Classifying Cardiac Arrhythmia

Balaji Mediboina

Koushik Srivastav Lala

Group 12

MA5790

December 2023

Table of Contents:

1. Abstract…………………………………………………………………… 3

2. Background……………………………………………………………... 5

3. Dataset…………………………………………………………………….. 5

4. Preprocessing ……………………………………………………………….. 10

    a. Correlations…………………………………………………………… 10
    b. Transformations………………………………………………………… 11

5. Data Splitting…………………………………………………………… 14

6. Model Fitting…………………………………………………………… 14
   Linear Classification Model………………………………………………. 16
   Non- Linear Classification Model………………………………………… 17

7. Summary ……………………………………………………………….. 22
   Appendix 1: Tuning parameter plot for linear classification models……… 23
   Appendix 2: Tuning parameter plot for nonlinear classification models….. 24

8. References………………………………………………………………… 27

9. R- Code…………………………………………………………………… 27

1. Abstract

The process of diagnosing a cardiac arrhythmia entails using an electrocardiogram (ECG) to measure heart activity for irregular heartbeats and then analyzing the recorded data. Doctors can then identify arrhythmia and its category by combining these parameters with patient information.

A few difficulties in recognizing arrhythmias are:

- ❖ While many of the existing algorithms are implemented using rules, the classification of the cardio log is superior and different.
- ❖ A physician would be unable to detect even the smallest steeps and irregularities because there are so many parameters (more than 270) involved.
- ❖ In critical care units, 90% of clinical alarms may be erroneous. Patients and clinical staff are negatively impacted by this high percentage. Additionally, true alarms may be disregarded as a result of desensitization brought on by the alarm overload.

In order to distinguish between the presence and absence of cardiac arrhythmia and to classify it in one of 16 groups using ECG data of new patients, our goal is to develop a classification model that uses the cardio log data as a gold standard. This will support accomplishing the following.

- ❖ Reducing false alarms which in turn helps clinical staff to focus on the attention required areas.
- ❖ Accurate detection to expedite patient's treatment.

2. Background

A heart arrhythmia is a ventricular fibrillation (uh-RITH-me-uh). When the electrical impulses that instruct the heart to beat malfunction, a heart arrhythmia happens. Too fast or too slow heartbeats are possible. Alterations in the heartbeat's rhythm are also possible. An arrhythmia of the heart can feel like a racing, pounding, or fluttering heartbeat. Heart arrhythmias can sometimes be benign. Others might result in symptoms that are fatal.Either a fast or slow heartbeat is acceptable at certain moments.

For instance, the heart may beat more quickly when exercising or more slowly when you sleep.Treatment options for heart arrhythmias can include medications, implants like pacemakers, operations, or surgery. Controlling or eliminating fast, sluggish, or other irregular heartbeats are the main objectives of treatment.

Types

Heart arrhythmias are generally categorized based on heart rate variability. As an illustration:

"Tachycardia" is the pronunciation of "fast heartbeat." There are more than 100 beats per minute in the heart rate.

A bradycardia is characterized by a slow heartbeat. There are less than 60 beats per minute in the heart rate.

Symptoms

There could be no symptoms associated with a heart arrhythmia. For some other reason, a health checkup may reveal an irregular heartbeat.

Symptoms of an arrhythmia may include:

- A fluttering, pounding or racing feeling in the chest.
- A fast heartbeat.
- A slow heartbeat.
- Chest pain.
- Shortness of breath.

Other symptoms may include:

- Anxiety.
- Feeling very tired.
- Lightheadedness or dizziness.
- Sweating.
- Fainting or almost fainting.

3. Dataset

The data contains 452 observations and 279 features for each observation. There are 110 categorical variables, 169 numerical observations and 408 missing values. The dataset can be found at kaggle.

No. of Features	Type	Example Features
110	Categorical	Sex, Rag_R_Nom, Diph_R_Nom, Diph_P
169	Numeric	Age, weight, Height, QRS_Dur

In the dataset, class 01 refers to normal which constitutes 54% of the data. The contribution for various classes of arrhythmia is less when compared to class 01.

Class Label	Class Description
1	Normal
2	Ischemic changes (Coronary Artery Disease)
3	Old Anterior Myocardial Infarction
4	Old Inferior Myocardial Infarction
5	Sinus tachycardy
6	Sinus bradycardy
7	Ventricular Premature Contraction (PVC)
8	Supraventricular Premature Contraction
9	Left bundle branch block
10	Right bundle branch block

11	1 degree AV block
12	2 degree AV block
13	3 degree AV block
14	Left ventricular hypertrophy
15	Atrial Fibrillation or Flutter
16	Others

i. Missing Values

```
         T          P       QRST          J heart_rate
         8         22          1        376          1
>
```

Fig: Table representation of null values

From the above figure, we can see that columns T, P, QRST, J, heart_rate contains missing values with the J column having the majority of missing values.

