Customer Segmentation Project

# Project Overview

* Name: Rajiv Yadav
* Project Title: Customer Segmentation Project
* Business/Problem Context: Understanding customer spending patterns is critical for targeted marketing, personalized offers, and revenue optimization.
* Objective Statement: Cluster customers based on annual spending in various product categories to derive meaningful business insights.

# Problem Understanding

* Problem Type: Unsupervised Learning (Clustering)
* Target Variable: None (unsupervised)
* Key Inputs: Annual spending in categories ['Fresh', 'Milk', 'Grocery', 'Frozen', 'Detergents\_Paper', 'Delicassen']
* Assumptions/Constraints:  
  - Dataset has no missing values or anomalies.  
  - Spending values are annual and comparable across features.  
  - Clusters represent meaningful customer segments for business insights.

# Data Handling

* Dataset Source: Wholesale Customers Dataset (UCI ML Repository)
* Data Description: 440 rows, 8 columns; features: Channel, Region, Fresh, Milk, Grocery, Frozen, Detergents\_Paper, Delicassen; no missing values; no duplicates
* Preprocessing Steps:  
  - Dropped categorical features 'Channel' and 'Region'  
  - Checked for missing values and duplicates  
  - Log transformation applied to skewed numerical features  
  - Scaled features using StandardScaler (for analysis) and MinMaxScaler (for modeling)
* Exploratory Data Analysis (EDA):  
  - Histograms to understand distributions  
  - Boxplots for outlier detection  
  - Correlation heatmap to identify relationships  
  - KDE plots comparing original, Standard Scaled, and MinMax Scaled data

# Technical Implementation

* Approach Selection: KMeans clustering chosen for segmenting customers based on spending patterns due to simplicity and interpretability.
* Baseline Model: Initial KMeans with K=2 clusters; metrics recorded for comparison.
* Feature Engineering:  
  - Total spending per customer calculated  
  - Clusters analyzed for dominant and weak categories
* Model Development:  
  - KMeans  
  - Agglomerative Clustering  
  - DBSCAN (for comparison)  
  - Gaussian Mixture Model (for comparison)
* Hyperparameter Tuning: Tested cluster sizes from 2 to 9 using Elbow Method and silhouette score
* Pipelines/Automation: Models and scalers saved to 'models/' directory

# Code Quality & Documentation

* Code Structure:  
  customer-clustering/  
  ├─ data/ (dataset files)  
  ├─ models/ (saved models and scalers)  
  ├─ notebooks/ (EDA and experimentation notebooks)  
  ├─ train.py (train and save model/scaler)  
  ├─ predict.py (predict single customer cluster)  
  ├─ batch\_predict.py (predict multiple customers)  
  ├─ requirements.txt (Python dependencies)  
  ├─ README.md (project documentation)
* Comments/Docstrings: Functions documented clearly
* Reproducibility: Scripts can be executed sequentially to reproduce results
* Version Control: Git used for code tracking

# Results Analysis & Interpretation

* Evaluation Metrics:  
  - Silhouette Score: Measures cohesion/separation (higher is better)  
  - Davies-Bouldin Index: Measures similarity between clusters (lower is better)  
  - Calinski-Harabasz Index: Measures variance ratio between and within clusters (higher is better)
* Model Comparison: KMeans provided best interpretability and meaningful clusters
* Error Analysis: Misclassified customers may be due to overlapping spending patterns
* Business Interpretation: Clusters help define high-value, medium-value, and specialized customers for marketing focus

# Presentation & Communication

* Clarity: Documentation and code structure make the solution easy to follow
* Visualization: Histograms, boxplots, heatmaps, and cluster plots included
* Storytelling: Problem → Approach → Solution → Insights connection maintained

# Peer Review

* Feedback Criteria: Clarity, originality, missing considerations, reproducibility

# Time Management

* Solution delivered within deadline; work properly paced

# Installation

* Clone the repository:
* git clone <repository-url>  
  cd customer-clustering

# How to Run

* Train the Model:  
  python train.py  
  Saves kmeans\_model.pkl and minmax\_scaler.pkl to models/ folder.
* Predict a Single Customer:  
  python predict.py  
  Input a dictionary with customer spending in categories ['Fresh', 'Milk', 'Grocery', 'Frozen', 'Detergents\_Paper', 'Delicassen'].  
  Outputs predicted cluster, segment, total spending, and recommendations.
* Batch Prediction:  
  python batch\_predict.py  
  Reads multiple customer records from new\_customers\_sample.csv.  
  Saves predictions to new\_customers\_sample\_with\_predictions.csv.

# 12. Assumptions

- Dataset has no missing values or anomalies  
- Spending values are annual and comparable across features  
- Clusters represent meaningful customer segments for business insights

# 13. References

- Wholesale Customers Dataset (UCI Machine Learning Repository)  
- Scikit-learn documentation for clustering algorithms and evaluation metrics