Experiment 1

Aim: Introduction to:

(A) Jupyter Notebook IDE
(B) Spyder IDE
(C) NumPy and Pandas
(D) MatplotLib and SeaBorn, and
(E) Scikit Learn

Theory:

(A) Jupyter Notebook IDE: Jupyter Notebook is an open-source, web-based interactive development environment (IDE) that allows users to create and share documents containing live code, equations, visualizations and explanatory texts, it supports over 40 programming languages, with python being the most widely used.

Features:

- **Interactive Cells**: Code is executed in isolated cells, allowing for incremental testing and debugging.
- **Markdown Support:** Markdown cells enable the inclusion of headings, lists, links, and other formatting options to create structured documentation.
- **Data Visualization**: Direct integration with libraries like Matplotlib, Seaborn, and Plotly allows inline plotting of data.
- **Reproducibility**: Notebooks can be saved in .ipynb format and shared for reproducibility of results.
- **Kernel Support**: Jupyter supports various programming kernels like Python, R, Julia, and more.

Use Cases:

- Data analysis and visualization
- Machine Learning and deep learning
- Educational tutorials and presentations

(B) Spyder IDE: Spyder(Scientific Python Development Environment) is a powerful open-source python IDE designed for scientific computing and data analysis. It is part of the Anaconda distribution and is widely used by data scientists and engineers.

Features:

- **Multi-panel Layout**: Includes a code editor, console, file explorer, and variable explorer.
- **Integrated Debugger**: Allows setting breakpoints and stepping through code.
- **Variable Explorer**: Displays data structures like arrays and dataframes, enabling real-time inspection and modification.
- **IPython Console**: Provides interactive testing and debugging.
- **Integrated Documentation**: Inline help and documentation support improve code efficiency.

Use Cases:

- Scientific Programming
- Data Analysis
- Debugging complex machine learning models

(C) NumPy and Pandas: NumPy (Numerical Python) and Pandas are the two most widely used Python libraries for numerical computing and data manipulation.

NumPy: NumPy provides support for large, multi-dimensional arrays and matrices along with a collection of mathematical functions to operate on these arrays.

Features:

- Efficient handling of arrays and matrices
- Supports element-wise operations
- Offers linear algebra, Fourier transforms, and random number capabilities.

Pandas: Pandas is used for data manipulation and analysis. It introduces two main data structures:

- Series: One dimensional labeled array
- DataFrame: Two-dimensional labeled data structures

```python
import numpy as np

# Array Creation
a1 = np.array([1, 2, 3])
a2 = np.array([[1, 2], [3, 4]])
z = np.zeros((2, 2))
o = np.ones((2, 2))
i = np.eye(3)
l = np.linspace(0, 1, 5)
r = np.arange(0, 10, 2)

# Properties
print("Shape:", a2.shape, "Type:", a2.dtype, "Size:", a2.size)

# Indexing & Slicing
print("a1[1]:", a1[1], "a2[0]:", a2[0], "a1[1:3]:", a1[1:3], "a1[-1]:", a1[-1])

# Reshaping
reshaped = np.arange(1, 13).reshape(3, 4)

# Math Ops
b1, b2 = np.array([1, 2, 3]), np.array([4, 5, 6])
print("Add:", b1 + b2, "Mul:", b1 * b2, "Square:", np.square(b1), "Sum:", b1.sum(), "Mean:", b1.mean())

# Broadcasting
b3 = np.array([[1], [2], [3]])
b4 = np.array([10, 20, 30])
print("Broadcast:\n", b3 + b4)
```

```python
# Stacking & Concatenation
print("VStack:\n", np.vstack((b1, b2)))
print("Concat:", np.concatenate((b1, b2)))

# Logical Ops
print("b1 > 2:", b1 > 2)

# Random
print("Rand:\n", np.random.rand(2, 2))
print("RandInt:\n", np.random.randint(0, 10, (2, 2)))

# Transpose & Dot Product
print("Transpose:\n", a2.T)
m1 = np.array([[1, 2], [3, 4]])
m2 = np.array([[5, 6], [7, 8]])
print("Dot:\n", np.dot(m1, m2))

# Flattening
print("Flatten:", reshaped.flatten())

# Sorting
unsorted = np.array([3, 1, 4])
print("Sorted:", np.sort(unsorted))

# Universal Functions
u = np.array([1, 2, 3])
print("Exp:", np.exp(u), "Log:", np.log(u), "Sin:", np.sin(u))
```

```python
import pandas as pd
import os
import numpy as np

#Set working directory
os.chdir("")

df= pd.read_csv("Toyota.csv", index_col=0, na_values=["###","??","????"])


print("First 5 rows of the dataset: ")
print(df.head())

print("\nData types in each column: ")
print(df.dtypes)

print("\nCount of unique data types: ")
print(df.dtypes.value_counts())

numericData= df.select_dtypes(include=["int64", "float64"])
objectData= df.select_dtypes(include=["object"])
print("\nNumeric Data Columns: ")
print(numericData.head)
print("\nObject Data Columns: ")
print(objectData.head)
```

```python
print("\nDataFrame Info: ")
print(df.info)

for col in df.columns:
    print(f"\nUnique Values in'{col}':")
    print(df[col].unique())

df["Doors"]= df["Doors"].replace({"three":3, "Four":4, "Five":5}).astype("float64")
df["MetColor"] = df["MetColor"].astype("object")
df["Automatic"] = df["Automatic"].astype("object")

print("\nMissing Values Count in Each Column")
print(df.isnull.sum())

df.to_csv("Toyota_Cleaned.csv", index=False)

print("\nData Cleaning and analysis completed Cleaned data saved as 'Toyota_Cleaned.csv'. ")


