Experiment No 12

* 1. **Aim/Purpose of the Experiment**

To familiarize the students with model evaluation methods using Confusion Matrix for K-Means Clustering algorithm.

* 1. **Learning Outcomes**

Knowledge of the model evaluation methods using Confusion Matrix for K-Means clustering.in python.

* 1. **Prerequisites**

Basic knowledge of programming, python syntax, matplotlib, seaborn, different libraries.

* 1. **Materials/Equipment/Apparatus / Devices/Software required**

Jupyter Notebook.

* 1. **Introduction and Theory**

Importing Libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

import warnings

warnings.filterwarnings('ignore')

Importing Data

df = pd.read\_csv('College\_Data.csv')

Check the head of the data

df.head(3)

df.info()

df.isnull().sum()

df.describe()

EDA

sns.set\_style('darkgrid')

sns.lmplot('Room.Board','Grad.Rate',data=df, hue='Private',palette='magma\_r',size=6,aspect=1,fit\_reg=False)

scatterplot of Grad.Rate versus Room.Board where the points are colored by the Private column.

sns.set\_style('darkgrid')

sns.lmplot('Outstate','F.Undergrad',data=df, hue='Private',palette='magma\_r',size=6,aspect=1,fit\_reg=False)

scatterplot of F.Undergrad versus Outstate where the points are colored by the Private column.

sns.set\_style('dark')

h = sns.FacetGrid(df,hue="Private",palette='coolwarm',size=6,aspect=2)

h = h.map(plt.hist,'Outstate',bins=20,alpha=0.7)

A stacked histogram showing Out of State Tuition based on the Private column.

sns.set\_style('dark')

g = sns.FacetGrid(df,hue="Private",palette='coolwarm',size=6,aspect=2)

g = g.map(plt.hist,'Grad.Rate',bins=20,alpha=0.7)

A histogram for the Grad.Rate column.

We can see there seems to be a private school with a graduation rate of higher than 100%.

we need to set that school's graduation rate to 100 so it makes sense. we get a warning not an (error) , so we use dataframe operations and check the histogram visualization to make sure it actually changed.

df.set\_value(95, 'Grad.Rate', 100)

df[df['Grad.Rate'] > 100]

sns.set\_style('darkgrid')

g = sns.FacetGrid(df,hue="Private",palette='coolwarm',size=6,aspect=2)

g = g.map(plt.hist,'Grad.Rate',bins=20,alpha=0.7)

df=df.drop(['NameofInstitution',],axis=1)

Import KMeans from SciKit Learn.

from sklearn.cluster import KMeans

kmeans = KMeans(n\_clusters=2)

Fit the model to all the data except for the Private label.

kmeans.fit(df.drop('Private',axis=1))

The cluster center vectors for unlabeled data

means=kmeans.cluster\_centers\_

print(means)

Evaluation

There is no perfect way to evaluate clustering if we don't have the labels, however, we do have the labels, so we take advantage of this to evaluate our clusters.

Create a new column for df called 'Cluster', which is a 1 for a Private school, and a 0 for a public school.

def converter(cluster):

if cluster=='Yes':

return 1

else:

return 0

df['Cluster'] = df['Private'].apply(converter)

df.head(3)

df.Private.value\_counts()

Creating a confusion matrix and classification report to see how well the Kmeans clustering worked without being given any labels.

from sklearn.metrics import confusion\_matrix,classification\_report

print(confusion\_matrix(df['Cluster'],kmeans.labels\_))

print(classification\_report(df['Cluster'],kmeans.labels\_))

* 1. **Operating Procedure**
* Open Jupyter note book
* Take a new python file
* Type the code
* Run it
* Take inputs from user
* Observe the results
* Verify the results manually
* Store the note book file
  1. **Precautions and/or Troubleshooting**

**Precautions:**

* Save Your Work: Regularly save your Jupyter Notebook to avoid losing your work. You can save your notebook by clicking on the save icon or using the keyboard shortcut Ctrl + S (or Cmd + S on Mac).
* Restart Kernel: If you encounter unexpected behavior or errors, try restarting the kernel. This clears all the variables and imported modules, essentially resetting the notebook's state. You can restart the kernel by going to the "Kernel" menu and selecting "Restart."
* Clear Outputs: To reduce clutter and confusion, consider clearing the outputs of code cells that are no longer relevant. You can do this by selecting "Clear Outputs" from the "Edit" menu.
* Readability: Keep your code and comments clear and well-organized to make it easier to understand and maintain. Use markdown cells for explanations, headings, and documentation.
* Check Dependencies: If you're using external libraries or packages, ensure they are properly installed in your Jupyter environment. You can check the installed packages by running !pip list or !conda list in a code cell.
* Kernel Selection: Make sure you're using the correct kernel for your notebook. The kernel determines the programming language and environment in which your code runs. You can change the kernel by clicking on "Kernel" > "Change kernel" in the menu.
* Resource Usage: Be mindful of the resources your notebook is using, especially if you're working with large datasets or running intensive computations. Check system monitor tools to ensure you're not exhausting memory or CPU resources.

**Troubleshooting:**

* Syntax Errors: Check for syntax errors in your code. Python is sensitive to indentation and syntax, so ensure your code is properly formatted.
* Variable Scope: Be aware of variable scope issues, especially if you're reusing variable names or working with nested functions.
* Library Installation: If you encounter Module Not Found Error or similar errors, ensure that the required libraries are installed in your Jupyter environment. You can install libraries using !pip install <library> or !conda install <library> in a code cell.
* Kernel Crashes: If the kernel crashes frequently, consider reducing the complexity of your code or optimizing resource usage. Large datasets or intensive computations can sometimes overwhelm the kernel.
* Browser Issues: If you experience rendering or responsiveness issues in the notebook interface, try clearing your browser cache or using a different browser.
* Documentation: Consult the official Jupyter documentation and community forums for additional troubleshooting tips and solutions to common problems.
  1. **Observations**

Observe the results obtained in each operation.

* 1. **Calculations & Analysis**

Calculations should be given for each operation.

* 1. **Result & Interpretation**

Result should be printed and pasted in laboratory copy found from Jupyter note book.

* 1. **Follow-up Questions**
  + What is K means Clustering Algorithm?
  + Is Feature Scaling required for the K means Algorithm?
  + Why do you prefer Euclidean distance over Manhattan distance in the K means Algorithm?
  + Which metrics can you use to find the accuracy of the K means Algorithm?
  + What is a centroid point in K means Clustering?
  + Does centroid initialization affect K means Algorithm?
  + What are the advantages and disadvantages of the K means Algorithm?
  1. **Extension and Follow-up Activities (if applicable)**

NA

* 1. **Assessments**
  2. **Suggested reading**

NA