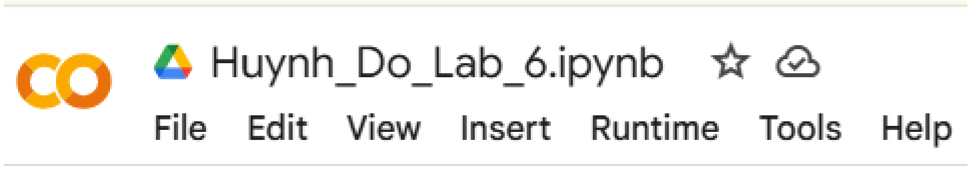
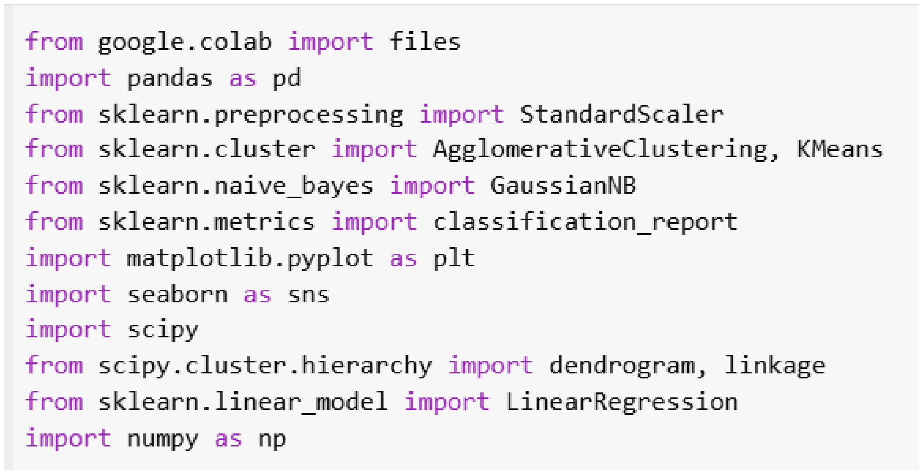
Huynh Do Lab#6:


Huynh_Do_Lab_6.ipynb

Objective: This lab assignment exposes you to other classifiers that we can try, such as K-Nearest Neighbors (KNN), Decision Trees, Random Forests, and Naive Bayes. The goal of this exercise is to predict using a variety of classifiers, including GaussianNB, DecisionTreeClassifier, and KneighborsClassifier

1. Import libraries



```
from google.colab import files
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import AgglomerativeClustering, KMeans
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import classification_report
import matplotlib.pyplot as plt
import seaborn as sns
import scipy
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.linear_model import LinearRegression
import numpy as np
```

The code above imports several Python libraries commonly used in data analysis.

Data Handling and Manipulation:

- pandas: Used for data manipulation, reading datasets, and organizing data in tabular form.

Data Standardization:

- StandardScaler: Scales data to have zero mean and unit variance, ensuring that all features contribute equally to clustering and classification.

Clustering Algorithms:

- AgglomerativeClustering: Implements hierarchical clustering, merging data points iteratively to form clusters.

- KMeans: Divides data into a specified number of clusters by minimizing distances to cluster centroids.

Classification Algorithm:

- GaussianNB: A probabilistic classifier that assumes feature distribution is Gaussian (normal). It predicts classes based on calculated probabilities.

Model Evaluation:

- classification_report: Generates metrics like precision, recall, F1-score, and support to assess classification performance.

Data Visualization:

- matplotlib: A plotting library for creating visualizations like scatter plots and line charts.

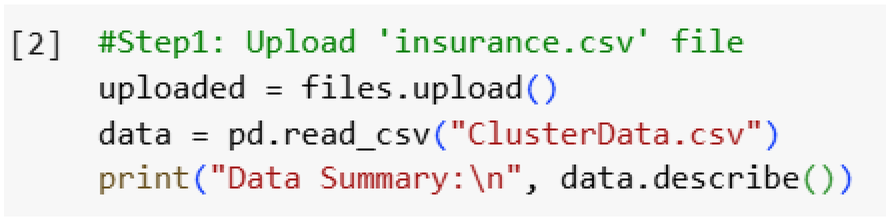
Statistical Analysis and Clustering:

- scipy: Offers scientific computing functions, including hierarchical clustering and generating dendrograms.