```

(D) MatplotLib and Seaborn: Matplotlib and Seaborn are Python libraries used for data visualization.

MatplotLib: Matplotlib is a low-level library that provides extensive plotting functions such as line plots, scatter plots, bar plots, and histograms.

Seaborn: Seaborn is built on top of Matplotlib and provides a high-level interface for drawing attractive statistical graphics. It integrates closely with Pandas data structures.

```python
import matplotlib.pyplot as plt
import numpy as np

# Line Plot
x = np.linspace(0, 10, 100)
plt.plot(x, np.sin(x), label="sin(x)", linestyle="--")
plt.title("Line Plot"); plt.legend(); plt.grid(); plt.show()

# Scatter Plot
plt.scatter(np.random.rand(50), np.random.rand(50),
            s=300*np.random.rand(50), c=np.random.rand(50), alpha=0.5)
plt.colorbar(); plt.title("Scatter Plot"); plt.show()

# Bar Plot
plt.bar(["A", "B", "C", "D", "E"], [3, 7, 1, 8, 5])
plt.title("Bar Plot"); plt.show()

# Histogram
plt.hist(np.random.randn(1000), bins=30, color="lightgreen", edgecolor="black")
plt.title("Histogram"); plt.show()

# Pie Chart
plt.pie([15, 30, 45, 10], labels=["A", "B", "C", "D"], autopct="%1.1f%%")
plt.title("Pie Chart"); plt.axis("equal"); plt.show()

```

```python
# Subplots
x = np.linspace(0, 10, 100)
fig, ax = plt.subplots(2, 1)
ax[0].plot(x, np.sin(x)); ax[0].set_title("sin(x)")
ax[1].plot(x, np.cos(x)); ax[1].set_title("cos(x)")
plt.tight_layout(); plt.show()

# Customized Plot
plt.plot(x, np.sin(x), color="green", linestyle="-.")
plt.xticks(np.arange(0, 11)); plt.yticks(np.arange(-1, 1.5, 0.5))
plt.title("Customized Plot"); plt.grid(); plt.show()

# Annotation
plt.plot(x, np.sin(x), color="purple")
plt.annotate("Max", xy=(1.5*np.pi, 1), xytext=(4, 1.2),
             arrowprops=dict(facecolor="red"))
plt.title("Annotated Plot"); plt.show()

# Save Plot
plt.plot(x, np.sin(x)); plt.title("Saved Plot")
plt.savefig("saved_plot.png", dpi=300); plt.show()

# Polar Plot
theta = np.linspace(0, 2*np.pi, 100)
r = np.abs(np.sin(2*theta))
plt.subplot(111, polar=True).plot(theta, r)
plt.title("Polar Plot"); plt.show()
```

```python
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

tips = sns.load_dataset('tips')
x = np.linspace(0, 10, 100)

# Line Plot
sns.lineplot(x=x, y=np.sin(x)); plt.title("Line Plot"); plt.show()

# Scatter Plot
sns.scatterplot(data=tips, x='total_bill', y='tip', hue='sex', size='size'); plt.title("Scatter"); plt.show()

# Bar Plot
sns.barplot(x='day', y='total_bill', data=tips); plt.title("Bar Plot"); plt.show()

# Histogram
sns.displot(tips['total_bill'], bins=20, kde=True, color="purple"); plt.title("Histogram"); plt.show()

# Box Plot
sns.boxplot(x='day', y='total_bill', data=tips); plt.title("Box Plot"); plt.show()

```

```python

# Violin Plot
sns.violinplot(x='day', y='total_bill', data=tips, hue='sex', split=True); plt.title("Violin Plot"); plt.show()

# Heatmap
sns.heatmap(tips.corr(), annot=True); plt.title("Heatmap"); plt.show()

# Pair Plot
sns.pairplot(tips, hue='sex'); plt.suptitle("Pair Plot", y=1.02); plt.show()

# Count Plot
sns.countplot(x='day', data=tips, hue='sex'); plt.title("Count Plot"); plt.show()

# KDE Plot
sns.kdeplot(tips['total_bill'], shade=True); plt.title("KDE Plot"); plt.show()

# Regression Plot
sns.lmplot(x='total_bill', y='tip', data=tips, hue='sex'); plt.title("Regression"); plt.show()

# Swarm Plot
sns.swarmplot(x='day', y='total_bill', data=tips, hue='sex'); plt.title("Swarm Plot"); plt.show()
```

(E) Scikit-Learn: Scikit-Learn is one of the most widely used machine learning libraries in Python. It provides simple and efficient tools for data mining and machine learning.

