LAB 1: INTRODUCTION TO PYTHON FOR MACHINE LEARNING

AIM:

To understand basic Python programming concepts and libraries used in Machine Learning.

INTRODUCTION:

Machine learning has become one of the fastest growing fields in computer science. It allows computers to learn from data and make predictions without being directly programmed for each task. Python has become the most popular language for machine learning because it is easy to learn and has powerful libraries that make complex tasks simple.

Python provides several important libraries for machine learning. NumPy is used for fast numerical calculations with arrays. Pandas helps us organize and analyze data in tables called DataFrames. Matplotlib creates graphs and charts to visualize data patterns. Scikit-learn provides ready-to-use machine learning algorithms.

Understanding these libraries is the first step in learning machine learning with Python. Once we know how to use them, we can start building our own models and solving real problems.

THEORY:

Python provides several key libraries that form the basis of machine learning work:

1. NumPy (Numerical Python):

NumPy is used for working with numbers and arrays. An array is like a list, but it can have multiple dimensions and supports fast mathematical operations. When we need to do calculations on thousands or millions of numbers, NumPy is much faster than regular Python.

Key features:

- Creates arrays of any dimension
- Performs element-wise operations
- Has functions for statistics like mean, median, standard deviation
- Supports linear algebra operations

2. Pandas (Panel Data):

Pandas gives us two main data structures - Series and DataFrame. A Series is like a single column, and a DataFrame is like a table with rows and columns.

Key features:

- Reads data from CSV, Excel, databases
- Handles missing data
- Filters and sorts data easily
- Groups data for analysis

3. Matplotlib:

Matplotlib creates static, animated, and interactive visualizations. It is similar to MATLAB's plotting functions, which is why it's called Matplotlib.

Key features:

- Line plots, scatter plots, bar charts
- Customizable labels, titles, colors
- Can save plots as image files
- Works well with NumPy and Pandas

4. Scikit-learn:

Scikit-learn provides simple tools for data analysis and modeling. It is built on NumPy, Pandas, and Matplotlib.

Key features:

- Classification algorithms (identifying categories)
- Regression algorithms (predicting numbers)
- Clustering algorithms (finding groups in data)
- Tools for splitting data into training and testing sets

PROCEDURE:

1. Install Python (version 3.7 or higher recommended)
2. Install required libraries using pip: `pip install numpy pandas matplotlib scikit-learn`
3. Open a Python editor or Jupyter Notebook
4. Import the libraries
5. Create simple examples using each library
6. Run the code and observe outputs
7. Create visualizations to understand the data

CODE EXAMPLES:

Example 1 - NumPy Arrays and Operations:

```
import numpy as np


# Creating arrays

arr1 = np.array([1, 2, 3, 4, 5])

arr2 = np.array([10, 20, 30, 40, 50])
```

```
print("Array 1:", arr1)

print("Array 2:", arr2)


# Basic operations

print("Sum:", arr1 + arr2)

print("Product:", arr1 * arr2)


# Statistical operations

print("Mean of arr1:", np.mean(arr1))

print("Standard deviation:", np.std(arr1))

print("Maximum value:", np.max(arr2))


# 2D array (matrix)

matrix = np.array([[1, 2, 3],

                   [4, 5, 6],

                   [7, 8, 9]])

print("Matrix:")

print(matrix)

print("Shape:", matrix.shape)
```

Example 2 - Pandas DataFrames:

```
import pandas as pd


# Creating a DataFrame

student_data = {
```

```
    'Name': ['Rahul', 'Priya', 'Amit', 'Sneha', 'Vikram'],
    'Age': [21, 22, 20, 21, 23],
    'Marks': [85, 92, 78, 88, 95],
    'City': ['Mumbai', 'Delhi', 'Pune', 'Mumbai', 'Delhi']
}

df = pd.DataFrame(student_data)
print("Student DataFrame:")
print(df)

# Basic operations
print("Average marks:", df['Marks'].mean())
print("Students from Mumbai:")
print(df[df['City'] == 'Mumbai'])

# Adding a new column
df['Grade'] = df['Marks'].apply(lambda x: 'A' if x >= 90 else ('B' if x >= 80 else 'C'))
print("DataFrame with grades:")
print(df)
```