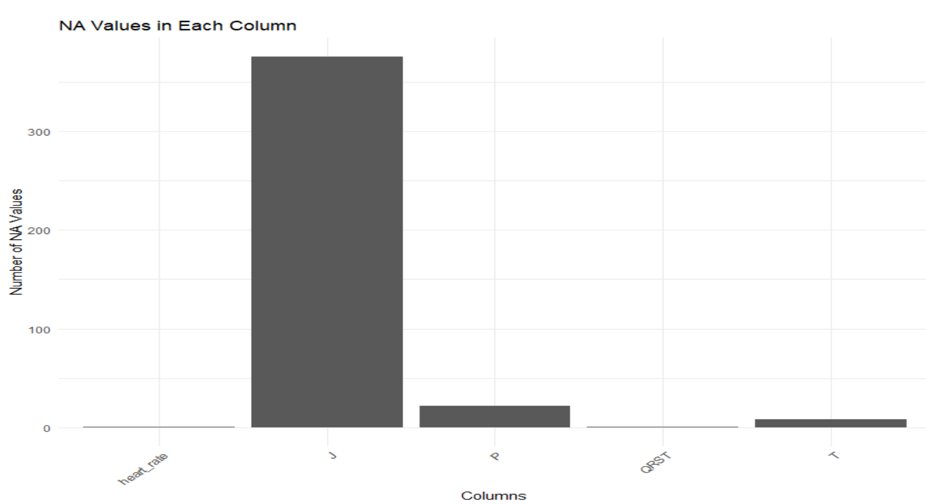


Fig: Barplot representation of null values of the columns

ii. Data Cleaning

Column	Action Performed
heart_rate	There are two records for a male patient with age 75, height 190 and weight 80 and the Heart_Rate for one is missing (table below), a look at ECG values also indicate they are close hence it is possible that both these records belong to same patient and hence I have substituted the missing Heart_Rate with the available heart rate of the other record.
J	J contributes to QRS duration which is available for all records and hence we will drop J with a notion that QRS has the details of J and hence its contribution to the data will not be entirely lost.
P	Used KNN Imputation to replace missing values
QRST	Used KNN Imputation to replace missing values
T	Used KNN Imputation to replace missing values

iii. Outlier Examination

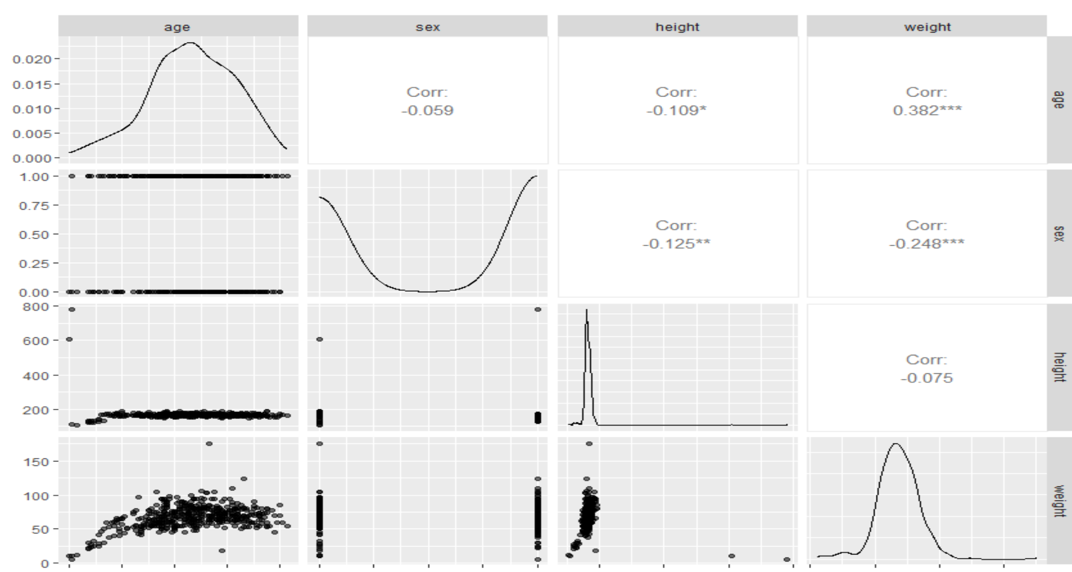


Fig: Outlier examination of observations

Boxplots

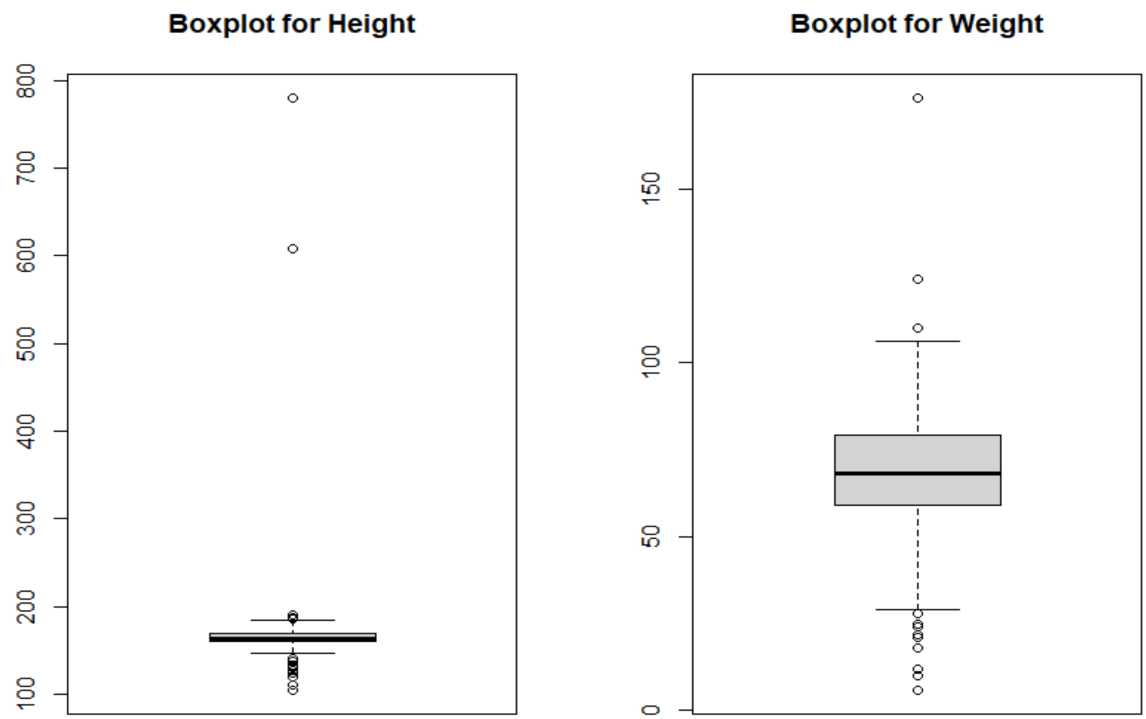


Fig: Boxplot representation of height and weight

From the boxplot, we can observe that there are some outliers in Height and Weight attributes.

Column	**Outliers**	**Action Performed**
Height	2 observations more than 600	is not practical for heights greater than 600 centimeters, so I have replaced them with the median of the remaining observations.
Weight	1 Observation around 160	No action taken as 160kgs looks realistic

Data Exploration

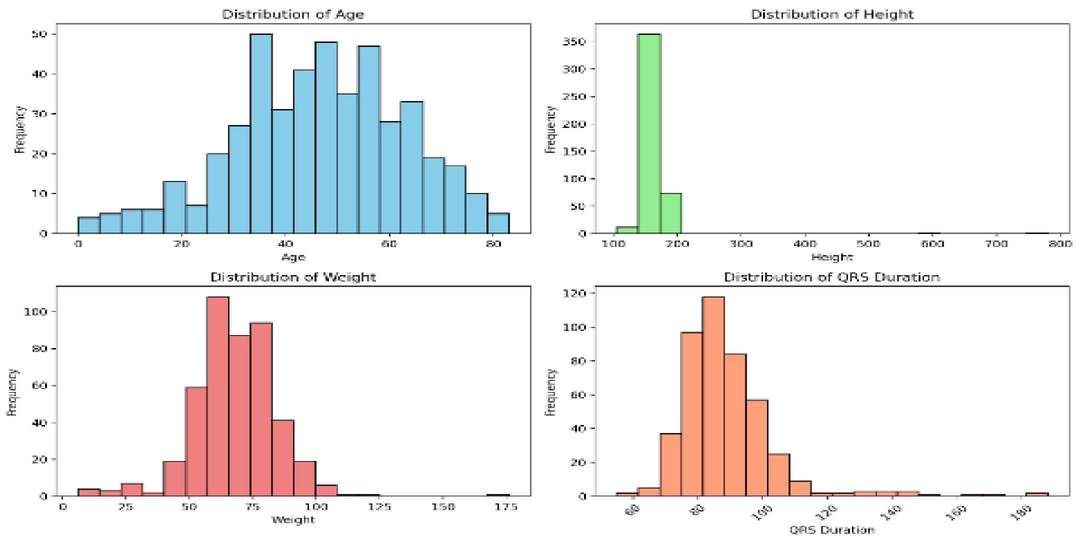


Fig: Histograms of numerical predictors

4. Preprocessing:

a. Correlations

A correlation of 47 predictors has been plotted using correlation plot. The positively correlated predictors are in red color and the negatively correlated predictors are in blue color.

