Experiment 1: Working with Python packages-Numpy, Scipy, Scikit-Learn, Matplotlib

Shinigdapriya Sathish

30 July 2025

Aim

To explore the basic functionalities of important Python libraries: NumPy, Pandas, SciPy, Scikit-learn, and Matplotlib, and understand machine learning workflows using publicly available datasets.

Libraries Used

- **NumPy**: Numerical computations and array manipulations.
- **Pandas**: Handling and processing tabular data.
- **SciPy**: Scientific computing including integration, optimization, and linear algebra.
- **Scikit-learn**: Machine learning algorithms and model evaluation.
- **Matplotlib**: Data visualization and plotting.

Mathematical / Theoretical Description

1. NumPy

NumPy provides multi-dimensional arrays and supports vectorized operations. Common operations include:

- Element-wise arithmetic: $x + y, x \cdot y$
- Aggregations: mean, standard deviation, etc.
- Array slicing and indexing.

2. Pandas

Pandas provides DataFrames for structured data. Key operations include:

- Handling missing values (e.g., using `fillna()`).
- Filtering rows with conditions.
- Grouping and summarizing data.

3. SciPy

Includes scientific operations such as:

- Integration: $\int_0^1 x^2 dx$
- Optimization: Finding roots of equations.
- Linear algebra: Solving systems of equations $Ax = b$

4. Scikit-learn

Supports full machine learning workflow:

- Dataset loading and splitting.
- Model training (e.g., Logistic Regression).
- Evaluation using metrics like accuracy.

5. Matplotlib

Used for:

- Line plots.
- Histograms.
- Customizing plots with labels, legends, and grids.

Results and Discussions

Code Snippets

##Section 1: NumPy — Numerical Python ###Description: NumPy is the foundational package for numerical computation in Python. It supports multi-dimensional arrays and a variety of mathematical operations.

[ ]:
```
import numpy as np

# Creating arrays
a = np.array([1, 2, 3])
b = np.zeros((2, 3))
c = np.ones((3, 3))
d = np.eye(3)

# Array indexing and slicing
print(a[1])        # 2
print(c[1:, :2])   # slicing

# Arithmetic
x = np.array([1, 2, 3])
y = np.array([4, 5, 6])
```

```
print(x + y)
print(x * y)

# Aggregation
print(np.mean(x))
print(np.std(y))
```

```
2
[[1. 1.]
 [1. 1.]]
[5 7 9]
[ 4 10 18]
2.0
0.816496580927726
```

##Section 2: Pandas — Data Handling ###Description: Pandas is used for handling tabular data with dataframes. It provides tools for cleaning, transforming, and analyzing data.

[ ]:
```
import pandas as pd

# Create DataFrame
df = pd.DataFrame({
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [25, 30, 35],
    'Score': [85.5, 90.3, np.nan]
})

# Basic operations
print(df.head())
print(df.describe())

# Handling missing data
df['Score'] = df['Score'].fillna(df['Score'].mean())

# Filtering
print(df[df['Age'] > 28])

# Grouping
grouped = df.groupby('Age')
print(grouped)
```

```
      Name  Age  Score
0    Alice   25   85.5
1      Bob   30   90.3
2  Charlie   35    NaN
             Age      Score
count   3.0   2.000000
mean   30.0  87.900000
std     5.0   3.394113
```

```
min    25.0  85.500000
25%    27.5  86.700000
50%    30.0  87.900000
75%    32.5  89.100000
max    35.0  90.300000
      Name  Age  Score
1      Bob   30   90.3
2  Charlie   35   87.9
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x7ba151f8b850>
```

##Section 3: SciPy — Scientific Computing ###Description: SciPy builds on NumPy and provides a wide range of mathematical tools such as optimization, integration, interpolation, eigenvalue problems, etc.

[ ]:
```python
from scipy import integrate, optimize, linalg, stats

# Integration
result, error = integrate.quad(lambda x: x**2, 0, 1)
print("Integral:", result)

# Root finding
root = optimize.root_scalar(lambda x: x**2 - 4, bracket=[1, 3])
print("Root:", root.root)

# Linear algebra
A = np.array([[3, 1], [1, 2]])
b = np.array([9, 8])
x = linalg.solve(A, b)
print("Solution:", x)

# Statistics
print("PDF at x=0.5 for normal dist:", stats.norm.pdf(0.5))
```

```
Integral: 0.33333333333333337
Root: 1.9999999999999987
Solution: [2. 3.]
PDF at x=0.5 for normal dist: 0.3520653267642995
```

##Section 4: Scikit-learn — Machine Learning ###Description: Scikit-learn is a library for machine learning that provides tools for classification, regression, clustering, model selection, and preprocessing.

