# CS5014 - Practical 2:
# Classification of objects using a radar signal and machine learning

**Student ID: 180029539**

## Problem:

Given a set of radar signals and the type of object(class) that generated these radar signals, create a classification model which can predict the type of object from a given set of radar signals.

## 1. Data load:

First, I opened the input and output data files and saw that the data had no headers. Using this information, I loaded the input data and output data into two variables(xBinData, yBinData) for binary-classification data and (xMultiData, yMultiData) for multi-class classification data. Then, I loaded the data to be classified into a separate variable(xBinToClassify) for binary classification and (xMultiToClassify) for multi-class classification.

## 2. Data Cleaning:

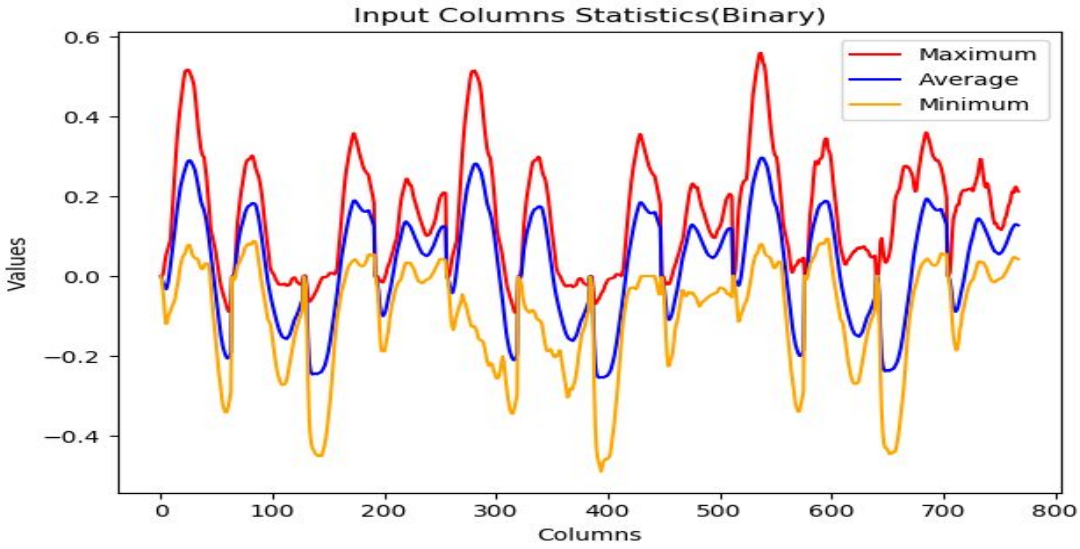
Then, I checked for missing values in the data of both the classification data sets. There were no missing values in the data. Hence, no cleaning of data was required. After visualising the data, we can confirm if any other anamolies are present in the data which requires data cleaning. After the data cleaning is done, I have split the data sets into training and test data sets with a split of 80-20. This helps to avoid the influence of test data on the model. When doing the split, I made sure that all the classes were equally split between the training and test data set. For this, I looped through each class and split it into training and test sets and then merged all individual training and test into a single training and test set. But, I made sure the dataset is not biased by picking up random rows when splitting into training and test data sets. When the program runs you can see the number of sentences in each data set.

(Continued on next page)

## 3. Data Visualization and Analysis:

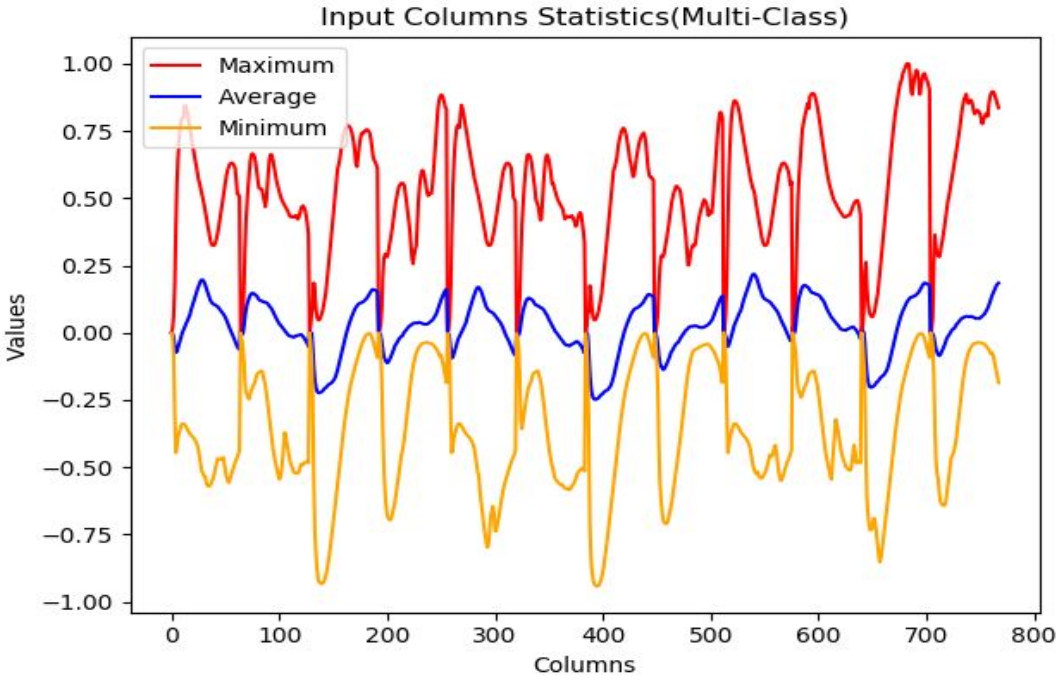
Next, I plotted a line graph with average(blue line), minimum(orange line) and maximum(red line) of all the columns for both classification data sets.

**The minimum, average and maximum of columns in binary data set:**




From this graph, you can see that all the values are bounded between -0.5 and +0.5. Hence, we don't have to scale or normalize the data.

**The minimum, average and maximum of columns in multi-class data set:**

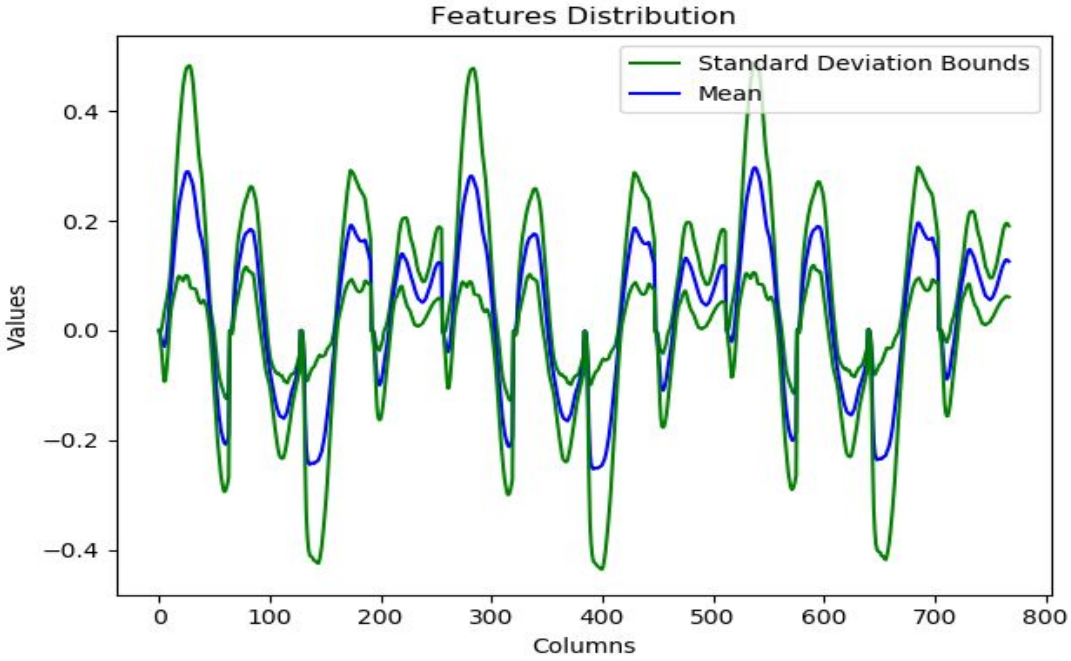



From this graph, you can see that all the values are bounded between -1.0 and +1.0. Hence, we don't have to scale or normalize the data.

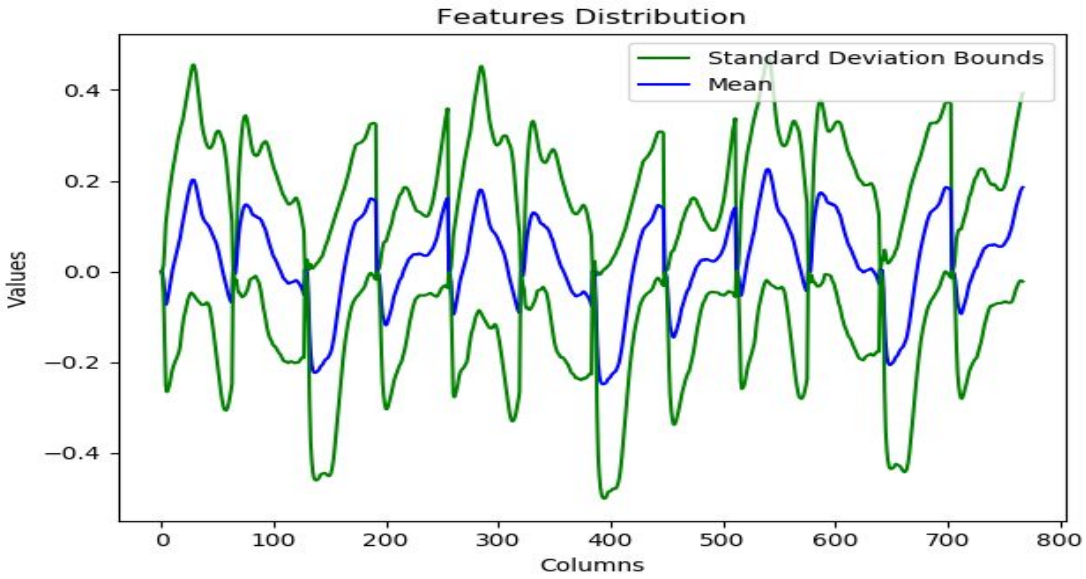
## 4. Understanding the Features and Feature Selection:

When doing feature selection, I first wanted to see the mean and standard deviation of each column to understand the feature distribution better. So, I drew a line graph having mean(blue line) and standard deviation bounds(red lines) for all the columns.

**The feature distribution of the binary data set:**




**The feature distribution of the multi-class data set:**




From these graphs, we can see that the trend for the columns from 0-255, 256-511 and 512-768 are same. This is due to the reason that the data is from the same wave and is split into three parts average, max and min signal of each component. Hence, we can work only on

the average signals part of the data and ignore the maximum and minimum signals. We will discuss this further in the last section of the report(critical discussion).

## 5. Selecting and Training Classification Models:

**Gaussian Naive Bayes Classifier:**

The main thing to remember before we select a classification model for our problem is the limited number of data samples that we have in the data sets. Due to this reason, I have decided to use the Gaussian Naive Bayes classifier to classify the datasets. The Naive Bayes classifier can work on a very limited number of data samples as it uses conditional probability model and determines the new class based on the assigned probability instances from the trained data set samples.

Let us take an example to better understand the classifier, let's assume we have a dataset of weight and heights of people to determine whether they are children or adults. We construct a Gaussian distribution for each class based on the training data samples and get data statistics like mean and standard deviations of each class. Next, when we are testing, we get a height and a weight of a new person. We compare that weight and height to the existing Gaussian distributions of each class. We calculate conditional probabilities of that data point being a particular class based on the mean and standard deviation from the Gaussian distributions of those classes. The class with the highest probability becomes the predicted class.

Another advantage of this classifier is that it does not have any parameters that need to be tuned unlike classifiers like Multi-Level Perceptrons. So, I trained a model using this classifier for both the binary and multi-class data sets. I will compare the accuracies of each classifier in the next section.

**Logistic Regression:**

The Logistic Regression model also uses probability models to evaluate the test samples as Naive bayes does but the difference is instead of Gaussian distribution it uses Bernoulli's distribution. It does a very good job splitting the binary data as it uses a logit function to make the prediction between two classes. But, the multi-class classification is a bit complicated when we use the logistic regression. We have to use the one vs all where it goes through class by class and compares the selected class with the rest of the data as a single class. By doing this it can identify the classes by calculating probability of that point belonging to each class similar to Naive Bayes classifier. The accuracy of the logistic regression as classifier is also shown in the next section, where I compare it to the other models.

The parameters that need to be passed to this function are solver and multi-class. The default value for solver is ‘liblinear’ and multi-class is ‘ovr’. I chose the default options as we are performing a one vs all classification. The other solvers support the multinomial classification.

**Decision Tree Classifier:**

The Decision Tree Classifier works on the basis of a tree structure. When you start training the root node splits into different nodes based on the most important feature. Next, it

splits those nodes based on the next important feature. It goes on until all the important features are used. The order of the importance of features is defined by the information gain or gini impurity. Information Gain is calculated based on the entropy of a particular feature and compare it to other features' information gain. The higher the information gain, the important the feature. Each node in the tree is marked with a Gini impurity. Gini Impurity can be defined as the the probability of predicting a test element belonging to a particular class. Gini impurity is used when classifying a new item into a class. We will see the accuracy of this model with respect to our dataset in the next section. The problem with this classifier is that the model might overfit due to the fact that we might go too deep into the tree which results in model over-learning the properties of each point.

The important parameters that need to be passed are criterion, splitter, max_depth, max_features,class_weight. I chose 'gini' as my criterion as both 'gini' and 'information gain' both give similar results but entropy is a little bit slower due to the mathematical complexity of the formula. Next, I chose 'best' as the splitter as the method will optimize the splitting conditions. Then, I set max_depth as 'none' due to the fact that I don't have any special requirement to restrict the depth of my tree. Then, I set the 'max_features' to none as I already filtered the features in the previous step. I set the 'class_weight' to none as the classes are already balanced. As all the values I want to set are the default values of these parameters, I did not pass any parameters to the method.

**Random Forest Classifier:**

This is a classifier that is built on the the Decision Tree concept. The random forest classifier during the training time takes subsets of the training data set and creates decision tree for each subset. Then, takes the mode of the classes to make the prediction. In our case its mode of the classes as we are classifying the data. We would use mean in case of a regression task. This reduces the problem of overfitting that we face in the decision tree because it is more generalised than decision tree and we don't use all the training points to create a decision tree. It avoids the over-learning of a data point as the data point won't be present in all the decision trees. We can see this by the reduction in the variance of this classifier compared to the decision tree classifier. We will compare the accuracy of this classifier with other classifiers in the next section.

The only extra parameter that I needed to pass to random forest that is not in the decision tree is the number of trees. I chose number of trees to be 10 rather than the default of 100 due to the fact that we have very limited number of data samples.All other parameters I used are same as the parameters I chose for the decision tree classifier.

(Continued on next page)

## 6. Evaluation and Comparison of Models:

The accuracies of all the models are shown in the table below. The training accuracies are the accuracies from the training set. These accuracies help us understand the bias of the model. Next, the testing accuracies are the accuracies by passing testing data set to the model. These help understand the variance of the model.

**Training and Testing Accuracies of binary dataset by each model:**

| Classification Model used | Training Accuracies | Testing Accuracies |
|---|---|---|
| Gaussian Naive-bayes | 100% | 100% |
| Logistic | 100% | 100% |
| Decision Tree | 100% | 100% |
| Random Forest | 100% | 100% |

**Training and Testing Accuracies of multi-class dataset by each model:**

| Classification Model used | Training Accuracies | Testing Accuracies |
|---|---|---|
| Gaussian Naive-bayes | 97.5% | 100% |
| Logistic | 96.88% | 100% |
| Decision Tree | 100% | 97.5% |
| Random Forest | 100% | 100% |

## 7. Critical Discussion:

- The accuracies of binary are 100 percent due to the fact that we had so many features compared to the number of classes. So, it could predict the values exactly.
- Next, all the training accuracies are almost equal to the testing accuracies showing that all the models have a good bias-variance balance and all the models are well trained.
- Next, we can see that the decision trees have less testing accuracies compared to the random forest which might show the evidence of overfitting in decision trees
- I also compared the accuracies by considering all the columns as discussed in the ‘Feature Selection’ Section and found the accuracies to be approximately same as the filtered data set. By removing the extra columns we did not lose any accuracy and decreased the computational power required to train the model.

**Accuracies before Feature Reduction:**

```
Accuracies with all the columns before Feature Reduction:

Accuracy of Gaussian classifier on training set: 96.25%
Accuracy of Guassian classifier on test set: 100.00%
Accuracy of Logistic regression classifier on training set: 99.38%
Accuracy of Logistic regression classifier on test set: 100.00%
Accuracy of Decision Tree classifier on training set: 100.00%
Accuracy of Decision Tree classifier on test set: 100.00%
Accuracy of Random Forest classifier on training set: 100.00%
Accuracy of Random Forest classifier on test set: 97.50%
```

**Accuracies after Feature Reduction:**

```
Accuracy of Gaussian classifier on training set: 97.50%
Accuracy of Guassian classifier on test set: 100.00%
Accuracy of Logistic regression classifier on training set: 96.88%
Accuracy of Logistic regression classifier on test set: 100.00%
Accuracy of Decision Tree classifier on training set: 100.00%
Accuracy of Decision Tree classifier on test set: 97.50%
Accuracy of Random Forest classifier on training set: 100.00%
Accuracy of Random Forest classifier on test set: 100.00%
```

- Next, if we look at the accuracies of logistic regression is fairly lower in comparison to the other models as we are not training in respect to each class but comparing each class to the rest of the classes.
- Due to the accuracies being 95% to 100%, we can be sure all of our models can predict almost all the values to be classified correctly.
- But for predicting the values, I used the Random Forest classifier as it overcomes overfitting and has better accuracies.

## Appendix:

The predicted values for the binary data set is given below.

**Binary dataset predictions:**

[1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]

**Multi-class dataset predictions:**

[2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 3. 3. 3. 3.
 3. 3. 3. 3. 3. 3. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 4. 4. 4. 4. 4. 4. 4. 4.
 4. 4.]

## References:


1. Ng, Andrew Y., and Michael I. Jordan. "On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes." *Advances in neural information processing systems*. 2002.
2. Pal, Mahesh. "Random forest classifier for remote sensing classification." *International Journal of Remote Sensing* 26.1 (2005): 217-222.
3. Swain, Philip H., and Hans Hauska. "The decision tree classifier: Design and potential." *IEEE Transactions on Geoscience Electronics* 15.3 (1977): 142-147.