Machine learning LAB-05

Perumalla Dharan
AP21110010201

1)

Distance functions

Euclidean $$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$

Manhattan $$\sum_{i=1}^{k}|x_i - y_i|$$

Minkowski $$\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$$

```
# Take two 3D coordinates from the user. Find out the
distance between these points

# x and y are vectors representing the coordinates of two
points.
# Here, k=3

import pandas as pd
from scipy.spatial import distance

# Load the Iris dataset
file_path = r'E:\SRM\Machine Learning\Lab\Lab-5\iris.csv'
iris_data = pd.read_csv(file_path)

# Select two random rows (data points) from the dataset
```

```python
point1 = iris_data.sample(1, random_state=42).iloc[0,
:-1].values  # Exclude the last column (species)
point2 = iris_data.sample(1, random_state=99).iloc[0,
:-1].values  # Exclude the last column (species)

# Calculate Manhattan distance
manhattan_dist = distance.cityblock(point1, point2)

# Calculate Euclidean distance
euclidean_dist = distance.euclidean(point1, point2)

# Calculate Minkowski distance (with p=3)
minkowski_dist = distance.minkowski(point1, point2, p=3)

# Print the calculated distances
print(f"Manhattan distance: {manhattan_dist}")
print(f"Euclidean distance: {euclidean_dist}")
print(f"Minkowski distance (p=3): {minkowski_dist}")
```

Output-

```
PS E:\SRM\Machine Learning> python -u "e:\SRM\
Manhattan distance: 2.7000000000000001
Euclidean distance: 1.4247806848775015
Minkowski distance (p=3): 1.1722813257645157
PS E:\SRM\Machine Learning>
```

---

2)

```python
# Import the Iris dataset. Write a program to obtain the
Euclidian Distance Matrix for
# all the data samples in the feature space. Distance
metric is a 2D array, where the
```

```
# (i,j)th entry represents the distance between the ith and
jth sample points in the feature
# space.

import pandas as pd
from scipy.spatial.distance import pdist, squareform

# Load the Iris dataset
file_path = r'E:\SRM\Machine Learning\Lab\Lab-5\iris.csv'
iris_data = pd.read_csv(file_path)

# Extract features (attributes) from the dataset
X = iris_data.iloc[:, :-1]  # Exclude the last column
(species)

# Calculate pairwise Euclidean distances
distances = pdist(X, metric='euclidean')

# Convert pairwise distances to a square distance matrix
euclidean_distance_matrix = squareform(distances)

# Print the Euclidean Distance Matrix
print("Euclidean Distance Matrix:")
print(euclidean_distance_matrix)
```

Output-

```
PS E:\SRM\Machine Learning> python -u "e:\SRM\Machine Learning\Lab\Lab-5\ques
Euclidean Distance Matrix:
[[0.         0.53851648 0.50990195 ... 4.45982062 4.65080638 4.14004831]
 [0.53851648 0.         0.3        ... 4.49888875 4.71805044 4.15331193]
 [0.50990195 0.3        0.         ... 4.66154481 4.84871117 4.29883705]
 ...
 [4.45982062 4.49888875 4.66154481 ... 0.         0.6164414  0.64031242]
 [4.65080638 4.71805044 4.84871117 ... 0.6164414  0.         0.76811457]
 [4.14004831 4.15331193 4.29883705 ... 0.64031242 0.76811457 0.        ]]
PS E:\SRM\Machine Learning>
```

---

3)

```
# Import the Iris dataset. Prepare a dataset considering
samples belong to any two
# output classes. Draw the scatter plot for all the samples
in the new dataset considering
# any two input attributes. Examine the scatter plot to
find the equation of a line that
# can separate sample of two classes.

import pandas as pd
import matplotlib.pyplot as plt

# Read the Iris dataset from CSV
iris_df = pd.read_csv('E:\SRM\Machine
Learning\Lab\Lab-5\iris.csv')

# Select two classes from the dataset
class1 = 'setosa'
class2 = 'versicolor'

# Filter the dataset to include only the selected classes
selected_df = iris_df[(iris_df['Species'] == class1) |
(iris_df['Species'] == class2)]
```