Example 3 - Matplotlib Visualizations:

```
import matplotlib.pyplot as plt

# Line plot
months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun']
temperature = [15, 17, 22, 28, 32, 35]
```

```
plt.figure(figsize=(8, 5))
plt.plot(months, temperature, marker='o', color='red', linewidth=2)
plt.xlabel('Month')
plt.ylabel('Temperature (°C)')
plt.title('Monthly Temperature')
plt.grid(True)
plt.show()


# Bar chart using student marks
names = df['Name']
marks = df['Marks']


plt.figure(figsize=(8, 5))
plt.bar(names, marks, color='green')
plt.xlabel('Student Name')
plt.ylabel('Marks')
plt.title('Student Marks Comparison')
plt.show()
```

Example 4 - Simple Machine Learning with Scikit-learn:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression


# Sample data - hours studied vs marks obtained
hours = np.array([1, 2, 3, 4, 5, 6, 7, 8]).reshape(-1, 1)
```

```
marks = np.array([45, 55, 60, 65, 75, 80, 85, 90])


# Split data into training and testing
X_train, X_test, y_train, y_test = train_test_split(hours, marks, test_size=0.25)


# Create and train model
model = LinearRegression()
model.fit(X_train, y_train)


# Make predictions
predictions = model.predict(X_test)
print("Predictions:", predictions)
print("Actual values:", y_test)
```

OUTPUT:

NumPy Output:

```
Array 1: [1 2 3 4 5]
Array 2: [10 20 30 40 50]
Sum: [11 22 33 44 55]
Product: [10 40 90 160 250]
Mean of arr1: 3.0
Standard deviation: 1.4142135623730951
Maximum value: 50
Matrix:
 [[1 2 3]
  [4 5 6]
```

```
  [7 8 9]]
Shape: (3, 3)
```

Pandas Output:

```
Student DataFrame:
     Name  Age  Marks    City
0   Rahul   21     85  Mumbai
1   Priya   22     92   Delhi
2    Amit   20     78    Pune
3   Sneha   21     88  Mumbai
4  Vikram   23     95   Delhi

Average marks: 87.6

Students from Mumbai:
    Name  Age  Marks    City
0  Rahul   21     85  Mumbai
3  Sneha   21     88  Mumbai

DataFrame with grades:
     Name  Age  Marks    City Grade
0   Rahul   21     85  Mumbai     B
1   Priya   22     92   Delhi     A
2    Amit   20     78    Pune     C
3   Sneha   21     88  Mumbai     B
4  Vikram   23     95   Delhi     A
```

Matplotlib Output:

Two graphs are displayed - a line graph showing temperature rise over months and a bar chart comparing student marks.

Scikit-learn Output:

```
Predictions: [88.5 62.3]

Actual values: [90 60]
```

OBSERVATIONS:

1. NumPy arrays are much faster than Python lists for mathematical operations. They use less memory and provide many built-in functions for calculations.
2. Pandas DataFrames make it easy to work with structured data. We can filter, sort, and analyze data with just a few lines of code.
3. Matplotlib creates clear visualizations that help us understand patterns in data. Visual representation makes it easier to spot trends and outliers.
4. Scikit-learn provides simple interfaces for machine learning. Even complex algorithms can be used with just a few lines of code.
5. All these libraries work well together. We can use NumPy for calculations, Pandas for data handling, Matplotlib for visualization, and Scikit-learn for building models.
6. The syntax is beginner-friendly. Even without deep programming knowledge, we can start doing useful work with these libraries.
7. Real-world machine learning projects use all these libraries together. Understanding each one separately helps in building complete solutions.

CONCLUSION:

Python has become the standard language for machine learning because of its powerful libraries. NumPy provides fast numerical computing, Pandas makes data handling easy, Matplotlib creates useful visualizations, and Scikit-learn gives us ready-to-use machine learning algorithms.

Learning these libraries is important for anyone interested in machine learning. They save time by providing tested and optimized code for common tasks. Instead of writing everything from scratch, we can focus on solving the actual problem.

The examples in this lab show how each library works and how they complement each other. NumPy handles the numbers, Pandas organizes the data, Matplotlib shows the patterns, and Scikit-learn builds the models. Together, they provide everything needed to start working on machine learning projects.

As we continue learning machine learning, these libraries will be used in every project. Getting comfortable with them now will make advanced topics easier to understand later. The best way to learn is by practicing with different datasets and trying out various functions that these libraries offer.

Python's machine learning ecosystem continues to grow, with new libraries and tools being added regularly. However, NumPy, Pandas, Matplotlib, and Scikit-learn remain the core foundation that every machine learning practitioner should master.