VIVEKANANDA INSTITUTE OF MANAGEMENT

RAJAJINAGAR BANGALORE

MACHINE LEARNING LAB MANUAL.

6TH SEM BCA

Prepared by: NIHARIKA

Under the guidance of Girish Sir

1.Install and set up Python and essential libraries like NumPy and pandas.

a)Installing Python:

i)Download Python:-

Go to the official Python website download the latest version suitable for your operating system (Windows, macOS, or Linux).

ii)Install Python:-

For Windows: Download python software and install it.

iii)Verify Installation:-

- Open a command prompt (Windows)
- python --version

b)Pip upgrade: Using pip:

- Pip is Python's package manager.
- It usually comes installed with Python.
- Open a terminal/command prompt.

Py –m install pip –upgrade pip

c)Install NumPy :

pip install numpy

press Enter. This command will download and install NumPy.

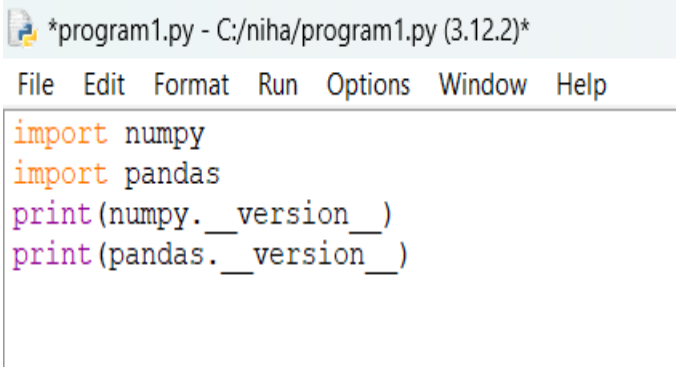
d)Install pandas:

pip install pandas

press Enter. This command will download and install pandas.

e)Verify installations:

Open a IDLE



OUTPUT:

```
1.26.3
2.2.0
```

2) Introduce scikit-learn as a machine learning library.

- Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python.
- It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python.
- This library, which is largely written in Python, is built upon NumPy, pandas,SciPy and Matplotlib.

Installation

If you already installed NumPy and Scipy, the following are the two easiest ways to install scikit-learn –

Using pip

The following command can be used to install scikit-learn via pip

pip install scikit-learn

Features

Rather than focusing on loading, manipulating and summarizing data , Scikit-learn library is focused on modeling the data. Some of the most popular groups of models provided by Sklearn are as follows .

Supervised Learning algorithms − Almost all the popular supervised learning algorithms, like Linear Regression, Support Vector Machine (SVM), Decision Tree etc., are the part of scikit-learn.

Unsupervised Learning algorithms − On the other hand, it also has all the popular unsupervised learning algorithms from clustering, factor analysis, PCA (Principal Component Analysis) to unsupervised neural networks.

Clustering − This model is used for grouping unlabeled data.

Cross Validation − It is used to check the accuracy of supervised models on unseen data.

3) Install and set up scikit-learn and other necessary tools.

Pip upgrade:

Using pip:

- Pip is Python's package manager.
- It usually comes installed with Python.
- Open a terminal/command prompt.

Py –m install pip - -upgrade pip

a)Install NumPy :-

pip install numpy

press Enter. This command will download and install NumPy.

b)Install pandas:-

pip install pandas

press Enter. This command will download and install pandas.

c)Install matplotlib:-

pip install matplotlib

press Enter. This command will download and install matplotlib.

d)Install scipy:-

pip install scipy press Enter. This command will download and install scipy

e)Install scikit-learn(sklearn):-

pip install scikit-learn

press Enter. This command will download and install scikit-learn

Verify installations:

Open a IDLE

program3.py - C:/niha/program3.py (3.12.2)

File Edit Format Run Options Window Help

```
import sklearn
import numpy
import pandas
import matplotlib
print(sklearn.__version__)
print(numpy.__version__)
print(pandas.__version__)
print(matplotlib.__version__)
```

OUTPUT:

```
1.4.2
1.26.3
2.2.0
3.8.2
```

4. Write a program to Load and explore the dataset of .CSV and excel files using pandas.

program4.py - C:/niha/program4.py (3.12.2)

File Edit Format Run Options Window Help

```
import pandas as pd
def explore_dataset(file_path):
# Check if the file is a CSV or Excel file
 if file_path.endswith('.csv'):
# Load CSV file into a pandas DataFrame
  df = pd.read_csv("C:/Users/Dell/iris.csv")
 elif file_path.endswith('.xlsx'):
# Load Excel file into a pandas DataFrame
  df = pd.read_excel("C:/Users/Dell/iris.xlsx")
 else:
  print("Unsupported file format. Please provide a CSV or Excel file.")
  return
# Display basic information about the DataFrame
 print("Dataset information:")
 print(df.info())
# Display the first few rows of the DataFrame
 print("\nFirst few rows of the dataset:")
 print(df.head())
# Display summary statistics for numerical columns
 print("\nSummary statistics:")
 print(df.describe())
# Display unique values for categorical columns
 print("\nUnique values for categorical columns:")
 for column in df.select_dtypes(include='object').columns:
     print(f"{column}: {df[column].unique()}")
# Example usage
file_path = 'iris.csv'
# Change this to the path of your CSV or Excel file
explore_dataset("C:/Users/Dell/iris.csv")
```

OUTPUT:

```
Dataset information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
None

First few rows of the dataset:
   Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm      Species
0   1            5.1           3.5            1.4           0.2  Iris-setosa
1   2            4.9           3.0            1.4           0.2  Iris-setosa
2   3            4.7           3.2            1.3           0.2  Iris-setosa
3   4            4.6           3.1            1.5           0.2  Iris-setosa
4   5            5.0           3.6            1.4           0.2  Iris-setosa

Summary statistics:
              Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
count  150.000000     150.000000    150.000000     150.000000    150.000000
mean    75.500000       5.843333      3.054000       3.758667      1.198667
std     43.445368       0.828066      0.433594       1.764420      0.763161
min      1.000000       4.300000      2.000000       1.000000      0.100000
25%     38.250000       5.100000      2.800000       1.600000      0.300000
50%     75.500000       5.800000      3.000000       4.350000      1.300000
75%    112.750000       6.400000      3.300000       5.100000      1.800000
max    150.000000       7.900000      4.400000       6.900000      2.500000

Unique values for categorical columns:
Species: ['Iris-setosa' 'Iris-versicolor' 'Iris-virginica']
```

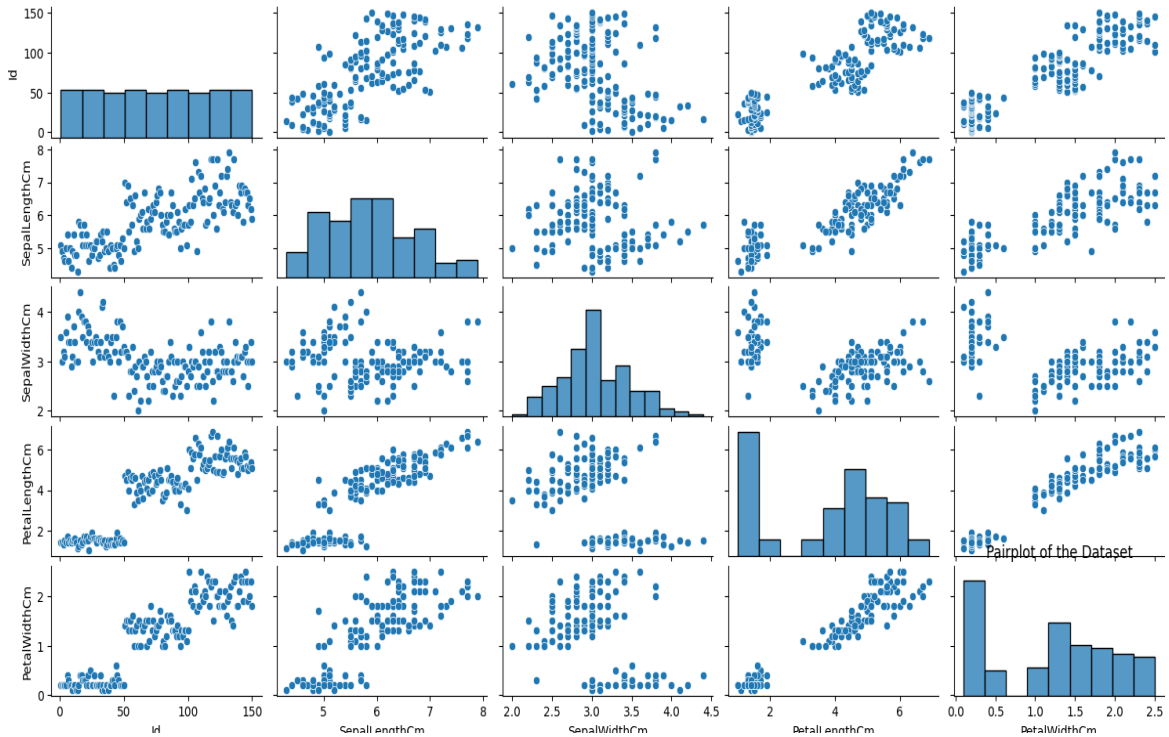
5. Write a program to Visualize the dataset to gain insights using Matplotlib or Seaborn by plotting scatter plots, and bar charts.

