])

# Scikit-learn - ML Workflows
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
features = [[50, 2000], [60, 2200], [80, 2500]]
normalized_data = scaler.fit_transform(features)
print("Min-Max Normalized Features:\n", normalized_data)

# Matplotlib - Data Visualization
import matplotlib.pyplot as plt
x_vals = [0, 1, 2, 3, 4]
y_vals = [5, 15, 10, 25, 20]
```

```
plt.bar(x_vals, y_vals, color='skyblue')
plt.title("Bar Chart of Sample Values")
plt.xlabel("Index")
plt.ylabel("Value")
plt.show()
```



```
Initial Array:
 [[ 7  8  9]
 [10 11 12]]
Transformed Shape (2x3 to 3x2):
 [[ 7  8]
 [ 9 10]
 [11 12]]
Overall Mean Value: 9.5

First Few Rows of DataFrame:
   Student  Score
0    Alice   85.0
1      Bob    NaN
2  Charlie   90.0
After Filling Missing Scores with Median:
   Student  Score
0    Alice   85.0
1      Bob   87.5
2  Charlie   90.0

Most Frequent Exam Score: 92
Min-Max Normalized Features:
 [[0.         0.        ]
 [0.33333333 0.4       ]
 [1.         1.        ]]
```



Experiment 2 - Exploring Public Repositories and Identifying ML Models

To download datasets and identify suitable ML models (Supervised, Unsupervised, Classification, Regression).

Dataset	Source	ML Type	Model
Loan Prediction	Kaggle	Supervised	Classification (Logistic Regression))
Handwritten Character Recognition	UCI	Supervised	Classification (CNN/SVM)
Email Spam Classification	UCI	Supervised	Classification (Naive Bayes/ SVM)
Diabetes Prediction	UCI	Supervised	Classification (Logistic Regression))
Iris Dataset	UCI	Supervised	Classification (KNN)

Learning Outcome

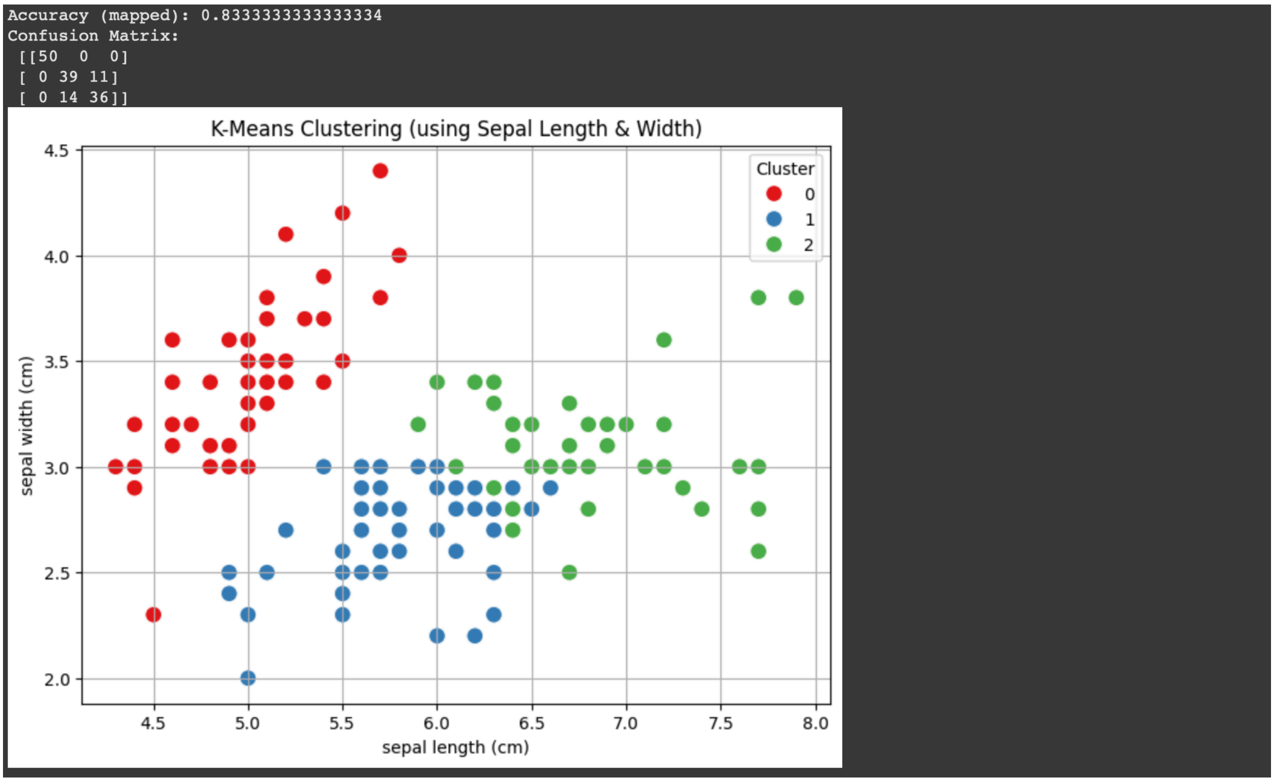
- Understood data concept and selected appropriate ML type and model.

Experiment 3 - ML Workflow with Iris dataset

To explore iris dataset using appropriate ML model

```
from google.colab import drive
drive.mount('/content/drive')

# Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.datasets import load_iris
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import mode

# Load the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target
feature_names = iris.feature_names
target_names = iris.target_names

# Standardize the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Apply K-Means clustering
kmeans = KMeans(n_clusters=3, random_state=42, n_init=10)
kmeans.fit(X_scaled)
y_kmeans = kmeans.labels_

# Map cluster labels to true labels using majority vote
labels = np.zeros_like(y_kmeans)
for i in range(3):
    mask = (y_kmeans == i)
    labels[mask] = mode(y[mask])[0]

# Evaluate
```

```
print("Accuracy (mapped):", accuracy_score(y, labels))

print("Confusion Matrix:\n", confusion_matrix(y, labels))


# Plot clusters using 2 original features (sepal length vs sepal width)

plt.figure(figsize=(8, 6))

sns.scatterplot(

    x=X[:, 0], y=X[:, 1], hue=labels, palette='Set1', s=100

)

plt.title("K-Means Clustering (using Sepal Length & Width)")

plt.xlabel(feature_names[0])  # Sepal length

plt.ylabel(feature_names[1])  # Sepal width

plt.legend(title="Cluster")

plt.grid(True)

plt.show()
```



```
Accuracy (mapped): 0.8333333333333334
Confusion Matrix:
 [[50  0  0]
 [ 0 39 11]
 [ 0 14 36]]
```

```
df=pd.read_csv('/content/drive/MyDrive/loantrain.csv')
df.head()

df=pd.read_csv('/content/drive/MyDrive/spam_ham_dataset.csv')
df.head()

df=pd.read_csv('/content/sample_data/mnist_train_small.csv')
df.head()
```

```
from sklearn.datasets import load_diabetes

import pandas as pd

diabetes = load_diabetes()

df = pd.DataFrame(data=diabetes.data, columns=diabetes.feature_names)

df['target'] = diabetes.target

print(df.head())
```

Inference

EDA - Clear feature separation in visualizations

Preprocessing - Features Standardized

Feature Selection - All 4 features selected

Evaluation - 83% Accuracy

Learning

- Practical Application of feature selection
- Improved visualization and model evaluation skills