KMeans and DBscan Clustering

Please ensure you update all the details:
Name: ULLI VENKATA SAI KUMAR Batch ID: 04072024HYD10AM
Topic: KMeans and DBscan Clustering

Hints:

1. Business Problem

- **1.1. What is the business objective?**
- **1.2. What are the constraints?**
- **1.3. Define success criteria**

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

Name of Feature	Description	Type	Relevance
ID	Customer ID	Quantitative, Nominal	Irrelevant, ID does not provide useful information

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.

```
import pandas as pd

df = pd.read_csv('AirTraffic_Passenger_Statistics.csv')

print(df.info())

print(df.head())

from sklearn.preprocessing import StandardScaler

from sklearn.cluster import KMeans

from sklearn.metrics import silhouette_score
```

```
features = df[['Passenger Count']]

scaler = StandardScaler()

scaled_features = scaler.fit_transform(features)

kmeans = KMeans(n_clusters=3, random_state=42)

clusters = kmeans.fit_predict(scaled_features)

df['Cluster'] = clusters

silhouette_avg = silhouette_score(scaled_features, clusters)

print(f'Silhouette Score: {silhouette_avg}')
```

Questions to Trigger Your Thoughts:

Q1. What is sklearn.preprocessing is mostly used for?

The sklearn.preprocessing module is primarily used for scaling, normalizing, and transforming data. This step is crucial for machine learning algorithms that are sensitive to feature scaling, such as K-Means, SVM, and logistic regression.

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

- StandardScaler (Standardization to zero mean and unit variance)
- MinMaxScaler (Normalization to a specific range, e.g., 0 to 1)
- RobustScaler (Scaling robust to outliers using the median and IQR)

Q3. What library does the groupby function belong to?

```
import pandas as pd

df.groupby('column_name').mean()
```

Q4. How to save an output file?

```
df.to_csv('output_file.csv', index=False)
```

Or use pickle or joblib for saving machine learning models:

```
import pickle

pickle.dump(model, open('model.pkl', 'wb'))
```

Q5. How to perform scaling without using inbuilt functions?

You can manually scale the data using the formula for normalization or standardization.

- **Normalization**: Rescale between 0 and 1:

  df_scaled = (df - df.min()) / (df.max() - df.min())

- **Standardization**: Rescale to have a mean of 0 and standard deviation of 1:
  df_scaled = (df - df.mean()) / df.std()

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

In sklearn, use the inverse_transform() function to reverse the scaling effect.

scaler = StandardScaler()

scaled_data = scaler.fit_transform(data)

original_data = scaler.inverse_transform(scaled_data)

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

- info(): Provides a concise summary of the DataFrame, including data types, non-null counts, and memory usage.
- describe(): Provides descriptive statistics for numerical columns (e.g., mean, std, min, max).

Q8. How to write the X-axis label?

In matplotlib, you can use the xlabel() function to label the X-axis.

import matplotlib.pyplot as plt

plt.xlabel('X-Axis Label')

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

To fix label overlap, you can rotate the labels using plt.xticks():

plt.xticks(rotation=45) # Rotate by 45 degrees

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

The **elbow method** helps in determining the optimal number of clusters by plotting the explained variance or inertia (within-cluster sum of squares) for different numbers of clusters. The point where the plot bends (like an elbow) indicates the best number of clusters.

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

The clustering algorithm itself (e.g., KMeans, DBSCAN) assigns labels, accessible via .labels_ or .predict().

- kmeans.labels_

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

Yes, for many clustering algorithms (e.g., **KMeans**), scaling is important because the algorithm is distance-based and unscaled data can lead to biased clustering.

Q13. Which libraries are used to implement DBscan clustering?

The DBSCAN algorithm is available in **scikit-learn**.

- from sklearn.cluster import DBSCAN

Q14. What is the DBscan algorithm?

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a clustering algorithm that groups points that are close together (density-based) and marks outliers as noise. It is well-suited for datasets with varying densities.

Q15. What are the parameters used in DBscan?

Key parameters:

- eps: Maximum distance between two samples to be considered neighbors.
- min_samples: Minimum number of samples to form a cluster.

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

Some common evaluation metrics for clustering:

1. **Silhouette Score**: Measures how similar points are within a cluster and how different they are from points in other clusters.

2. **Davies-Bouldin Index**: Measures the average ratio of intra-cluster and inter-cluster distances.

Q17. Which attribute gives labels for the clusters?

In clustering algorithms, use .labels_ for cluster labels (e.g., kmeans.labels_ or dbscan.labels_).

Q18. How to visualize the clusters?

Use **scatter plots** or **heatmaps** with different colors for clusters:

```
import matplotlib.pyplot as plt
plt.scatter(X[:, 0], X[:, 1], c=kmeans.labels_, cmap='viridis')
plt.show()
```

Q19. What are the files required for flask deployment?

The essential files include:

1. app.py (Flask app)
2. templates/ folder for HTML files
3. static/ folder for CSS and JavaScript
4. Model files (e.g., model.pkl for machine learning models)

Q20. What are the necessary libraries for flask?

Key libraries:

- **Flask**: For web application framework.
- **Pickle**: For loading the model.
- **Pandas**: For data handling.
- **Sklearn**: For applying ML models.
- pip install flask pandas scikit-learn

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

You can use **joblib** for saving and loading large models more efficiently.

```
from joblib import dump, load

dump(model, 'model.joblib')

model = load('model.joblib')
```

Q22. Why we are using @app.route?

@app.route is a **decorator** in Flask that maps a specific URL to a Python function. It defines the routes for different web pages in the app.

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

The '/' refers to the **root URL** of the Flask web application, typically the homepage.

Q24. What is GET & POST request I flask?

- **GET**: Requests data from the server (e.g., rendering a webpage).
- **POST**: Sends data to the server (e.g., submitting a form).

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

You can collect input data via request.form (for form data) or request.args (for URL parameters).

```
from flask import request

user_input = request.form['input_name']
```

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

Use render_template() to call an HTML file stored in the templates/ folder.

```
from flask import render_template

@app.route('/')
```

```
def index():
    return render_template('index.html')
```

Q27. Why do we use debug= true?

Setting debug=True enables **debug mode** in Flask, which:

- Automatically restarts the server when code changes are made.
- Displays detailed error messages in the browser during development.