Mini Project: <u>Dry Bean Type Classification</u>

Objective

An agriculture company approaches you with a challenge: they are currently **classifying different types of dry beans manually**, a process that is **labour-intensive, prone to errors and inefficient at scale**. They want to automate this classification process using **Artificial Intelligence** to improve **accuracy**, **reduce operational costs** and ensure **consistent quality** in packaging and distribution.

As a **data scientist**, your role is to help them build a machine learning solution that can **accurately classify bean types** based on physical characteristics such as area, perimeter, shape and compactness. By doing this, you'll:

- **Automate the bean classification process** using supervised learning techniques
- Help the company **reduce manual labour and cost**
- Improve the **speed and reliability** of quality control operations
- Deliver a scalable solution for **real-time classification in industrial settings**

This project not only enhances your understanding of supervised machine learning but also demonstrates how AI can be directly applied to **Agri-tech and food processing industries** for tangible business impact.

Dataset

Use the attached data

Dry_Bean_Dataset.xlsx

Data Dictionary

All configurations (like area, shape and roundness) of data were gathered **through camera-based systems using computer vision algorithms**—a method commonly used in modern agricultural quality control setups.

Attribute Information:

1. Area (A): The area of a bean zone and the number of pixels within its boundaries.
2. Perimeter (P): Bean circumference is defined as the length of its border.
3. Major axis length (L): The distance between the ends of the longest line that can be drawn from a bean.
4. Minor axis length (l): The longest line that can be drawn from the bean while standing perpendicular to the main axis.
5. Aspect ratio (K): Defines the relationship between L and l.
6. Eccentricity (Ec): Eccentricity of the ellipse having the same moments as the region.

7. Convex area (C): Number of pixels in the smallest convex polygon that can contain the area of a bean seed.
8. Equivalent diameter (Ed): The diameter of a circle having the same area as a bean seed area.
9. Extent (Ex): The ratio of the pixels in the bounding box to the bean area.
10. Solidity (S): Also known as convexity. The ratio of the pixels in the convex shell to those found in beans.
11. Roundness (R): Calculated with the following formula: (4piA)/(P^2)
12. Compactness (CO): Measures the roundness of an object: Ed/L
13. ShapeFactor1 (SF1)
14. ShapeFactor2 (SF2)
15. ShapeFactor3 (SF3)
16. ShapeFactor4 (SF4)
17. **Class (Seker, Barbunya, Bombay, Cali, Dermosan, Horoz and Sira)**

Tasks

1. Import and Load the Data

- Import necessary libraries (pandas, numpy, matplotlib, seaborn, sklearn etc.)
- Load the dataset and explore it using .head(), .info(), and .describe()

---

2. Exploratory Data Analysis (EDA)

- Visualize distributions of features using histograms and boxplots
- Analyze the class distribution (check for **class imbalance**)
- Plot feature correlations (eg heatmap)
- Visualize multivariate relationships (pairplot)
- Summarize key findings

---

3. Missing Values & Outlier Treatment

- Check for and handle missing values
- Detect and treat outliers if needed (Z-score / IQR methods/boxplots)

---

4. Feature Engineering & Preprocessing

- Scale numerical features (StandardScaler / MinMaxScaler)
- Encode categorical variables if necessary
- Check and treat skewness if required
- Split data into **train/test** sets (use **stratified sampling** while splitting)

---

5. Model Building: Try Multiple Classifiers

Train and test the dataset on a variety of **supervised classification algorithms**:

- **Logistic Regression**
- **Decision Tree Classifier**
- **Random Forest Classifier**
- **K-Nearest Neighbors (KNN)**
- **Support Vector Machine (SVM)**
- **Ensemble Learning Methods**
- **Naive Bayes** and more…

Use Cross validation techniques to check if model performance is improved.

---

6. Handling Class Imbalance

- Apply techniques like:
    - SMOTE (Synthetic Minority Over-sampling)
    - Random Oversampling / Undersampling
    - Class weighting
- Evaluate if performance improves on minority classes

---

7. Model Evaluation & Overfitting Check

Use appropriate classification metrics:

- **Accuracy**
- **Precision, Recall, and F1-Score** (for each class)
- **Confusion Matrix**

Check for Overfitting:
Compare **training vs test accuracy and performance metrics**.

---

8. Hyperparameter Tuning

- Use **GridSearchCV** or **RandomizedSearchCV** to optimize parameters for top-performing models
- Document the best parameters and performance improvement

9. Model Comparison Table

Model	Train Accuracy	Test Accuracy	F1 Score	Overfitting (Y/N)
Logistic Regression				
Decision Tree				
Random Forest				
SVM				
KNN				
and other algos..				
Best Model				

10. Build a Simple Classifier App

- Use **Streamlit** to create a basic UI
- Input physical measurements of a bean and get predicted class