Name : Vaishnavi | Roll No. : 74
Course: BCA (B1) | Semester: VI
Subject: Fundamentals of Machine Learning Lab | Course Code: PBC 602

9. Uploading an excel/csv data file (containing Student_ID, Student_Name, Gender, Sub1, Sub2, Sub3 with the marks of 30 students). Perform the following tasks:

1) Check for missing values, and replace them with suitable replacement.
2) Create two DataFrames containg Student_ID, Student_Name of male and female students.
3) Add a new column in the DataFrame 'Percentage' showing total percentage of each student.
4) Normalizing the marks of each subject.
5) Draw a bar diagram showing number of male and female students in the class.
6) Draw a pie chart showing the number of students having percentage (a) > = 60 (b) > =50 and < 60 (c) < 50

9.1. Python Program:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler

df = pd.read_excel("student_data.xlsx")
df.fillna(df.mean(numeric_only=True), inplace=True)
male_students = df[df["Gender"] == "Male"][["Student_ID",
"Student_Name"]]
female_students = df[df["Gender"] == "Female"][["Student_ID",
"Student_Name"]]
df["Total"] = df[["Sub1", "Sub2", "Sub3"]].sum(axis=1)
df["Percentage"] = df["Total"] / 3
scaler = MinMaxScaler()
df[["Sub1", "Sub2", "Sub3"]] = scaler.fit_transform(df[["Sub1", "Sub2",
"Sub3"]])
gender_counts = df["Gender"].value_counts()
gender_counts.plot(kind='bar', color=['skyblue', 'lightpink'])
plt.title("Number of Male and Female Students")
plt.xlabel("Gender")
```

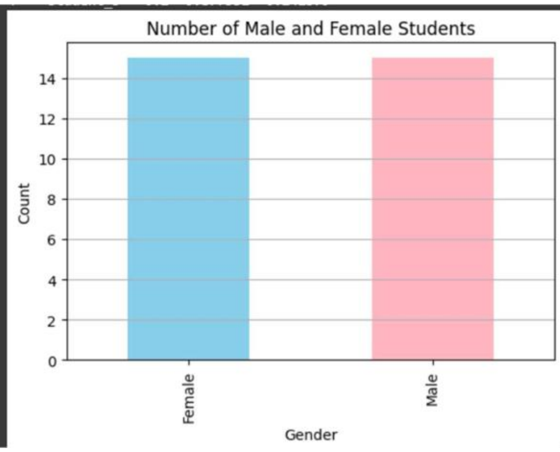
```
plt.ylabel("Count")
plt.grid(axis='y')
plt.show()
def percentage_category(p):
    if p >= 60:
        return ">=60"
    elif p >= 50:
        return "50-59"
    else:
        return "<50"
df["Category"] = df["Percentage"].apply(percentage_category)
category_counts = df["Category"].value_counts()
category_counts.plot(kind="pie", autopct='%1.1f%%', startangle=90,
colors=['lightgreen', 'orange', 'red'])
plt.title("Percentage Category Distribution")
plt.ylabel("")
plt.show()
```

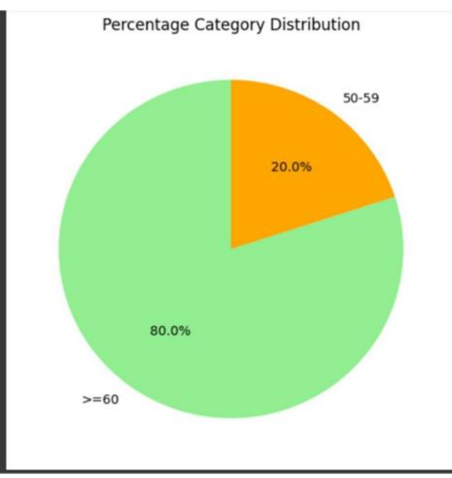
9.2. Output:

```
Original DataFrame:
   Student_ID Student_Name  Gender  Sub1  Sub2  Sub3
0           1    Student_1  Female    55    56    59
1           2    Student_2    Male    70    67    68
2           3    Student_3  Female    70    78    77
3           4    Student_4    Male    85    77    58
4           5    Student_5  Female    50    88    57

Missing values before replacement:
Student_ID      0
Student_Name    0
Gender          0
Sub1            0
Sub2            0
Sub3            0
dtype: int64
```

```
Missing values after replacement:
Student_ID      0
Student_Name    0
Gender          0
Sub1            0
Sub2            0
Sub3            0
dtype: int64

Male Students:
   Student_ID Student_Name
1           2    Student_2
3           4    Student_4
5           6    Student_6
7           8    Student_8
9          10   Student_10
```

```
Female Students:
   Student_ID Student_Name
0           1    Student_1
2           3    Student_3
4           5    Student_5
6           7    Student_7
8           9    Student_9

DataFrame with Percentage:
  Student_Name  Total  Percentage
0    Student_1    170   56.666667
1    Student_2    205   68.333333
2    Student_3    225   75.000000
3    Student_4    220   73.333333
4    Student_5    195   65.000000

Normalized Subjects:
  Student_Name  Sub1      Sub2      Sub3
0    Student_1   0.3  0.224490  0.310345
1    Student_2   0.6  0.448980  0.620690
2    Student_3   0.6  0.673469  0.931034
3    Student_4   0.9  0.653061  0.275862
4    Student_5   0.2  0.877551  0.241379
```





Name : Vaishnavi | Roll No. : 74
Course: BCA (B1) | Semester: VI
Subject: Fundamentals of Machine Learning Lab | Course Code: PBC 602

10. Create a .txt file in your directory with three lines as follows: Hi how are you? I am fine. I hope that you are also fine.

1) Display the content of the file as a string.
2) Display each line as an element of a list. III. Display the number of characters in the file.
3) Number of characters in first line.
4) 2nd to 5th characters of second last line. VI. Rename the file VII. Delete the file

10.1. Python Program:

```
import os
with open("myfile.txt", "w") as f:
    f.write("Hi how are you?\n")
    f.write("I am fine.\n")
    f.write("I hope that you are also fine.\n")
with open("myfile.txt", "r") as f:
    content = f.read()
with open("myfile.txt", "r") as f:
    lines = f.readlines()
char_count = len(content)
first_line_chars = len(lines[0])
if len(lines) >= 2:
    second_last_line = lines[-2]
    print(second_last_line[1:5])
os.rename("myfile.txt", "renamed_file.txt")
os.remove("renamed_file.txt")
```

10.2. Output:

```
File Content as String:
 Hi how are you?
I am fine.
I hope that you are also fine.


Lines as List:
 ['Hi how are you?\n', 'I am fine.\n', 'I hope that you are also fine.\n']

Total Characters in File: 58
Characters in First Line: 16
2nd to 5th characters of 2nd last line:  am

File renamed to 'renamed_file.txt'.
File deleted.
```

11.Write a program to implement the k-means clustering algorithm by taking a suitable dataset.

11.1. Algorithm:

1) Load dataset.
2) Standardize the data (op onal but recommended).
3) Choose the number of clusters (k).
4) Ini alize centroids randomly.
5) Repeat un l convergence:
   - Assign each data point to the nearest centroid.
   - Recalculate centroids as the mean of assigned points. 6) Output cluster labels.

11.2. Python Program:

```
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import load_iris

iris = load_iris()
iris_df = pd.DataFrame(iris.data, columns=iris.feature_names)
scaler = StandardScaler()
scaled_data = scaler.fit_transform(iris_df)
kmeans = KMeans(n_clusters=3, random_state=42)
clusters = kmeans.fit_predict(scaled_data)
iris_df['cluster'] = clusters
print(kmeans.cluster_centers_)
```

11.3. Output:

```
Cluster Centers:
 [[ 0.57100359 -0.37176778  0.69111943  0.66315198]
 [-0.81623084  1.31895771 -1.28683379 -1.2197118 ]
 [-1.32765367 -0.373138   -1.13723572 -1.11486192]]
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2

   cluster
0        1
1        2
2        2
3        2
4        1
```

12. Write a Program to implement the DBSCAN algorithm.

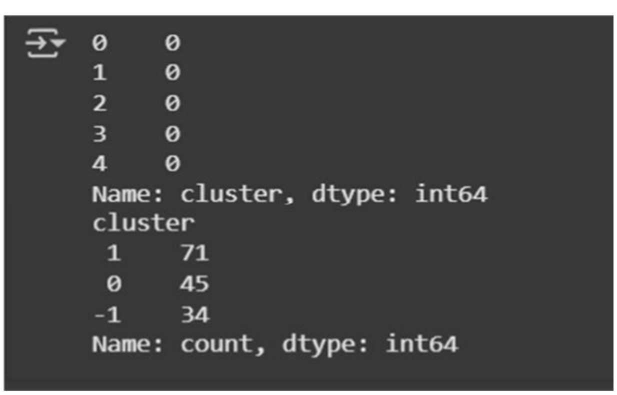
12.1. Algorithm:

1) Loads the Iris dataset.
2) Converts it into a pandas DataFrame.
3) Standardizes the features using StandardScaler (important for DBSCAN).
4) Applies DBSCAN with eps=0.5 and min_samples=5.
5) Adds the predicted cluster labels back to the DataFrame.
6) Prints the first 5 cluster labels.

12.2. Python Program:

```
import pandas as pd from sklearn.cluster
import DBSCAN from
sklearn.preprocessing import
StandardScaler import numpy as np
from sklearn.datasets import load_iris
iris = load_iris()
iris_df = pd.DataFrame(iris.data, columns=iris.feature_names)
scaler = StandardScaler()
scaled_data = scaler.fit_transform(iris_df)
dbscan = DBSCAN(eps=0.5, min_samples=5)
clusters = dbscan.fit_predict(scaled_data)
iris_df['cluster'] = clusters
print(iris_df['cluster'].head())
print(iris_df['cluster'].value_counts())
```

12.3. Output:



```
0    0
1    0
2    0
3    0
4    0
Name: cluster, dtype: int64
cluster
 1    71
 0    45
-1    34
Name: count, dtype: int64
```

Name : Vaishnavi Roll No. : 74
Course: BCA (B1) Semester: VI
Subject: Fundamentals of Machine Learning Lab Course Code: PBC 602

13. Write a program to implement low variance filter.

13.1. Algorithm:

1) Loads the Iris dataset from the UCI repository.
2) Reads it into a DataFrame with no header.
3) Splits features (X) and labels (y).
4) Applies VarianceThreshold to remove low-variance features (threshold = 0.5).
5) Selects high-variance features based on the threshold.
6) Creates a new DataFrame with only selected features. 7) Displays the first 5 rows of the filtered data.

13.2. Python Program:

```
import pandas as pd
from sklearn.feature_selec on import VarianceThreshold
from sklearn.datasets import load_iris
iris = load_iris(as_frame=True)
iris_df = iris.frame
iris_df['class'] =
iris.target
feature_columns = iris.feature_names
selector = VarianceThreshold(threshold=0.5)
selector.fit(iris_df[feature_columns])
selected_features = [feature_columns[i]
for i in
selector.get_support(indices=True)]
print("Selected
Features:",selected_features)
filtered_iris_df = iris_df[selected_features + ['class']]
print(filtered_iris_df.head())
```

13.3. Output:

```
Selected Features: ['sepal length (cm)', 'petal length (cm)', 'petal width (cm)']
   sepal length (cm)  petal length (cm)  petal width (cm)  class
0                5.1                1.4               0.2      0
1                4.9                1.4               0.2      0
2                4.7                1.3               0.2      0
3                4.6                1.5               0.2      0
4                5.0                1.4               0.2      0
```

Name : Vaishnavi | Roll No. : 74
Course: BCA (B1) | Semester: VI
Subject: Fundamentals of Machine Learning Lab | Course Code: PBC 602

14. Write a program to implement the single linkage hierarchical clustering method.

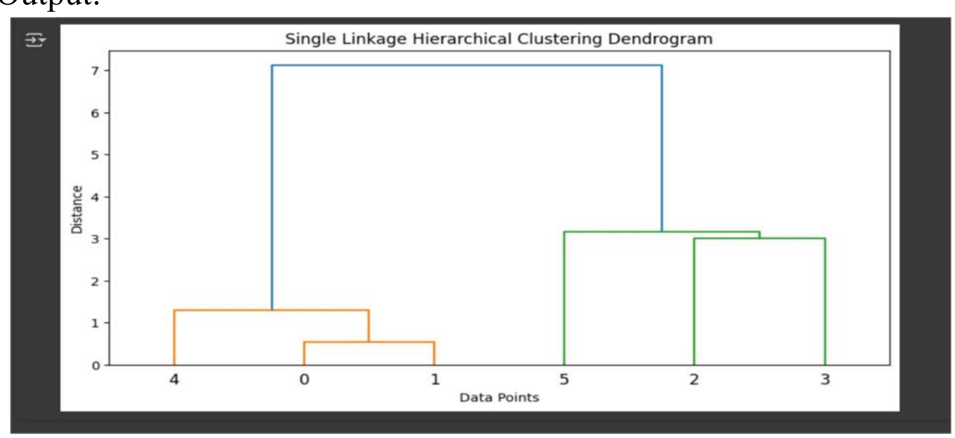
14.1. Algorithm:

1) Start with each point as its own cluster.
2) Calculate distances between all clusters.
3) Merge the closest clusters based on the smallest distance.
4) Repeat un l all points form one cluster.
5) Create a dendrogram to visualize the process.

14.2. Python Program:

```
import numpy as np
from scipy.cluster.hierarchy import linkage, dendrogram
import matplotlib.pyplot as plt
def single_linkage_clustering(data):
  linkage_matrix = linkage(data, method='single')
  plt.figure(figsize=(10, 5))
dendrogram(linkage_matrix)
  plt. tle('Single Linkage Hierarchical Clustering Dendrogram')
  plt.xlabel('Data Points')
plt.ylabel('Distance')
  plt.show()
  return linkage_matrix
data = np.array([[1, 2], [1.5, 1.8], [5, 8], [8, 8], [1, 0.6], [9, 11]])
linkage_matrix = single_linkage_clustering(data)
```

14.3. Output:



15. WAP to implement linear regression.

15.1. Algorithm:

1) Load dataset (e.g., experience vs salary).
2) Split the dataset into training and test sets.
3) Fit a linear model using the training data.
4) Predict values for the test data.
5) Evaluate using Mean Squared Error or similar metric.
6) Output the predicted values and performance.

15.2. Python Program:

```
from sklearn.linear_model import LinearRegression from
sklearn.model_selec on import train_test_split from
sklearn.metrics import mean_squared_error import
pandas as pd
data = {'Experience': [1, 2, 3, 4, 5], 'Salary': [10000, 20000, 30000,
40000,
50000]}
df = pd.DataFrame(data)
X = df[['Experience']]
y = df['Salary']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test) print("Predic
ons:", y_pred)
print("MSE:", mean_squared_error(y_test, y_pred))
```

15.3. Output:

```
Predictions: [50000.]
MSE: 0.0
```

Name : Vaishnavi Roll No. : 74
Course: BCA (B1) Semester: VI
Subject: Fundamentals of Machine Learning Lab Course Code: PBC 602

16. WAP to implement Decision Tree algorithm.

16.1. Algorithm:

1) Load dataset (e.g., Iris dataset).
2) Split data into training and test sets.
3) Train Decision Tree classifier on the training data.
4) Predict labels for test data.
5) Evaluate accuracy of the model.
6) Output predicted labels and accuracy score.

16.2. Python Program:

```
from sklearn.tree import DecisionTreeClassifier from
sklearn.datasets import load_iris from
sklearn.model_selec on import train_test_split
from sklearn.metrics import accuracy_score

iris = load_iris()
X, y = iris.data, iris.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
```

16.3. Output:

```
Accuracy: 0.9555555555555556
```

Name : Vaishnavi Roll No. : 74
Course: BCA (B1) Semester: VI
Subject: Fundamentals of Machine Learning Lab Course Code: PBC 602

17. WAP to implement Naive Bayes algorithm.

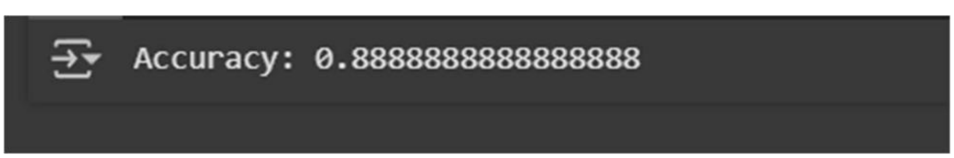
17.1. Algorithm:

1) Load dataset (e.g., Iris dataset).
2) Split data into training and tes ng sets.
3) Train Naive Bayes model (e.g., GaussianNB) on training data.
4) Predict class labels on test data.
5) Calculate accuracy score.
6) Output predicted results and model performance.

17.2. Python Program:

```
from sklearn.naive_bayes import GaussianNB from
sklearn.datasets import load_iris from
sklearn.model_selec on import train_test_split
from sklearn.metrics import accuracy_score

iris = load_iris() X, y =
iris.data, iris.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

model = GaussianNB()
model.fit(X_train, y_train)

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

17.3. Output:



18. WAP to implement SVM.

18.1. Algorithm:

1) Load dataset (e.g., Iris dataset).
2) Split data into training and test sets.
3) Train SVM classifier using the training set.
4) Make predic ons on the test data.
5) Evaluate using accuracy or other metrics.
6) Output the results and performance.

18.2. Python Program: from sklearn import svm from sklearn.datasets import load_iris from sklearn.model_selec on import train_test_split

```
from sklearn.metrics import accuracy_score

iris = load_iris()
X, y = iris.data, iris.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

clf = svm.SVC()
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
```

18.3. Output:

```
Accuracy: 0.9333333333333333
```

Name : Vaishnavi Roll No. : 74
Course: BCA (B1) Semester: VI
Subject: Fundamentals of Machine Learning Lab Course Code: PBC 602

19. WAP to Implement principal component analysis (PCA).

19.1. Algorithm:

1) Load dataset (e.g., Iris dataset).
2) Standardize features to have zero mean and unit variance.
3) Compute covariance matrix of the data.
4) Find eigenvectors and eigenvalues of the covariance matrix.
5) Sort eigenvectors by largest eigenvalues to get principal components.
6) Project original data onto top N principal components.
7) Visualize reduced data (usually 2D or 3D).
8) Output transformed data and possibly a plot.

19.2. Python Program:

```
from sklearn.decomposi on import PCA from sklearn.datasets
import load_iris
import matplotlib.pyplot as plt

iris = load_iris() X = iris.data y =
iris.target
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)
plt.figure(figsize=(8,6))
sca er = plt.sca er(X_pca[:,0], X_pca[:,1], c=y, cmap='viridis') plt. tle('PCA on Iris
Dataset')
plt.xlabel('PC 1') plt.ylabel('PC 2')
plt.legend(*sca er.legend_elements(), tle="Classes") plt.show()
```

Name : Vaishnavi | Roll No. : 74
Course: BCA (B1) | Semester: VI
Subject: Fundamentals of Machine Learning Lab | Course Code: PBC 602

19.3. Output: