**KMeans and DBscan Clustering**

**Instructions:**

Please share your answers wherever applicable in line with the word document. Submit code separately wherever applicable.

Please ensure you update all the details:

**Name: Vishvash C Batch ID: 23012024**

**Topic: KMeans and DBscan Clustering**

**Guidelines:**

**1. An assignment submission is considered complete only when correct and executable code(s) is submitted along with the documentation, explaining the method and results. Failing to submit either of those will be considered an invalid submission and will not be considered a correct submission.**

**2. Ensure that you submit your assignments correctly and in full. Resubmission is not allowed.**

**3. Post the submission you can evaluate your work by referring to the keys provided. (Will be available only post the submission).**

**Hints:**

**1. Business Problem**

* 1. **What is the business objective?**
  2. **What are the constraints?**
  3. **Define success criteria**

**2. Work on each feature of the dataset to create a data dictionary as displayed in the below image:**

Table

Description automatically generated

**3. Exploratory Data Analysis (EDA):**

**3.1. Univariate analysis.**

**3.2. Bivariate analysis.**

**4. Data Pre-processing**

**4.1 Data Cleaning, Feature Engineering, etc.**

**5. Model Building**

**5.1 Build the model on the scaled data (try multiple options).**

**5.2 Perform the KMeans and DBscan clustering and find out the best model that minimizes Within the Sum of Squares. Compare the result with Hierarchical Clustering methods.**

**5.3 Validate the clusters (try with the different numbers of clusters), label the clusters, and derive insights (compare the results from multiple approaches).**

**6. Write about the benefits/impact of the solution - in what way does the business (client) benefit from the solution provided?**

**7. Deploy the best model using Python Flask on the local machine.**

**Problem Statements:**

Global air travel has seen an upward trend in recent times. The maintenance of operational efficiency and maximizing profitability are crucial for airlines and airport authorities. Businesses need to optimize airline and terminal operations to enhance passenger satisfaction, improve turnover rates, and increase overall revenue.

The airline companies with the available data want to find an opportunity to analyze and understand travel patterns, customer demand, and terminal usage.

**CRISP-ML(Q) process model describes six phases:**

1. Business and Data Understanding

2. Data Preparation

3. Model Building

4. Model Evaluation

5. Deployment

6. Monitoring and Maintenance

**Objective**: Maximize the Sales

**Constraints**: Minimize the Customer Retention

**Success Criteria:**

Business Success Criteria: Increase the Sales by 10% to 12% by targeting cross-selling opportunities on current customers.

ML Success Criteria: Achieve a Silhouette coefficient of at least 0.6

Economic Success Criteria: The insurance company will see an increase in revenues by at least 8%

Data: Refer to the ‘AirTraffic\_Passenger\_Statistics.csv’ dataset.

**Questions to Trigger Your Thoughts:**

Q1. What is sklearn.preprocessing mostly used for?

sklearn.preprocessing is mostly used for data preprocessing tasks like scaling, normalization, encoding categorical variables, etc.

Q2. What are three scaling functions that can be imported from sklearn.preprocessing?

Three scaling functions from sklearn.preprocessing are StandardScaler, MinMaxScaler, and RobustScaler.

Q3. What library does the groupby function belong to?

The groupby function belongs to the pandas library.

Q4. How to save an output file?

You can save an output file using various methods like using the .to\_csv() function for CSV files, .to\_excel() for Excel files, or using file handling methods like open() and write().

Q5. How to perform scaling without using inbuilt functions?

Scaling without using inbuilt functions can be done manually by subtracting the mean and dividing by the standard deviation for standardization or by subtracting the minimum and dividing by the range for min-max scaling.

Q6. What is the function used to reverse (inverse) the scaling function?

The function used to reverse (inverse) the scaling function is inverse\_transform().

Q7. How are info() and describe() functions different?

The info() function provides a concise summary of a DataFrame including the data types and memory usage, while the describe() function gives descriptive statistics of numerical columns in the DataFrame.

Q8. How to write the X-axis label?

To write the X-axis label, you can use plt.xlabel('label\_name') if you're using Matplotlib.

Q9. How to visually see more row labels when they are overlapped by one another?

To visually see more row labels without overlapping, you can rotate the labels using plt.xticks(rotation='vertical') in Matplotlib.

Q10. Why plot elbow or scree plot to define the number of clusters?

Plotting an elbow or scree plot helps in defining the number of clusters by identifying the point where the rate of decrease of inertia (sum of squared distances from each point to its assigned cluster center) sharply changes, indicating an optimal number of clusters.

Q11. What inbuilt Python function is used to label the clusters?

The inbuilt Python function used to label the clusters depends on the clustering algorithm used. For K-means clustering, labels are assigned using the .labels\_ attribute of the KMeans object.

Q12. Is it important to scale data before clustering it?

Yes, it is important to scale data before clustering to ensure that variables with larger scales do not dominate the distance metrics used in clustering algorithms.

Q13. Which libraries are used to implement DBscan clustering?

Libraries used to implement DBSCAN clustering include scikit-learn (sklearn) and other libraries like HDBSCAN.

Q14. What is the DBscan algorithm?

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a clustering algorithm that groups together points that are closely packed together based on a density criterion.

Q15. What are the parameters used in DBscan?

Parameters used in DBSCAN include epsilon (eps) which defines the radius of the neighborhood around a point and min\_samples which specifies the minimum number of points required to form a dense region.

Q16. How to perform cluster evaluation? Which are the techniques used for cluster evaluation?

Cluster evaluation can be performed using techniques like silhouette score, Davies-Bouldin index, and visual inspection of cluster plots.

Q17. Which attribute gives labels for the clusters?

The attribute that gives labels for the clusters depends on the clustering algorithm used. In scikit-learn, it's typically the .labels\_ attribute for many clustering algorithms.

Q18. How to visualize the clusters?

Clusters can be visualized using scatter plots where each cluster is represented by a different color or shape.

Q19. What are the files required for flask deployment?

Files required for Flask deployment include the main Python file (app.py or similar), HTML templates, static files (like CSS, JavaScript), and any other dependencies.

Q20. What are the necessary libraries for flask?

Necessary libraries for Flask include Flask itself (flask), and optionally, libraries like Flask-WTF for forms, Flask-SQLAlchemy for database integration, etc.

Q21. Apart from Pickle which library we can use for saving and loading the model?

Apart from Pickle, another library that can be used for saving and loading models is joblib.

Q22. Why we are using @app.route?

@app.route is used in Flask to define URL routes that map to specific functions or views in the application.

Q23. Why are we using '/' this inside the root?

'/' inside the root is used to represent the root URL of the application.

Q24. What is GET & POST request in Flask?

In Flask, GET request is used to request data from a specified resource, and POST request is used to submit data to be processed to a specified resource.

Q25. How do you collect the input data in the Flask?

Input data in Flask can be collected using request methods like request.form.get() for form data or request.args.get() for URL query parameters.

Q26. How would you call an HTML file in Flask?

You can call an HTML file in Flask using the render\_template() function, providing the name of the HTML file as an argument.

Q27. Why do we use debug= true?

debug=True in Flask enables the debug mode, which provides helpful debugging information like stack traces in case of errors and automatically reloads the application when changes are made in the code during development.



**KMeans Clustering Algorithm**

# -\*- coding: utf-8 -\*-

"""

Created on Wed Mar 20 20:37:32 2024

@author: Lenovo

"""

'''

# K-Means Clustering Algorithm - Data Mining (Machine Learning) Unsupervised learning Algorithm

Problem Statements:

Global air travel has seen an upward trend in recent times. The maintenance of operational efficiency and maximizing profitability are crucial for airlines and airport authorities. Businesses need to optimize airline and terminal operations to enhance passenger satisfaction, improve turnover rates, and increase overall revenue.

The airline companies with the available data want to find an opportunity to analyze and understand travel patterns, customer demand, and terminal usage.

CRISP-ML(Q) process model describes six phases:

1. Business and Data Understanding

2. Data Preparation

3. Model Building

4. Model Evaluation

5. Deployment

6. Monitoring and Maintenance

Objective: Maximize the Sales

Constraints: Minimize the Customer Retention

Success Criteria:

Business Success Criteria: Increase the Sales by 10% to 12% by targeting cross-selling opportunities on current customers.

ML Success Criteria: Achieve a Silhouette coefficient of at least 0.6

Economic Success Criteria: The insurance company will see an increase in revenues by at least 8%

Data: Refer to the ‘AirTraffic\_Passenger\_Statistics.csv’ dataset.

'''

# #### Install the required packages if not available

# !pip install feature\_engine

# !pip install sklearn\_pandas

# \*\*Importing required packages\*\*

# import numpy as np

import pandas as pd

import sweetviz

import matplotlib.pyplot as plt

from sklearn.pipeline import Pipeline

from sklearn.impute import SimpleImputer

from sklearn.preprocessing import MinMaxScaler

from sklearn.compose import ColumnTransformer

from sklearn.preprocessing import OneHotEncoder

from sklearn.cluster import KMeans

from sklearn import metrics

import joblib

import pickle

# \*\*Import the data\*\*

from sqlalchemy import create\_engine, text

traff = pd.read\_csv(r"C:/Users/Lenovo/Downloads/Study material/Data Science/Clustering Segmentation 2/Assignments/Data Set/Data Set (5)/AirTraffic\_Passenger\_Statistics.csv")

# Credentials to connect to Database

user = 'root' # user name

pw = '1234' # password

db = traff\_db' # database name

engine = create\_engine(f"mysql+pymysql://{user}:{pw}@localhost/{db}")

# to\_sql() - function to push the dataframe onto a SQL table.

traff.to\_sql('traff\_tbl', con = engine, if\_exists = 'replace', chunksize = 1000, index = False)

###### To read the data from MySQL Database

sql = 'select \* from traff\_tbl;'

df = pd.read\_sql\_query(text(sql), engine.connect())

df1 = df.drop(['Operating Airline IATA Code', 'Boarding Area', 'Year', 'Month'], axis = 1)

# Data types

df1.info()

# EXPLORATORY DATA ANALYSIS (EDA) / DESCRIPTIVE STATISTICS

# \*\*\*Descriptive Statistics and Data Distribution Function\*\*\*

df1.describe()

# ## We have to check unique values for categorical data

df1["Operating Airline"].unique()

df1["Operating Airline"].unique().size

df1["Operating Airline"].value\_counts()

# AutoEDA

# Automated Libraries

# import sweetviz

my\_report = sweetviz.analyze([df1, "df1"])

my\_report.show\_html('Report.html')

# Missing Data

# Checking Null Values

df1.isnull().sum()

# Segregate Numeric and Non-numeric columns

df1.info()

# \*\*By using Mean imputation null values can be impute\*\*

numeric\_features = df1.select\_dtypes(exclude = ['object']).columns

numeric\_features

# Non-numeric columns

categorical\_features = df1.select\_dtypes(include = ['object']).columns

categorical\_features

# Define Pipeline to deal with Missing data and scaling numeric columns

num\_pipeline = Pipeline([('impute', SimpleImputer(strategy = 'mean')), ('scale', MinMaxScaler())])

num\_pipeline

# Fit the numeric data to the pipeline. Ignoring State column

processed = num\_pipeline.fit(df1[numeric\_features])

# Save the pipeline

joblib.dump(processed, 'processed1')

# Transform the data with pipeline on numeric columns to get clean data

# traff\_num = pd.DataFrame(processed.transform(df1[numeric\_features]), columns = numeric\_features)

# traff\_num

# Encoding Non-numeric fields

# \*\*Convert Categorical data "State" to Numerical data using OneHotEncoder\*\*

categ\_pipeline = Pipeline([('OnehotEncode', OneHotEncoder(drop = 'first'))])

categ\_pipeline

# Using ColumnTransfer to transform the columns of an array or pandas DataFrame.

# This estimator allows different columns or column subsets of the input to be

# transformed separately and the features generated by each transformer will

# be concatenated to form a single feature space.

preprocess\_pipeline = ColumnTransformer([('categorical', categ\_pipeline, categorical\_features),

('numerical', num\_pipeline, numeric\_features)],

remainder = 'passthrough') # Skips the transformations for remaining columns

preprocess\_pipeline

# Pass the raw data through pipeline

processed2 = preprocess\_pipeline.fit(df1)

# ## Save the Imputation and Encoding pipeline

# import joblib

joblib.dump(processed2, 'processed2')

# File gets saved under current working directory

import os

os.getcwd()

# Clean and processed data for Clustering

traff = pd.DataFrame(processed2.transform(df1).toarray(), columns = list(processed2.get\_feature\_names\_out()))

traff

# Clean data

traff.describe()

# # CLUSTERING MODEL BUILDING

# ### KMeans Clustering

# Libraries for creating scree plot or elbow curve

# from sklearn.cluster import KMeans

# import matplotlib.pyplot as plt

###### scree plot or elbow curve ############

TWSS = []

k = list(range(2, 15))

for i in k:

kmeans = KMeans(n\_clusters = i)

kmeans.fit(traff)

TWSS.append(kmeans.inertia\_)

TWSS

# ## Creating a scree plot to find out no.of cluster

plt.plot(k, TWSS, 'ro-'); plt.xlabel("No\_of\_Clusters"); plt.ylabel("total\_within\_SS")

# ## Using KneeLocator

List = []

for k in range(2, 15):

kmeans = KMeans(n\_clusters = k, init = "random", max\_iter = 30, n\_init = 10)

kmeans.fit(traff)

List.append(kmeans.inertia\_)

# !pip install kneed

from kneed import KneeLocator

kl = KneeLocator(range(2, 15), List, curve = 'convex')

# kl = KneeLocator(range(2, 9), List, curve='convex', direction = 'decreasing')

kl.elbow

plt.style.use("seaborn")

plt.plot(range(2, 15), List)

plt.xticks(range(2, 15))

plt.ylabel("Interia")

plt.axvline(x = kl.elbow, color = 'r', label = 'axvline - full height', ls = '--')

plt.show()

# Not able to detect the best K value (knee/elbow) as the line is mostly linear

# Building KMeans clustering

model = KMeans(n\_clusters = 100)

yy = model.fit(traff)

# Cluster labels

model.labels\_

# ## Cluster Evaluation

# \*\*Silhouette coefficient:\*\*

# Silhouette coefficient is a Metric, which is used for calculating

# goodness of clustering technique and the value ranges between (-1 to +1).

# It tells how similar an object is to its own cluster (cohesion) compared to

# other clusters (separation).

# A score of 1 denotes the best meaning that the data point is very compact

# within the cluster to which it belongs and far away from the other clusters.

# Values near 0 denote overlapping clusters.

# from sklearn import metrics

metrics.silhouette\_score(traff, model.labels\_)

# \*\*Calinski Harabasz:\*\*

# Higher value of CH index means cluster are well separated.

# There is no thumb rule which is acceptable cut-off value.

metrics.calinski\_harabasz\_score(traff, model.labels\_)

# \*\*Davies-Bouldin Index:\*\*

# Unlike the previous two metrics, this score measures the similarity of clusters.

# The lower the score the better the separation between your clusters.

# Vales can range from zero and infinity

metrics.davies\_bouldin\_score(traff, model.labels\_)

# ### Evaluation of Number of Clusters using Silhouette Coefficient Technique

from sklearn.metrics import silhouette\_score

silhouette\_coefficients = []

for k in range (2, 50):

kmeans = KMeans(n\_clusters = k)

kmeans.fit(traff)

score = silhouette\_score(traff, kmeans.labels\_)

k = k

Sil\_coff = score

silhouette\_coefficients.append([k, Sil\_coff])

silhouette\_coefficients

sorted(silhouette\_coefficients, reverse = True, key = lambda x: x[1])

# silhouette coefficients shows the number of clusters 'k = 2' as the best value

# Building KMeans clustering

bestmodel = KMeans(n\_clusters = 12)

result = bestmodel.fit(traff)

# ## Save the KMeans Clustering Model

# import pickle

pickle.dump(result, open('Clust\_traff.pkl', 'wb'))

import os

os.getcwd()

# Cluster labels

bestmodel.labels\_

mb = pd.Series(bestmodel.labels\_)

# Concate the Results with data

df\_clust = pd.concat([mb, df1], axis = 1)

df\_clust = df\_clust.rename(columns = {0:'cluster\_id'})

df\_clust.head()

# Aggregate using the mean of each cluster

cluster\_agg = df\_clust.select\_dtypes(exclude = ['object']).groupby(df\_clust.cluster\_id).mean()