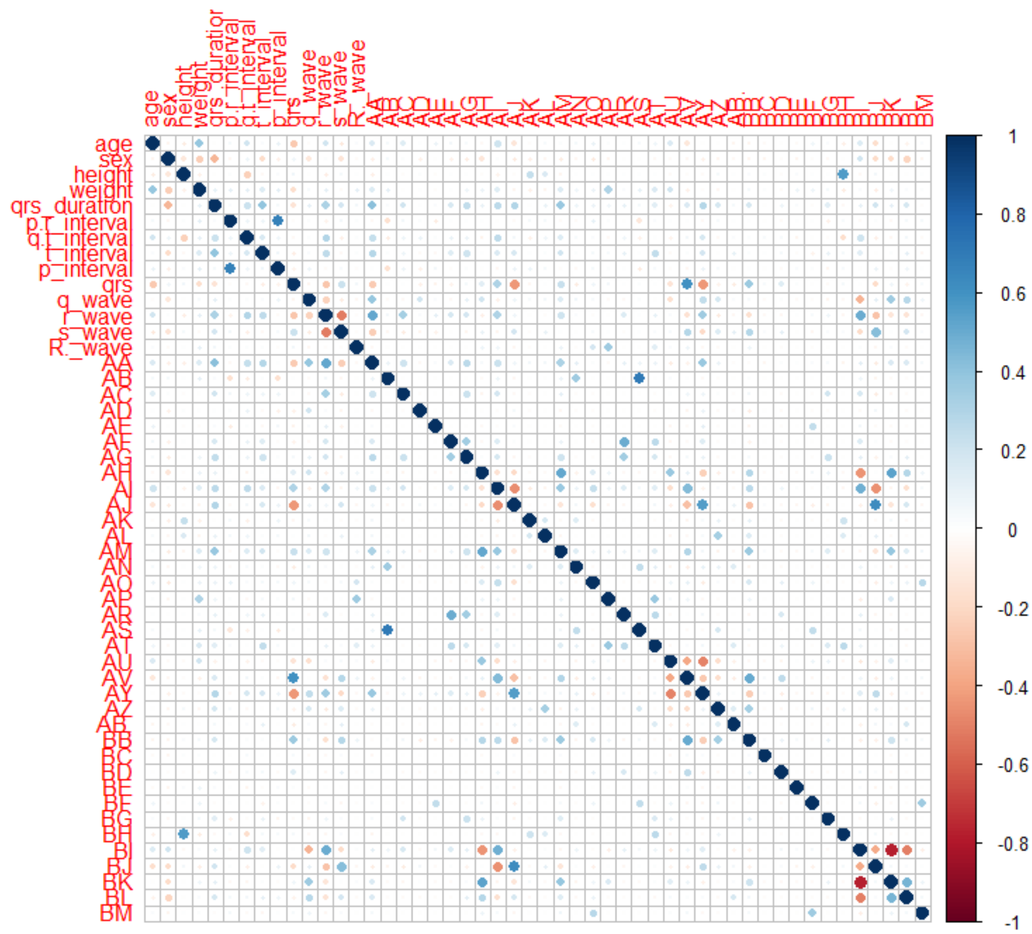


Fig: Correlation Matrix

b. Transformations

To fit any model to the data we should transform the data in order to remove skewness from the data. Below are the histograms for a few variables after transformation.

