DRY BEANS CLASSIFICATION

Team members

Name	ID
Yasmine Khaled atta	20201701154
Ahmed Esmail Mohamed	20201700024
Ebrahim Ahmed Abdelmaboud	20201700009
Ahmed Mohsen Fathelbab	20201700067
Ahmed Mohamed Kamal	20201700080

Table of contents

Introduction: Dry Beans Classification

The "Dry Beans Classification" project aims to predict and classify three distinct categories of dry beans: Bombay, Cali, and Sira. This classification is based on measurements of five significant features: Area, Perimeter, MajorAxisLength, MinorAxisLength, and Roundness, each measured in millimeters. The dataset contains a total of 150 samples, with 50 samples from each of the three dry bean varieties.

- ## Objective

The primary objective of this project is to build a machine learning model that can accurately classify dry beans into one of the three categories (Bombay, Cali, or Sira) based on the provided measurements.

- ## Data Dictionary

The dataset includes the following features:

1. Class: The categorical variable representing the three categories of dry beans – Bombay, Cali, and Sira.
2. Area: The area of a dry bean sample, measured in millimeters squared (mm^2).
3. Perimeter: The perimeter or circumference of a dry bean sample, measured in millimeters (mm).
4. MajorAxisLength: The length of the major axis of a dry bean, measured in millimeters (mm).
5. MinorAxisLength: The length of the minor axis of a dry bean, measured in millimeters (mm).
6. Roundness: A measurement indicating the roundness or shape of the dry bean. The specific method of calculation may vary, but it represents a dimensionless value related to the bean's shape.

Data Exploration

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Area             150 non-null    int64  
 1   Perimeter        150 non-null    float64
 2   MajorAxisLength  150 non-null    float64
 3   MinorAxisLength  149 non-null    float64
 4   roundnes         150 non-null    float64
 5   Class            150 non-null    object 
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

```
# finding no of rows and no of columns in data set
print('number of rows:',data.shape[0])
print('number of columns:',data.shape[1])
```

```
number of rows: 150
number of columns: 6
```

```
data.dtypes
```

```
Area                int64
Perimeter         float64
MajorAxisLength   float64
MinorAxisLength   float64
roundnes          float64
Class              object
dtype: object
```

```
data.columns
```

```
Index(['Area', 'Perimeter', 'MajorAxisLength', 'MinorAxisLength', 'roundnes',
       'Class'],
      dtype='object')
```

```
data.describe()
```

	Area	Perimeter	MajorAxisLength	MinorAxisLength	roundnes
count	150.000000	150.000000	150.000000	149.000000	150.000000
mean	75557.933333	999.372293	370.564985	240.245914	0.875078
std	44232.034170	293.752695	105.115378	75.490630	0.034295
min	31519.000000	668.106000	233.804968	157.802740	0.688618
25%	35139.250000	708.690000	264.743366	175.379706	0.855589
50%	56756.500000	914.957000	352.010221	206.618773	0.880003
75%	132879.500000	1369.277750	497.101354	338.364472	0.893303
max	144079.000000	1463.258000	540.677823	376.550241	0.954104

```
data['Class'].unique()
```

```
array(['BOMBAY', 'CALI', 'SIRA'], dtype=object)
```

```
data['Class'].value_counts()
```

```
Class
BOMBAY    50
CALI      50
SIRA      50
Name: count, dtype: int64
```

```
data.isnull().sum().sort_values(ascending=False)
```

```
MinorAxisLength    1
Area               0
Perimeter          0
MajorAxisLength    0
roundnes           0
Class              0
dtype: int64
```

```
data.duplicated()
```

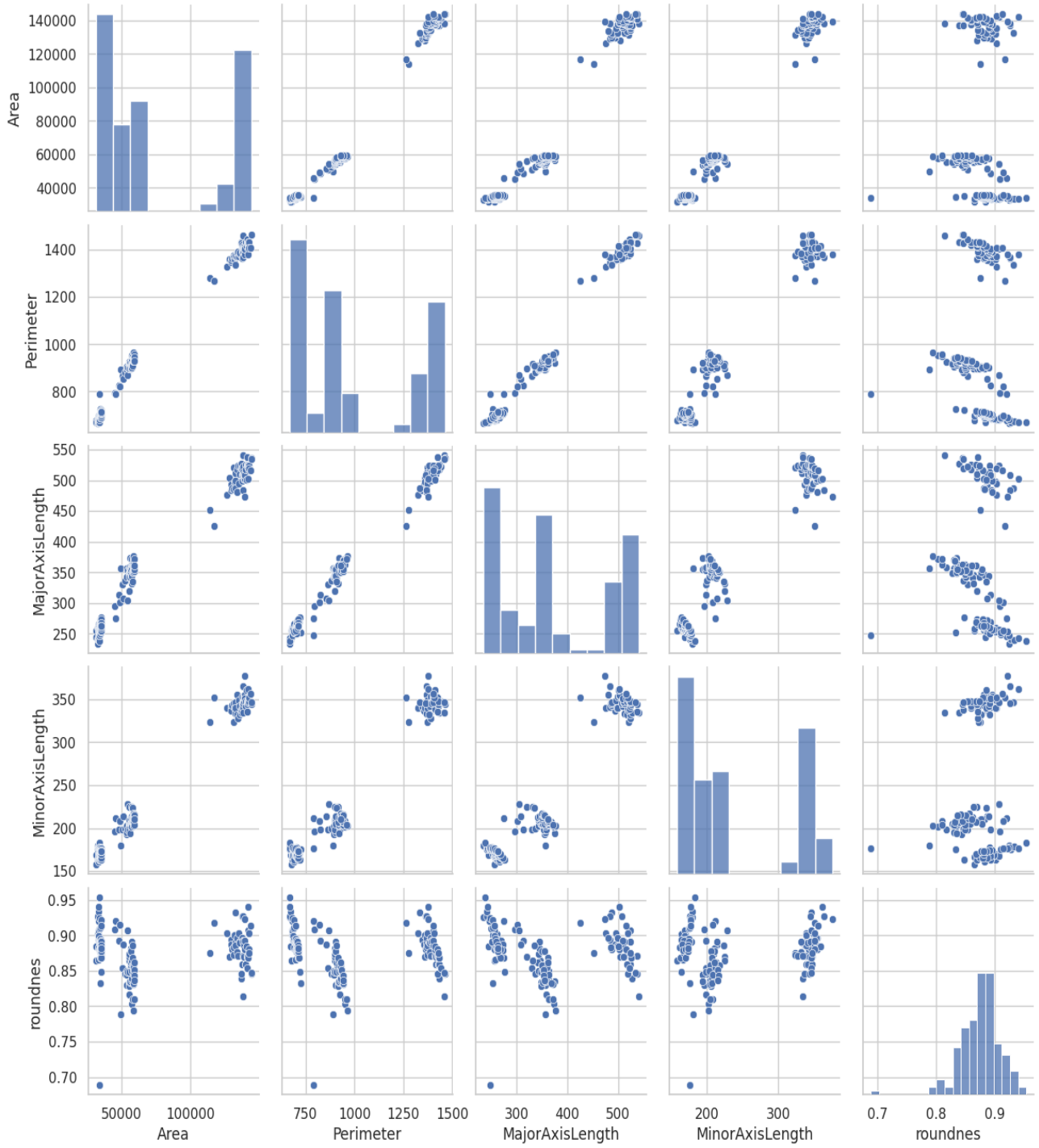
```
0      False
1      False
2      False
3      False
4      False
       ...
145    False
146    False
147    False
148    False
149    False
Length: 150, dtype: bool
```

```
data.duplicated().sum()
```

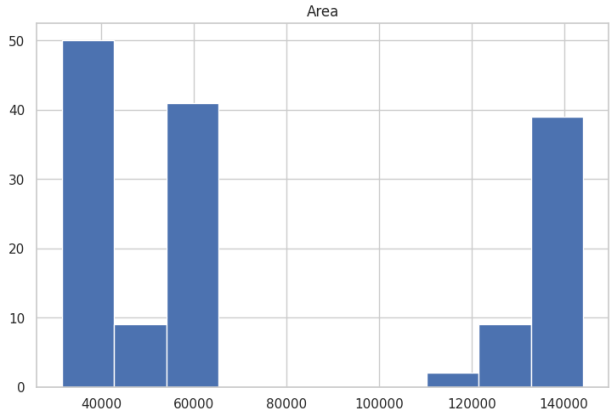
```
0
```

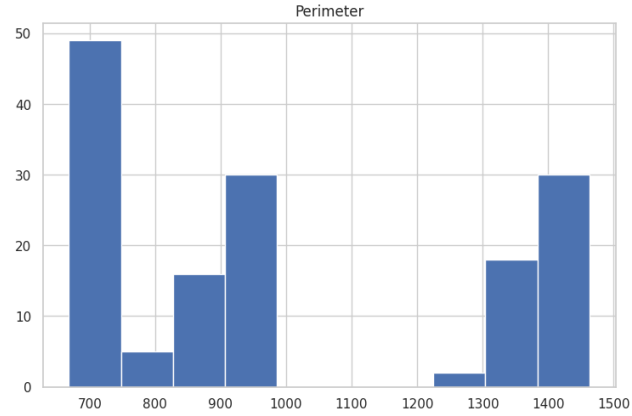
Data Visualization

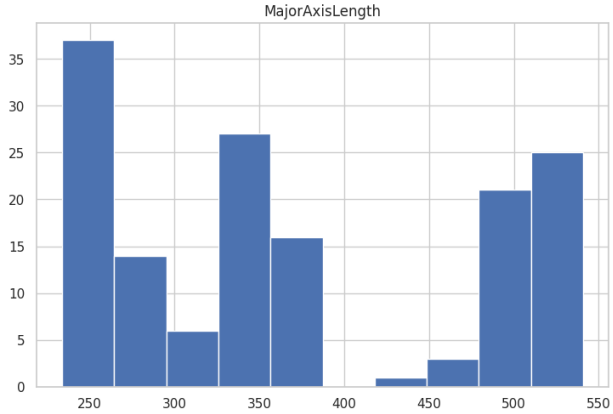
- **Relationships between Features dataset**

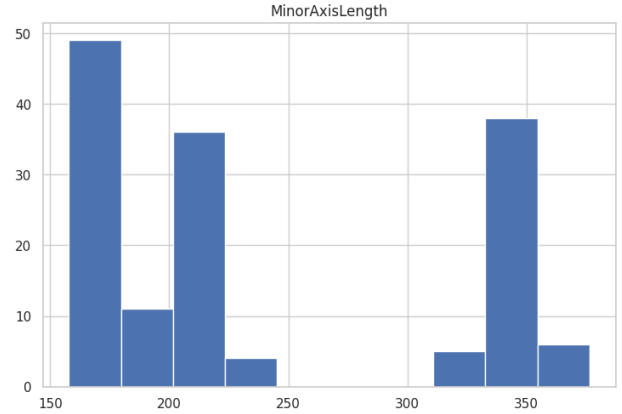


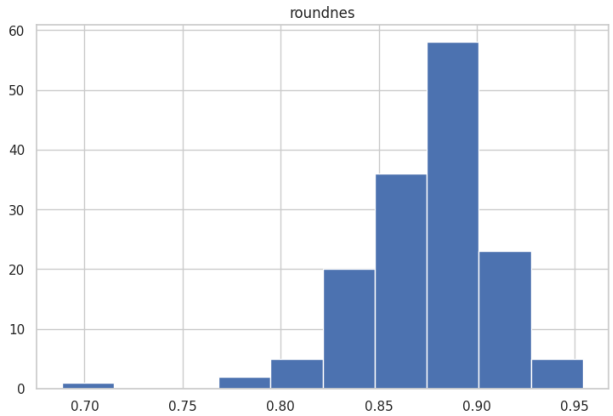
- **Display distribution of values in each feature**



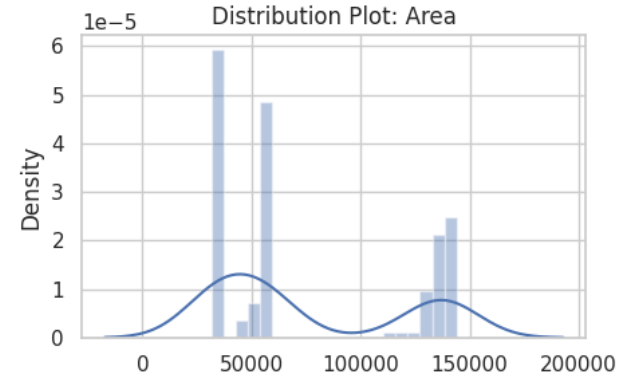


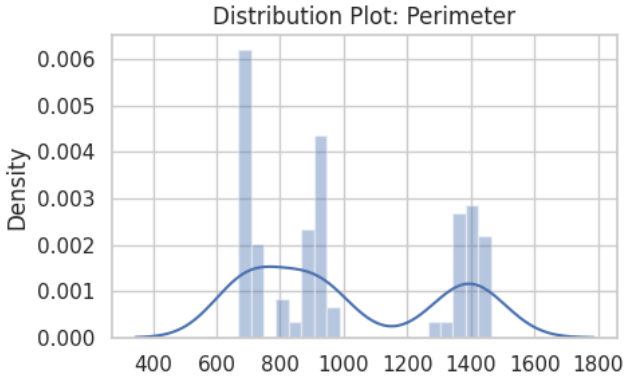


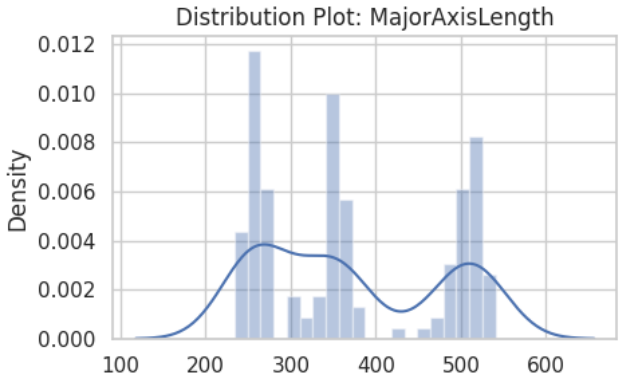


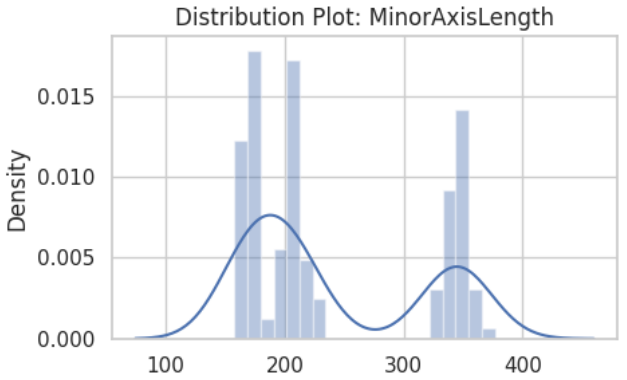


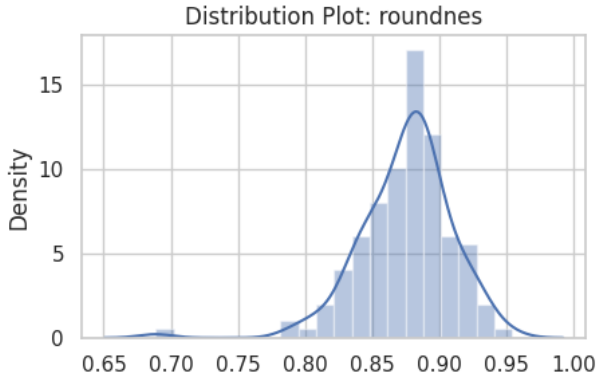
- **Display distribution of values in each feature**



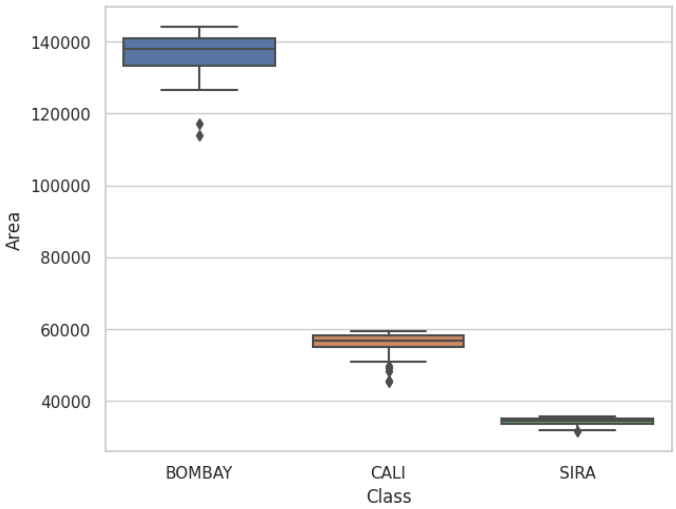


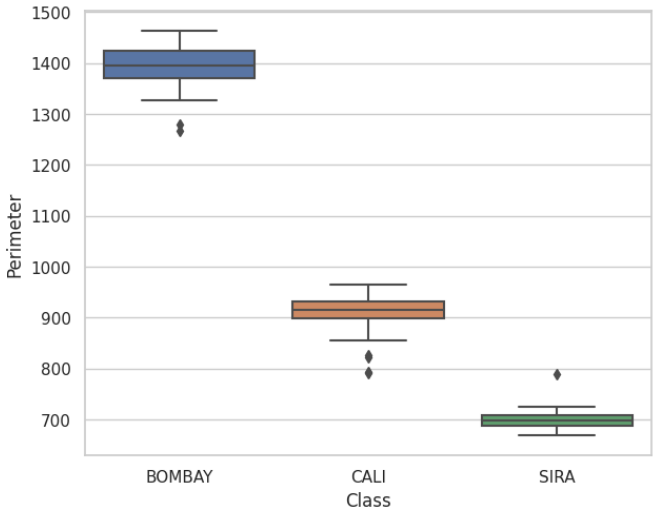


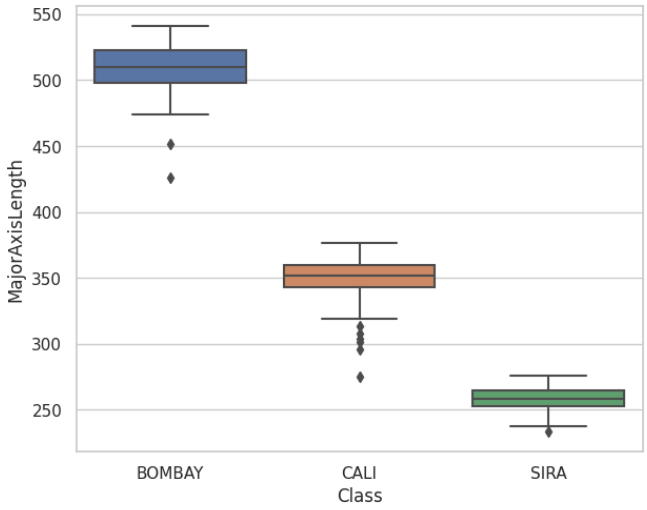


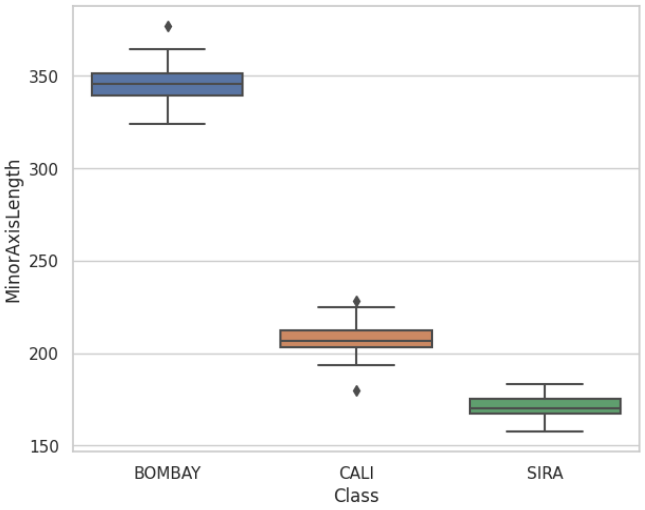


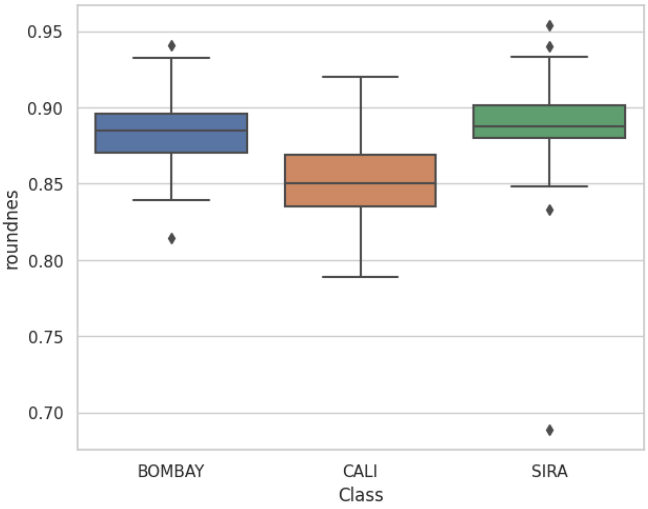
- **Boxplot of numerical features for each type of bean**