[ ]:
```python
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# Load dataset
iris = load_iris()
```

```
X, y = iris.data, iris.target

# Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Train model
model = LogisticRegression(max_iter=200)
model.fit(X_train, y_train)

# Predict & evaluate
y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
```

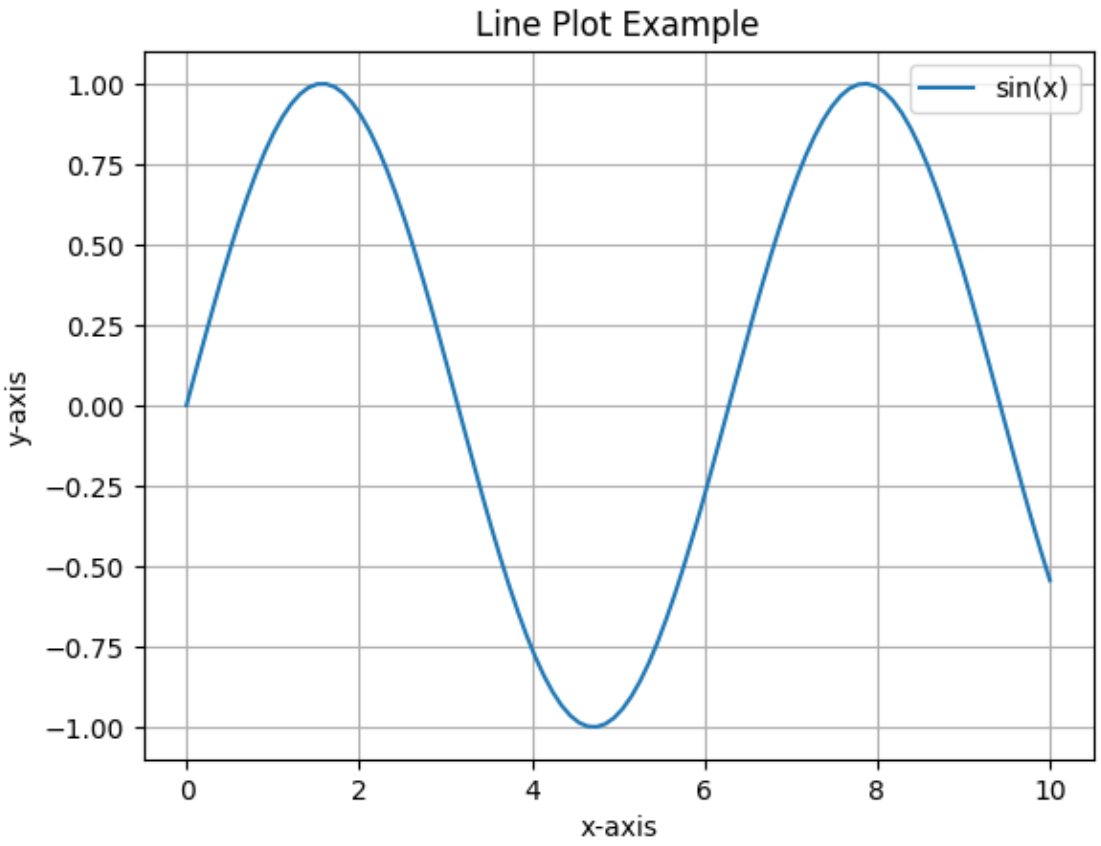
```
Accuracy: 0.9666666666666667
```

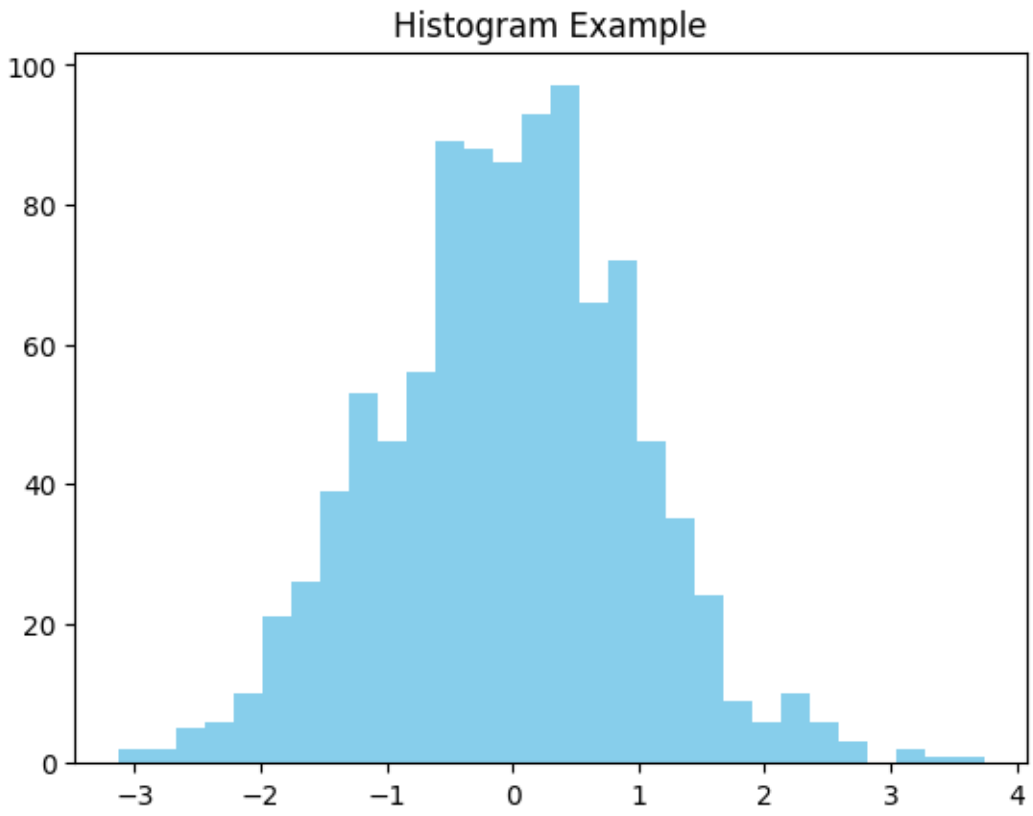
##Section 5: Matplotlib — Data Visualization ###Description: Matplotlib is used to create static, interactive, and animated plots.

[ ]:
```
import matplotlib.pyplot as plt

# Line plot
x = np.linspace(0, 10, 100)
y = np.sin(x)
plt.plot(x, y, label='sin(x)')
plt.title('Line Plot Example')
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.legend()
plt.grid(True)
plt.show()

# Histogram
data = np.random.randn(1000)
plt.hist(data, bins=30, color='skyblue')
plt.title("Histogram Example")
plt.show()
```


Line Plot Example
sin(x)
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
y-axis
0
2
4
6
8
10
x-axis



- NumPy: Array operations, indexing, slicing, arithmetic.
- Pandas: DataFrame creation, handling NaN values, filtering.
- SciPy: Calculated integrals, found roots, solved linear equations.
- Scikit-learn: Loaded Iris dataset, trained a logistic regression model, calculated accuracy.
- Matplotlib: Plotted line graph and histogram with appropriate titles and legends.

Dataset Exploration Table

Dataset	**Type of ML Task**	**Suitable ML Algorithm**
Iris Dataset	Classification	Logistic Regression, KNN, SVM
Loan Amount Prediction	Regression	Linear Regression, Random Forest
Predicting Diabetes	Classification	Logistic Regression, Decision Tree
Email Spam Classification	Classification	Naive Bayes, SVM

MNIST Handwritten Digits	Classification	CNN (Deep Learning), KNN, SVM

Learning Practices

- Understood NumPy operations for numerical computation.
- Explored Pandas for handling tabular data and missing values.
- Used SciPy for solving calculus and linear algebra problems.
- Trained and evaluated ML models using Scikit-learn.
- Created visualizations using Matplotlib for data understanding.
- Identified ML task types for common datasets.
- Applied preprocessing, feature selection, and model evaluation techniques.

Conclusion

This experiment provided hands-on experience with essential Python libraries used in data science and machine learning. We gained insight into the complete machine learning workflow—from loading and preprocessing data to model building and visualization.