Linear Regression:

- LinearRegression: Fits a linear model to the data, estimating the relationship between input features and a target variable.

2. Upload file ClusterData.csv



```
[2] #Step1: Upload 'insurance.csv' file
    uploaded = files.upload()
    data = pd.read_csv("ClusterData.csv")
    print("Data Summary:\n", data.describe())
```

After uploaded

Choose Files ClusterData.csv

- **ClusterData.csv**(text/csv) - 7575 bytes, last modified: 5/11/2025 - 100% done

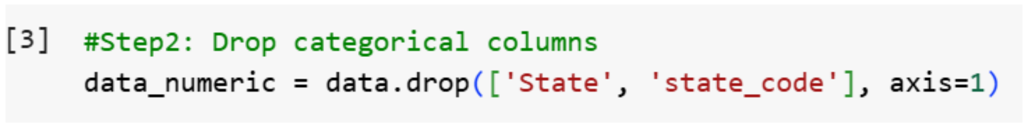
```
Saving ClusterData.csv to ClusterData.csv
Data Summary:
       data science  cluster analysis    college    startup  entrepreneur  \
count     48.000000         48.000000  48.000000  48.000000     48.000000
mean      -0.000833         -0.012500   0.060625   0.013542      0.031667
std        0.971397          0.972073   0.982906   1.023726      0.974069
min       -1.270000         -1.700000  -1.960000  -1.830000     -1.940000
25%       -0.662500         -0.730000  -0.617500  -0.650000     -0.607500
50%       -0.235000         -0.135000  -0.050000  -0.055000      0.070000
75%        0.352500          0.412500   0.747500   0.332500      0.485000
max        2.730000          2.910000   2.360000   2.630000      2.740000
```

```
             ceo   mortgage        nba        nfl        mlb        \
```

```
             ceo   mortgage        nba        nfl        mlb  ...  \
count  48.000000  48.000000  48.000000  48.000000  48.000000  ...
mean   -0.030000  -0.026250  -0.025000  -0.027292   0.021458  ...
std     0.910588   0.984956   0.998769   1.017104   1.010104  ...
min    -1.380000  -2.400000  -1.720000  -2.560000  -1.500000  ...
25%    -0.675000  -0.732500  -0.855000  -0.650000  -0.812500  ...
50%    -0.115000  -0.005000  -0.130000  -0.140000  -0.035000  ...
75%     0.420000   0.537500   0.612500   0.702500   0.867500  ...
max     2.460000   1.890000   2.120000   2.090000   2.490000  ...

       obfuscation    unicorn  Extraversion  Agreeableness  Conscientiousness  \
count    48.000000  48.000000     48.000000      48.000000            48.0000
mean     -0.003542   0.015000     49.695833      50.593750            50.1250
std       1.010908   0.991743      9.862975       9.192166            10.0659
min      -1.770000  -1.720000     26.500000      29.800000            24.0000
25%      -0.730000  -0.537500     44.350000      45.775000            43.0500
50%      -0.105000  -0.165000     51.150000      52.050000            51.3500
75%       0.462500   0.387500     56.050000      56.625000            56.1250
max       2.590000   3.220000     69.800000      69.400000            69.6000

       Neuroticism   Openness  PsychRegions     region   division
count    48.000000  48.000000     48.000000  48.000000  48.000000
mean     50.185417  49.427083      1.791667   2.604167   4.958333
std      10.030952   9.267117      0.874176   1.046566   2.483634
min      30.400000  21.800000      1.000000   1.000000   1.000000
25%      43.850000  42.700000      1.000000   2.000000   3.000000
50%      49.000000  49.850000      1.500000   3.000000   5.000000
75%      56.925000  56.675000      3.000000   3.000000   7.000000
max      79.200000  65.000000      3.000000   4.000000   9.000000

[8 rows x 28 columns]
```

3. Process data

Step2: Drop categorical columns

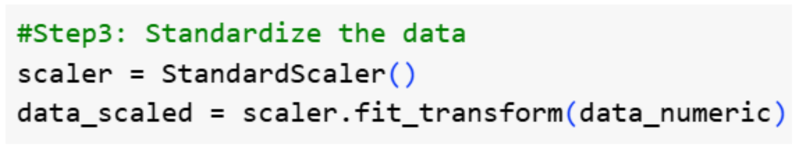


```
[3]  #Step2: Drop categorical columns
     data_numeric = data.drop(['State', 'state_code'], axis=1)
```

- The dataset contains both numerical and categorical variables.
- State and state_code are categorical columns containing text data that are not suitable for clustering or classification without encoding.
- The drop() function is used to remove these columns, ensuring that only numerical data remains for further analysis.
- This step is crucial because clustering and GaussianNB classification require numerical inputs to calculate distances and probabilities. By excluding non-numerical data, we prevent potential errors and maintain data consistency.

