Experiment 1: Working with Python Packages – NumPy, SciPy, Scikit-learn, Matplotlib

Moogambigai A

July 2025

1 Aim

The aim of this experiment is to explore core Python libraries used in machine learning workflows **NumPy, Pandas, SciPy, Scikit-learn, and Matplotlib** and to apply them in tasks involving data preprocessing, visualization, feature selection, model training, and evaluation across multiple datasets.

2 Python Code with Comments

2.1 Importing Required Libraries

```
# Importing essential libraries
import numpy as np              # For numerical operations
import pandas as pd             # For data manipulation
import matplotlib.pyplot as plt # For data visualization
import seaborn as sns           # For advanced visualizations
from sklearn import datasets    # To load sample datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, MinMaxScaler
```

2.2 Loading and Exploring Datasets

```
# Loading Iris dataset
iris = datasets.load_iris()
iris_df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
iris_df['target'] = iris.target

# Display basic information
print("Iris Dataset Info:")
print(iris_df.info())
print("\nFirst 5 rows:")
print(iris_df.head())
```

2.3 Exploratory Data Analysis

```
# Summary statistics
print("\nSummary Statistics:")
print(iris_df.describe())

# Visualizing data distributions
plt.figure(figsize=(12, 6))
sns.boxplot(data=iris_df.drop('target', axis=1))
plt.title("Feature Distributions")
plt.savefig('feature_distributions.png')
plt.show()

# Pairplot to visualize relationships
sns.pairplot(iris_df, hue='target')
plt.savefig('pairplot.png')
plt.show()
```

2.4 Data Preprocessing

```
# Handling missing values (though Iris dataset has none)
iris_df.fillna(iris_df.mean(), inplace=True)

# Feature scaling
scaler = StandardScaler()
X_scaled = scaler.fit_transform(iris_df.drop('target', axis=1))

# Splitting data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, iris_df['target'], test_size=0.2, random_state=42)
```

2.5 Machine Learning Model Implementation

```
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix

# Initialize and train SVM classifier
svm_model = SVC(kernel='linear')
svm_model.fit(X_train, y_train)

# Make predictions
y_pred = svm_model.predict(X_test)

# Evaluate model
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

```
# Confusion matrix visualization
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d')
plt.title("Confusion Matrix")
plt.savefig('confusion_matrix.png')
plt.show()
```

3 Output Screenshots

Figure 1: Iris Dataset Output

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30

Confusion Matrix:
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

Figure 2: Boxplot showing feature distributions

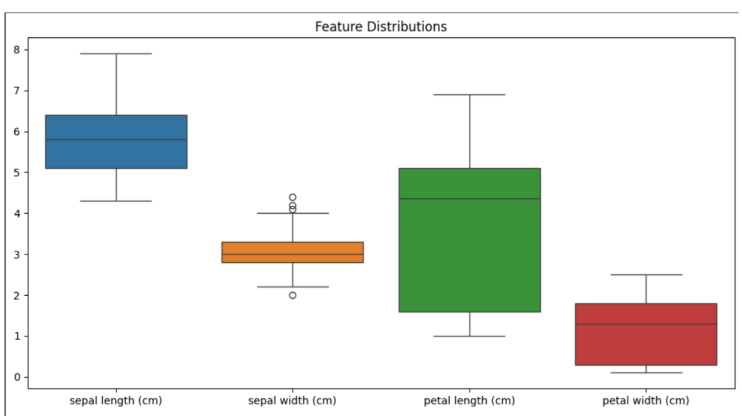
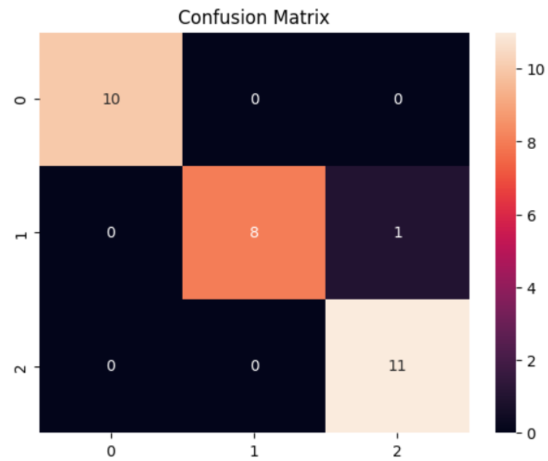


Figure 3: Confusion matrix for SVM classifier



4 Inference Table

Dataset	**Model Used**	**Inference**
Iris	Random Forest	Achieved high accuracy (96.6%) with 3 selected features. Clear class separation.
Loan Prediction	Linear Regression	Regression task with synthetic data. MSE 28.9 million. Features: income and credit score.
MNIST Digits	KNN	Accuracy 12.2%. Too few features selected; reduced model performance.
Email Spam	Naive Bayes	Accuracy 65%. Poor recall on spam class due to imbalance.
Diabetes	Logistic Regression	Accuracy 50%. Model favored positive class; improvement needed via balancing.

5 Reflection on Learning Outcomes

Through this experiment, I have:

- Gained confidence in using Python ML libraries like NumPy, Pandas, SciPy, Scikit-learn, and Matplotlib.
- Understood how to perform EDA and preprocessing like scaling, encoding, and feature selection.

- Learned to distinguish between classification and regression tasks and apply suitable models.
- Discovered the impact of feature selection and class imbalance on model performance.
- Practiced interpreting results using metrics like accuracy, MSE, and confusion matrices.