Rice Classification

Presented by:

Mohammad AL-Zahrani
Turki Alhannawi
Feras Alsadat

INTRODUCTION

Our aim for this project is to use and apply what we learned in this course and prove the practices developed by combining different Machine Learning algorithms to analyze our data.

TASKS

01 Read the dataset

02 Explore the dataset

03 Modify the data

04 Normalize the data.

05 Split the dataset

06 ML algorithms

07 Data tuning to improve the results

08 Results comparison

READ THE DATASETs

1. Importing libraries
2. Reading (Rice Classification) dataset

```
In [3]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt

In [4]: df = pd.read_csv("riceClassification.csv")

In [5]: df.head()

Out[5]:
```

	id	Area	MajorAxisLength	MinorAxisLength	Eccentricity	ConvexArea	EquivDiameter	Extent	Perimeter	Roundness	Asp
0	1	4537	92.229316	64.012769	0.719916	4677	76.004525	0.657536	273.085	0.764510	
1	2	2872	74.691881	51.400454	0.725553	3015	60.471018	0.713009	208.317	0.831658	
2	3	3048	76.293164	52.043491	0.731211	3132	62.296341	0.759153	210.012	0.868434	
3	4	3073	77.033628	51.928487	0.738639	3157	62.551300	0.783529	210.657	0.870203	
4	5	3693	85.124785	56.374021	0.749282	3802	68.571668	0.769375	230.332	0.874743	

EXPLORE THE DATASET

3. Exploring the data's information:

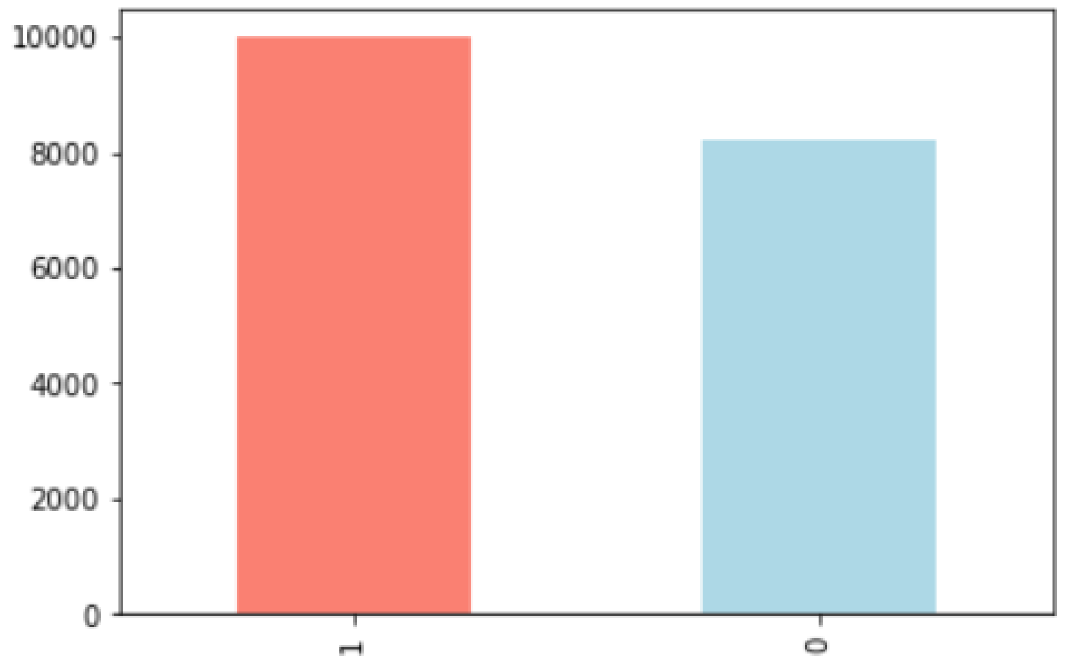
- Number and names of features
- Data types

```
In [8]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18185 entries, 0 to 18184
Data columns (total 12 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   id               18185 non-null  int64  
 1   Area             18185 non-null  int64  
 2   MajorAxisLength  18185 non-null  float64
 3   MinorAxisLength  18185 non-null  float64
 4   Eccentricity     18185 non-null  float64
 5   ConvexArea       18185 non-null  int64  
 6   EquivDiameter    18185 non-null  float64
 7   Extent           18185 non-null  float64
 8   Perimeter        18185 non-null  float64
 9   Roundness        18185 non-null  float64
 10  AspectRation     18185 non-null  float64
 11  Class            18185 non-null  int64  
dtypes: float64(8), int64(4)
memory usage: 1.7 MB
```