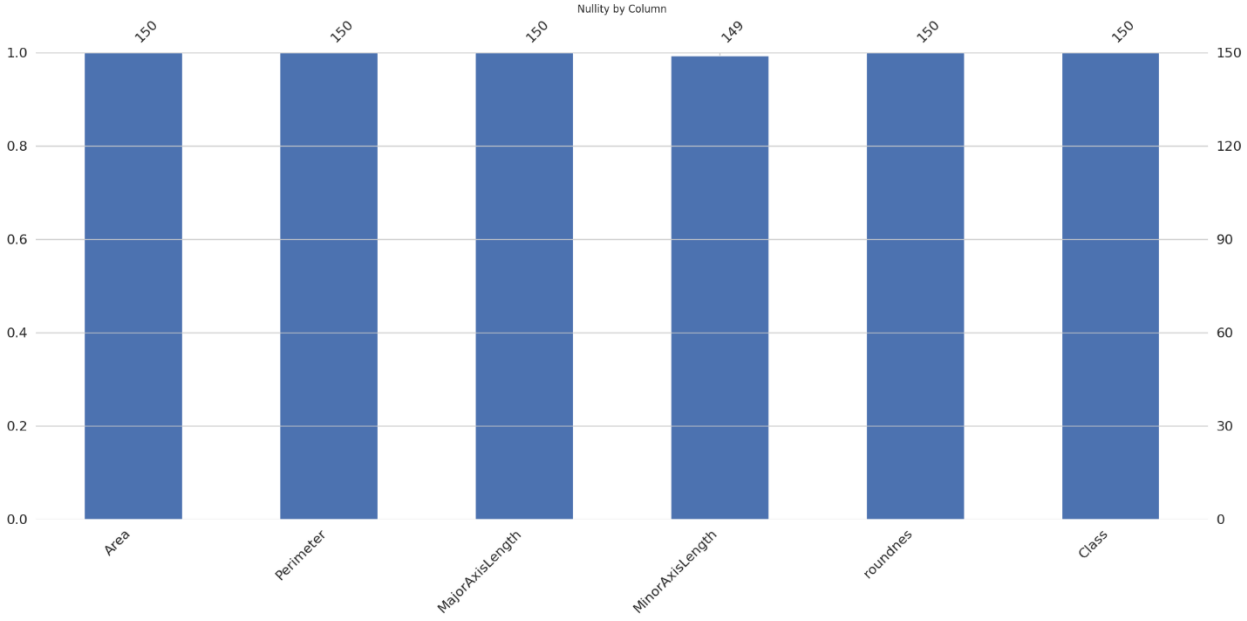






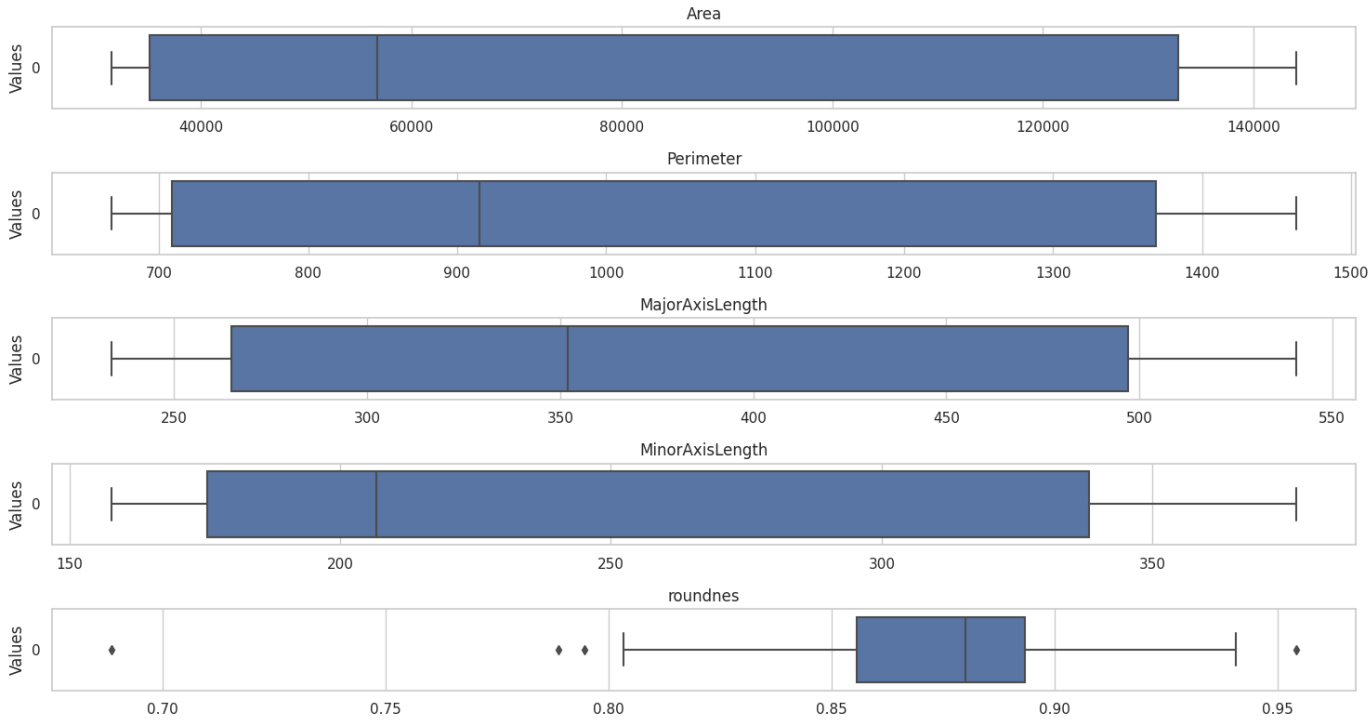


- **Check for Missing values**

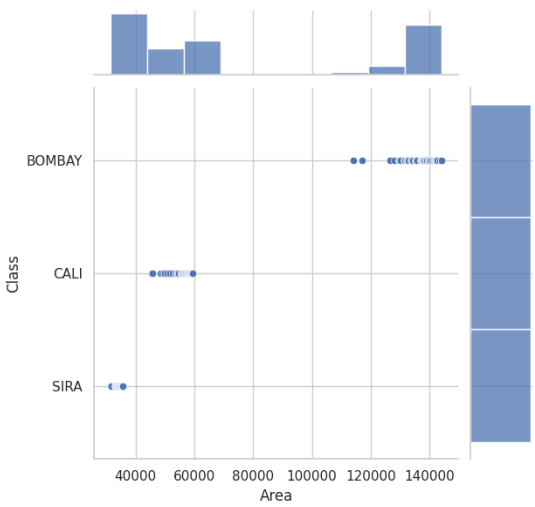


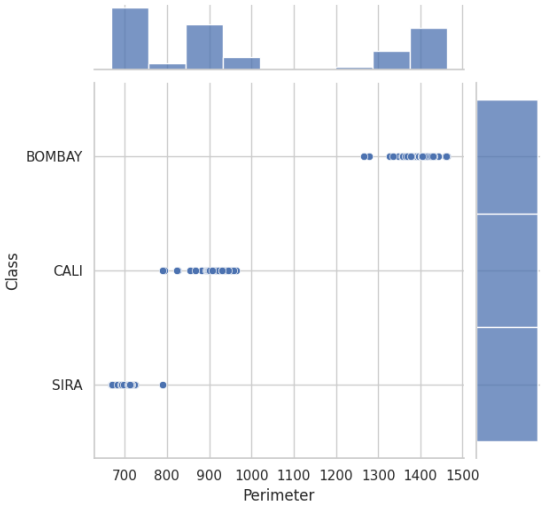
There is a missing value in MinorAxisLength column

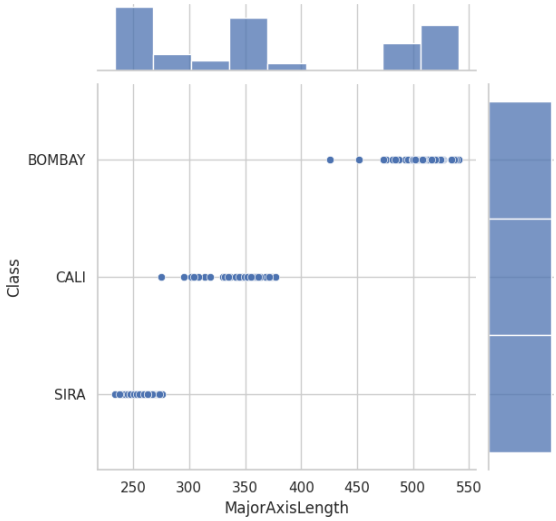
- **Check for outliers**

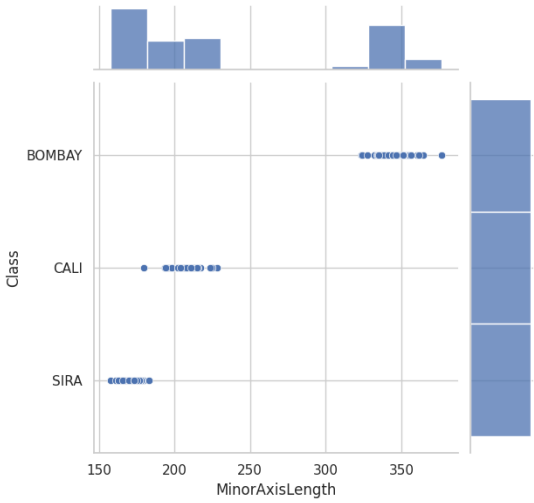


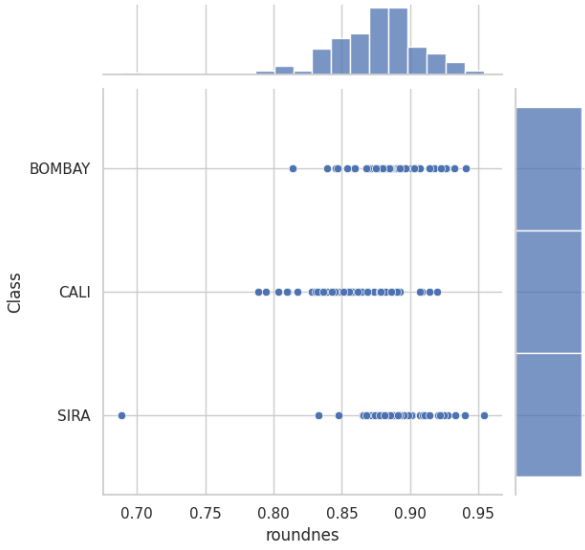
- **visualize the relationship between Features and Class**



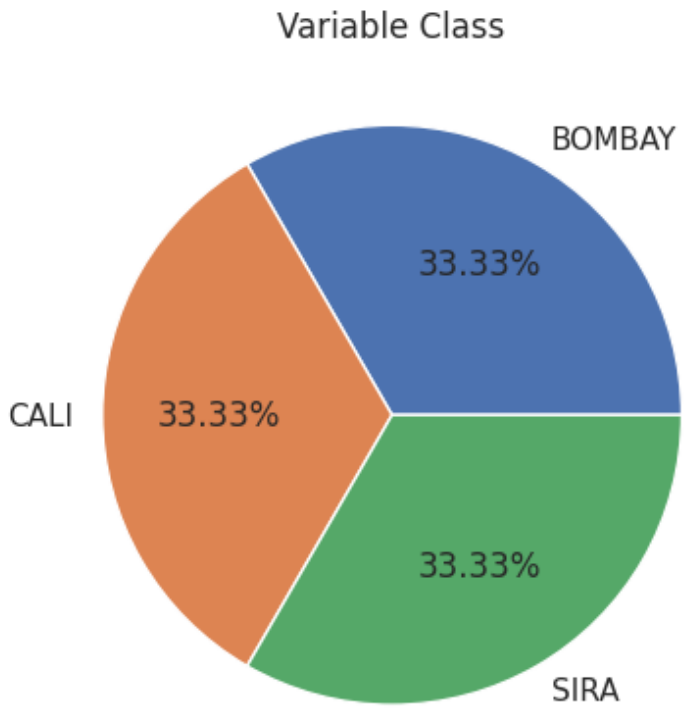


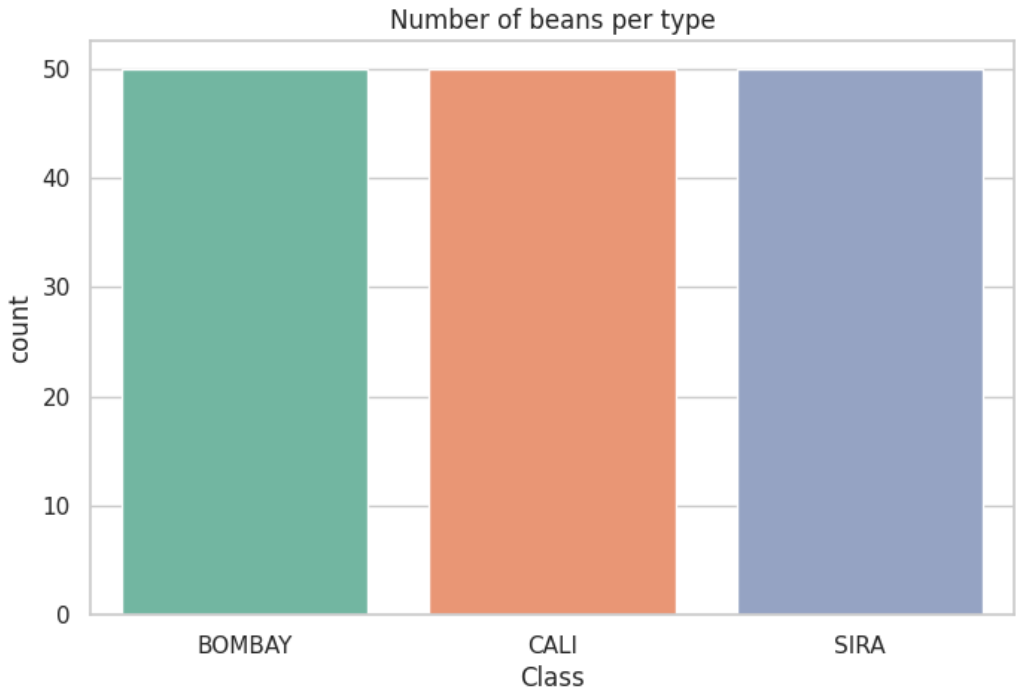




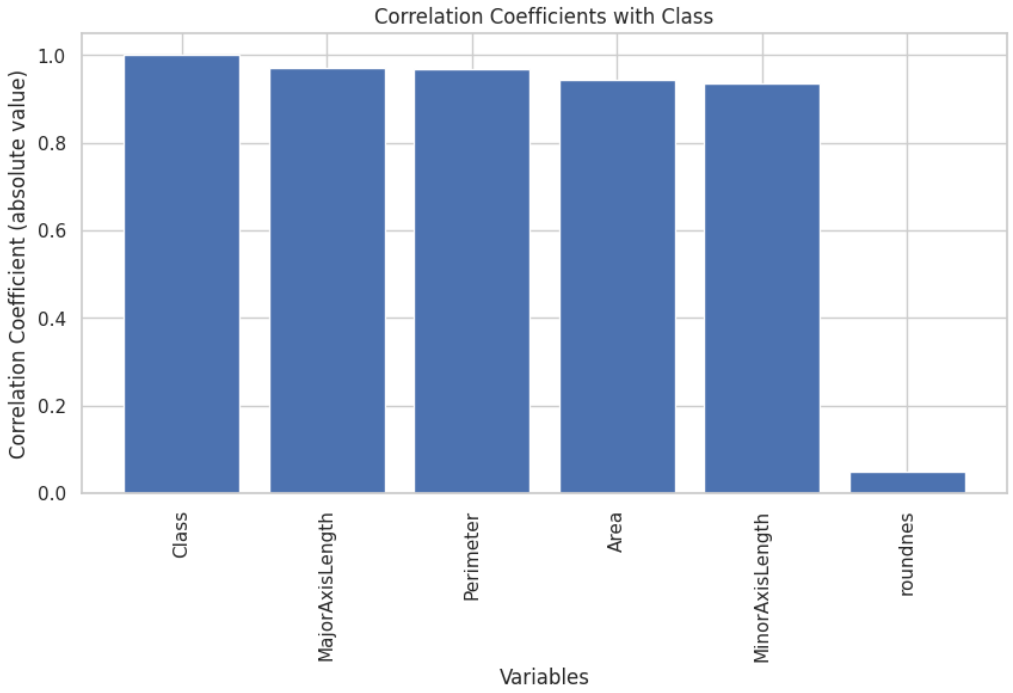


- **Display beans per type**

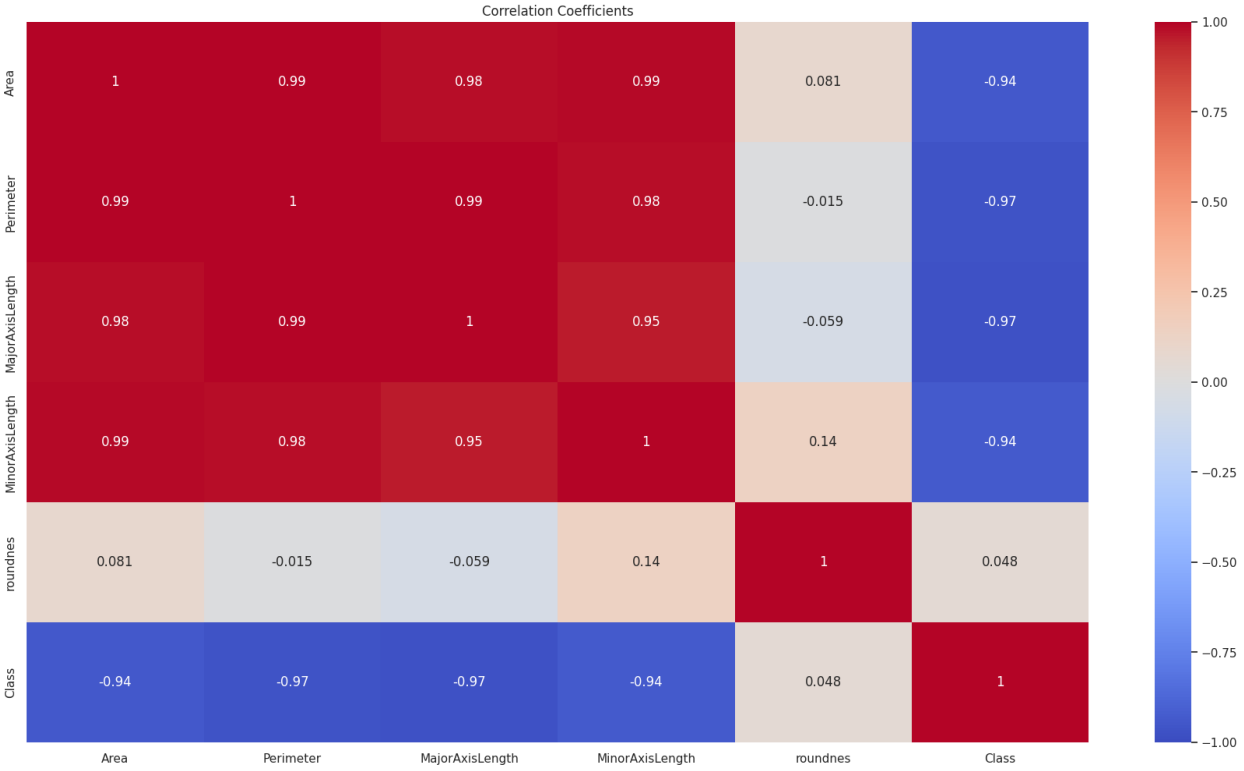




- **Visualize the correlation between the columns (bar graph)**



- **Visualize the correlation between the columns (heatmap)**



Data Preprocessing

We prepared the data for model training by handling missing values, scaling, and encoding categorical target.

