CS5483 Project 2

Exploring Data Preprocessing by Dry Bean Classification

Group 26: Law Kiu Tsz (57812965)

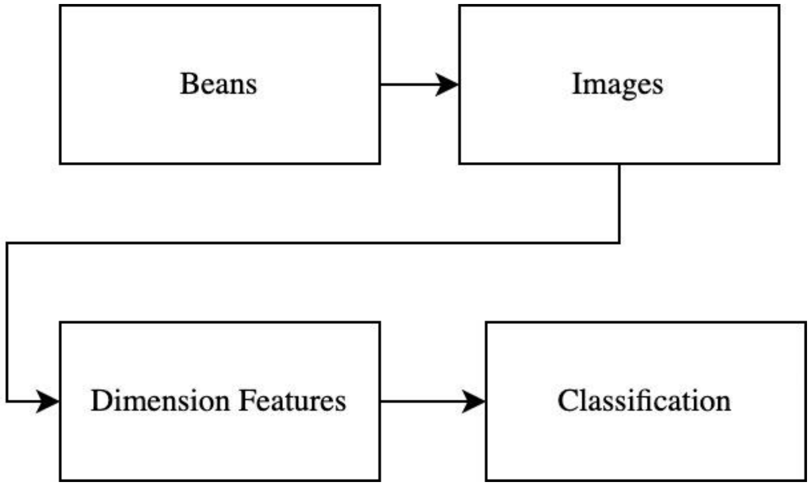
Tam Ho Fung (55773779)

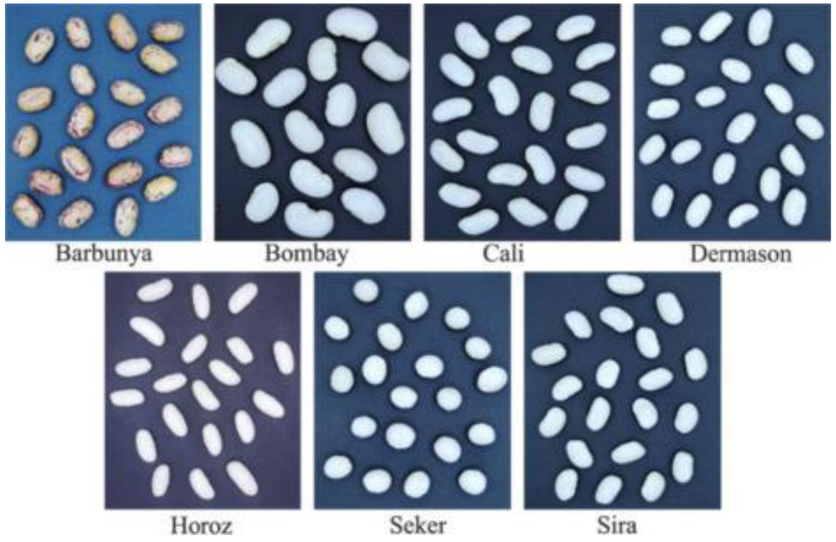
Nam Ho Sing (56681717)

Introduction

Introduction

- Classification of bean type by images
- Project initiated by M. Koklu and I.A. Ozkan
  - *"Multiclass classification of dry beans using computer vision and machine learning techniques"*
- Use computer vision system to obtain images
- From image processing to obtain dimension features





Literature Review

Literature Review

1. “Multiclass classification of dry beans using computer vision and Machine Learning Techniques”

M. Koklu and I. A. Ozkan

- Initialize this project
- Classification: SVM, DT, kNN, MLP

2. “Comparison of multiclass classification techniques using dry bean dataset”

M. Salauddin Khan et al.

- Preprocessing: ADASYN
- - Classifiers: LR, KNN, DT, RF, SVM, NB, XGB, MLP

Literature Review

3. “Dry bean cultivars classification using Deep CNN features and Salp Swarm algorithm based Extreme Learning Machine”

M. Dogan et al.

- Focus on convolutional neural network (CNN) and Extreme learning machine (ELM)

4. “Data mining approach for dry bean seeds classification”

J. C. Macuácua, J. A. Centeno, and C. Amisse

- Preprocessing: SMOTE, feature selection, PCA
- Classifiers: RF, SVM, KNN

Our Initiatives

1. Various classifications has been explored
   - Traditional classifiers: LR, KNN, DT, RF, SVM, NB, XGB, MLP
   - Deep Learning: MLP, CNN, ELM
2. Not much exploration on preprocessing
   - Preprocessing techniques: **SMOTE**, Adaptive Synthetic (ADASYN)
   - Most of the previous research uses the original data to do classification
3. Explore more on preprocessing techniques

Dataset

Dataset

Kaggle - Dry Bean Dataset Classification

https://www.kaggle.com/datasets/nimapourmoradi/dry-bean-dataset-classification/data

NIMA POURMORADI · UPDATED 2 MONTHS AGO

76 New Notebook Download (1 MB)

Dry Bean Dataset Classification

Images of 13,611 grains of 7 different registered dry beans.

Data Card Code (7) Discussion (0) Suggestions (0)

About Dataset

Relevant Information:

Seven different types of dry beans were used in this research, taking into account the features such as form, shape, type, and structure by the market situation. A computer vision system was developed to distinguish seven different registered varieties of dry beans with similar features in order to obtain uniform seed classification. For the classification model, images of 13,611 grains of 7 different registered dry beans were taken with a high-resolution camera. Bean images obtained by computer vision system were subjected to segmentation and feature extraction stages, and a total of 16 features; 12 dimensions and 4 shape forms, were obtained from the grains.

Usability
10.00

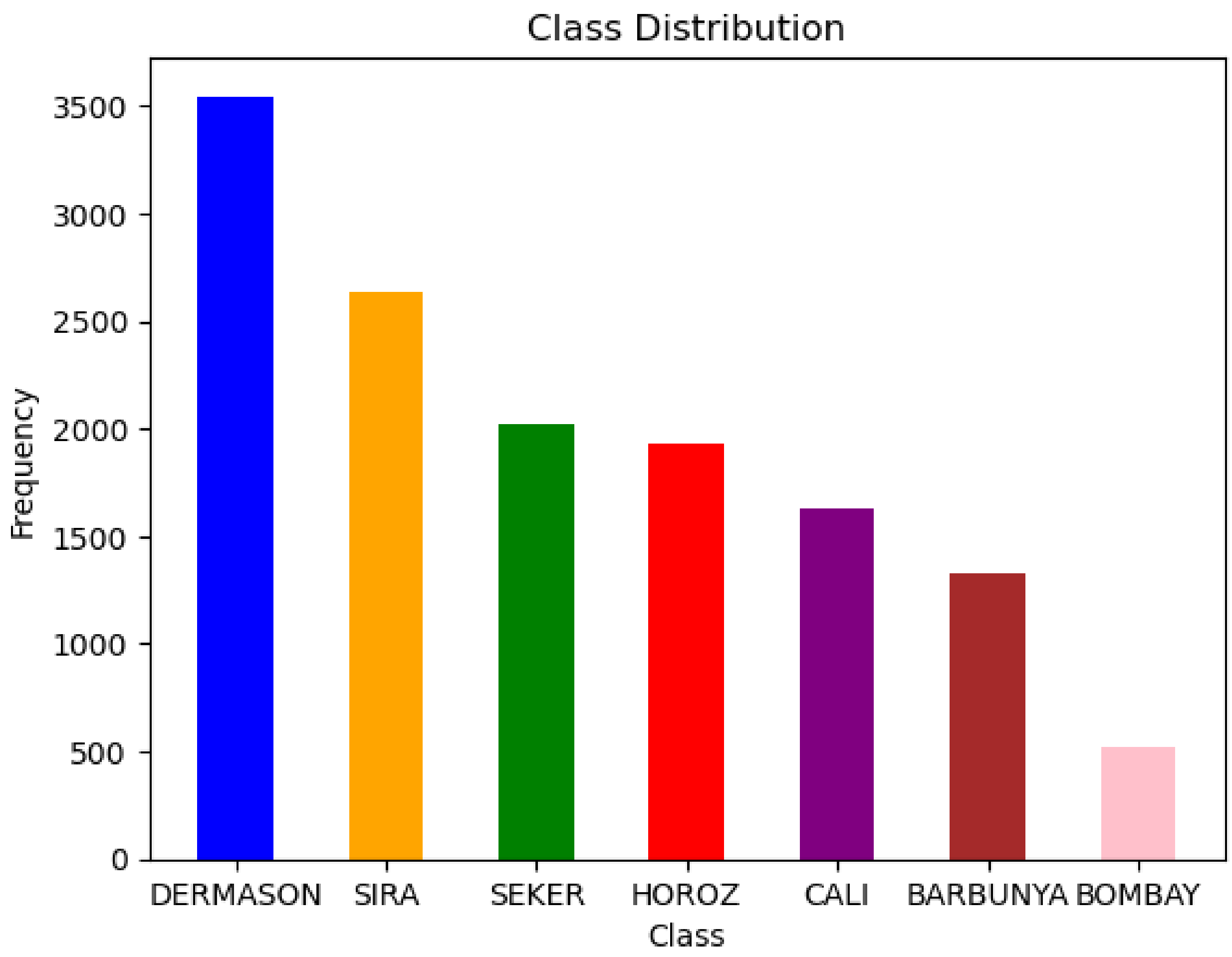
License
Other (specified in description)

Expected update frequency
Never

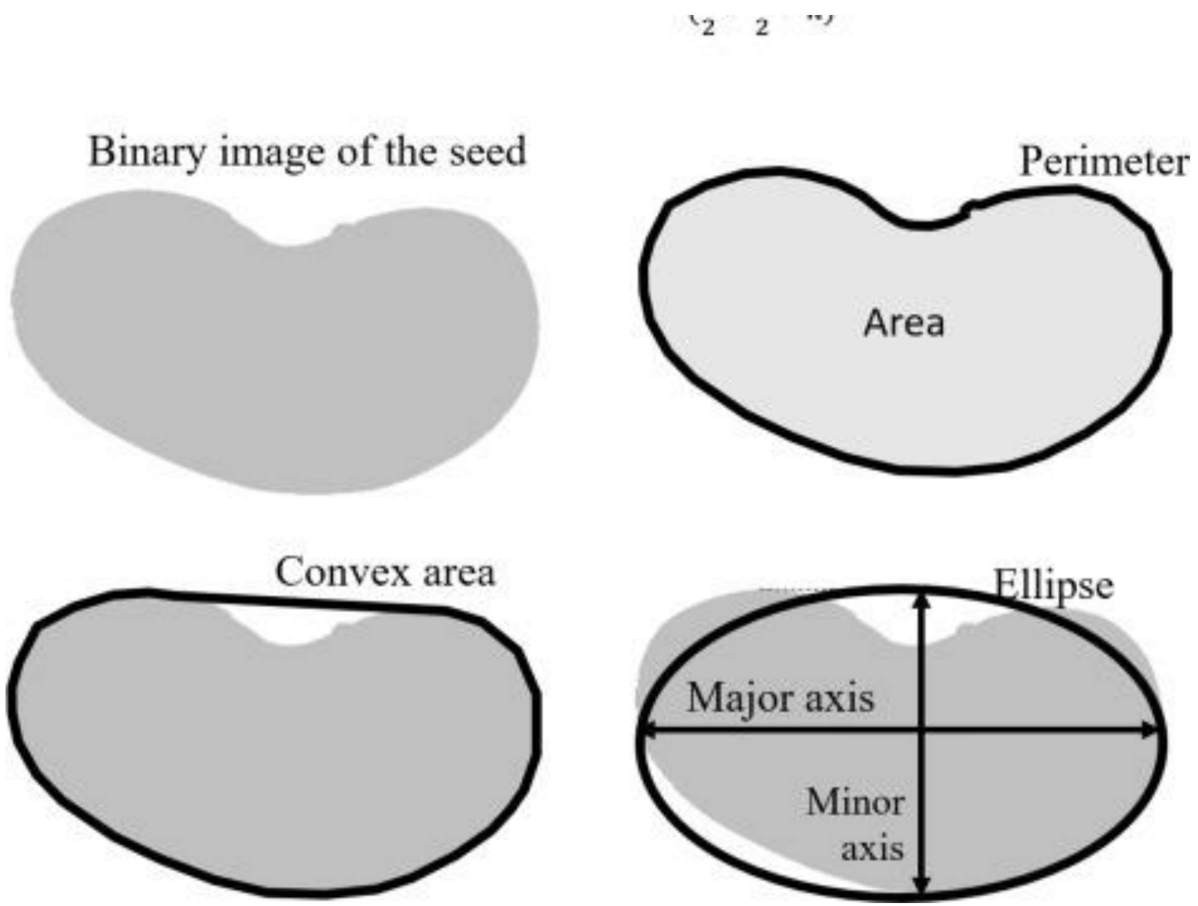
Tags
Earth and Nature · Food · Classification · Nutrition · Multiclass Classification

Dataset - Classes

Class	**Count**
DERMASON	3546
SIRA	2636
SEKER	2027
HOROZ	1928
CALI	1630
BARBUNYA	1322
BOMBAY	522
Total	13611



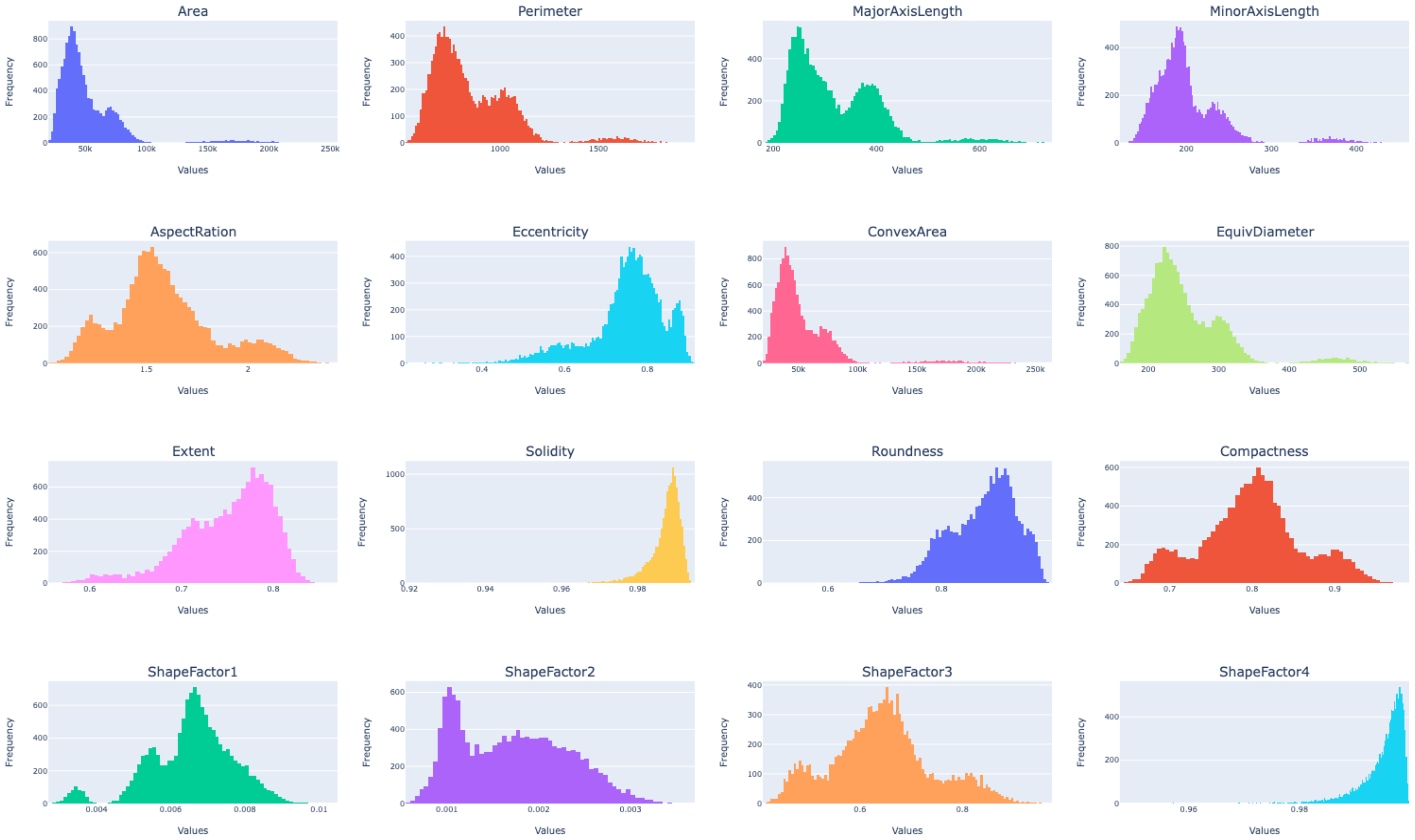
Dataset - Features



Dataset - Features

No	Feature	mean	std	min	0.25	0.5	0.75	max
1	Area	53048.28455	29324.09572	20420	36328	44652	61332	254616
2	Perimeter	855.283459	214.289696	524.736	703.5235	794.941	977.213	1985.37
3	MajorAxisLength	320.141867	85.694186	183.601165	253.303633	296.883367	376.495012	738.860154
4	MinorAxisLength	202.270714	44.970091	122.512653	175.84817	192.431733	217.031741	460.198497
5	AspectRation	1.583242	0.246678	1.024868	1.432307	1.551124	1.707109	2.430306
6	Eccentricity	0.750895	0.092002	0.218951	0.715928	0.764441	0.810466	0.911423
7	ConvexArea	53768.20021	29774.91582	20684	36714.5	45178	62294	263261
8	EquivDiameter	253.06422	59.17712	161.243764	215.068003	238.438026	279.446467	569.374358
9	Extent	0.749733	0.049086	0.555315	0.718634	0.759859	0.786851	0.866195
10	Solidity	0.987143	0.00466	0.919246	0.98567	0.988283	0.990013	0.994677
11	Roundness	0.873282	0.05952	0.489618	0.832096	0.883157	0.916869	0.990685
12	Compactness	0.799864	0.061713	0.640577	0.762469	0.801277	0.83427	0.987303
13	ShapeFactor1	0.006564	0.001128	0.002778	0.0059	0.006645	0.007271	0.010451
14	ShapeFactor2	0.001716	0.000596	0.000564	0.001154	0.001694	0.00217	0.003665
15	ShapeFactor3	0.64359	0.098996	0.410339	0.581359	0.642044	0.696006	0.974767
16	ShapeFactor4	0.995063	0.004366	0.947687	0.993703	0.996386	0.997883	0.999733

Dataset - Features



Methodology

Methodology

- **Data Preprocessing**
  - **Feature Selection**
    - CFS
    - InfoGain
  - **Normalization**
    - Z-score
    - Min-max
  - **Noise**
    - Gaussian
  - **Class Imbalance**
    - SMOTE
- **Classification**
  - Random Forest
  - AdaBoost
  - KNN
  - Decision Tree (CART)
- **Evaluation**
  - PRC
  - ROC

Preprocessing - Feature Selection

- No feature selection in the previous researches

Feature Selection

1. Correlation-based Feature Selection Subset Evaluation

"Evaluates the worth of a subset of attributes by ***considering the individual predictive ability of each feature*** *along with the degree of redundancy between them.*

Subsets of features that are highly correlated with the class while having low intercorrelation are preferred."

2. Information Gain

$$Entropy(S) \equiv \sum_{i=1}^{c} -p_i log_2 p_i$$

Preprocessing - Normalization

1. Z-score Normalization

- Calculate z-score for each data point

$$Z = \frac{(x - \mu)}{\sigma}$$

- Z is the Z-score of the data point.
- x is the original value of the data point.
- μ is the mean of the feature.
- σ is the standard deviation of the feature.

2. Min-max Normalization

- Calculate Min-Max for each data point

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

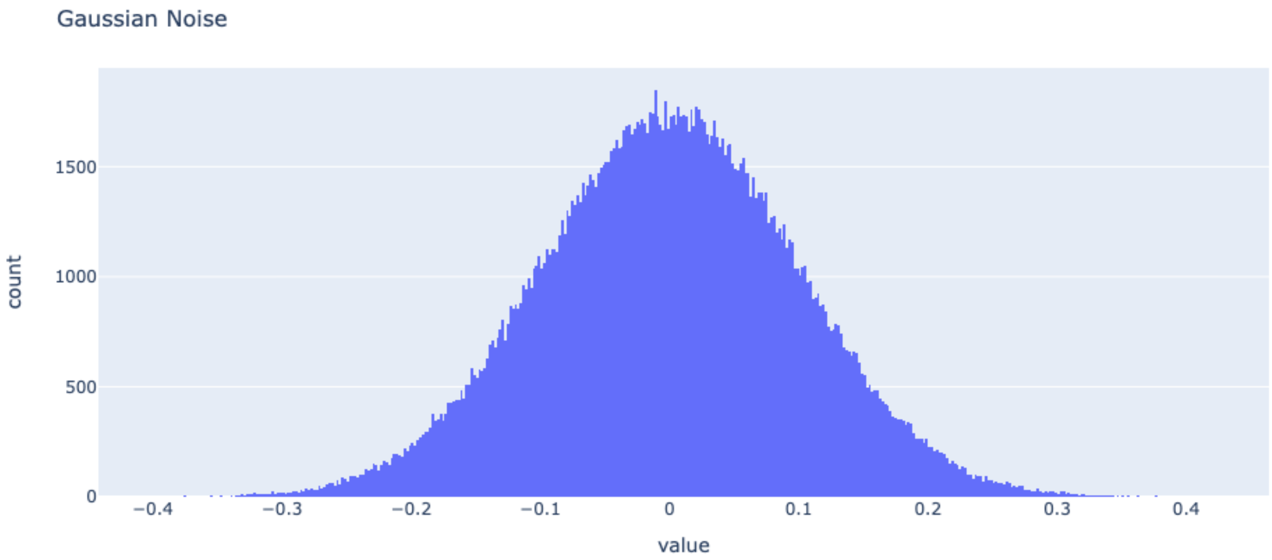
- X' is the normalized value of the data point.
- X is the original value of the data point.
- X_{min} is the minimum value of the feature.
- X_{max} is the maximum value of the feature.

Differences

- Z-score normalization
  - Handles the outliners better
  - Standard deviation were used
    - individual values will carry less weight
- Min-Max normalization
  - The outliner itself should be the values used to normalize the data
  - Will impact the result of normalization

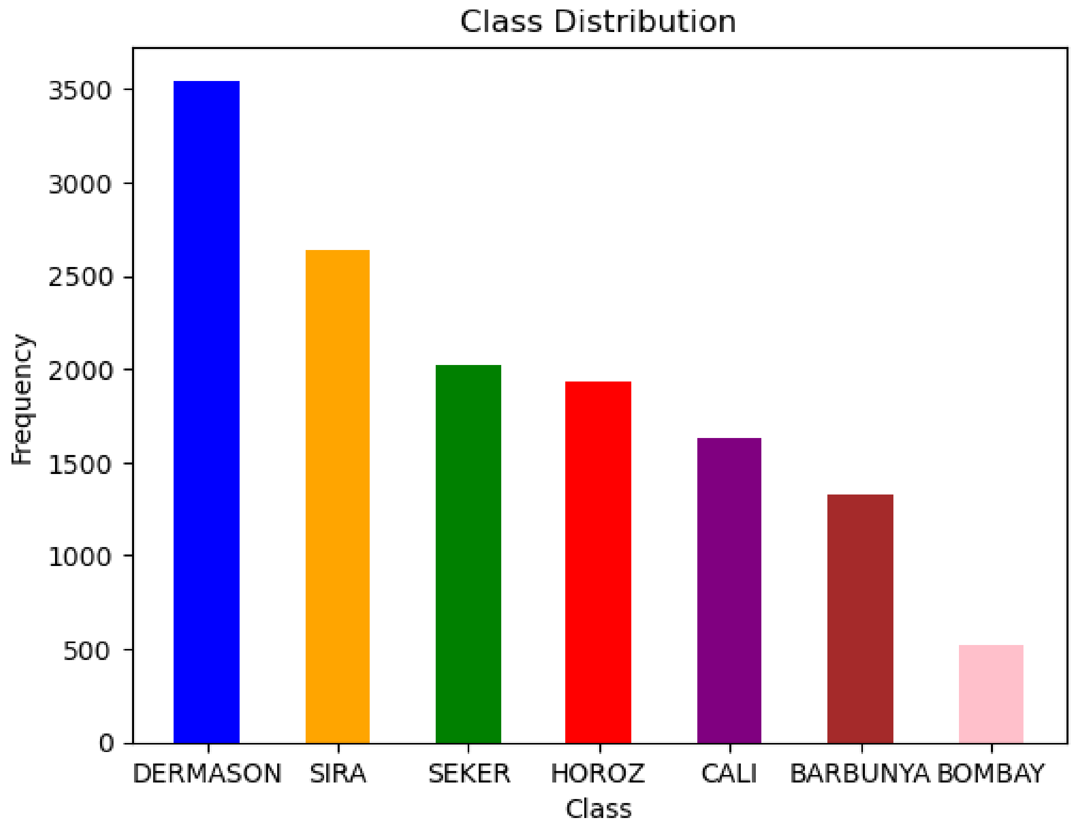
Preprocessing - Noise

- Inspired by image processing
- Noise is introduced to make the classifier more robust
  - Prevent overfitting
- **Introduce Gaussian Noise to Z-score normalized data**
- **10% Gaussian Noise**
  - Mean: 0
  - StdDiv: 0.1



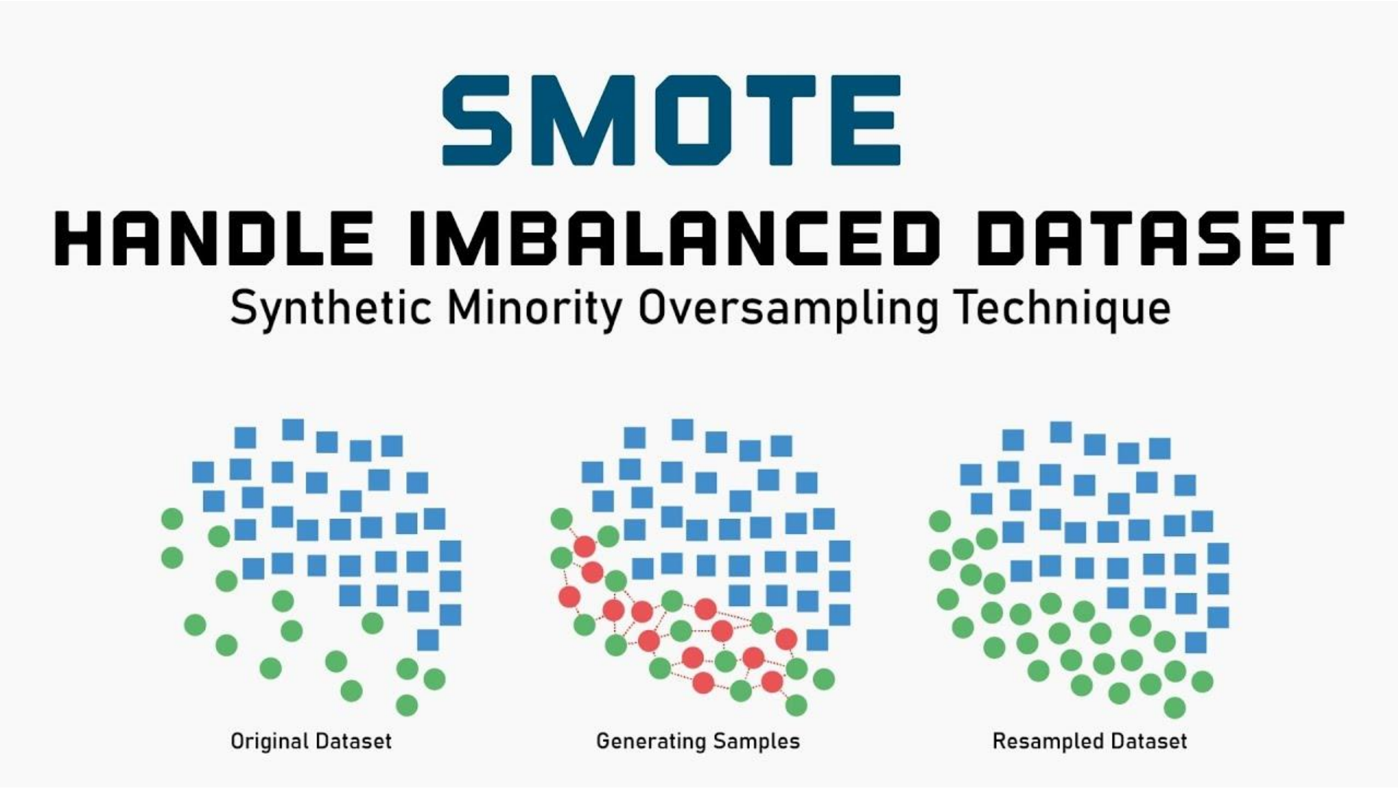
Preprocessing - SMOTE

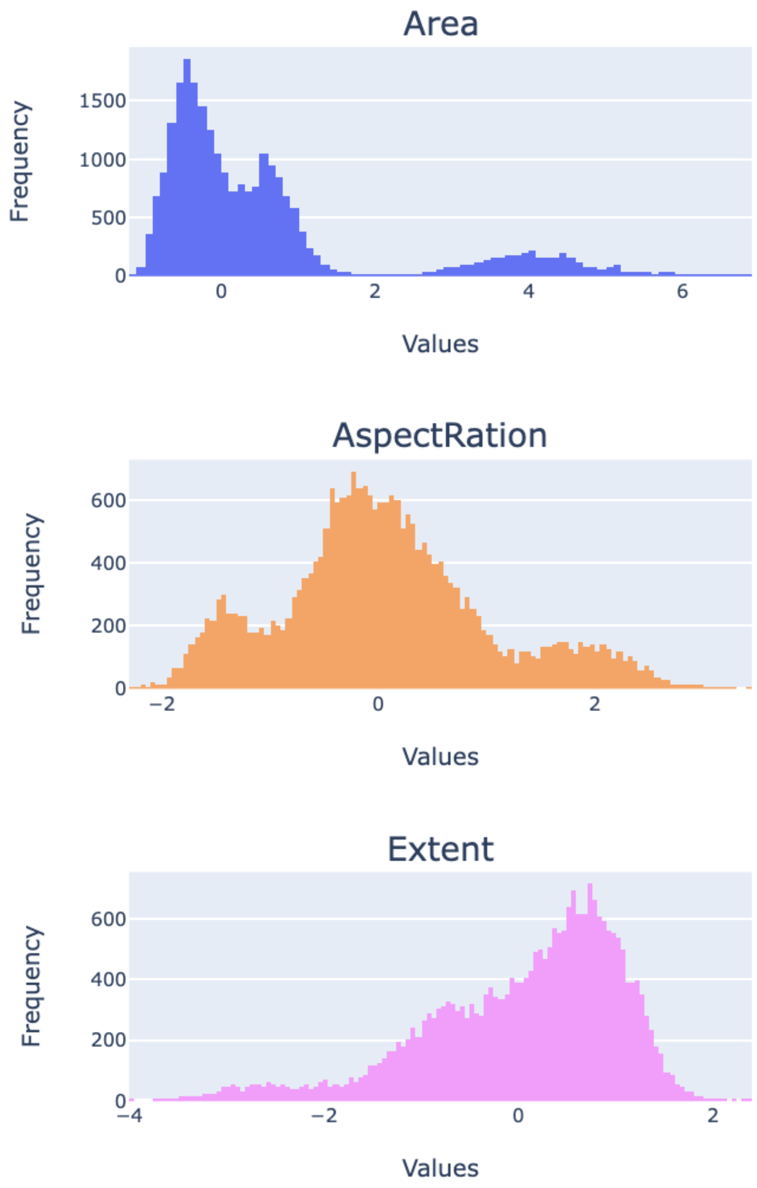
- There are class imbalance
  - BOMBWAY
- Too few samples
  - May not able to learn the decision boundary effectively.



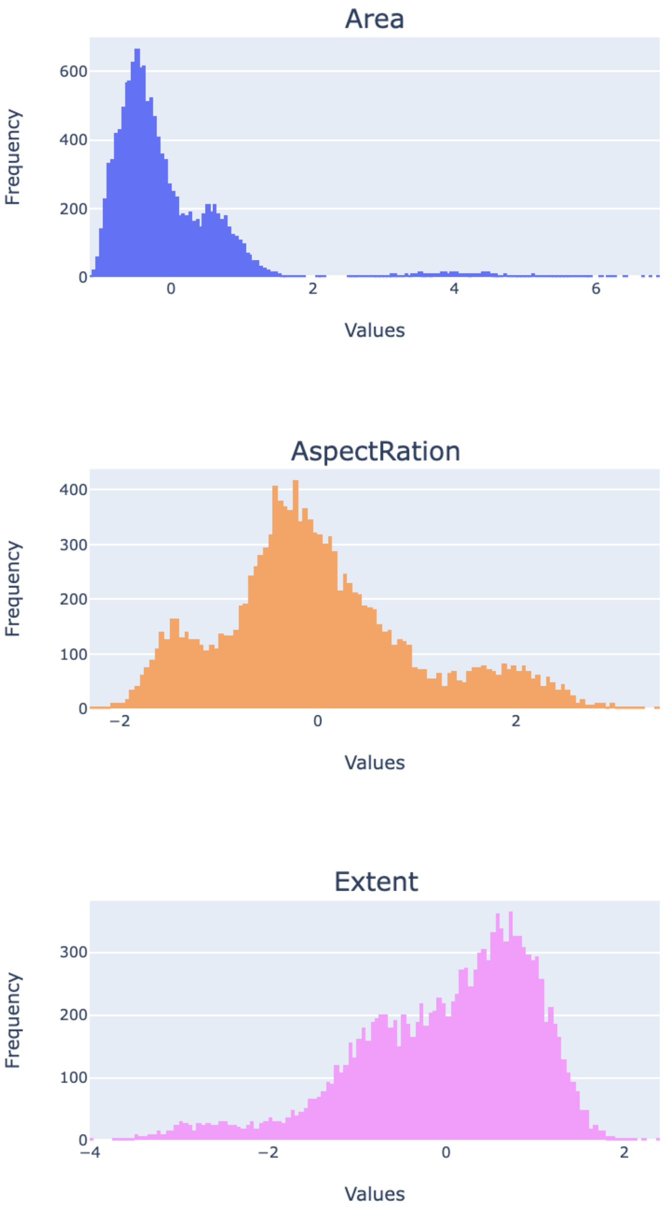
Preprocessing - SMOTE

- Generate samples for the minority classes
  - Select examples that are close in the feature space
  - Generate a new sample between two selected samples
  - Add the samples randomly to the points until the data imbalanced were solved





Z-Score



Z-Score + SMOTE(training data)

Limitations

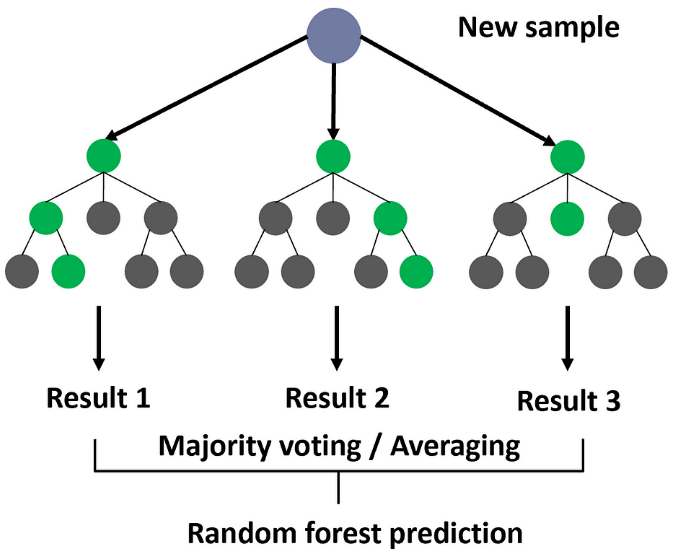
- The generated data may not able to accurately represent the data pattern in reality
  - May cause overfitting
- SMOTE assumes that the data in minority class are close.
  - if the data quality is poor
  - samples created may not be representative
- Validations are important

Methodology - Classification

- Experiments were conducted with 4 classifiers
  - Random Forest
  - AdaBoost (*Adaptive Boosting)*
  - KNN (*k-nearest Neighbors)*
  - Decision Tree (*CART)*

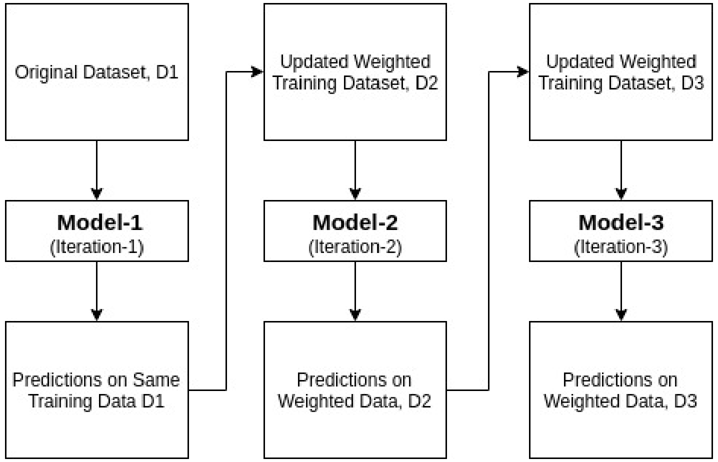
Classification - Random Forest

- Randomly sample from dataset with replacement
- Constructs multiple decision trees
- Uses majority vote among all trees to determine class
- Used in fields like banking and medicine



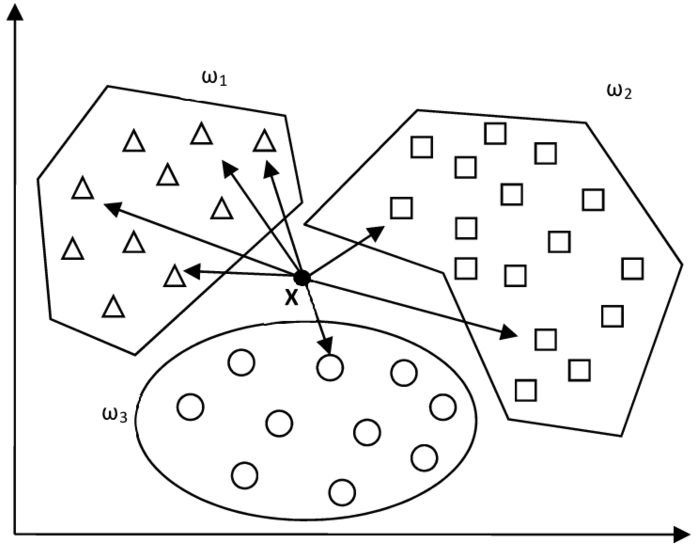
Classification - AdaBoost

- Train model repeatedly via multiple iterations
- In each iteration incorrectly classified result gain higher weight
- Weight increases probability of them being used in next iteration
- Repeat iterations until max number or when data fit with no error
- Used in computer vision and natural language processing



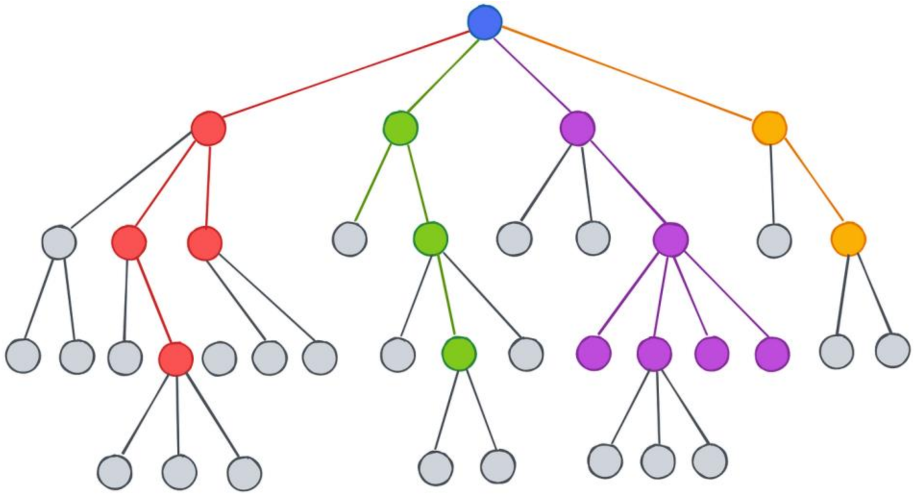
Classification - KNN

- Use distance between data points to determine a target's neighbor
- The number k specifies the number of neighbors
- Majority vote in neighbors to determine class of target
- Used for image/video recognition and handwriting detection



Classification - CART

- Decision Tree uses nodes and branches to determine class
- Branches represent decision split points
- Nodes represent class labels
- Uses pruning techniques to avoid overfit
- Used in analysing customer data for marketing decisions



Evaluation

1. **Precision-Recall Curve (PRC)**
   - Precision vs Recall

2. **Receiver Operating Characteristic (ROC)**
   - TPR vs FPR

Experiment Results

Preprocessing - Feature Selection

CFSSubsetEval (11/16)

No	Feature
2	Perimeter
3	MajorAxisLength
4	MinorAxisLength
5	AspectRation
7	ConvexArea
9	Extent
11	Roundness
12	Compactness
13	ShapeFactor1
14	ShapeFactor2
16	ShapeFactor4

```
=== Attribute Selection on all input data ===

Search Method:
        Best first.
        Start set: no attributes
        Search direction: forward
        Stale search after 5 node expansions
        Total number of subsets evaluated: 137
        Merit of best subset found:    0.682

Attribute Subset Evaluator (supervised, Class (nominal): 17 Class):
        CFS Subset Evaluator
        Including locally predictive attributes

Selected attributes: 2,3,4,5,7,9,11,12,13,14,16 : 11
                     Perimeter
                     MajorAxisLength
                     MinorAxisLength
                     AspectRation
                     ConvexArea
                     Extent
                     Roundness
                     Compactness
                     ShapeFactor1
                     ShapeFactor2
                     ShapeFactor4
```

Preprocessing - Feature Selection

InfoGain: Select the top 13 attributes

```
=== Attribute Selection on all input data ===

Search Method:
        Attribute ranking.

Attribute Evaluator (supervised, Class (nominal): 17 Class):
        Information Gain Ranking Filter

Ranked attributes:
 1.524    2 Perimeter
 1.49     7 ConvexArea
 1.484    1  Area
 1.484    8 EquivDiameter
 1.437    3 MajorAxisLength
 1.376   14 ShapeFactor2
 1.329   13 ShapeFactor1
 1.328    4 MinorAxisLength
 1.192   15 ShapeFactor3
 1.192   12 Compactness
 1.175    5 AspectRation
 1.175    6 Eccentricity
 1.147   11 Roundness
 0.533   16 ShapeFactor4
 0.34    10 Solidity
 0.284    9 Extent

Selected attributes: 2,7,1,8,3,14,13,4,15,12,5,6,11,16,10,9 : 16
```

Ranked	No	attributes
1.524	2	Perimeter
1.49	7	ConvexArea
1.484	1	Area
1.484	8	EquivDiameter
1.437	3	MajorAxisLength
1.376	14	ShapeFactor2
1.329	13	ShapeFactor1
1.328	4	MinorAxisLength
1.192	15	ShapeFactor3
1.192	12	Compactness
1.175	5	AspectRation
1.175	6	Eccentricity
1.147	11	Roundness
0.533	16	~~ShapeFactor4~~
0.34	10	~~Solidity~~
0.284	9	~~Extent~~

Preprocessing - Normalization

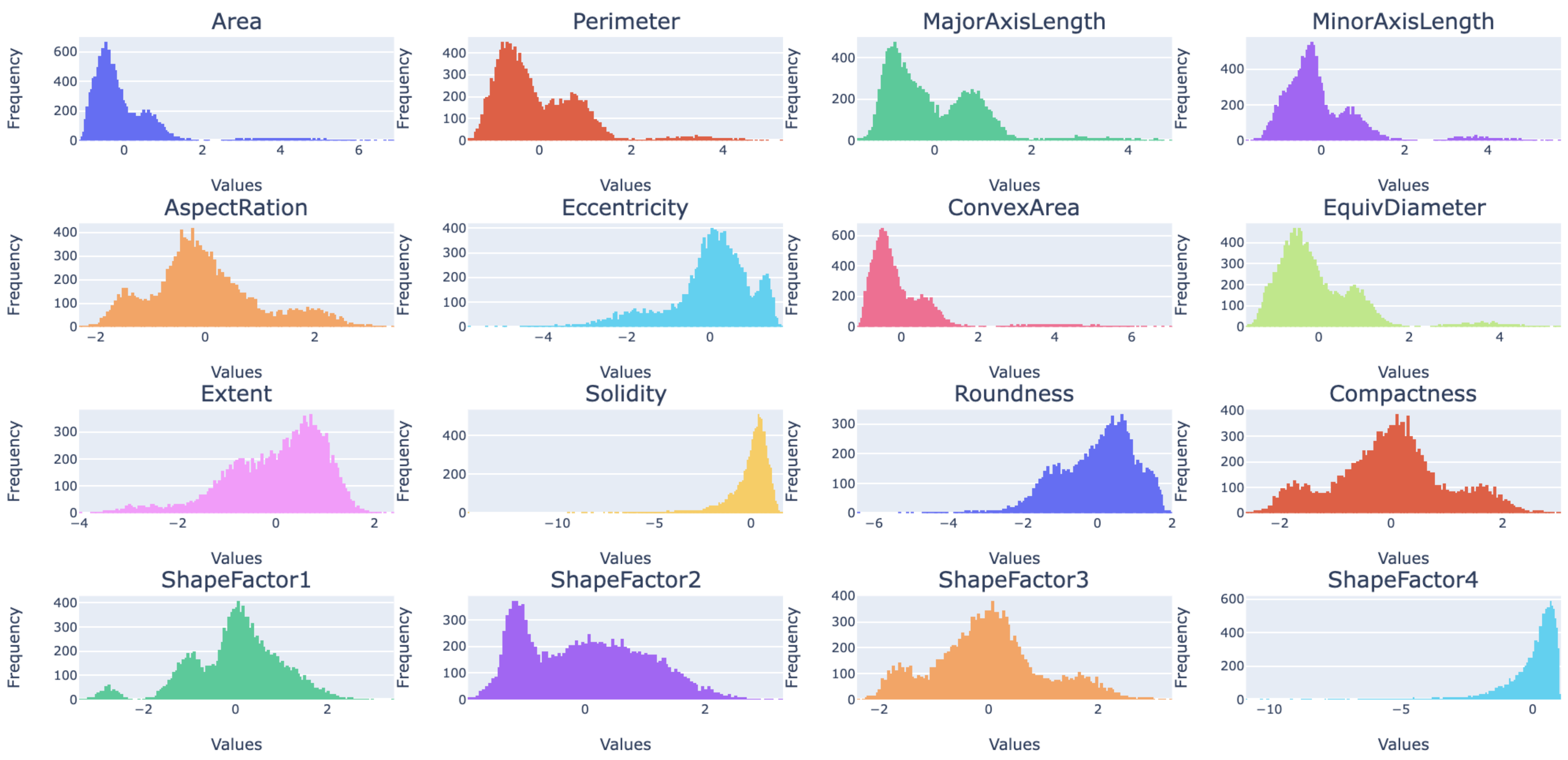
- Dealing with attributes in different scale

	Area	Perimeter	MajorAxisLength	MinorAxisLength
0	28395	610.291	208.178117	173.888747
1	28734	638.018	200.524796	182.734419
2	29380	624.110	212.826130	175.931143

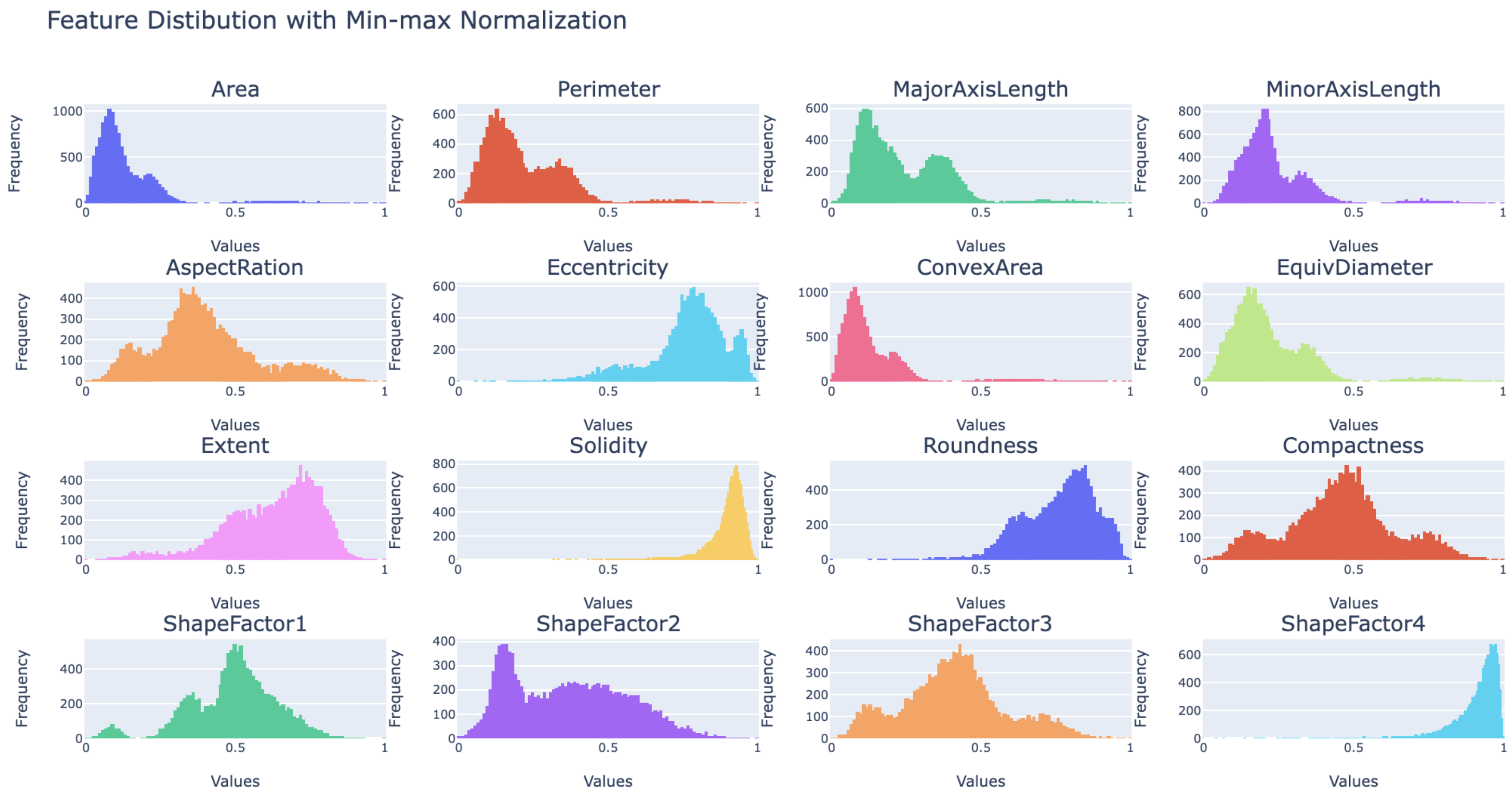
- improve the performance of different algorithms (e.g. KNN) by scaling the input features to a common scale
- without changing the distribution of the data
- If in different scale
  - Effectiveness / importance of the numbers will be different
  - As some numbers are in larger scale

Preprocessing - Z-score Normalization

Feature Distibution with Z-score Normalization

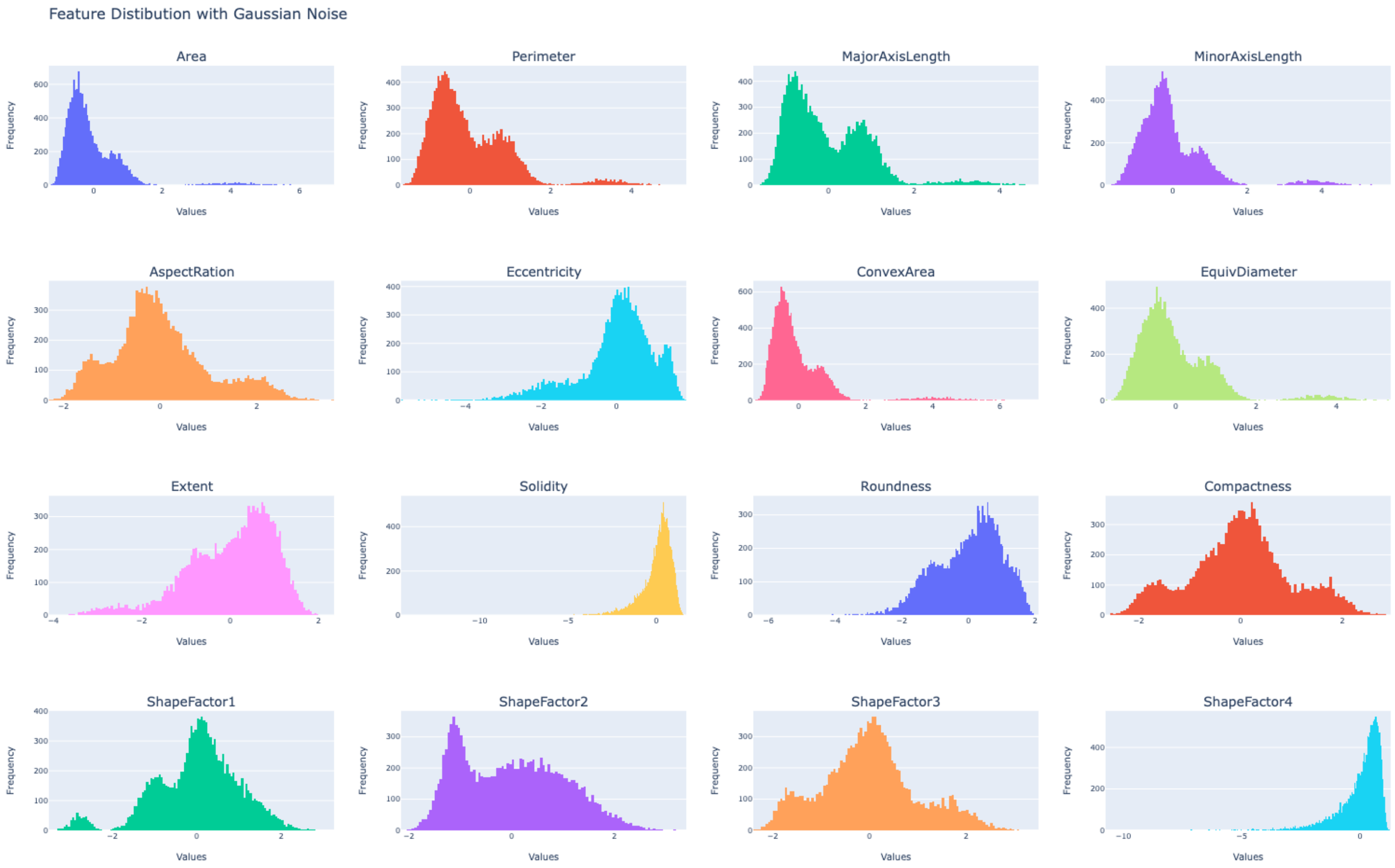


Preprocessing - Min-Max Normalization

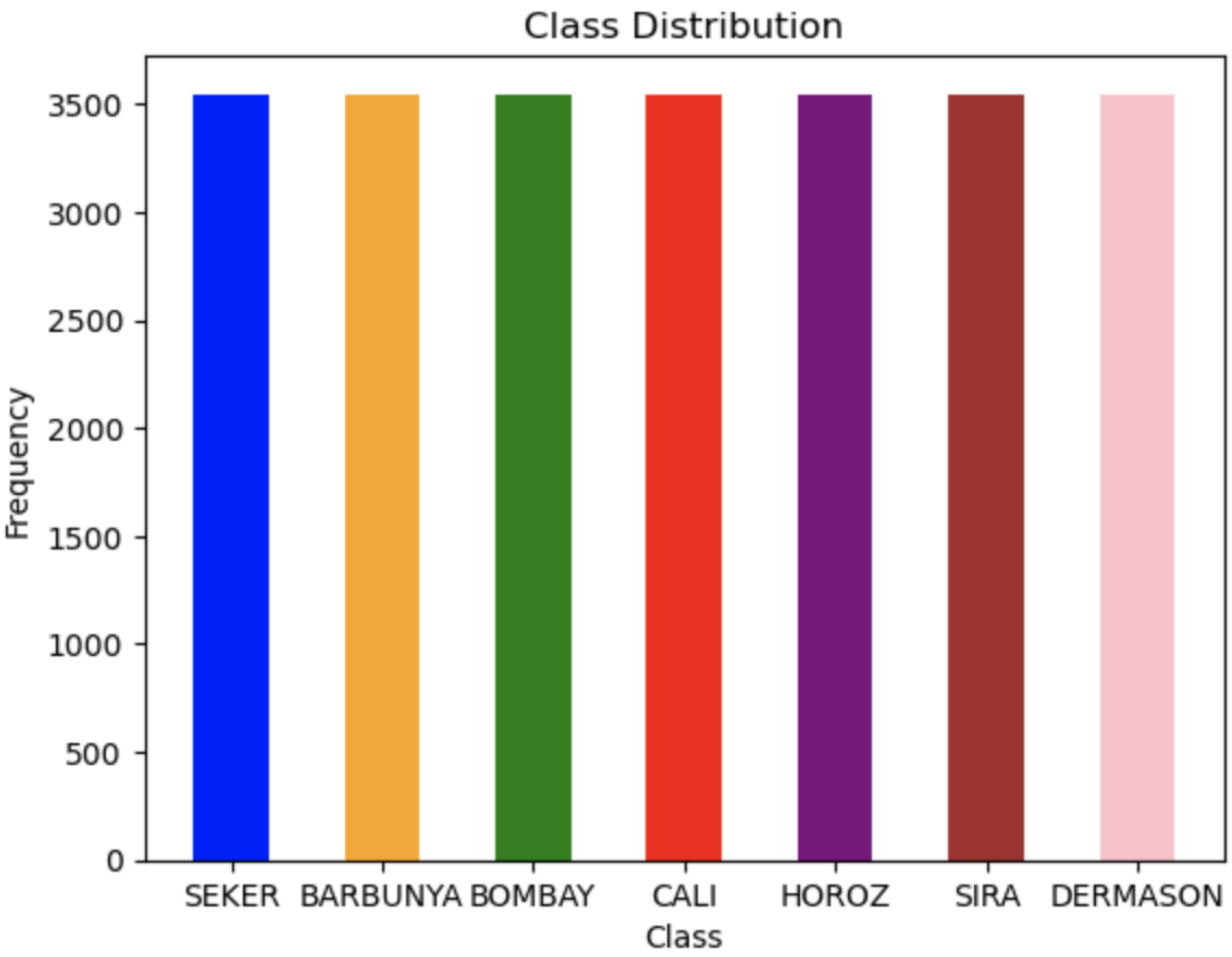


Preprocessing - Noise

- Add 10% Gassian Noise



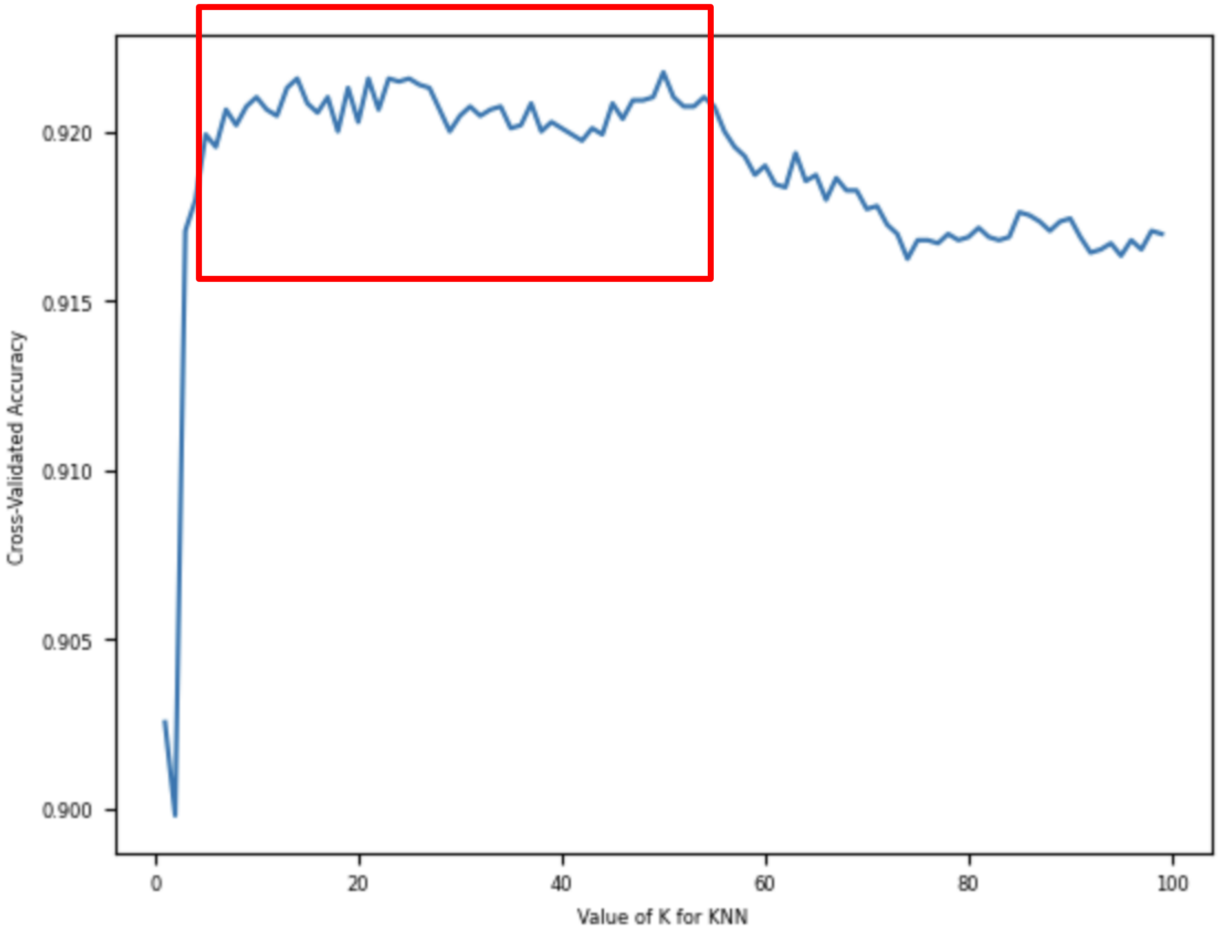
Preprocessing - SMOTE



Cross-validation and Fine Tuning

- Validation is important because we do not know how the model works when facing unseen data
- Cross validation
  - Divide the data into folds
  - Do training on most of the folds
  - But use one of the folds data as validation
  - Prevent overfitting
  - CV = 10
- Train and test split
  - Similarly, split the dataset into training and testing data
  - Simulate the unseen data
  - Training data = 0.8, Testing data = 0.2

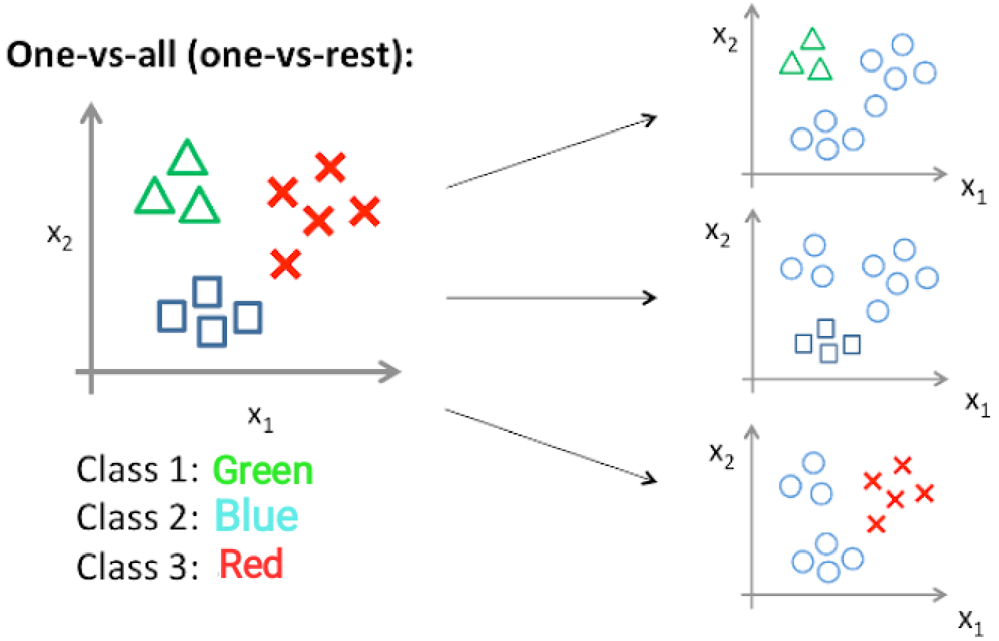
Cross-validation and Fine Tuning



- Use KNN as an example

Cross-validation and Fine Tuning

- **GridSearchCV**
  - Grid Search + Cross validation
  - Search over the parameter
  - Look for the parameters that have the highest cross validation accuracy
  - Get the best combinations
- **OneVsRestClassifier**
  - Common in multi-class classification
    - Fit one classifier per class
      - the class is fitted against all the other classes
    - Handle multi-class classification by binary classifiers



Cross-validation and Fine Tuning

- Can get the best parameter
- Make sure the classifier performs good without overfitting or underfitting

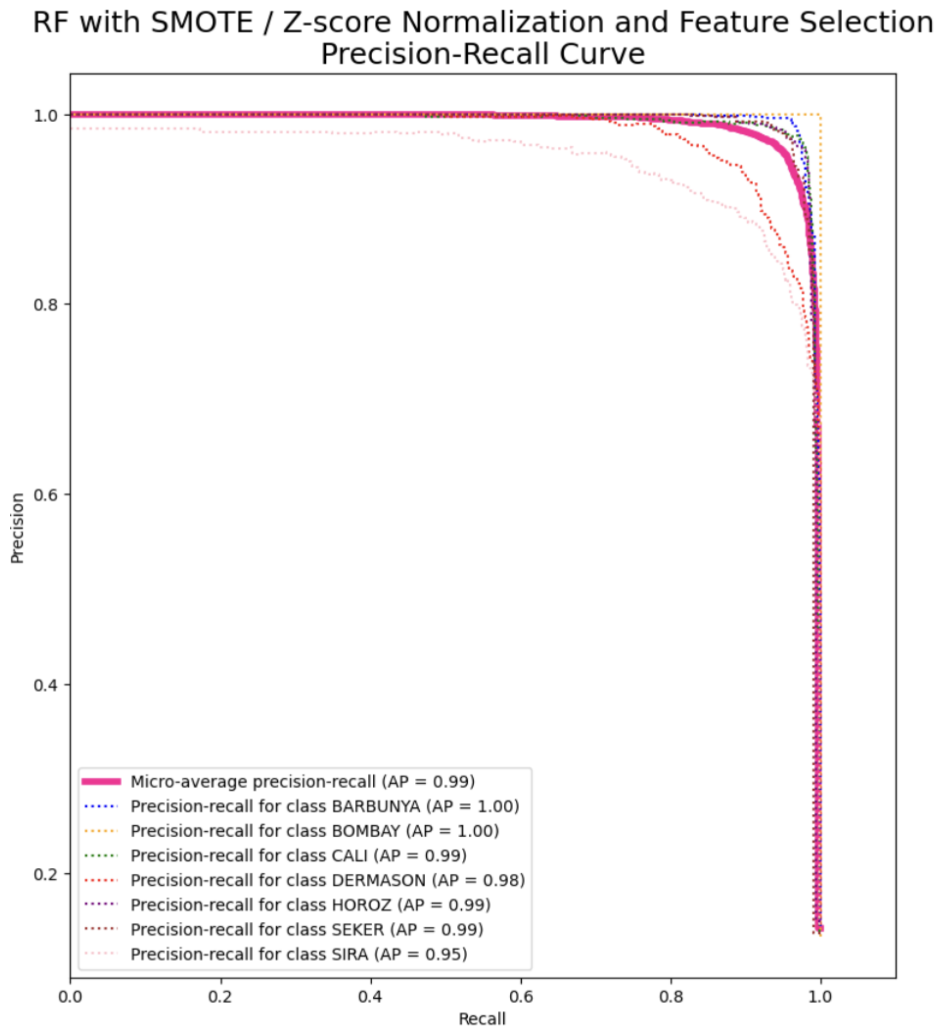
```
=== Best Parameter ===
Classifier for class BARBUNYA: {'n_estimators': 100}
Classifier for class BOMBAY: {'n_estimators': 10}
Classifier for class CALI: {'n_estimators': 200}
Classifier for class DERMASON: {'n_estimators': 200}
Classifier for class HOROZ: {'n_estimators': 50}
Classifier for class SEKER: {'n_estimators': 50}
Classifier for class SIRA: {'n_estimators': 200}
```

Result

Visualize the performance of the classifiers among different classes

$$P = \frac{T_p}{T_p + F_p} \qquad R = \frac{T_p}{T_p + F_n}$$

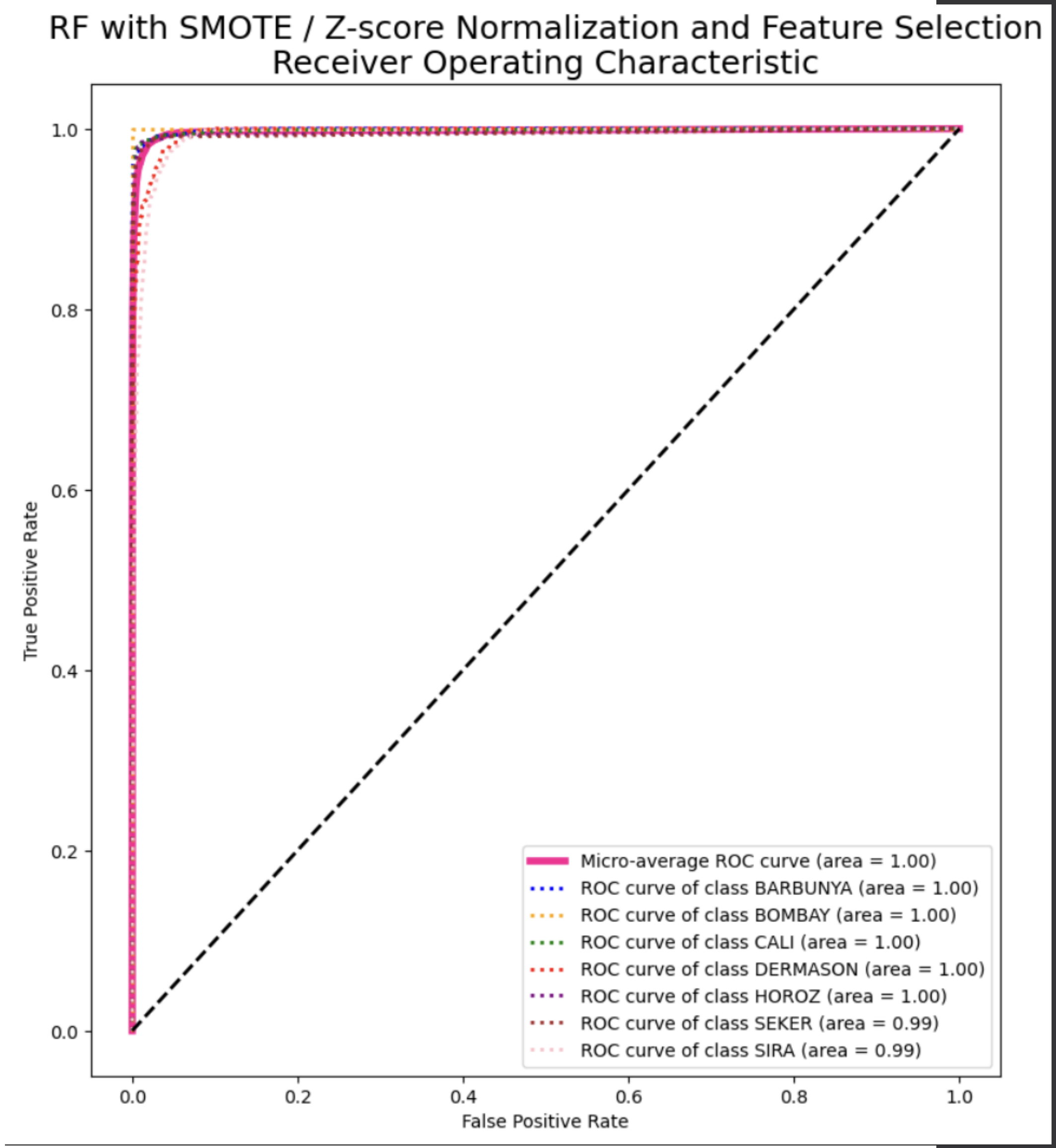
- PRC (our focus)
- should have high precision and high recall
  - all predictions are correct
- high recall & low precision
  - many results
  - but most of its predicted results are incorrect
- high precision & low recall
  - few results
  - but most of its predicted results are correct



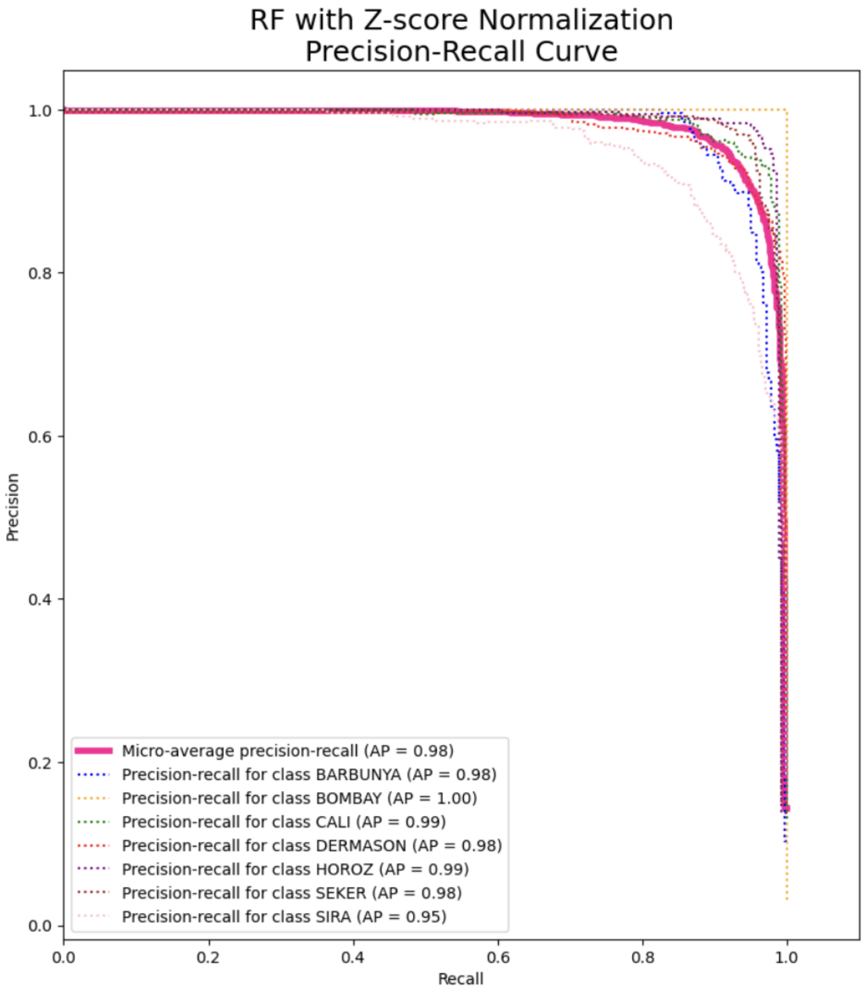
Result

$$TPR = \frac{TP}{TP + FN} \qquad FPR = \frac{FP}{FP + TN}$$

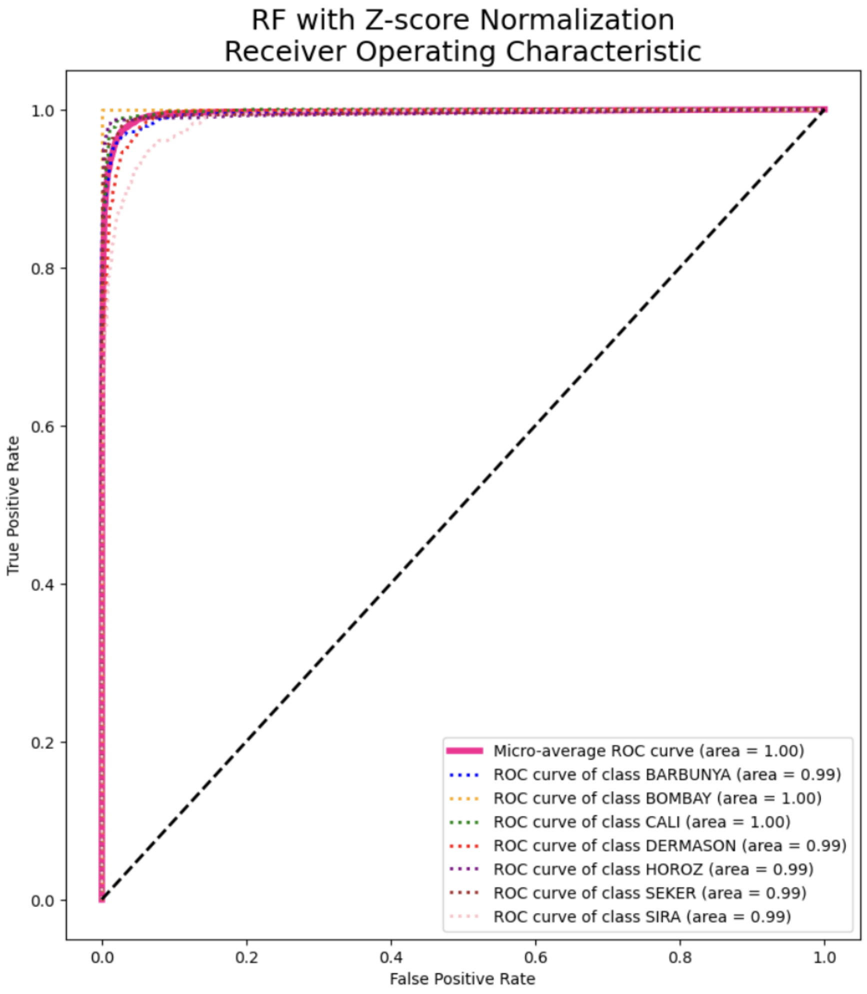
- ROC
  - X-axis = False Positive Rate
  - Y-axis = True Positive Rate
  - Classifier perform better when it is closer to the point (0, 1)
  - It is because it means
    - No False Positive
    - All the predictions are True Positive



Result

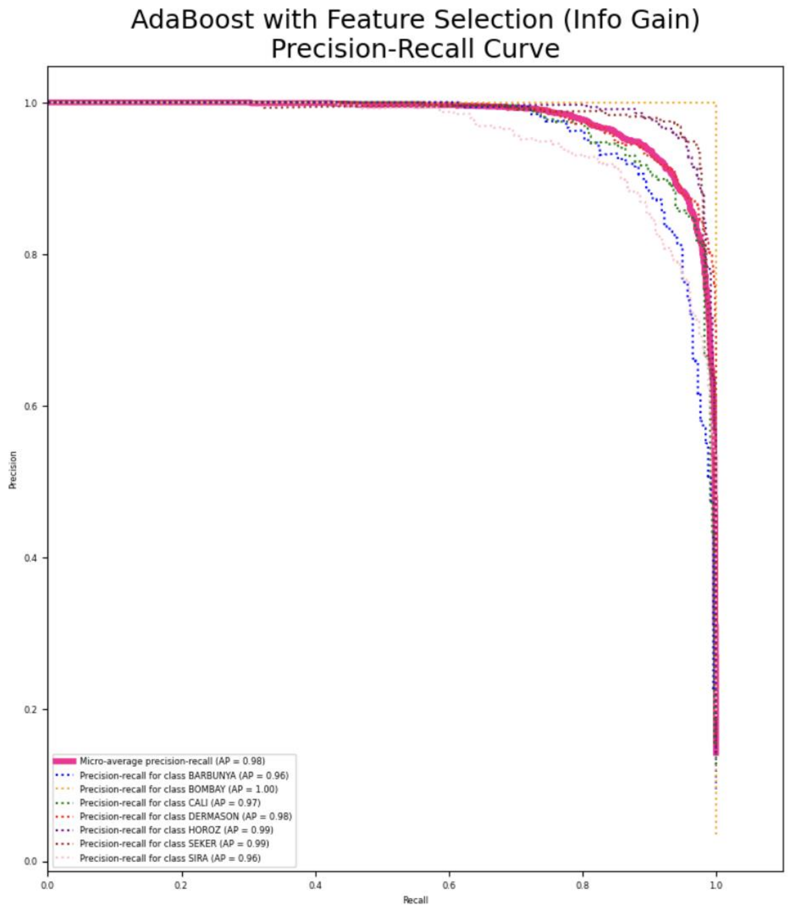


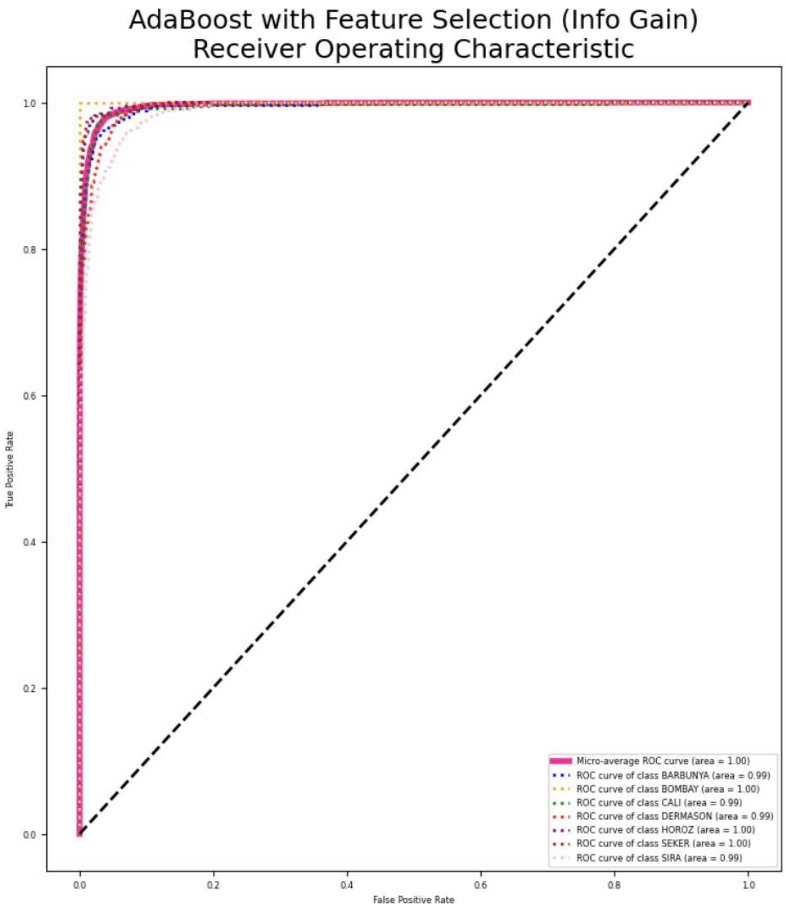
RF with Z-score Normalization
Precision-Recall Curve
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Micro-average precision-recall (AP = 0.98)
Precision-recall for class BARBUNYA (AP = 0.98)
Precision-recall for class BOMBAY (AP = 1.00)
Precision-recall for class CALI (AP = 0.99)
Precision-recall for class DERMASON (AP = 0.98)
Precision-recall for class HOROZ (AP = 0.99)
Precision-recall for class SEKER (AP = 0.98)
Precision-recall for class SIRA (AP = 0.95)



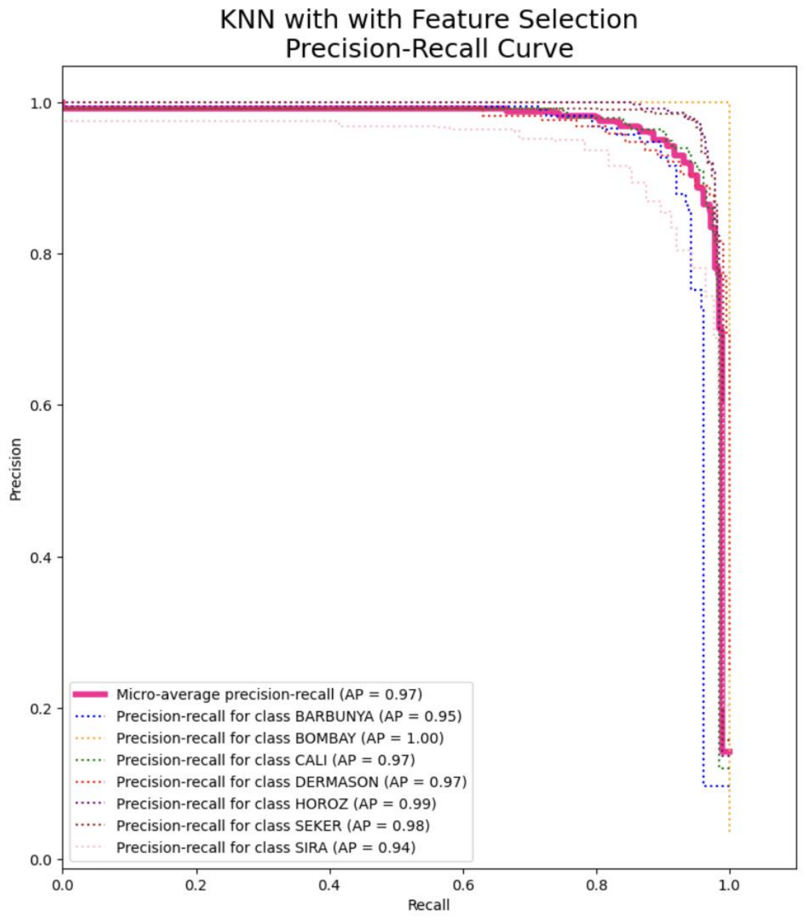
RF with Z-score Normalization
Receiver Operating Characteristic
True Positive Rate
False Positive Rate
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Micro-average ROC curve (area = 1.00)
ROC curve of class BARBUNYA (area = 0.99)
ROC curve of class BOMBAY (area = 1.00)
ROC curve of class CALI (area = 1.00)
ROC curve of class DERMASON (area = 0.99)
ROC curve of class HOROZ (area = 0.99)
ROC curve of class SEKER (area = 0.99)
ROC curve of class SIRA (area = 0.99)

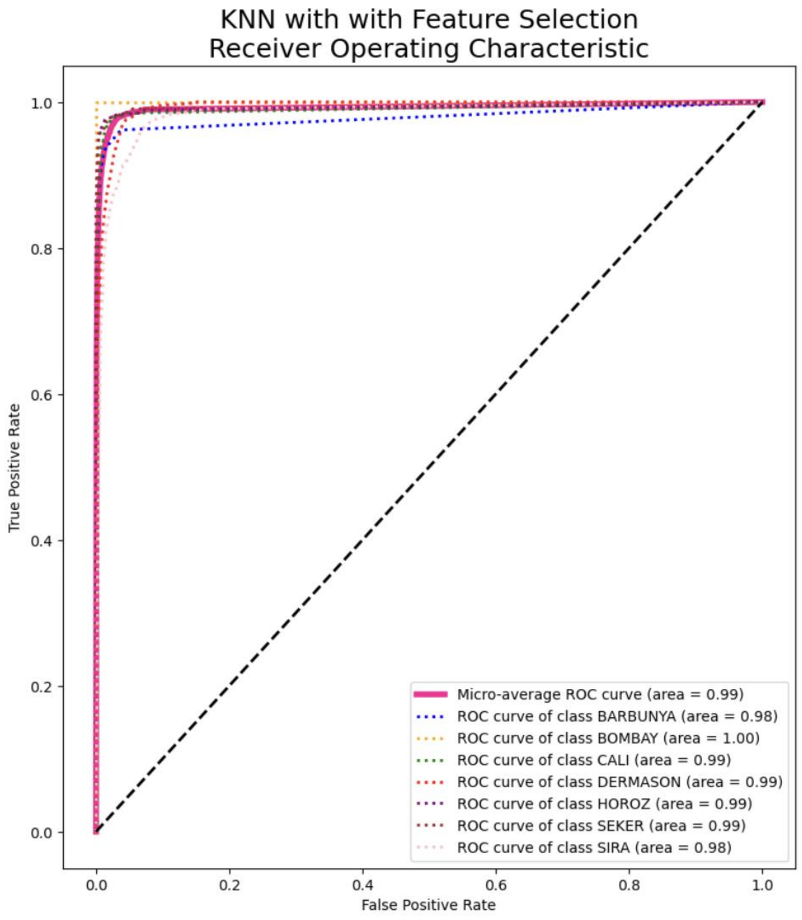
Result


AdaBoost with Feature Selection (Info Gain)
Precision-Recall Curve
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Micro-average precision-recall (AP = 0.98)
Precision-recall for class BARBUNYA (AP = 0.96)
Precision-recall for class BOMBAY (AP = 1.00)
Precision-recall for class CALI (AP = 0.97)
Precision-recall for class DERMASON (AP = 0.98)
Precision-recall for class HOROZ (AP = 0.99)
Precision-recall for class SEKER (AP = 0.99)
Precision-recall for class SIRA (AP = 0.96)

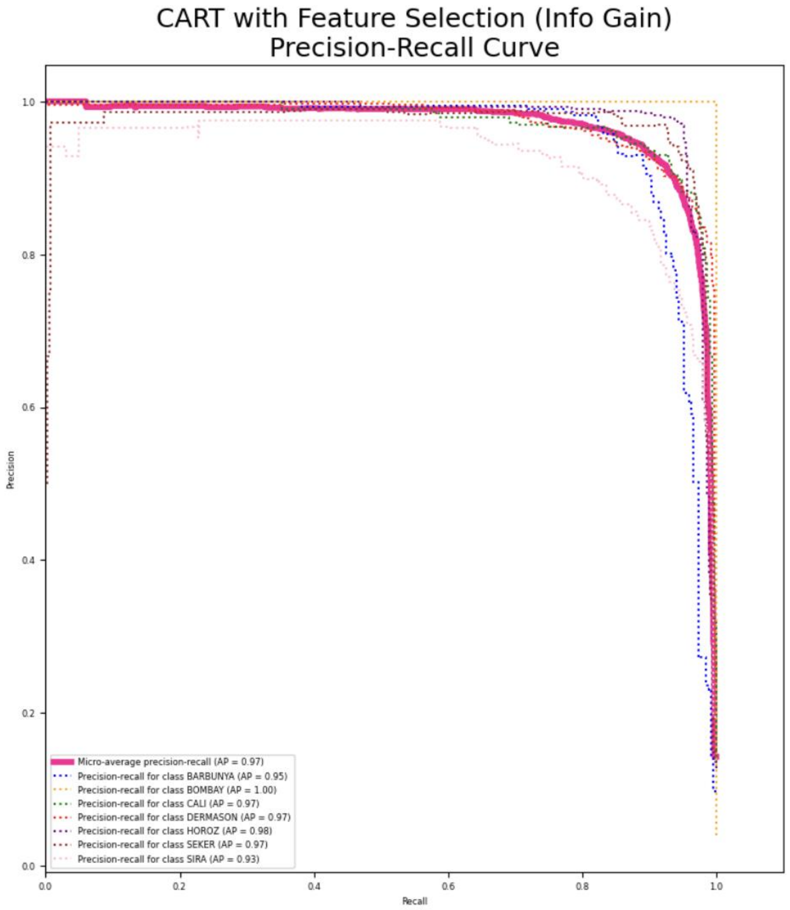

AdaBoost with Feature Selection (Info Gain)
Receiver Operating Characteristic
True Positive Rate
False Positive Rate
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Micro-average ROC curve (area = 1.00)
ROC curve of class BARBUNYA (area = 0.99)
ROC curve of class BOMBAY (area = 1.00)
ROC curve of class CALI (area = 0.99)
ROC curve of class DERMASON (area = 0.99)
ROC curve of class HOROZ (area = 1.00)
ROC curve of class SEKER (area = 1.00)
ROC curve of class SIRA (area = 0.99)

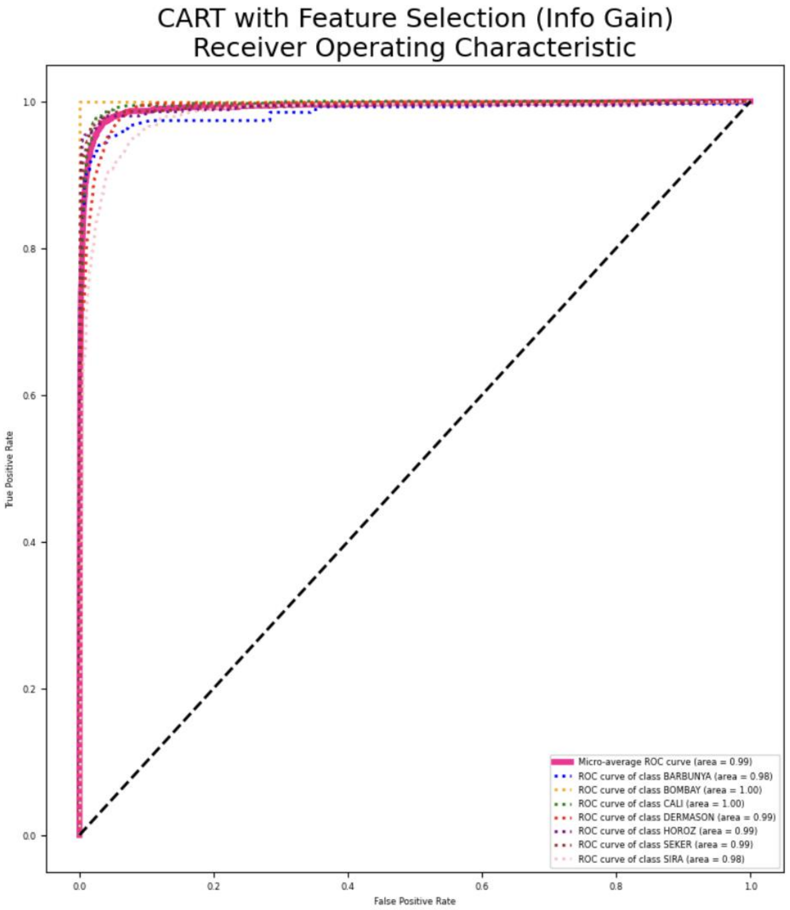
Result


KNN with with Feature Selection
Precision-Recall Curve
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Micro-average precision-recall (AP = 0.97)
Precision-recall for class BARBUNYA (AP = 0.95)
Precision-recall for class BOMBAY (AP = 1.00)
Precision-recall for class CALI (AP = 0.97)
Precision-recall for class DERMASON (AP = 0.97)
Precision-recall for class HOROZ (AP = 0.99)
Precision-recall for class SEKER (AP = 0.98)
Precision-recall for class SIRA (AP = 0.94)


KNN with with Feature Selection
Receiver Operating Characteristic
True Positive Rate
False Positive Rate
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Micro-average ROC curve (area = 0.99)
ROC curve of class BARBUNYA (area = 0.98)
ROC curve of class BOMBAY (area = 1.00)
ROC curve of class CALI (area = 0.99)
ROC curve of class DERMASON (area = 0.99)
ROC curve of class HOROZ (area = 0.99)
ROC curve of class SEKER (area = 0.99)
ROC curve of class SIRA (area = 0.98)

Result


CART with Feature Selection (Info Gain)
Precision-Recall Curve
Precision
Recall
Micro-average precision-recall (AP = 0.97)
Precision-recall for class BARBUNYA (AP = 0.95)
Precision-recall for class BOMBAY (AP = 1.00)
Precision-recall for class CALI (AP = 0.97)
Precision-recall for class DERMASON (AP = 0.97)
Precision-recall for class HOROZ (AP = 0.98)
Precision-recall for class SEKER (AP = 0.97)
Precision-recall for class SIRA (AP = 0.93)


CART with Feature Selection (Info Gain)
Receiver Operating Characteristic
True Positive Rate
False Positive Rate
Micro-average ROC curve (area = 0.99)
ROC curve of class BARBUNYA (area = 0.98)
ROC curve of class BOMBAY (area = 1.00)
ROC curve of class CALI (area = 1.00)
ROC curve of class DERMASON (area = 0.99)
ROC curve of class HOROZ (area = 0.99)
ROC curve of class SEKER (area = 0.99)
ROC curve of class SIRA (area = 0.98)

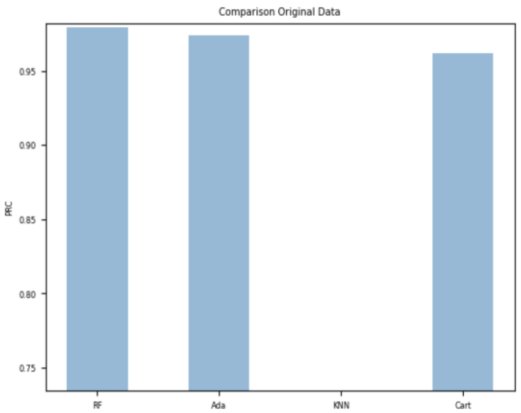
Discussion

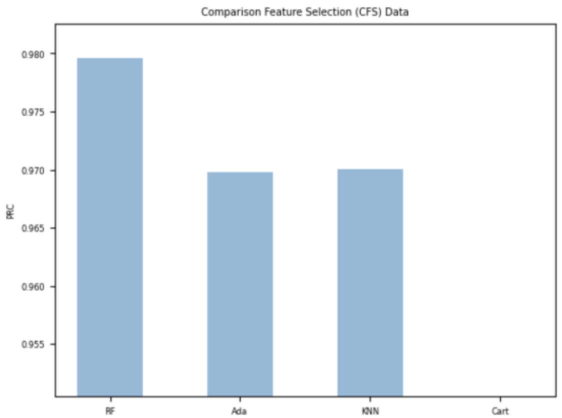
Discussion – Summarized Results

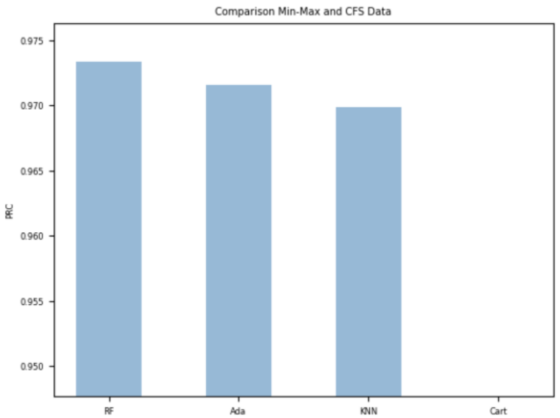
	PRC			
	RF	**AdaBoost**	**kNN**	**CART**
Original	0.979144699	0.973678676	0.73434203	0.961590026
Z-score	**0.98048767**	0.976913639	0.964942969	0.961022414
Min-max	0.97350926	0.975952883	0.969369673	0.953725505
CFS	0.979585829	0.969781191	**0.970041488**	0.950480018
Infogain	0.969800689	**0.978043618**	0.964988369	**0.968217017**
Noisy	0.977489563	0.970487715	0.961325432	0.929492055
Min-max & CFS	0.973315627	0.971533892	0.96984866	0.947659283
Z-score & CFS	0.975020048	0.975377604	0.969417381	0.94973238
Noise & CFS	0.974383935	0.968319736	0.966193701	0.941901072
Min-max & Infogain	0.967004399	0.970086957	0.959338112	0.946196496
Score & Infogain	0.967557197	0.969273899	0.965619545	0.961483048
Noise & Infogain	0.967832047	0.962632966	0.950987514	0.936518344
SMOTE, Z-score and feature selection	**0.989352833**	**0.986172399**	**0.973625952**	**0.972122992**

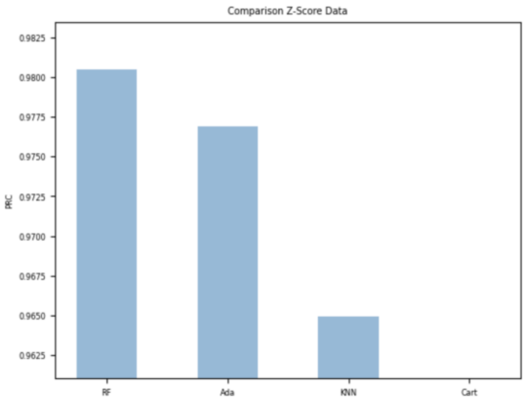
Discussion - Preprocessing

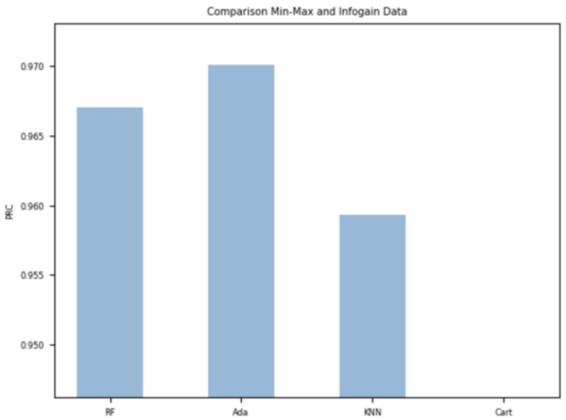
- Among all the classifiers, with different data preprocessing techniques

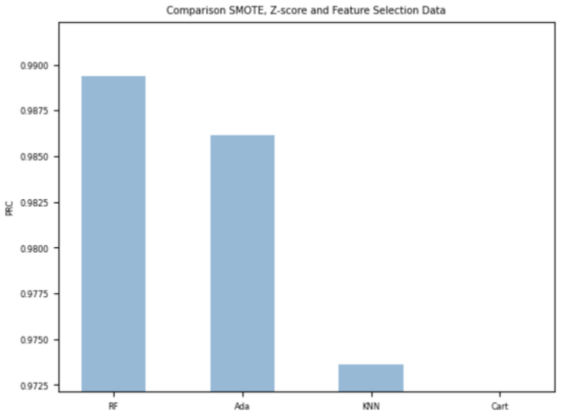












Discussion - Preprocessing

- **Different preprocessing techniques suits different classifiers**
  - Without SMOTE, Random Forest with Z-score normalization performs the best
  - For kNN, normalization gradually enhance its performance (~23%)
  - Feature selection performs well
  - Gaussian noise lower the performance

- **Class Imbalance**
  - SMOTE gives the best performance for each classifiers

	PRC			
	RF	**AdaBoost**	**kNN**	**CART**
Original	0.979144699	0.973678676	0.73434203	0.961590026
Z-score	**0.98048767**	0.976913639	0.964942969	0.961022414
Min-max	0.97350926	0.975952883	0.969369673	0.953725505
CFS	0.979585829	0.969781191	**0.970041488**	0.950480018
Infogain	0.969800689	**0.978043618**	0.964988369	**0.968217017**
Noisy	0.977489563	0.970487715	0.961325432	0.929492055
Min-max & CFS	0.973315627	0.971533892	0.96984866	0.947659283
Z-score & CFS	0.975020048	0.975377604	0.969417381	0.94973238
Noise & CFS	0.974383935	0.968319736	0.966193701	0.941901072
Min-max & Infogain	0.967004399	0.970086957	0.959338112	0.946196496
Score & Infogain	0.967557197	0.969273899	0.965619545	0.961483048
Noise & Infogain	0.967832047	0.962632966	0.950987514	0.936518344
SMOTE, Z-score and feature selection	**0.989352833**	**0.986172399**	**0.973625952**	**0.972122992**

Discussion - Evaluation

- Compare to the result of J. C. Macuácua et al
  - Ours perform slightly better with z-score normalization

Our Experiment (with SMOTE)

	Precision
BARBUNYA	0.98
BOMBAY	1
CALI	0.98
DERMASON	0.91
HOROZ	0.98
SEKER	0.98
SIRA	0.91

J. C. Macuácua et al. (with SMOTE)

	Precision
BARBUNYA	0.987
BOMBAY	1
CALI	0.983
DERMASON	0.889
HOROZ	0.976
SEKER	0.978
SIRA	0.908

Conclusion

Conclusion

- This project aims at exploring preprocessing on the dry bean dataset
  - It is successful by an improvement to the performance
  - Best processing techniques:
    - SMOTE (Class imbalance)
    - Z-score (Normalization)
    - CfsSubsetEval (Feature selection)

End

Thanks