EXPLORE THE DATASET

4. Checking the data shape of column and row numbers
5. Check if any of our data is missing/null

```
In [12]: # try to count is the two class balanced ?
         df['Class'].value_counts().plot(kind="bar", color=["salmon", "lightblue"]);
```

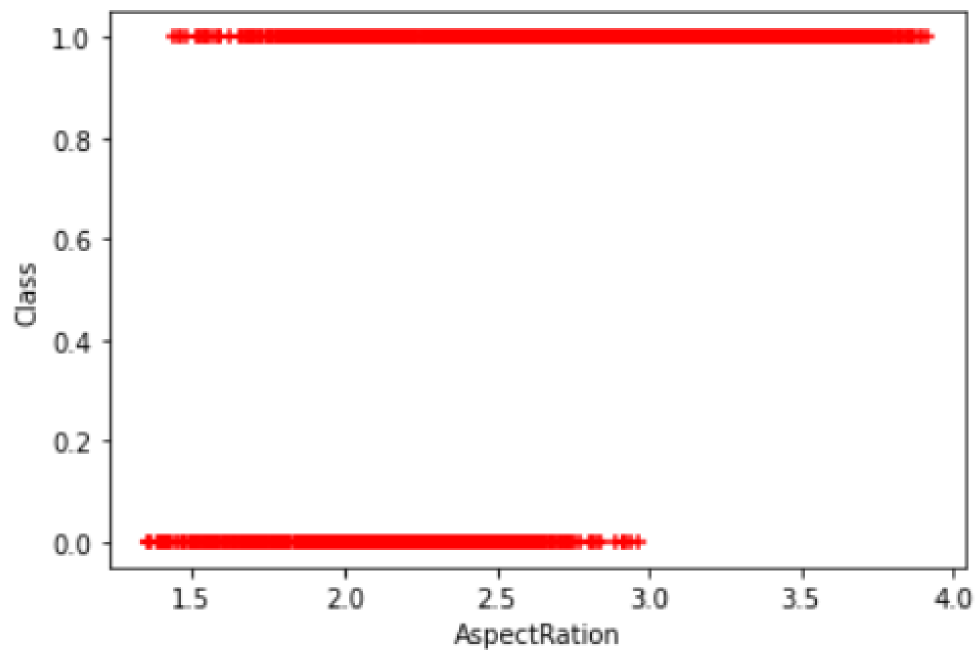



```
In [11]: # see if there's any missing data in
         df.isnull().sum()

Out[11]: id                 0
         Area               0
         MajorAxisLength    0
         MinorAxisLength    0
         Eccentricity       0
         ConvexArea         0
         EquivDiameter      0
         Extent             0
         Perimeter          0
         Roundness          0
         AspectRation       0
         Class              0
         dtype: int64
```

EXPLORE THE DATASET

- Plotting the numbers of each class
- Plotting one of our data's features

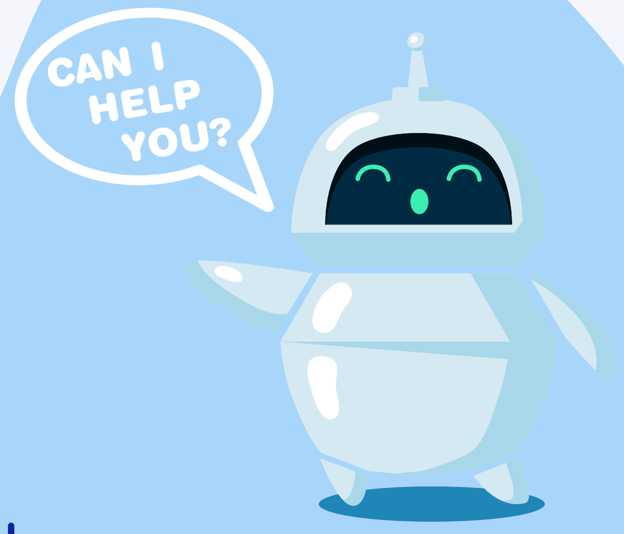
```
In [13]: df['Class'].value_counts()

Out[13]: 1    9985
         0    8200
         Name: Class, dtype: int64

In [14]: # Plot the data to get idea
         plt.scatter(df.AspectRation,df.Class,marker='+',color='red')
         plt.ylabel('Class')
         plt.xlabel('AspectRation')

Out[14]: Text(0.5, 0, 'AspectRation')
```



MODIFY THE DATA

- No need to MODIFY THE DATA !

Class
1
1
1
1
1



NORMALIZE THE DATA

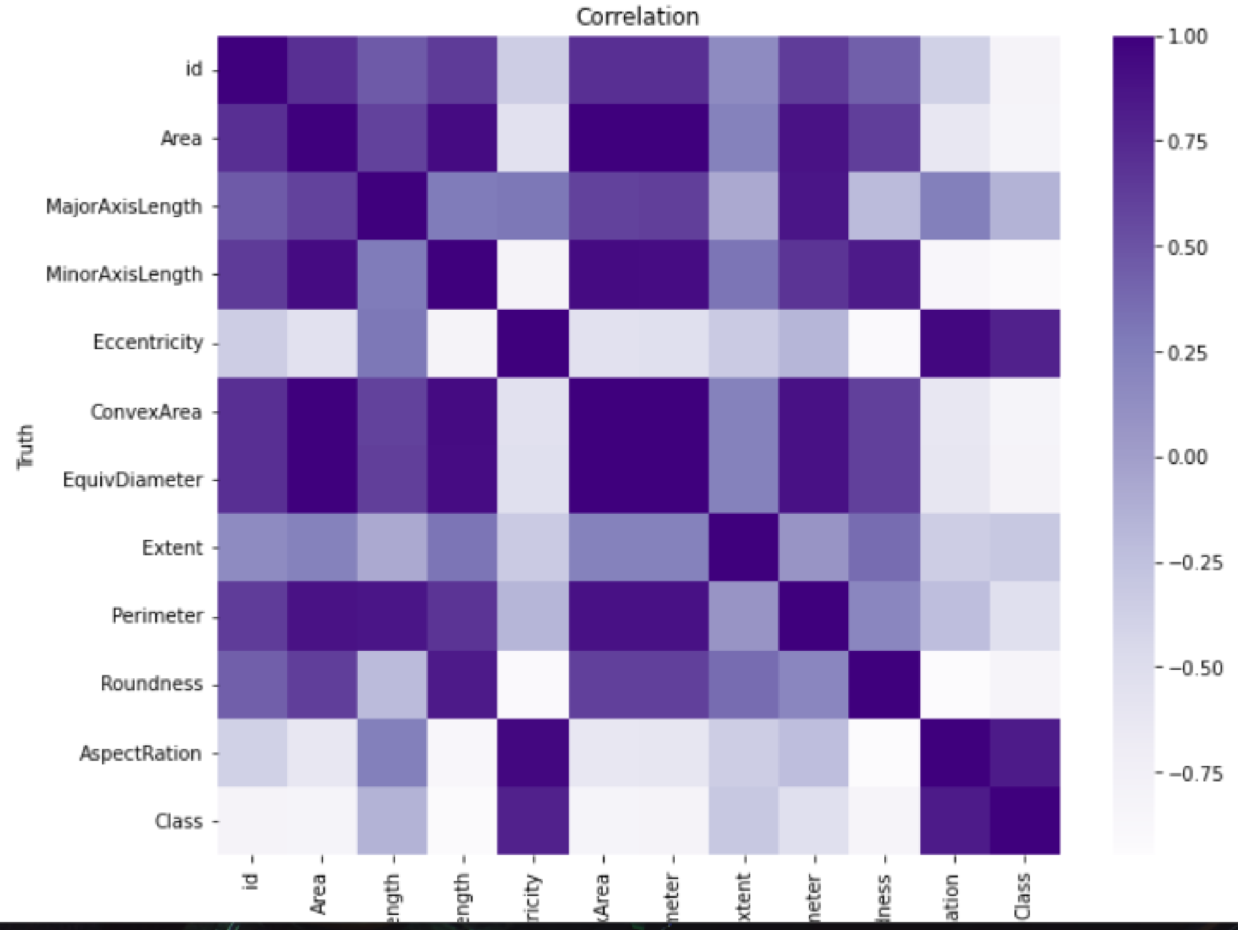
Normalizing the data using the Min-Max Scaler so that values are shifted and rescaled ranging from 0 to 1.

```
x = (x_df-np.min(x_df)) / (np.max(x_df)-np.min(x_df))
```

EXPLORE THE DATASET

- Correlation to visualize our data's best

```
In [10]: #correlation value between features
         import seaborn as sns
         import matplotlib.pyplot as plt
         corr = df.corr()
         fig = plt.figure(figsize=(10,8))
         r = sns.heatmap(corr, cmap='Purples')
         r.set_title("Correlation")
         plt.xlabel('Predicted')
         plt.ylabel('Truth')

Out[10]: Text(69.0, 0.5, 'Truth')
```



ML ALGORITHMS USED

- Support Vector Machine (SVM)
- Artificial Neural Network (ANN)
- Random Forest Classifier
- Logistic Regression

SPLIT THE DATASET

Splitting the dataset into 80% training and 20% testing set

SPLIT THE DATASET

```
In [15]: from sklearn.model_selection import train_test_split
         from sklearn.ensemble import RandomForestClassifier
         X = df.drop(['id','Class'], axis=1)
         y = df['Class']
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

         # Instantiate Random Forest Classifier
         clf = RandomForestClassifier(n_estimators=1000)

         # Fit the model to the data (training the machine learning model)
         clf.fit(X_train, y_train)

Out[15]: RandomForestClassifier(n_estimators=1000)
```

Dropping the column (class and id) so the machine doesn't train it

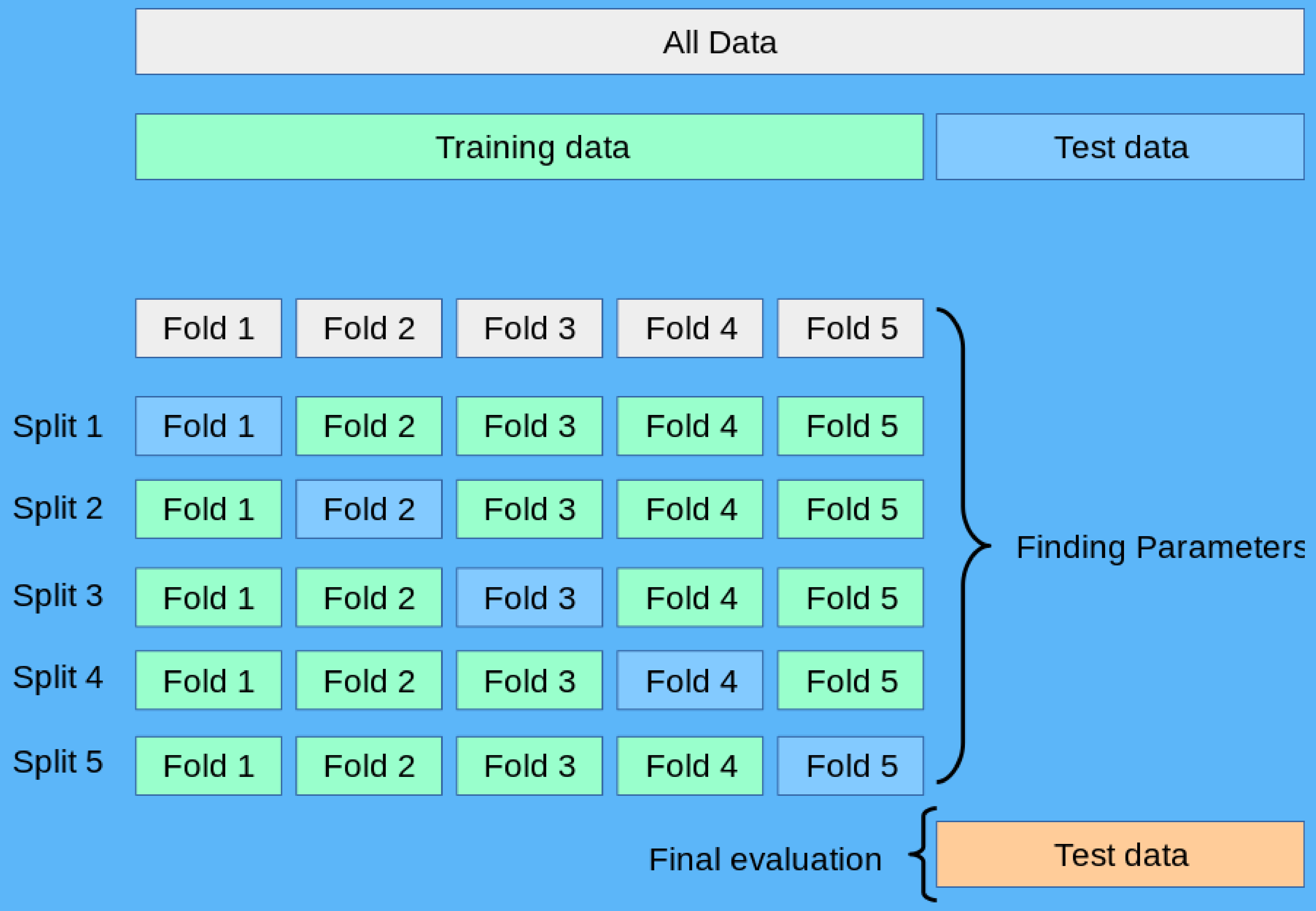
RANDOM FOREST CLASSIFIER

```
clf.fit(X_train, y_train)

Out[15]: RandomForestClassifier(n_estimators=1000)

In [16]: # The highest value for the .score() method is 1.0, the lowest is 0.0
         clf.score(X_train, y_train)

Out[16]: 1.0

In [17]: clf.score(X_test, y_test)

Out[17]: 0.9923013472642288
```

K-FOLD

```
In [19]: from sklearn.model_selection import cross_val_score
         cross_val_score(clf, X, y, cv=5)

Out[19]: array([0.47236734, 0.84465219, 0.93923563, 0.99752543, 0.58207314])

In [20]: cross_val_score(clf, X, y, cv=10)

Out[20]: array([0.45134689, 0.68554151, 1.        , 0.99945025, 0.99890049,
                0.99944994, 0.99944994, 0.99669967, 0.99724972, 0.56050605])

In [21]: # Take the mean of 5-fold cross-validation score
         clf_cross_val_score = np.mean(cross_val_score(clf, X, y, cv=5))
         # Compare the two
         clf_cross_val_score

Out[21]: 0.7662359087159747
```

Remember


All Data
Training data
Test data
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Split 1
Split 2
Split 3
Split 4
Split 5
Finding Parameters
Final evaluation
Test data

SVM

01 Importing the SVM model and fitting the data into the model to be trained

02 Printing accuracy scores of training and testing data

```
In [29]: # Import the SVC which makes support vector classification by using SVM and create the SVM classifier.
         from sklearn.svm import SVC
         svm = SVC(random_state = 1)

In [30]: #Train the model using the training sets.
         svm.fit(X_train , y_train)

Out[30]: SVC(random_state=1)

In [31]: #Calculate the accuracy of the model on the training data and in testing data.
         print("train accuracy :" , svm.score(X_train , y_train))
         print("test accuracy : " , svm.score(X_test , y_test))

         train accuracy : 0.9898267803134452
         test accuracy :  0.9903766840802859
```

IMPROVING RESULTS: (SVM)

01 Plotting the confusion matrix

02 New accuracy using best features

```
test accuracy :  0.9903766840802859
```

```
In [37]: # improve the accuracy of prediction
         from sklearn.feature_selection import SelectKBest
         from sklearn.feature_selection import f_classif
         accuracy_list_train = []
         number_of_features = np.arange(1,12,1)
         for each in number_of_features :
             X_new = SelectKBest(f_classif ).fit_transform(X_train , y_train)
             svm.fit(X_new , y_train)
             accuracy_list_train.append(svm.score(X_new , y_train))

             plt.plot(accuracy_list_train , color = "green" , label = "train")
             plt.xlabel("number of features")
             plt.ylabel("train accuracy")
             plt.legend()
             plt.show()
```

```
x train features :  ['Area' 'MinorAxisLength' 'ConvexArea' 'Roundness' 'AspectRation']
x test features :  ['Area' 'MinorAxisLength' 'ConvexArea' 'Roundness' 'AspectRation']
```

```
In [41]: # Re-train and re-calculate the model accuracy using the new arrangement of features
         from sklearn.svm import SVC
         svm = SVC (random_state = 1 )
         svm.fit(X_new , y_train)

         print("train accuracy : " , svm.score(X_new , y_train))
         print ("test accuracy : " , svm.score(X_new_test , y_test ))

         train accuracy :  0.989689304371735
         test accuracy :  0.9898267803134452
```

SVM

01 Applying the Cross-validation method to give us a better understanding of the model performance

```
In [42]: from sklearn.model_selection import cross_val_score
         cross_val_score(svm, X, y, cv=5)

Out[42]: array([0.87929612, 0.94088535, 0.94968381, 0.95023371, 0.9056915 ])

In [43]: cross_val_score(svm, X, y, cv=10)

Out[43]: array([0.83562397, 0.92413414, 0.93952721, 0.94282573, 0.95876855,
                0.94224422, 0.94444444, 0.95544554, 0.93784378, 0.87623762])

In [44]: np.random.seed(42)

         # Single training and test split score

         # Take the mean of 5-fold cross-validation score
         svm_cross_val_score = np.mean(cross_val_score(svm, X, y, cv=10))

         # Compare the two
         svm_cross_val_score

Out[44]: 0.9257095225740277
```

LOGISTIC REGRESSION

01 Importing the LR model and fitting the data into the model to be trained

02 Printing accuracy scores of training and testing data

```
In [46]: from sklearn.linear_model import LogisticRegression
         lgrgmodel=LogisticRegression()

In [47]: lgrgmodel.fit(X_train,y_train)

Out[47]: LogisticRegression()

In [48]: Lgg_predict1=lgrgmodel.predict(X_test)
         Lgg_predict1

Out[48]: array([0, 1, 1, ..., 0, 1, 1], dtype=int64)

In [49]: print("train accuracy :" , lgrgmodel.score(X_train , y_train))
         print("test accuracy : " , lgrgmodel.score(X_test , y_test))

         train accuracy : 0.9872834753918064
         test accuracy :  0.9859774539455596

In [50]: ################################################################
```

LOGISTIC REGRESSION

1 Applying the Cross-validation method to give us a better understanding of the model performance

```
Out[53]: array([0.6520066 , 0.99450247, 1.        , 1.        , 0.99945025,
                1.        , 1.        , 0.99449945, 0.99559956, 0.91309131])

In [54]: Lr_cv = np.mean(cross_val_score(lgrgmodel, X, y, cv=10))
         Lr_cv
```

ARTIFICIAL NEURAL NETWORK

01 Importing tenserflow

02 Defining the number of layers that matches our data's features (10)

03 Deviding the dataset into (100) epochs

```
import tensorflow as tf
from tensorflow import keras
```

```
X_train.shape
```

```
(14548, 10)
```

```
y_train.shape
```

```
(14548,)
```

```
#Defining the model and adding the layers
model = keras.Sequential([
    keras.layers.Dense(10,input_shape=(10,), activation='sigmoid')
])
```

```
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

```
model.fit(X_train,y_train , epochs=100)
```

ARTIFICIAL NEURAL NETWORK

Model evaluation of the test set:

01 Data loss

02 Data accuarcy

```
[ ]  model.evaluate(X_test,y_test)

     114/114 [==============================] - 1s 4ms/step - loss: 0.0402 - accuracy: 0.9857
     [0.040153853595256805, 0.9857025146484375]
```

COMPARING RESULTS

Status	RF	SVM	LR	NN
Test Accuracy	0.99	0.99	0.985	0.9857
Train Accuracy	1	0.989	0.987	
After improve test		0.9898		
K-fold Score	0.766236	0.925	0.954	

Thank You