- **Handle Missing Values**

```
mean_minor = data['MinorAxisLength'].mean()
data['MinorAxisLength'].fillna(mean_minor, inplace=True)
```

```
data.isnull().sum().sort_values(ascending=False)
```

```
Area              0
Perimeter         0
MajorAxisLength   0
MinorAxisLength   0
roundnes          0
Class             0
dtype: int64
```

- **Train Test Split**

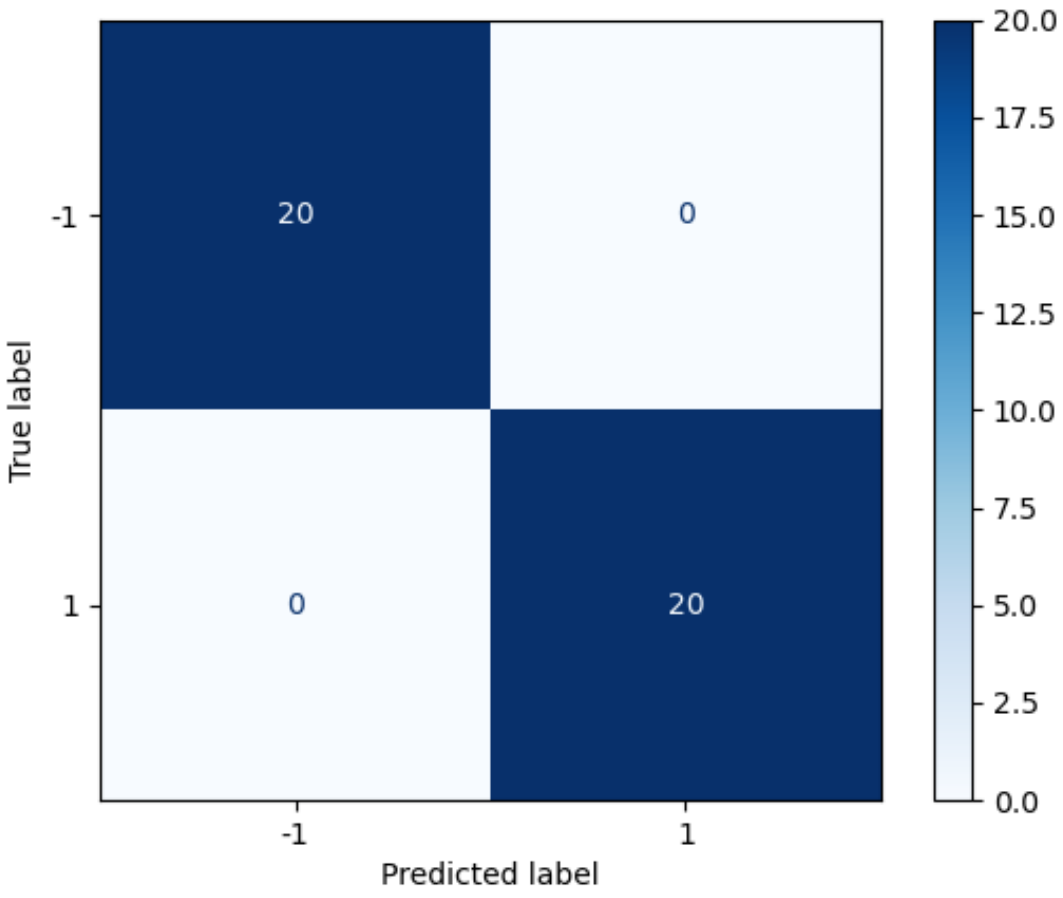
Splitting the data into training and testing sets (Each class has 50 samples: train NN with 30 non-repeated samples randomly selected and test it with the remaining 20 samples).

Model Engineering & Evaluation

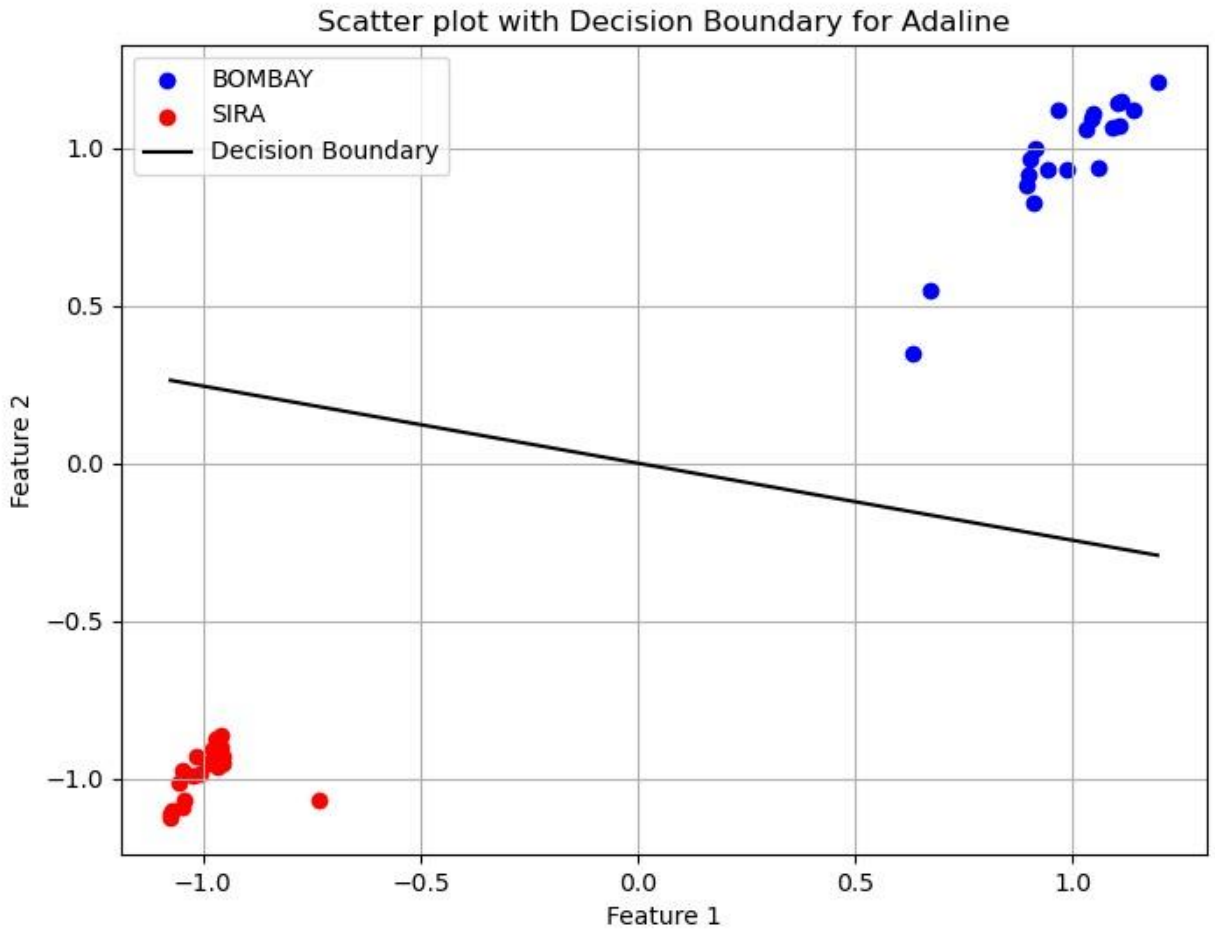
- **Adaline Algorithm**

We implemented the Adaline (Adaptive Linear Neuron) algorithm from scratch. Adaline is a single-layer neural network that uses linear activation and minimizes Mean Squared Error (MSE) during training. We coded the algorithm to handle the training process, weight updates, bias handling and predictions.