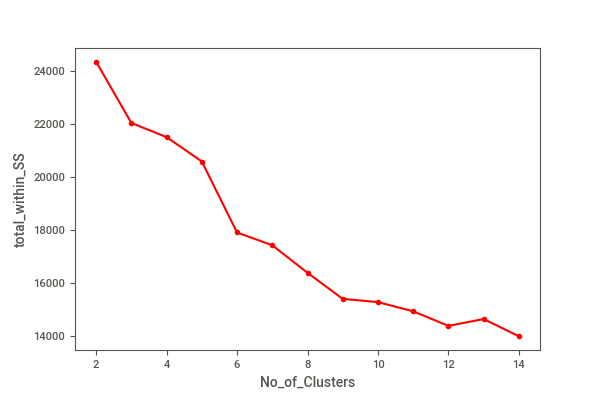
cluster\_agg

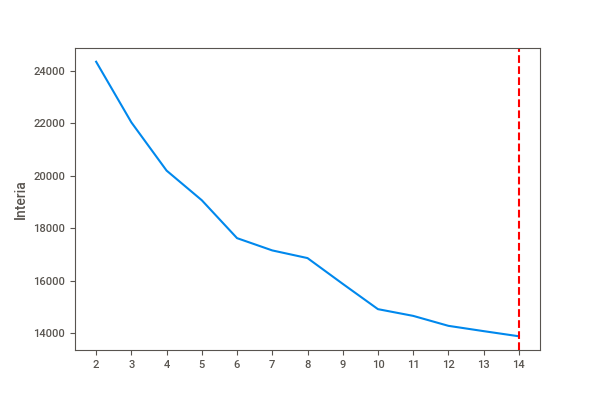
# Save the Results to a CSV file

df\_clust.to\_csv('KMeans\_traff.csv', encoding = 'utf-8', index = False)

import os

os.getcwd()

**Output:**



TWSS = []

k = list(range(2, 9))

for i in k:

kmeans = KMeans(n\_clusters = i)

kmeans.fit(traff)

TWSS.append(kmeans.inertia\_)

TWSS

Out[89]:

[24338.954246524016,

22841.274517583894,

21229.379390424372,

20019.415818083253,

19573.69811522043,

17168.603144055145,

15757.00098384532]

silhouette\_coefficients = []

for k in range (2, 50):

kmeans = KMeans(n\_clusters = k)

kmeans.fit(traff)

score = silhouette\_score(traff, kmeans.labels\_)

k = k

Sil\_coff = score

silhouette\_coefficients.append([k, Sil\_coff])

silhouette\_coefficients

sorted(silhouette\_coefficients, reverse = True, key = lambda x: x[1])

Out[161]:

[[49, 0.5612294348905238],

[48, 0.5592689275342583],

[47, 0.5470528500082168],

[46, 0.5410239758330019],

[45, 0.5310446854554752],

[43, 0.5125941957887442],

[44, 0.5081015515055182],

[41, 0.4970561420150224],

[42, 0.49427432406507293],

[39, 0.48727570355828237],

[40, 0.46638629589393676],

[38, 0.4573695293645861],

[37, 0.4494455522833752],

[36, 0.4379994137907618],

[35, 0.4361706200656429],

[34, 0.4172574247715371],

[32, 0.4032199720754645],

[33, 0.3999410280275558],

[31, 0.3803865300277675],

[30, 0.3637233291873617],

[29, 0.35980664968957937],

[28, 0.33422057146308026],

[26, 0.32935670312122456],

[27, 0.3260009987418636],

[24, 0.30354691047601434],

[25, 0.3018252894836216],

[22, 0.2876436557094971],

[19, 0.2841982922619446],

[23, 0.2724608180835593],

[21, 0.27176233879365835],

[20, 0.25351472237026174],

[16, 0.25236884563929485],

[12, 0.23871747406837404],

[10, 0.23620352308123885],

[17, 0.23382337856646598],

[18, 0.22987115161034533],

[14, 0.22747766020733118],

[11, 0.22002792094788146],

[7, 0.1964336983372859],

[15, 0.19569596590283786],

[13, 0.19387938293473797],

[9, 0.1905316606917133],

[6, 0.18870695817339947],

[8, 0.1875340235153071],

[4, 0.17520339304005147],

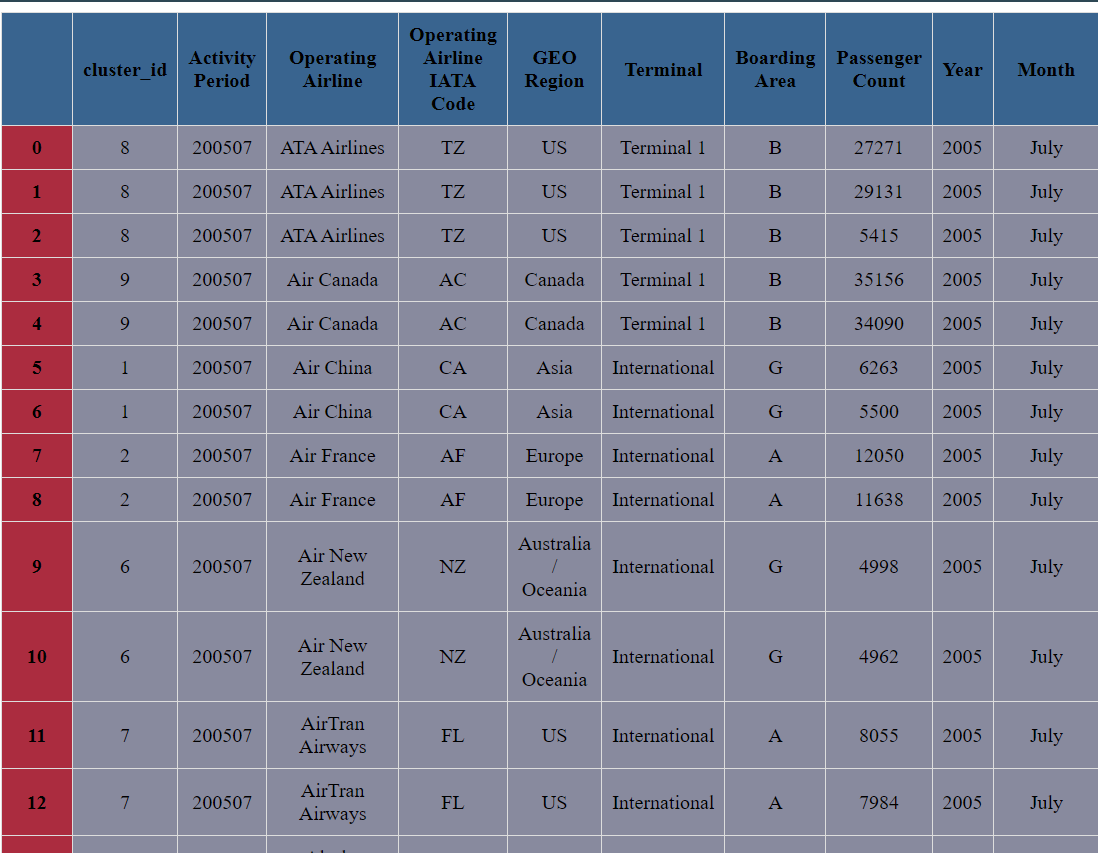
[5, 0.1691832824218696],

[2, 0.16231021526109732],

[3, 0.15812927787121422]]



**Deployed using Flask**



**DBSCAN Clustering Algorithm**

# -\*- coding: utf-8 -\*-

"""

Created on Fri Mar 22 17:05:30 2024

@author: Lenovo

"""

'''

# K-Means Clustering Algorithm - Data Mining (Machine Learning) Unsupervised learning Algorithm

Problem Statements:

Global air travel has seen an upward trend in recent times. The maintenance of operational efficiency and maximizing profitability are crucial for airlines and airport authorities. Businesses need to optimize airline and terminal operations to enhance passenger satisfaction, improve turnover rates, and increase overall revenue.

The airline companies with the available data want to find an opportunity to analyze and understand travel patterns, customer demand, and terminal usage.

CRISP-ML(Q) process model describes six phases:

1. Business and Data Understanding

2. Data Preparation

3. Model Building

4. Model Evaluation

5. Deployment

6. Monitoring and Maintenance

Objective: Maximize the Sales

Constraints: Minimize the Customer Retention

Success Criteria:

Business Success Criteria: Increase the Sales by 10% to 12% by targeting cross-selling opportunities on current customers.

ML Success Criteria: Achieve a Silhouette coefficient of at least 0.6

Economic Success Criteria: The insurance company will see an increase in revenues by at least 8%

Data: Refer to the ‘AirTraffic\_Passenger\_Statistics.csv’ dataset.

'''

import pandas as pd # data manipulation

import sweetviz # autoEDA

import matplotlib.pyplot as plt # data visualization

from sklearn.cluster import KMeans, AgglomerativeClustering, DBSCAN # machine learning algorithms

from sklearn.metrics import silhouette\_score

from sqlalchemy import create\_engine, text # connect to SQL database

# Load Airtraffic data set

traff = pd.read\_csv(r"C:/Users/Lenovo/Downloads/Study material/Data Science/Clustering Segmentation 2/Assignments/Data Set/Data Set (5)/AirTraffic\_Passenger\_Statistics.csv")

# Credentials to connect to Database

user = 'root' # user name

pw = '1234' # password

db = 'traff\_db' # database name

engine = create\_engine(f"mysql+pymysql://{user}:{pw}@localhost/{db}")