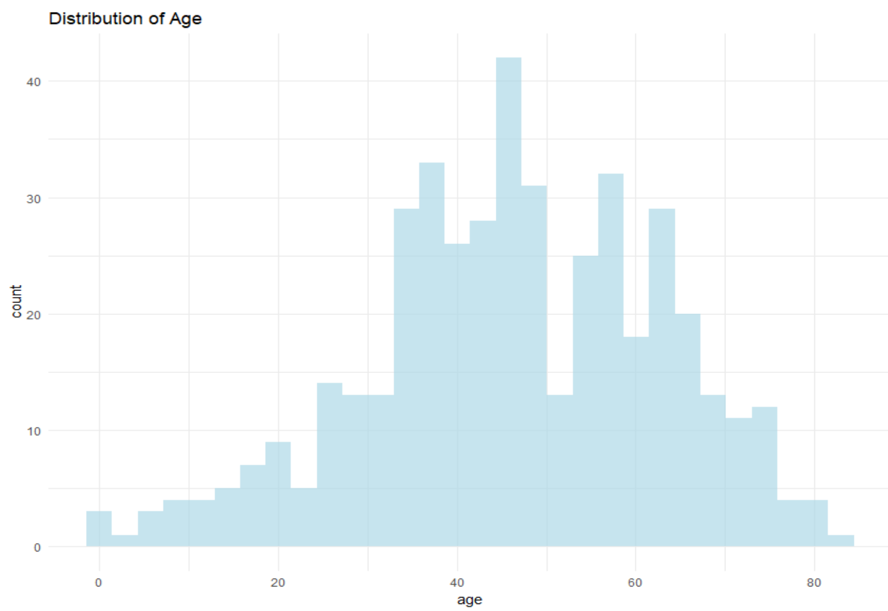


Fig: Histogram of Age

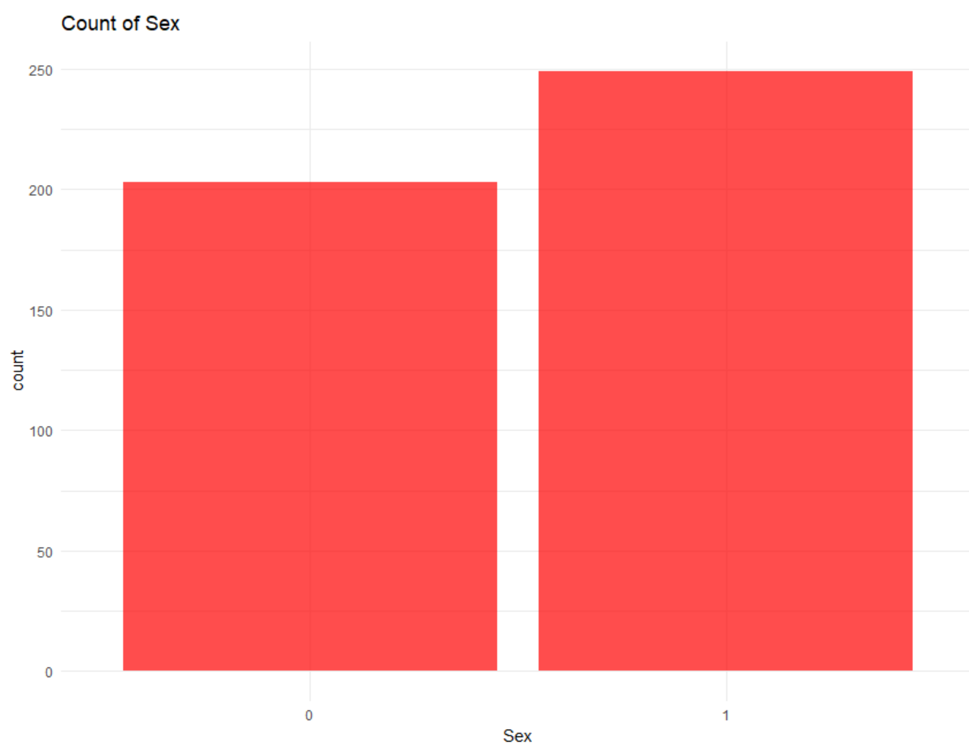


Fig: Histogram of Sex

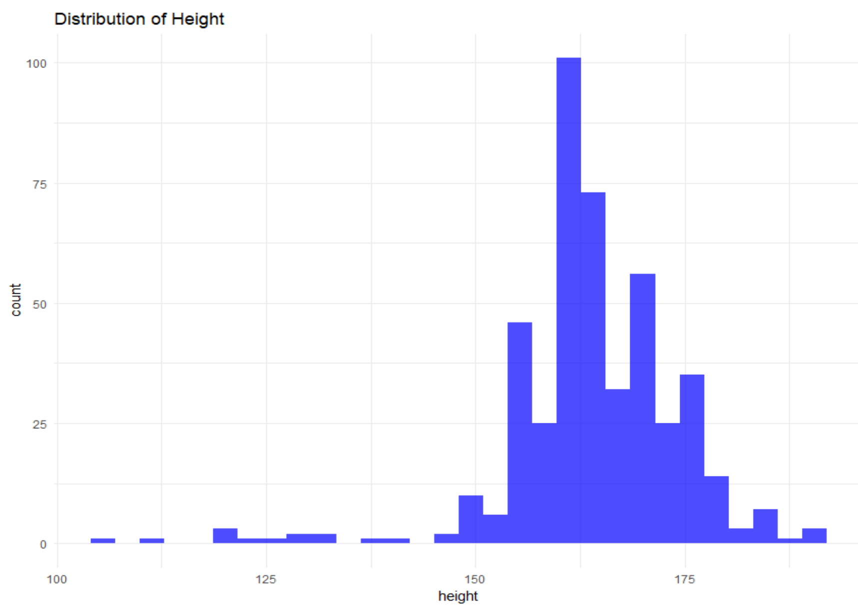


Fig: Histogram of Height

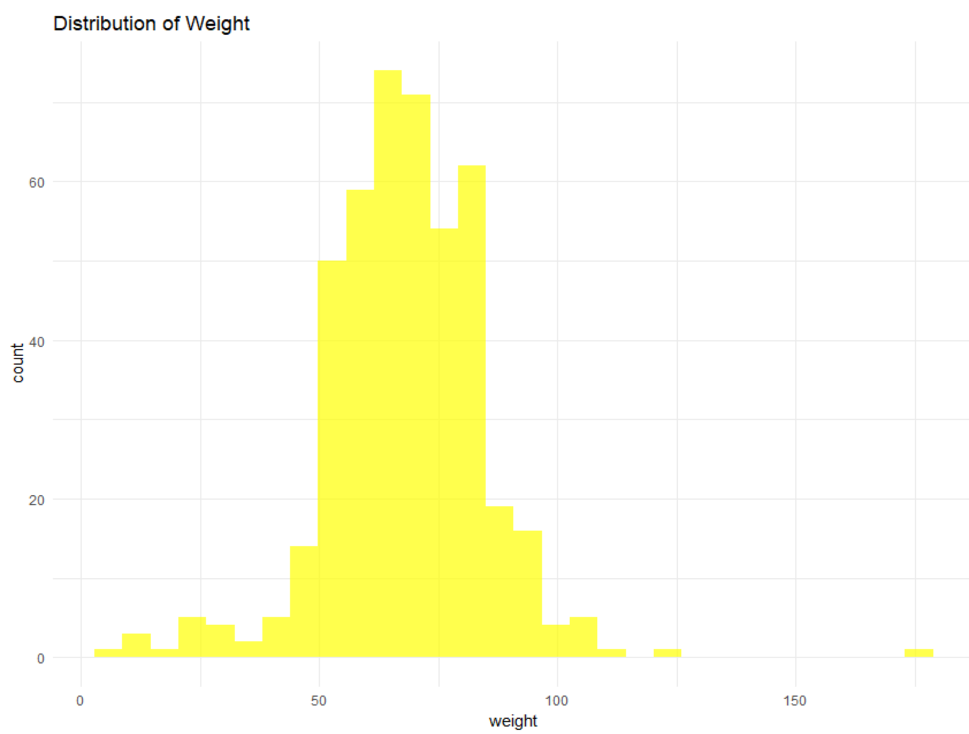


Fig: Histogram of weights

5. Data Splitting

The classes are imbalanced and stratified random sampling has been used for splitting the data into a training and testing set. We used 70% of the data in training set and 30% in the testing set.

6. Model Fitting

The classification models are built and the evaluation metric is **Kappa**. The models are:

1. Linear Classification Models
    a. Linear Discriminant Analysis (LDA)
    b. Partial Least Square Discriminant Analysis (PLSDA)
    c. Penalized Models
    d. Logistic Regression
    e. Nearest Shrunken Centroids
2. Non-Linear Classification Models
    a. Nonlinear Discriminant Analysis
        i. Regularized Discriminant Analysis
        ii. Mixture Discriminant Analysis
    b. Neural Networks
    c. Flexible Discriminant Analysis
    d. Support Vector Machine
    e. K-Nearest Neighbors
    f. Naive Bayes

Model fitting is the process of adjusting a model's parameters to most accurately represent the underlying relationships in a dataset. This entails reducing the discrepancy between the actual data points and the model's predictions.

Outcomes for Linear Classification models:

Classification Model	Training Kappa	Testing Kappa	Accuracy	Best Tuning Parameter
Linear Discriminant Analysis (LDA)	0.4509618	0.5277	0.6674419	N/A
Partial Least Square Discriminant Analysis (PLSDA)	0.3789236	0.4954	0.6558828	ncomp=15
Penalized Model	0.43484420	0.5985	0.6772093	alpha=0.2, lambda=0.01
Logistic Regression	0.3126730	0.3076	0.5302326	decay=1e-01
Nearest Shrunken Centroid	0.346917893	0.4563	0.6609912	Threshold=0.5

Outcomes for Non-Linear Classification models:

Classification Model	**Training Kappa**	**Testing Kappa**	**Accuracy**	**Best Tuning Parameter**
Regularized Discriminant Analysis	0.315940886	0.4343	0.53133175	gamma=1, lambda=1
Mixture Discriminant Analysis	0.3984889	0.5138	0.6247636	subclasses=1
Neural Networks	0.3370664908	0.4913	0.5862480	size=9, decay=0.1
Flexible Discriminant Analysis	0.45501974	0.5123	0.6828431	degree=1, nprune=9
Support Vector Machine	0.3281037	0.276	0.6293008	sigma=0.006724775 c=0.25
K-Nearest Neighbors	0.218172595	0.1988	0.5295972	k=1
Naive Bayes	0.3571407	0.1988	0.6345138	Tuning parameter"fL" was held constant at value of 2

The top two models were selected to predict on the test set for final model selection. It can be seen from the table that Penalized models gave the highest kappa value when predicting on a test set followed by Mixture Discriminant Analysis. The kappa as an evaluation metric is important in this case because of the imbalance classes present in our dataset. This makes kappa a more reliable measure of performance in such cases and also helps us to classify the presence or absence of cardiac arrhythmia.

The top two models have been selected to predict on the test set and below are the accuracy rates and kappa values.

Model	Accuracy	Kappa
Penalized Model	0.6772093	0.5985
Mixture Discriminant Analysis	0.6247636	0.5138

```
Confusion Matrix and Statistics

          Reference
Prediction X1 X10 X15 X16 X2 X3 X4 X5 X6 X9
       X1  46   3   0   3  1  1  2  2  2  0
       X10  0   7   0   1  1  0  0  0  0  0
       X15  0   0   0   0  0  0  0  0  0  0
       X16  1   0   0   1  0  0  0  0  0  0
       X2   1   0   1   0  7  0  0  0  1  1
       X3   0   0   0   0  0  2  0  0  0  0
       X4   0   0   0   0  0  0  1  0  0  0
       X5   0   0   0   0  0  0  0  1  0  0
       X6   1   0   0   0  0  0  0  0  2  0
       X9   0   0   0   0  0  0  0  0  0  1

Overall Statistics

               Accuracy : 0.7556
                 95% CI : (0.6536, 0.84)
    No Information Rate : 0.5444
    P-Value [Acc > NIR] : 2.879e-05

                  Kappa : 0.5985

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: X1 Class: X10 Class: X15 Class: X16 Class: X2 Class: X3
Sensitivity             0.9388    0.70000    0.00000    0.20000   0.77778   0.66667
Specificity             0.6585    0.97500    1.00000    0.98824   0.95062   1.00000
Pos Pred Value          0.7667    0.77778        NaN    0.50000   0.63636   1.00000
Neg Pred Value          0.9000    0.96296    0.98889    0.95455   0.97468   0.98864
Prevalence              0.5444    0.11111    0.01111    0.05556   0.10000   0.03333
Detection Rate          0.5111    0.07778    0.00000    0.01111   0.07778   0.02222
Detection Prevalence    0.6667    0.10000    0.00000    0.02222   0.12222   0.02222
Balanced Accuracy       0.7987    0.83750    0.50000    0.59412   0.86420   0.83333
                     Class: X4 Class: X5 Class: X6 Class: X9
Sensitivity            0.33333   0.33333   0.40000   0.50000
Specificity            1.00000   1.00000   0.98824   1.00000
Pos Pred Value         1.00000   1.00000   0.66667   1.00000
Neg Pred Value         0.97753   0.97753   0.96552   0.98876
Prevalence             0.03333   0.03333   0.05556   0.02222
Detection Rate         0.01111   0.01111   0.02222   0.01111
Detection Prevalence   0.01111   0.01111   0.03333   0.01111
Balanced Accuracy      0.66667   0.66667   0.69412   0.75000
```

Fig: Confusion matrix and Statistics of penalized model

```
The final values used for the model were alpha = 0.2 and lambda = 0.01.
> varImp(glmn_Tune)
glmnet variable importance

  variables are sorted by maximum importance across the classes
  only 20 most important variables shown (out of 47)

         X1     X10      X15     X16        X2       X3      X4      X5      X6       X9
IG   0.0000  0.0000 66.80261 49.1615 100.00000 40.593198 96.1443 32.6824  8.2660  1.57452
sex 60.7931  0.0000  0.03286 25.7048  13.92678 59.059581 10.4669 57.5993 99.0705  0.00000
KG  38.6039 18.1531 76.33323 19.3805   8.46460  8.530744 18.3473  0.0000  0.0000  0.00000
IZ   0.0000  9.0281  0.00000 31.3556   3.51619  1.318635  0.0000  0.0000 68.1239  0.00000
JJ   3.6887 16.7402 30.06011 62.2016  23.95997 25.038058  0.0000  0.0000 67.5908  0.00000
HG   0.0000 43.7073 11.13701 36.6940  20.00479 16.199151  0.0000  4.9233  5.3698 21.18161
KH   5.8939  0.0000  0.89680  5.0544  36.90667  8.465554  1.6475  5.9161 19.3769  0.00000
KA  32.5720  0.0000  9.23469 26.2022   1.60031  0.000000  0.0000 31.4465 21.1383 13.84190
IT   0.3156  0.1978  0.00000  0.0000   5.40679  5.260762  1.0047 14.0735  9.3061  0.00000
JP   0.8946  2.9437  0.00000  2.0504   0.02296  0.868929  0.0000  5.1082 11.3136  0.00000
HM   0.0000  3.3393  0.00000  5.2695   7.93368  1.434472  8.8135  0.0000  9.9619  0.00000
GE   4.4168  9.7771  1.38011  5.7990   5.76576  0.000000  0.0000  1.3557  0.0000  0.19814
ES   2.9001  4.3136  4.48586  0.8274   0.62248  0.000000  2.1604  2.4292  0.0000  0.08759
CJ   1.6710  1.0489  0.00000  0.0000   0.00000  0.000000  4.2118  1.0995  0.0000  0.00000
IF   0.1241  0.9855  0.00000  2.2395   4.14687  0.000000  0.7877  1.1928  2.9323  0.00000
BB   0.0000  0.1239  0.14563  1.9735   1.60466  0.001083  3.8036  0.3284  0.0000  0.00000
DD   2.5743  3.7293  0.00000  0.6861   0.00000  0.718252  0.2402  0.0000  0.2977  0.00000
EB   0.0000  0.3713  0.00000  2.2291   0.22787  3.724217  0.0000  0.0000  0.0000  3.00261
DP   3.5830  0.8785  0.36459  0.6643   0.44311  0.000000  2.8193  0.0000  0.0000  0.00000
AM   1.0190  3.5315  2.01492  0.0000   0.80442  1.955815  2.8211  2.6856  1.4790  1.86219
```