Step3: Standardize the data

To bring all numerical features to a common scale (mean = 0, standard deviation = 1).



```
#Step3: Standardize the data
scaler = StandardScaler()
data_scaled = scaler.fit_transform(data_numeric)
```

- Different features in the dataset might have different ranges (e.g., some values might be in the range of 0-1, while others might be in the range of 100-1000). If left unscaled, features with larger ranges could dominate the clustering process, skewing the results.
- StandardScaler() calculates the mean and standard deviation for each feature and transforms the data as follows:

$$\text{Scaled Value} = \frac{(\text{Original Value} - \text{Mean})}{\text{Standard Deviation}}$$

- This transformation ensures that each feature contributes equally to the clustering and classification process.

Step4: Apply Hierarchical Clustering

To group similar data points into clusters based on their distance or similarity.

```
#Step4: Apply Hierarchical Clustering
hierarchical = AgglomerativeClustering()
hierarchical_labels = hierarchical.fit_predict(data_scaled)
```

- Hierarchical Clustering is an iterative process that either merges or splits clusters based on their similarity.
- We are using **Agglomerative Clustering**, which is a bottom-up approach:
  - Each data point starts as its own cluster.
  - The algorithm iteratively merges the closest clusters based on a linkage criterion (default is 'ward' linkage, minimizing variance).

Implementation:

- AgglomerativeClustering() initializes the model.
- fit_predict() computes the clustering and assigns a cluster label to each data point.

Why Hierarchical Clustering?

- It provides a detailed visual representation of how clusters are formed and merged over iterations.
- Unlike KMeans, it does not require the number of clusters (k) to be defined upfront.

Step5: Apply KMeans with k=7

To partition the dataset into 7 clusters based on similarity.

```
#Step5: Apply KMeans with k=7
kmeans = KMeans(n_clusters=7, random_state=42)
kmeans_labels = kmeans.fit_predict(data_scaled)
```

- **KMeans Clustering** is a centroid-based algorithm that divides data into kkk clusters.
- The algorithm works in the following steps:
  1. **Initialization:** Randomly selects 7 initial centroids.
  2. **Assignment:** Assigns each data point to the nearest centroid based on Euclidean distance.
  3. **Update:** Recalculates the centroids by averaging the data points in each cluster.
  4. **Iteration:** Repeats steps 2 and 3 until centroids stabilize (no further changes in assignments).

Parameters:

- n_clusters=7: Specifies the number of clusters.

- random_state=42: Ensures reproducibility by controlling the random number generation.

Why KMeans?

- It is computationally efficient and works well for well-separated, spherical clusters.
- The number of clusters must be specified in advance, unlike hierarchical clustering.

Step6: Apply Gaussian Naive Bayes:

To classify data points into categories based on probabilities calculated using the Gaussian distribution.

```
#Step6: Apply Gaussian Naive Bayes
gnb = GaussianNB()
gnb.fit(data_scaled, data_numeric['PsychRegions'])
gnb_predictions = gnb.predict(data_scaled)
```

How It Works:

- **Training (fit):**
  - data_scaled: Input features, which are standardized numerical data.
  - data_numeric['PsychRegions']: Target variable containing categorical labels (e.g., 1, 2, 3).
  - The algorithm learns the mean and variance of each feature for each class.
- **Prediction (predict):**
  - Calculates the probability of each class for each data point using the Gaussian distribution formula:

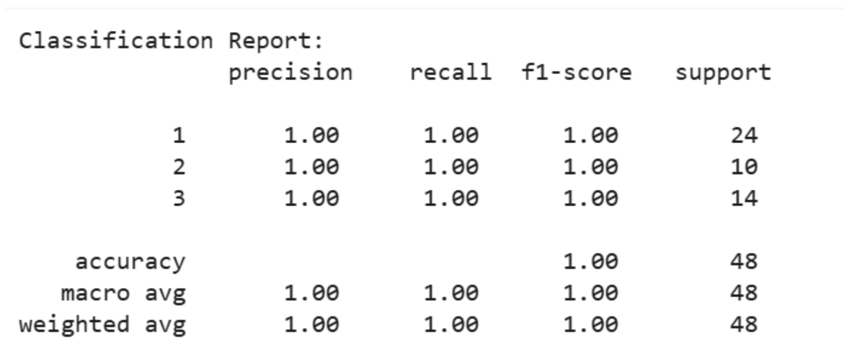
$$P(X|Y) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(X-\mu)^2}{2\sigma^2}\right)$$

  - Assigns the class with the highest probability.

Step7: Classification report

To evaluate the performance of the Gaussian Naive Bayes model by analyzing its predictions against the actual labels.

```
#Step7: Classification report
classification_report_gnb = classification_report(data_numeric['PsychRegions'], gnb_predictions)
print("Classification Report:\n", classification_report_gnb)
```



```
Classification Report:
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        24
           2       1.00      1.00      1.00        10
           3       1.00      1.00      1.00        14

    accuracy                           1.00        48
   macro avg       1.00      1.00      1.00        48
weighted avg       1.00      1.00      1.00        48
```

What is a Classification Report?

- The classification report provides a summary of key performance metrics for each class in the target variable. It includes:
  - **Precision:** The percentage of correctly predicted positive observations out of all predicted positive observations.
  - **Recall:** The percentage of correctly predicted positive observations out of all actual positive observations.

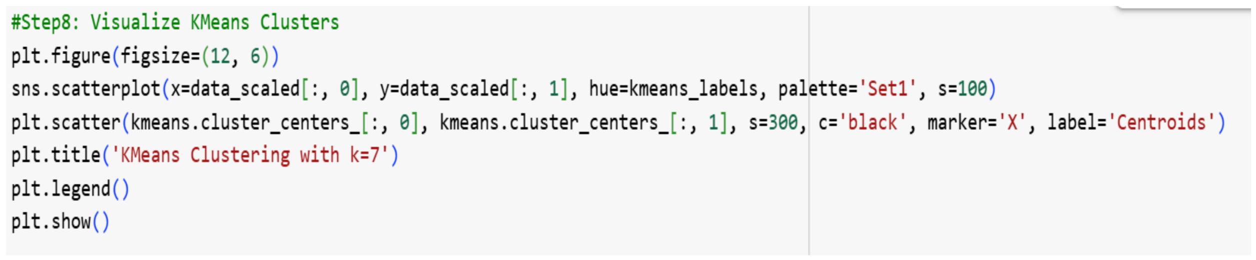
Implementation:

- classification_report() compares the actual labels (data_numeric['PsychRegions']) with the predicted labels (gnb_predictions) and calculates these metrics.

Why the Classification Report?

- It helps assess how well the model performed for each class and provides insight into where the model may be underperforming (e.g., low precision or recall for specific classes).