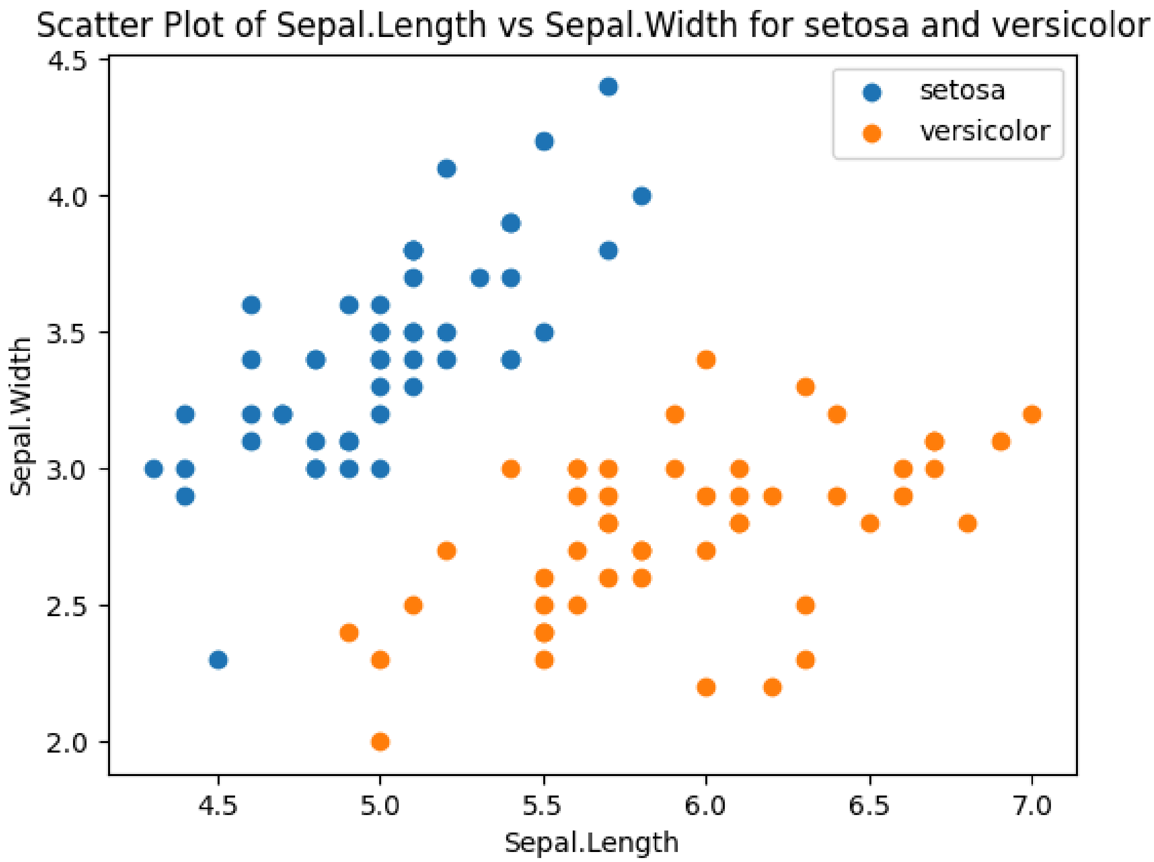
```python
# Select two input attributes for scatter plot
attribute1 = 'Sepal.Length'
attribute2 = 'Sepal.Width'

# Plot the scatter plot for the selected classes and
attributes
plt.scatter(selected_df[selected_df['Species'] ==
class1][attribute1],
            selected_df[selected_df['Species'] ==
class1][attribute2],
            label=class1)

plt.scatter(selected_df[selected_df['Species'] ==
class2][attribute1],
            selected_df[selected_df['Species'] ==
class2][attribute2],
            label=class2)

plt.xlabel(attribute1)
plt.ylabel(attribute2)
plt.title(f'Scatter Plot of {attribute1} vs {attribute2}
for {class1} and {class2}')
plt.legend()
plt.show()
```

Output-


Scatter Plot of Sepal.Length vs Sepal.Width for setosa and versicolor
setosa
versicolor
Sepal.Width
Sepal.Length
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0