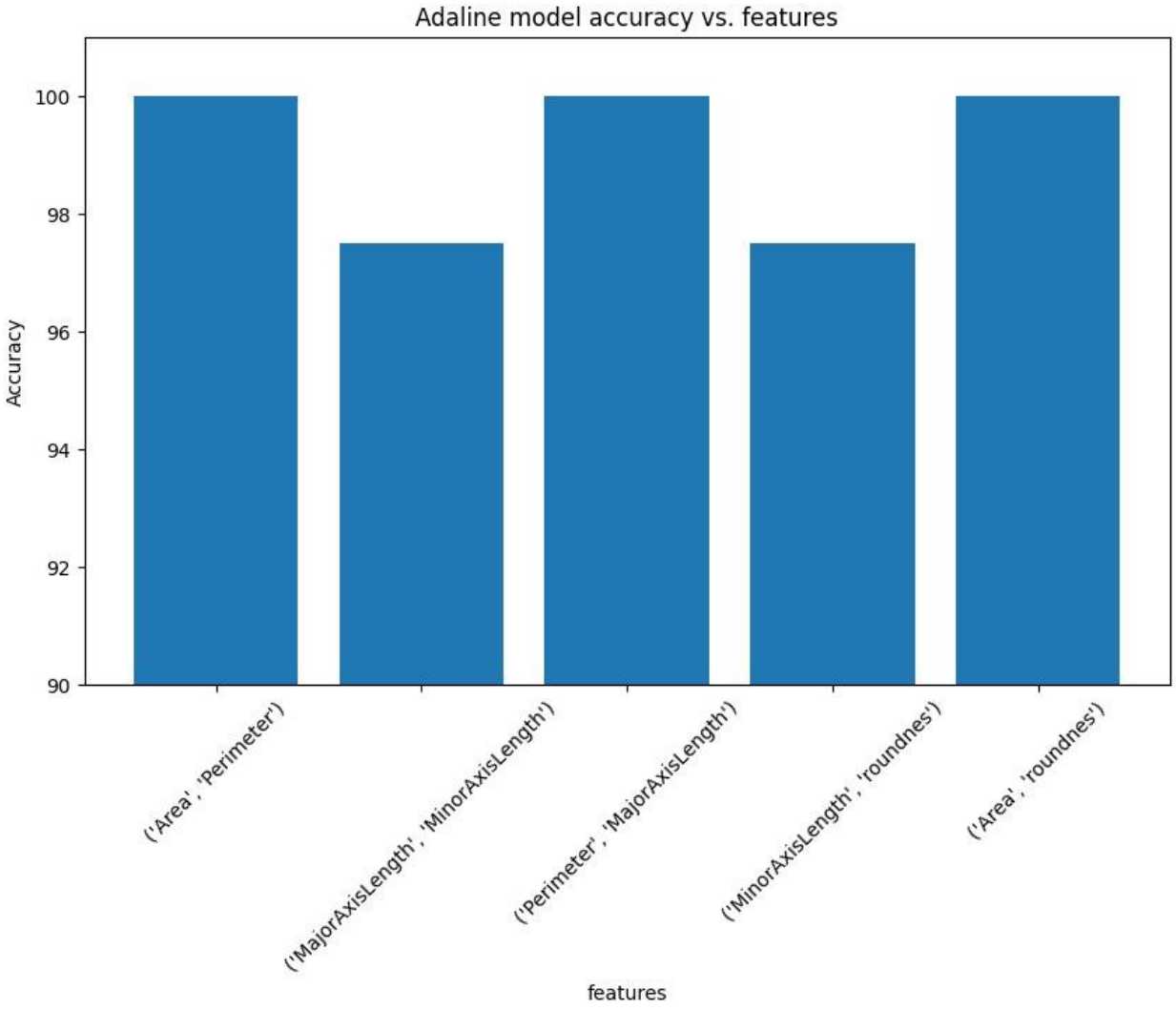
- **Adaline Algorithm Evaluation**
  - **Confusion matrix**



- **Plot the decision boundary and scatter plot**



- **Adaline model accuracy vs. features**

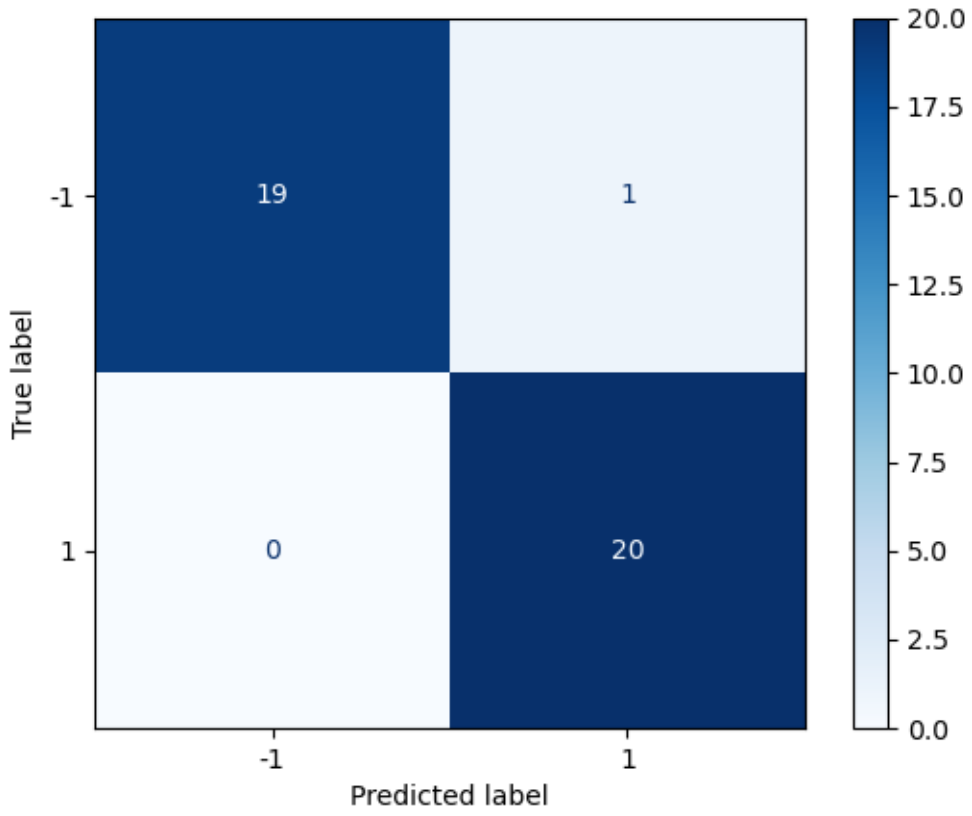


- **Hyperparameter Tuning for Adaline Algorithm**
  Best parameters that achieved the highest accuracy**:**
  1. **Learning Rate:** 0.001
  2. **Number of Epochs:** 200
  3. **Bias Term (Include/Exclude):** Include
  4. **Mean Squared Error (MSE) Threshold:** 0

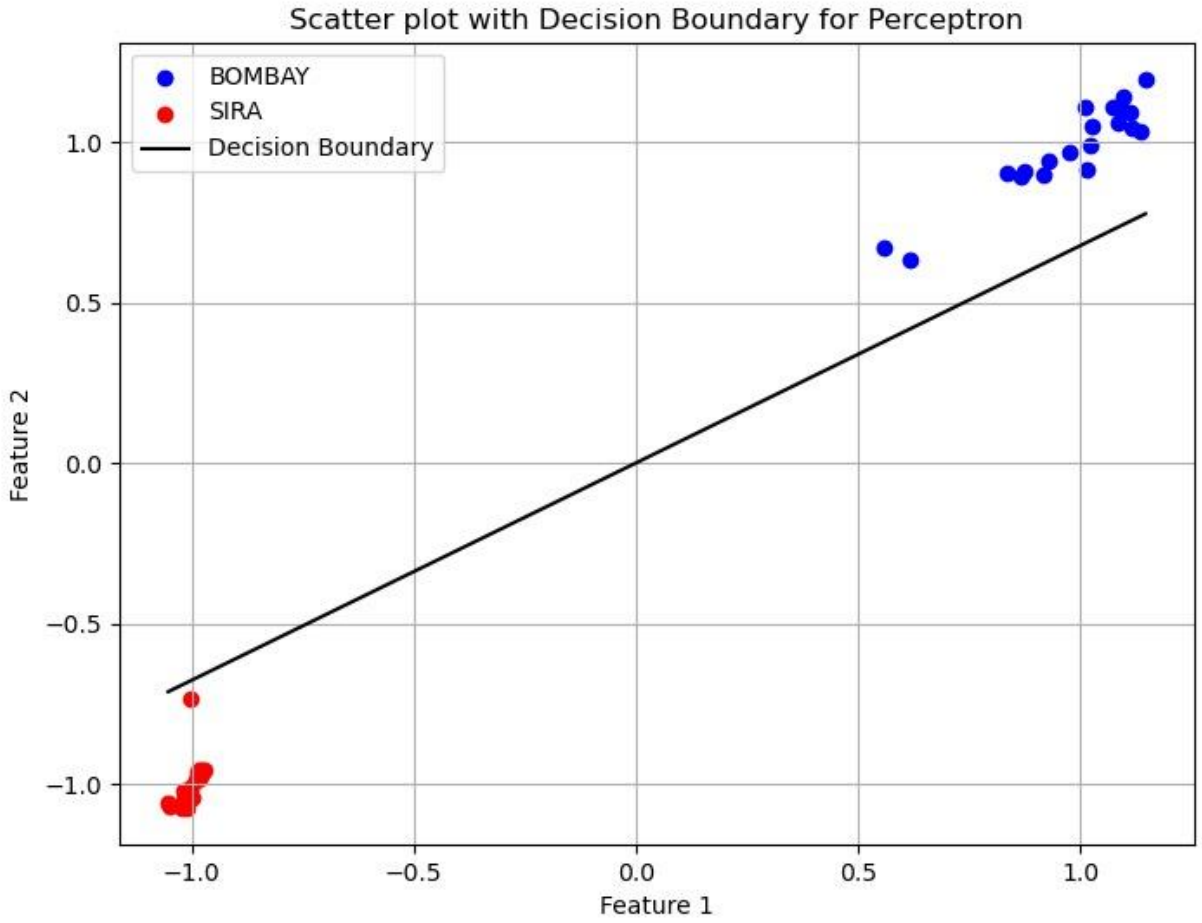
- **Perceptron Algorithm**

We implemented the Perceptron algorithm from scratch. The Perceptron is a simple binary classification algorithm that updates its weights and bias handling based on misclassified instances. We implemented the Perceptron algorithm to classify dry beans into the provided categories.

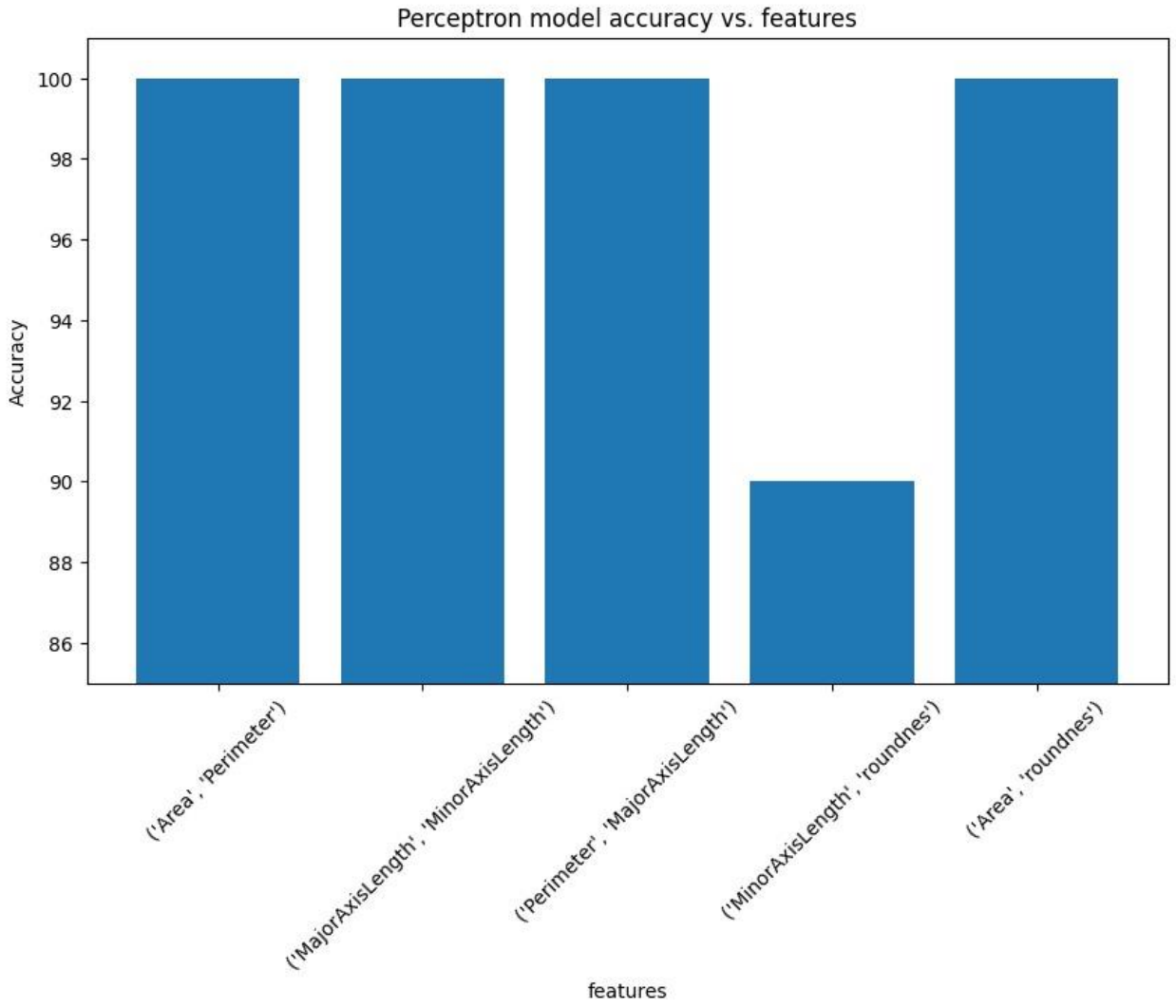
- **Perceptron Algorithm Evaluation**
  - **Confusion matrix**



- **Plot the decision boundary and scatter plot**



- **Perceptron model accuracy vs. features**



- **Hyperparameter Tuning for Perceptron Algorithm**
  Best parameters that achieved the highest accuracy**:**
  1. **Learning Rate:** 0.1
  2. **Number of Epochs:** 200
  3. **Bias Term (Include/Exclude):** Include

Identify High-Accuracy Features

We Compared the performance metrics of different feature subsets to identify which two features achieves the highest accuracy.

We found that the features (Perimeter - MajorAxisLength) and (Perimeter - Area) achieved the highest accuracy in the two models.

Model Deployment

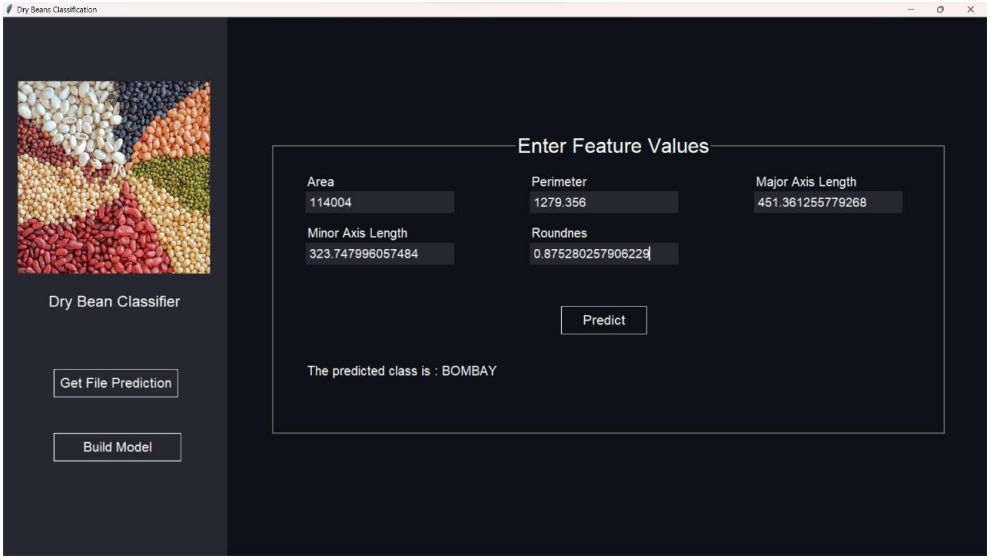
We developed a user-friendly GUI that allows users to interact with the model.

The GUI includes the following features:

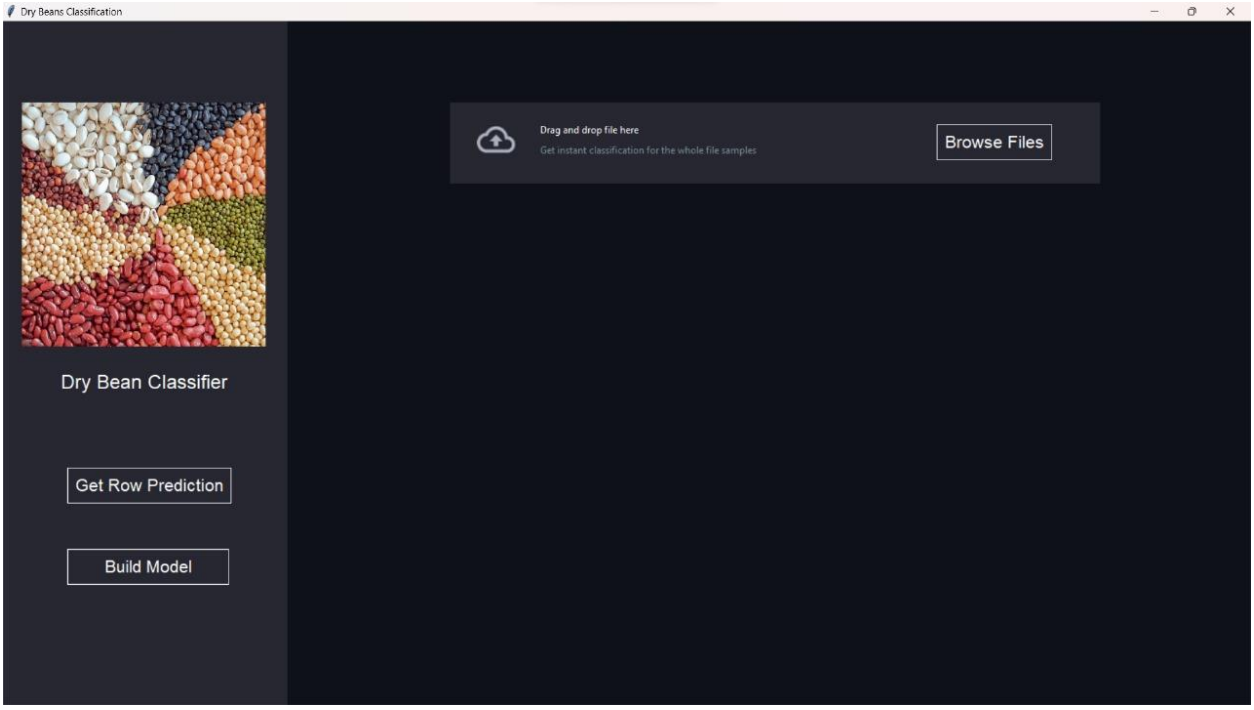
- **User Input:** Users can select two features, two classes, specify the learning rate (eta), set the number of epochs (m), enter an MSE threshold, and choose whether to add a bias term.
- **Algorithm Selection:** Users can choose between the Perceptron or Adaline algorithm for classification.
- **Test Data Input:** Users can browse and upload a test file or manually input data for prediction.
- **Prediction:** The GUI provides real-time prediction based on the selected features, classes, and model settings.
- **Overall Accuracy:** After making predictions, the GUI displays the overall accuracy of the model's predictions.
- **Intuitive Controls:** The GUI offers an intuitive interface for users to interact with the model without requiring programming knowledge.

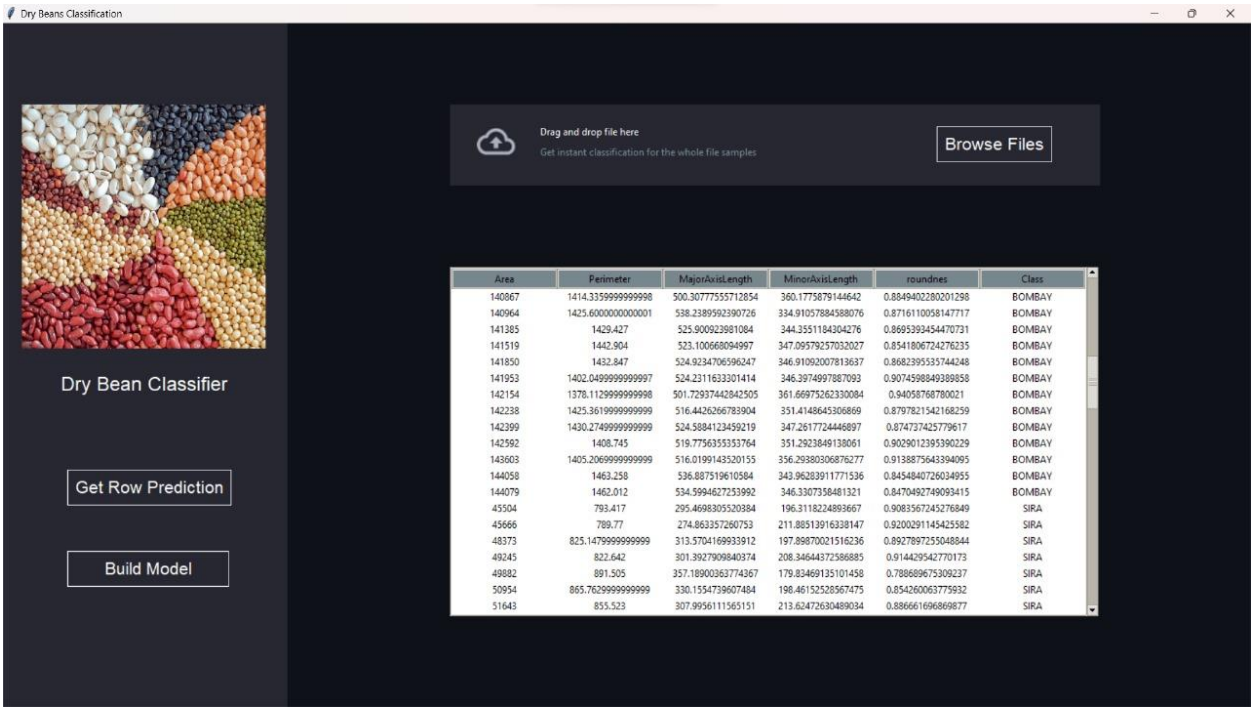
Screenshots:

1. **Single Prediction**



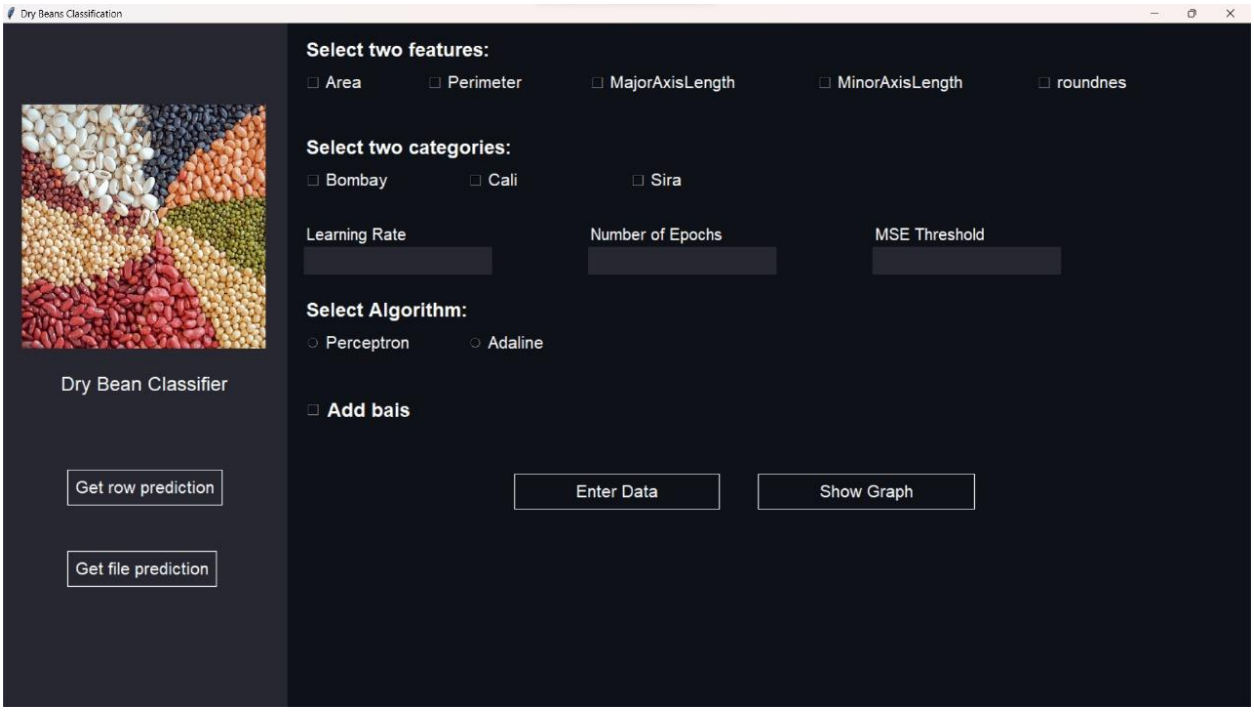
2. Batch Prediction using .xlsx file



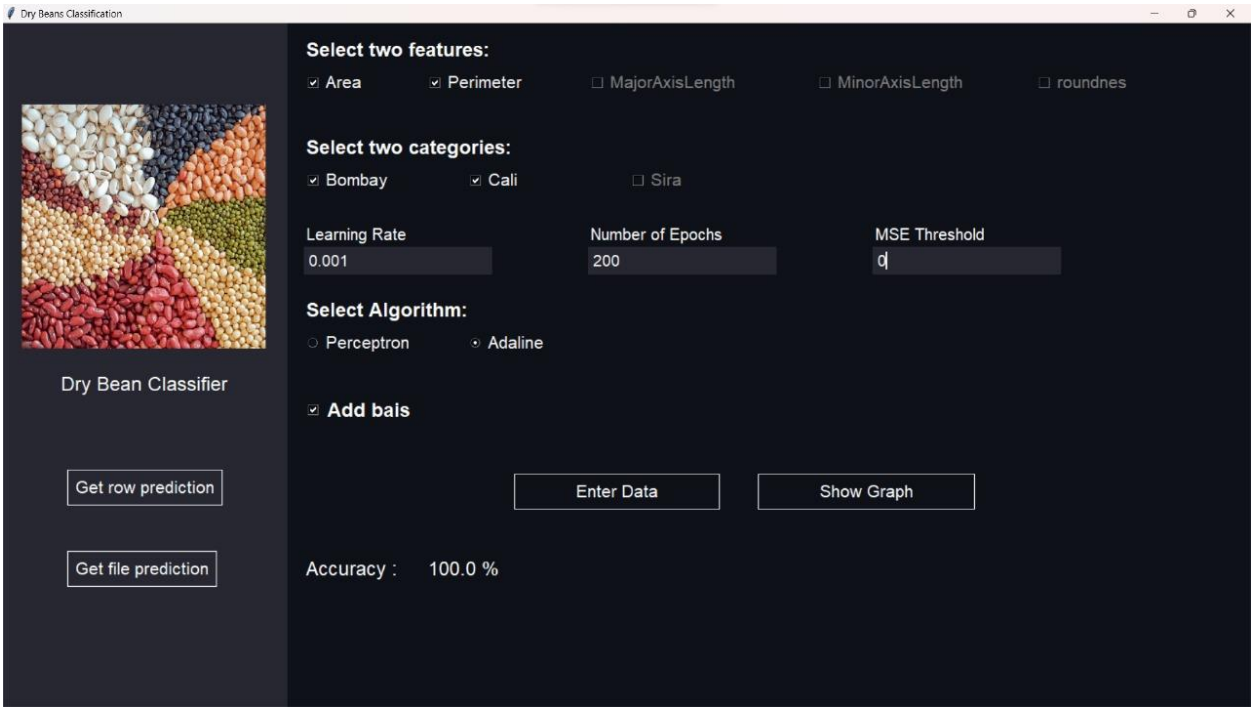


Area	Perimeter	MajorAxisLength	MinorAxisLength	roundnes	Class
140867	1414.3359999999998	500.30777555712854	360.1775879144642	0.8849402280201298	BOMBAY
140964	1425.6000000000001	538.2389592390726	334.91057884588076	0.8716110058147717	BOMBAY
141385	1429.427	525.900923981084	344.3551184304276	0.8695393454470731	BOMBAY
141519	1442.904	523.100668094997	347.09579257032027	0.8541806724276235	BOMBAY
141850	1432.847	524.9234706596247	346.91092007813637	0.8682395535744248	BOMBAY
141953	1402.0499999999997	524.2311633301414	346.3974997887093	0.9074598849389858	BOMBAY
142154	1378.1129999999998	501.72937442842505	361.66975262330084	0.94058768780021	BOMBAY
142238	1425.3619999999999	516.4426266783904	351.4148645306869	0.8797821542168259	BOMBAY
142399	1430.2749999999999	524.5884123459219	347.2617724446897	0.874737425779617	BOMBAY
142592	1408.745	519.7756355353764	351.2923849138061	0.9029012395390229	BOMBAY
143603	1405.2069999999999	516.0199143520155	356.29380306876277	0.9138875643394095	BOMBAY
144058	1463.258	536.887519610584	343.96283911771536	0.8454840726034955	BOMBAY
144079	1462.012	534.5994627253992	346.3307358481321	0.8470492749093415	BOMBAY
45504	793.417	295.4698305520384	196.3118224893667	0.9083567245276849	SIRA
45666	789.77	274.863357260753	211.88513916338147	0.9200291145425582	SIRA
48373	825.1479999999999	313.5704169933912	197.89870021516236	0.8927897255048844	SIRA
49245	822.642	301.3927909840374	208.34644372586885	0.914429542770173	SIRA
49882	891.505	357.18900363774367	179.83469135101458	0.788689675309237	SIRA
50954	865.7629999999999	330.1554739607484	198.46152528567475	0.8542600637759332	SIRA
51643	855.523	307.9956111565151	213.62472630489034	0.8866616968697877	SIRA

3. Build the Model



4. Model Prediction



Conclusion

The "Dry Beans Classification" project has successfully demonstrated the application of machine learning and classification algorithms to predict and classify three distinct categories of dry beans: Bombay, Cali, and Sira, based on their physical characteristics. This project has encompassed data exploration, model implementation, and the development of a user-friendly graphical user interface (GUI) to facilitate interaction with the models.