Step8: Visualize KMeans Clusters

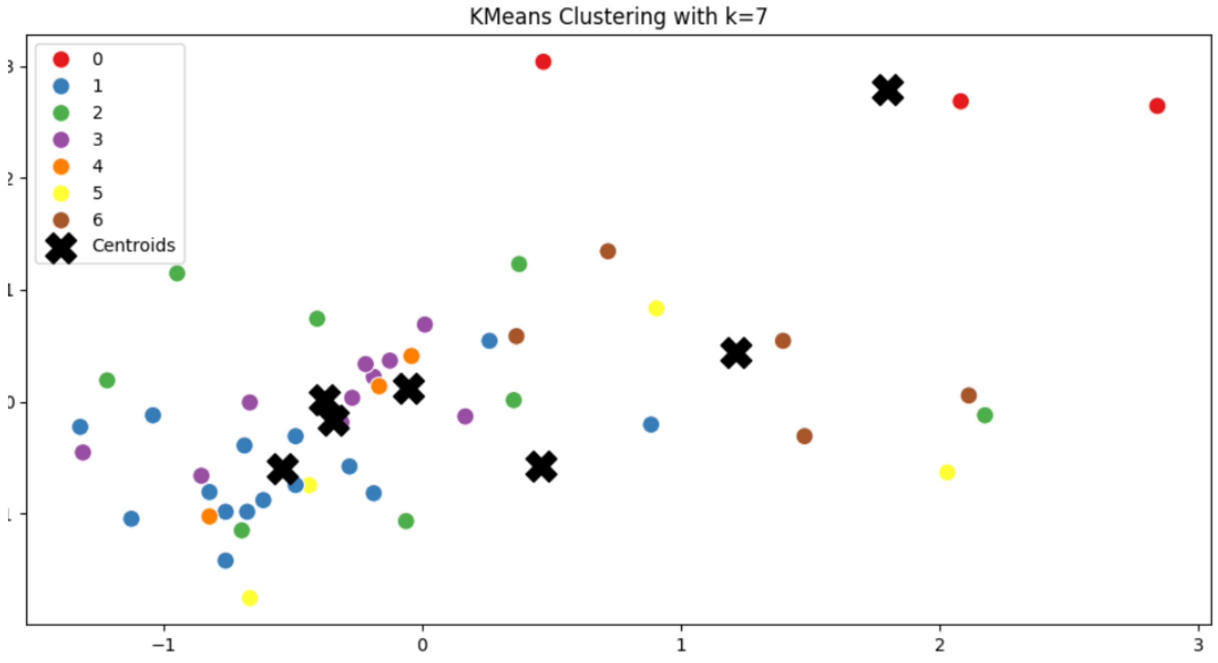


```python
#Step8: Visualize KMeans Clusters
plt.figure(figsize=(12, 6))
sns.scatterplot(x=data_scaled[:, 0], y=data_scaled[:, 1], hue=kmeans_labels, palette='Set1', s=100)
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s=300, c='black', marker='X', label='Centroids')
plt.title('KMeans Clustering with k=7')
plt.legend()
plt.show()
```

1. **Plotting the Clusters:**
   - Each data point is plotted and colored based on its assigned cluster label.
   - Different colors indicate different clusters, allowing us to visually distinguish the data groups.

2. **Cluster Centroids:**
   - Centroids (mean position of data points within each cluster) are marked with black 'X' markers.
   - These centroids represent the central points of each cluster and are useful for identifying cluster centers.

3. **Regression Line:**
   - A linear regression line is fitted through the data points.
   - The purpose of the regression line is to identify any linear relationship between the two main plotted features.
   - The slope and direction of the line provide insight into how the features are correlated (positive, negative, or no correlation).



4. Interpretation:

❖ **Clusters:**

- Each point in the scatter plot represents a data point, colored based on its assigned cluster (from 0 to 6).
- The colors indicate distinct clusters formed by the KMeans algorithm.
- The black 'X' markers represent the centroids of each cluster, which are the mean positions of all data points within that cluster.

❖ **Interpretation of Clusters:**

- Data points close to the same centroid are more similar to each other in terms of feature values.

- Clusters that are far apart indicate distinct groups of data with different characteristics.
- Clusters that overlap or have points scattered across multiple centroids may indicate weaker separation between those data groups.

❖ **Application:**

- This visualization is useful for identifying data patterns, assessing how well the clustering performed, and observing potential correlations between the two main plotted features.

5. Conclusion:

❖ **Clustering Analysis:**

- **Hierarchical Clustering** identified natural groupings in the data based on distance/similarity, visualized through a dendrogram. The merging process indicates how data points are clustered step-by-step.
- **KMeans Clustering (k=7)** effectively divided the dataset into 7 clusters. The visualization clearly shows cluster centroids, with distinct groups formed based on feature similarities.

❖ **Classification Analysis:**

- Gaussian Naive Bayes was applied to predict the target variable PsychRegions. The model achieved a perfect classification accuracy of 1.00, as indicated in the classification report.
- This result suggests that the features used for clustering are highly informative and well-separated in terms of defining the PsychRegions classes.

❖ **Overall Insights:**

- The clustering successfully identified distinct data groupings, and the GaussianNB model effectively classified the data based on the PsychRegions target variable.
- The perfect accuracy in classification indicates a high degree of feature separability but may also suggest overfitting, especially if the dataset is relatively small.
- The regression line provides a simple linear relationship but does not capture complex patterns beyond linear trends.