# to\_sql() - function to push the dataframe onto a SQL table.

traff.to\_sql('traff\_tbl', con = engine, if\_exists = 'replace', chunksize = 1000, index = False)

###### To read the data from MySQL Database

sql = 'select \* from traff\_tbl;'

df = pd.read\_sql\_query(text(sql), engine.connect())

traff\_df = df.iloc[:1000 , :]

traff\_df = df.select\_dtypes(exclude= ['object'])

traff\_df.drop(columns ="Year", inplace = True)

# Show sample of data set

traff\_df.head()

# ## EXPLORATORY DATA ANALYSIS (EDA) / DESCRIPTIVE STATISTICS

# \*\*\*Descriptive Statistics and Data Distribution Function\*\*\*

traff\_df.describe()

# \*\*\*1st Moment Business Decision (Measures of Central Tendency)\*\*\*

# 1) Mean

# 2) Median

# 3) Mode

# \*\*\*2nd Moment Business Decision (Measures of Dispersion)\*\*\*

# 1) Variance

# 2) Standard deviation

# 3) Range (maximum - minimum)

# \*\*\*3rd Business Moment Decision (Skewness)\*\*\*

# Measure of asymmetry in the data distribution

# traff\_df.skew()

# \*\*\*4th Business Moment Decision (Kurtosis)\*\*\*

# Measure of peakedness - represents the overall spread in the data

# traff\_df.kurt()

# AutoEDA

# ## Automated Libraries

# import sweetviz

my\_report = sweetviz.analyze([traff\_df, "traff\_df"])

my\_report.show\_html('Report.html')

# ## Data Preprocessing and Cleaning

# \*\*Typecasting\*\* :

#

# As Python automatically interprets the data types, there may be a requirement

# for the data type to be converted. The process of converting one data type

# to another data type is called Typecasting.

#

# Example:

# 1) int to float

# 2) float to int

traff\_df.info()

# \*\*Handling duplicates:\*\*

# If the dataset has multiple entries of the same record then we can remove the duplicate entries. In case of duplicates we will use function drop\_duplicates()

duplicate = traff\_df.duplicated() # Returns Boolean Series denoting duplicate rows.

print(duplicate)

sum(duplicate)

print(traff\_df.shape)

# Removing Duplicates

traff\_df = traff\_df.drop\_duplicates() # Returns DataFrame with duplicate rows removed.

print(traff\_df.shape)

# \*\*Missing Value Analysis\*\*

# \*\*\*IMPUTATION:\*\*\*

# The process of dealing with missing values is called Imputation.

# Most popular substitution based Imputation techniques are:

# 1) Mean imputation for numeric data

# 2) Mode imputation for non-numeric data

traff\_df.isnull().sum() # Check for missing values

# ### Outliers Analysis:

# Exceptional data values in a variable can be outliers. In case of outliers we can use one of the strategies of 3 R (Rectify, Retain, or Remove)

# \*\*Box Plot\*\*

# Visualize numeric data using boxplot for outliers

# Multiple boxplots in a single visualization.

# Columns with larger scales affect other columns.

# Below code ensures each column gets its own y-axis.

# pandas plot() function with parameters kind = 'box' and subplots = True

traff\_df.plot(kind = 'box', subplots = True, sharey = False, figsize = (10, 6))

'''sharey True or 'all': x- or y-axis will be shared among all subplots.

False or 'none': each subplot x- or y-axis will be independent.'''

# increase spacing between subplots

plt.subplots\_adjust(wspace = 0.75) # ws is the width of the padding between subplots, as a fraction of the average Axes width.

plt.show()

# \*\*No outliers observed\*\*

# ## Scatter Plot

plt.scatter(traff\_df.values[:, 0], traff\_df.values[:, 1])

plt.title("Airtraffic Dataset")

plt.xlabel("OD Reading")

plt.ylabel("Proline")

plt.show()

# Correlation Coefficient

traff\_df.corr()

# Generate clusters using Agglomerative Hierarchical Clustering

ac = AgglomerativeClustering(5, linkage = 'average')

ac\_clusters = ac.fit\_predict(traff\_df)

# Generate clusters from K-Means

km = KMeans(2)

km\_clusters = km.fit\_predict(traff\_df)

# Generate clusters using DBSCAN

db\_param\_options = [[5000, 20], [10000, 35], [20000, 15], [15000, 25], [8000, 30], [18000, 20]]

for ep, min\_sample in db\_param\_options:

db = DBSCAN(eps = ep, min\_samples = min\_sample)

db\_clusters = db.fit\_predict(traff\_df)

print("Eps: ", ep, "Min Samples: ", min\_sample)

print("DBSCAN Clustering: ", silhouette\_score(traff\_df, db\_clusters))

# Generate clusters using DBSCAN

db = DBSCAN(eps = 5000, min\_samples = 20)

db\_clusters = db.fit\_predict(traff\_df)

plt.figure(1)

plt.title("Airtraffic Clusters from Agglomerative Clustering")

plt.scatter(traff\_df.values[:, 0], traff\_df.values[:, 1], c = ac\_clusters, s = 50, cmap = 'tab20b')

plt.show()

plt.figure(2)

plt.title("Airtraffic Clusters from K-Means")

plt.scatter(traff\_df.values[:, 0], traff\_df.values[:, 1], c = km\_clusters, s = 50, cmap = 'tab20b')

plt.show()

plt.figure(3)

plt.title("Airtraffic Clusters from DBSCAN")

plt.scatter(traff\_df.values[:, 0], traff\_df.values[:, 1], c = db\_clusters, s = 50, cmap = 'tab20b')

plt.show()

# Calculate Silhouette Scores

print("Silhouette Scores for Airtraffic Dataset:\n")

print("Agg Clustering: ", silhouette\_score(traff\_df, ac\_clusters))

print("K-Means Clustering: ", silhouette\_score(traff\_df, km\_clusters))

print("DBSCAN Clustering: ", silhouette\_score(traff\_df, db\_clusters))

## saving dbscan

import pickle

pickle.dump(db, open('db.pkl', 'wb'))

model = pickle.load(open('db.pkl', 'rb'))

res = model.fit\_predict(traff\_df)

**Output:**

Eps: 5000 Min Samples: 20

DBSCAN Clustering: 0.8714839032870105

Eps: 10000 Min Samples: 35

DBSCAN Clustering: 0.8740328821638608

Eps: 20000 Min Samples: 15

DBSCAN Clustering: 0.9098292648449143

Eps: 15000 Min Samples: 25

DBSCAN Clustering: 0.8908792431516602

Eps: 8000 Min Samples: 30

DBSCAN Clustering: 0.8735442237590196

Eps: 18000 Min Samples: 20

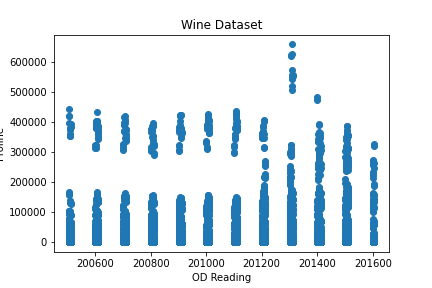
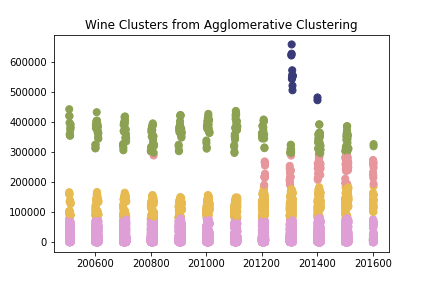
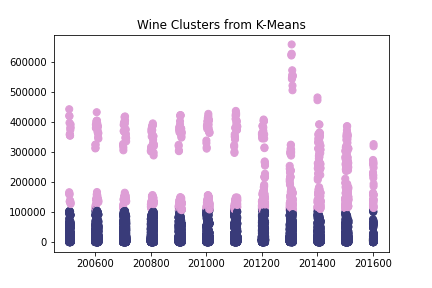
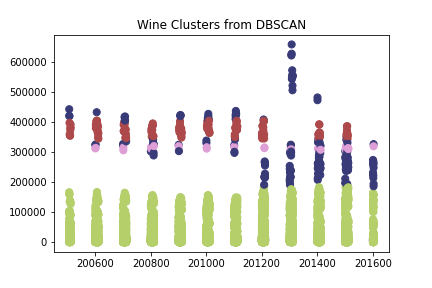
DBSCAN Clustering: 0.9098292648449143

Silhouette Scores for Airtraffic Dataset:

Agg Clustering: 0.8212379950092218

K-Means Clustering: 0.8537026098688284

DBSCAN Clustering: 0.8714839032870105

**Deployed using Flask**