program5.py - C:/niha/program5.py (3.12.2)

File Edit Format Run Options Window Help

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
def visualize_dataset(file_path):
# Load the dataset into a pandas DataFrame
 df = pd.read_csv("C:/Users/Dell/iris.csv")
# Assuming it's a CSV file, change accordingly if it's an Excel file
# Plot scatter plots
 sns.pairplot(df)
 plt.title("Pairplot of the Dataset")
 plt.show()
# Plot bar chart for categorical column (assuming the first column is categorical)
 if df.iloc[:, 0].dtype == 'object':
  sns.countplot(x=df.columns[0], data=df)
  plt.title("Bar Chart of Categorical Column")
  plt.xlabel(df.columns[0])
  plt.ylabel("Count")
  plt.show()
 else:
  print("No categorical column found to plot bar chart.")
# Example usage
file_path = 'iris.csv'
visualize_dataset("C:/Users/Dell/iris.csv")
```

OUTPUT:



6. Write a program to Handle missing data, encode categorical variables, and perform feature scaling.

program6.py - C:/niha/program6.py (3.12.2)

File Edit Format Run Options Window Help

```
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder, StandardScaler
# Load Iris dataset
iris = load_iris()
iris_df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
iris_df['target'] = iris.target
def preprocess_dataset(df):
# Handle missing data (Iris dataset doesn't have missing values, but we'll simulate some)
 df.iloc[::10, 0] = float('NaN')
# Simulate missing values in the first column
 imputer = SimpleImputer(strategy='mean')
 df[df.columns] = imputer.fit_transform(df[df.columns])
# Encode categorical variable (if applicable)
# Since Iris dataset doesn't have categorical variables, we'll skip this step
# Perform feature scaling
 scaler = StandardScaler()
 df[df.columns[:-1]] = scaler.fit_transform(df[df.columns[:-1]])
 return df
# Preprocess Iris dataset
preprocessed_df = preprocess_dataset(iris_df)
print("Preprocessed dataset:")
print(preprocessed_df.head())
```

OUTPUT:

```
Preprocessed dataset:
   sepal length (cm)  sepal width (cm)  ...  petal width (cm)  target
0       1.132764e-15          1.019004  ...         -1.315444     0.0
1      -1.196022e+00         -0.131979  ...         -1.315444     0.0
2      -1.451098e+00          0.328414  ...         -1.315444     0.0
3      -1.578636e+00          0.098217  ...         -1.315444     0.0
4      -1.068484e+00          1.249201  ...         -1.315444     0.0

[5 rows x 5 columns]
```

7. Write a program to implement a k-Nearest Neighbours (k-NN) classifier using scikit-learn and Train the classifier on the dataset and evaluate its performance.

program7.py - C:/niha/program7.py (3.12.2)

File Edit Format Run Options Window Help

```
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report
# Load Iris dataset
iris = load_iris()
X = iris.data
y = iris.target
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Initialize the k-NN classifier
k = 3 # Number of neighbors
knn_classifier = KNeighborsClassifier(n_neighbors=k)
# Train the classifier
knn_classifier.fit(X_train, y_train)
# Make predictions on the testing set
y_pred = knn_classifier.predict(X_test)
# Evaluate the classifier's performance
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
# Display classification report
print("Classification Report:")
print(classification_report(y_test, y_pred, target_names=iris.target_names))
```

OUTPUT:

```
Accuracy: 1.0
Classification Report:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        10
  versicolor       1.00      1.00      1.00         9
   virginica       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
```

8. Write a program to implement a linear regression model for regression tasks and Train the model on a dataset with continuous target variables.

program8.py - C:/niha/program8.py (3.12.2)

File Edit Format Run Options Window Help

```
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
# Load iris dataset
iris= load_iris()
X = iris.data
y = iris.target
# Convert the data to a pandas DataFrame for easier manipulation
iris_df = pd.DataFrame(data=X, columns=iris.feature_names)
iris_df['target'] = y
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
# Initialize Linear Regression model
linear_regression = LinearRegression()
# Train the model
linear_regression.fit(X_train, y_train)
# Make predictions on the testing set
y_pred = linear_regression.predict(X_test)
# Evaluate the model's performance
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print("Mean Squared Error:", mse)
print("R-squared Score:", r2)
```

OUTPUT:

```
Mean Squared Error: 0.03711379440797689
R-squared Score: 0.9468960016420045
```

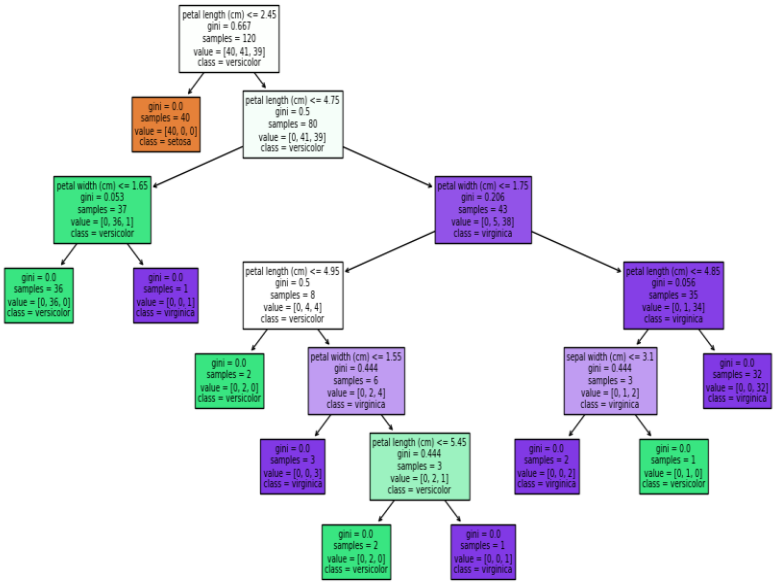
9. Write a program to implement a decision tree classifier using scikit-learn and visualize the decision tree and understand its splits.

program9.py - C:/niha/program9.py (3.12.2)

File Edit Format Run Options Window Help

```
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
import matplotlib.pyplot as plt
# Load Iris dataset
iris = load_iris()
X = iris.data
y = iris.target
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Initialize Decision Tree classifier
decision_tree = DecisionTreeClassifier()
# Train the classifier
decision_tree.fit(X_train, y_train)
# Visualize the decision tree
plt.figure(figsize=(12, 8))
plot_tree(decision_tree, feature_names=iris.feature_names, class_names=iris.target_names, filled=True)
plt.show()
```

OUTPUT:



10. Write a program to Implement K-Means clustering and Visualize clusters.

program10.py - C:/niha/program10.py (3.12.2)

File Edit Format Run Options Window Help

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
# Generate sample data
X, y = make_blobs(n_samples=500, centers=4, cluster_std=0.8, random_state=42)
# Create a K-Means clusterer with 4 clusters
kmeans = KMeans(n_clusters=4, random_state=42)
# Fit the data
kmeans.fit(X)
# Get cluster labels
labels = kmeans.labels_
# Plot the data with cluster labels
plt.figure(figsize=(8, 6))
plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='viridis')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s=100,
c='red', label='Centroids')
plt.title('K-Means Clustering')
plt.xlabel('X')
plt.ylabel('Y')
plt.legend()
plt.show()
```

OUTPUT: