Experiment 1: Exploring Python Libraries (Numpy, Pandas, Scipy, Scikit-learn, Matplotlib)

REPORT BY GOPIKA GANESAN

Aim: To such as array manipulations, data preprocessing, mathematical computing, machine learning workflows, and data visualization.

Code Snippets and Outputs:

```
# NumPy - Array Manipulations
import numpy as np
arr = np.array([[1, 2, 3], [4, 5, 6]])
print("Original Array:", arr)
print("Reshaped Array:", arr.reshape(3, 2))
print("Array Mean:", np.mean(arr))

# Pandas - Data Preprocessing
import pandas as pd
data = {'Name': ['Tom', 'Jerry', 'Mickey'], 'Age': [20, 21, np.nan]}
df = pd.DataFrame(data)
print("DataFrame Head:")
print(df.head())
df['Age'].fillna(df['Age'].mean(), inplace=True)
print("After Handling Missing Value:")
print(df)

# Scipy - Mathematical Computing
```

```
from scipy import stats
sample_data = [1, 2, 3, 4, 4, 5, 5, 5, 6]
mode_val = stats.mode(sample_data)
print("Mode:", mode_val)

# Scikit-learn - ML Workflows
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaled_data = scaler.fit_transform([[1, 2], [3, 4], [5, 6]])
print("Standardized Data:", scaled_data)

# Matplotlib - Data Visualization
import matplotlib.pyplot as plt
plt.plot([1, 2, 3, 4], [10, 20, 25, 30])
plt.title("Simple Line Plot")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.show()
```

Learning Outcome:

- Understood and used array manipulation (reshape, mean)
- Preprocessed data using Pandas
- Applied statistical operations from SciPy
- Scaled features using Scikit-learn
- Visualized data using Matplotlib

Experiment 2: Exploring Public Repositories and Identifying ML Models

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

Datasets and Suggested ML Tasks:

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

Learning Outcome:

- Understood dataset context and selected appropriate ML type
- Mapped datasets to real-world applications

Experiment 3: Machine Learning Workflow with Iris Dataset

Aim: To explore the ML workflow using the Iris dataset.

Code Snippets:

```
from sklearn.datasets import load_iris
```

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report

# Load dataset
data = load_iris()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['target'] = data.target

# EDA
print(df.describe())
sns.pairplot(df, hue='target')
plt.show()

# Preprocessing
X = df.drop('target', axis=1)
y = df['target']
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
# Feature Selection
selector = SelectKBest(score_func=f_classif, k=3)
X_selected = selector.fit_transform(X_scaled, y)

# Train-test Split
X_train, X_test, y_train, y_test = train_test_split(X_selected, y,
test_size=0.2, random_state=42)

# Model Training
clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

# Evaluation
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

Inference Table:

Step	Observation
EDA	Clear feature separation in visualizations
Preprocessing	Features standardized successfully
Feature Selection	Reduced features from 4 to 3
Evaluation	Achieved high accuracy (~93%)

Learning Reflection:

- Understood complete ML pipeline
- Learned practical application of feature selection

- Improved visualization and model evaluation skills