Features:

- Supervised and unsupervised learning algorithms
- Tools for model selection and evaluation
- Preprocessing and data transformation
- Support for regression, classification, clustering, and dimensionality reduction

```python
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.datasets import load_iris
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

data = load_iris()
X, y = data.data, data.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Linear Regression
lr = LinearRegression().fit(X_train, y_train)
print("LR Coefs:", lr.coef_)

# Logistic Regression
log_reg = LogisticRegression(max_iter=200).fit(X_train, y_train)
print("LogReg Acc:", accuracy_score(y_test, log_reg.predict(X_test)))

# Decision Tree
dt = DecisionTreeClassifier(max_depth=3).fit(X_train, y_train)
print("DT Acc:", accuracy_score(y_test, dt.predict(X_test)))

# Random Forest
```

```python
rf = RandomForestClassifier().fit(X_train, y_train)
y_pred = rf.predict(X_test)
print("RF Acc:", accuracy_score(y_test, y_pred))

# Confusion Matrix
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, xticklabels=data.target_names, yticklabels=data.target_names); plt.title("Confusion Matrix"); plt.show()

# Classification Report
print(classification_report(y_test, y_pred))

# Feature Importance
sns.barplot(x=data.feature_names, y=rf.feature_importances_); plt.title("Feature Importances"); plt.show()

# Cross-Validation
scores = cross_val_score(rf, X, y, cv=5)
print("CV Score Mean:", scores.mean())

# PCA
X_pca = PCA(n_components=2).fit_transform(X)
sns.scatterplot(x=X_pca[:, 0], y=X_pca[:, 1], hue=y); plt.title("PCA"); plt.show()

# KMeans Clustering
clusters = KMeans(n_clusters=3).fit_predict(X)
sns.scatterplot(x=X[:, 0], y=X[:, 1], hue=clusters); plt.title("KMeans Clustering"); plt.show()

# Decision Boundary (LogReg)
xx, yy = np.meshgrid(np.linspace(X[:, 0].min()-1, X[:, 0].max()+1, 200),
                     np.linspace(X[:, 1].min()-1, X[:, 1].max()+1, 200))
Z = log_reg.predict(np.c_[xx.ravel(), yy.ravel()]).reshape(xx.shape)
plt.contourf(xx, yy, Z, alpha=0.3)
sns.scatterplot(x=X[:, 0], y=X[:, 1], hue=y); plt.title("Decision Boundary"); plt.show()
```

Installation Instructions:

Step 1: Install Python:

1) Go to the official Python website: **https://www.python.org**
2) Download the latest version of Python (preferably Python 3.10 or higher).
3) During installation, check "Add Python to PATH" and proceed with the installation.
4) Verify installation using the command:

```
python —version
```

Step 2: Install Anaconda:

Alternatively, you can install the Anaconda distribution, which includes Jupyter Notebook, Spyder, and common scientific libraries pre-installed.

1) Download Anaconda from https://www.anaconda.com.
2) Install Anaconda following the on screen instructions.
3) Verify installation using:

```
conda —version
```

Step 3: Install Jupyter Notebook:

You can install Jupyter Notebook using either pip or conda:

Using pip:

```
pip install notebook
```

Using conda (if Anaconda is installed):

```
conda install -c conda-forge notebook
```

To run Jupyter Notebook:

```
jupyter notebook
```

Step 4: Install Spyder IDE:

You can install Spyder IDE using either pip or conda:

Using pip:

```
pip install spyder
```

Using conda (if Anaconda is installed):

```
conda install -c anaconda spyder
```

To run Spyder IDE:

```
spyder
```

Step 5: Install NpmPy and Pandas:

You can install NumPy and Pandas using pip or condo using either of the following commands:

Using pip:

```
pip install numpy pandas
```

Using conda:

```
conda install numpy pandas
```

Step 6: Install Matplotlib and Seaborn:

You can install Matplotlib and Seaborn using the following commands:

Using pip:

```
pip install matplotlib seaborn
```

Using conda:

```
conda install matplotlib seaborn
```

Step 7: Install Scikit-Learn:

You can install Scikit-Learn using the following commands:

Using pip:

```
pip install scikit-learn
```

Using conda:

```
conda install scikit-learn
```