Fig: Top 20 Important predictors of penalized model

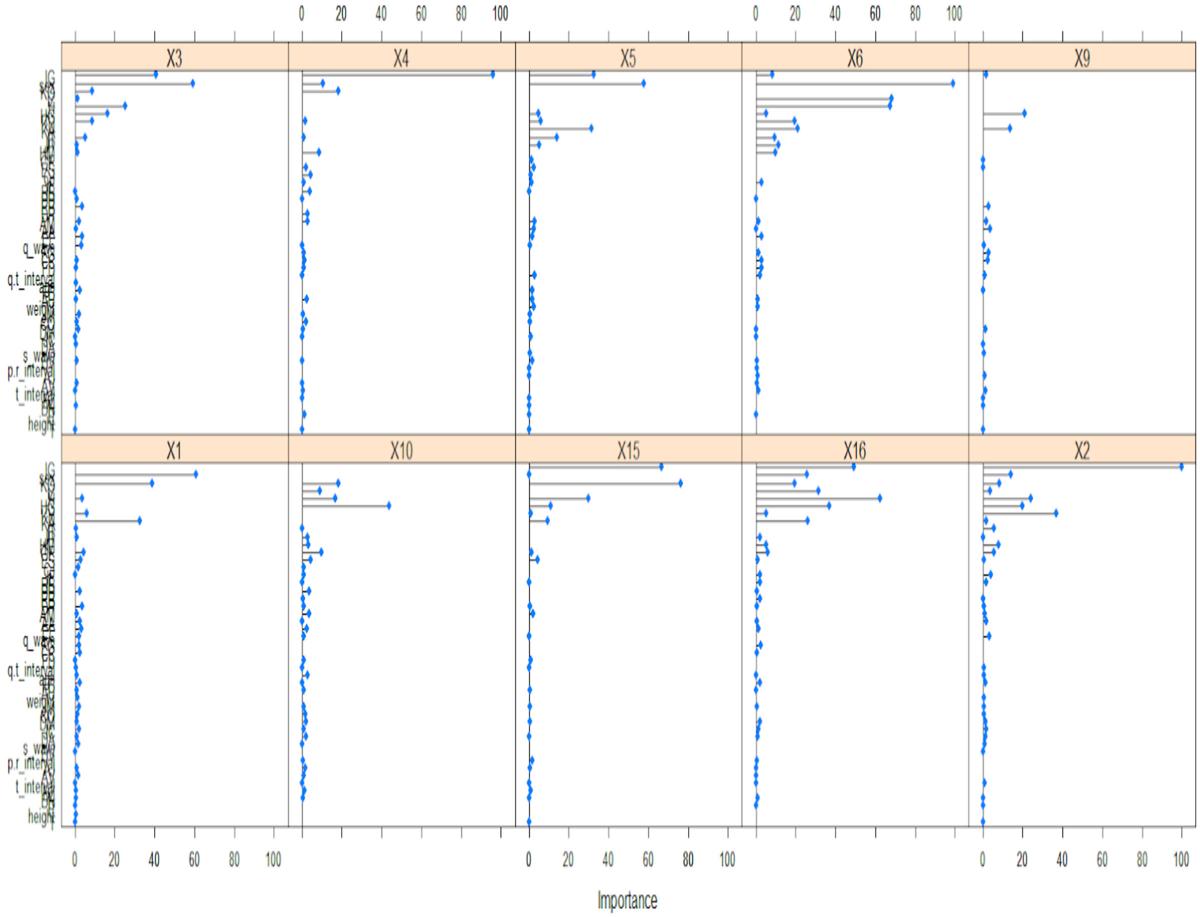


Fig: Plot of top 20 important predictors of penalized model

```
Confusion Matrix and Statistics

          Reference
Prediction X1 X10 X15 X16 X2 X3 X4 X5 X6 X9
       X1  43   4   0   3  2  0  2  2  3  0
       X10  1   6   1   1  1  0  0  0  0  0
       X15  0   0   0   0  1  0  0  0  0  0
       X16  2   0   0   1  0  0  0  0  0  0
       X2   0   0   0   0  5  0  1  0  0  0
       X3   0   0   0   0  0  3  0  0  0  0
       X4   0   0   0   0  0  0  0  0  0  0
       X5   0   0   0   0  0  0  0  1  0  0
       X6   3   0   0   0  0  0  0  0  2  0
       X9   0   0   0   0  0  0  0  0  0  2

Overall Statistics

               Accuracy : 0.7
                 95% CI : (0.5943, 0.7921)
    No Information Rate : 0.5444
    P-Value [Acc > NIR] : 0.001859

                  Kappa : 0.5138

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: X1 Class: X10 Class: X15 Class: X16 Class: X2 Class: X3
Sensitivity             0.8776    0.60000    0.00000    0.20000   0.55556   1.00000
Specificity             0.6098    0.95000    0.98876    0.97647   0.98765   1.00000
Pos Pred Value          0.7288    0.60000    0.00000    0.33333   0.83333   1.00000
Neg Pred Value          0.8065    0.95000    0.98876    0.95402   0.95238   1.00000
Prevalence              0.5444    0.11111    0.01111    0.05556   0.10000   0.03333
Detection Rate          0.4778    0.06667    0.00000    0.01111   0.05556   0.03333
Detection Prevalence    0.6556    0.11111    0.01111    0.03333   0.06667   0.03333
Balanced Accuracy       0.7437    0.77500    0.49438    0.58824   0.77160   1.00000
                     Class: X4 Class: X5 Class: X6 Class: X9
Sensitivity            0.00000   0.33333   0.40000   1.00000
Specificity            1.00000   1.00000   0.96471   1.00000
Pos Pred Value             NaN   1.00000   0.40000   1.00000
Neg Pred Value         0.96667   0.97753   0.96471   1.00000
Prevalence             0.03333   0.03333   0.05556   0.02222
Detection Rate         0.00000   0.01111   0.02222   0.02222
Detection Prevalence   0.00000   0.01111   0.05556   0.02222
Balanced Accuracy      0.50000   0.66667   0.68235   1.00000
```

Fig: Confusion matrix and statistics of mixture discriminant analysis

```
> varImp(mdaFit)
ROC curve variable importance

  variables are sorted by maximum importance across the classes
  only 20 most important variables shown (out of 47)

                 X1    X10    X15    X16     X2      X3     X4      X5     X6     X9
EB           42.137 19.884 37.207 84.398 10.805 18.2204 14.723 100.000 26.283 42.137
CC           15.777 15.777 15.777 48.438 36.908 15.7773 15.777  98.237 33.213 12.820
t_interval   61.168 10.233 14.091 38.922 14.663 33.2390 26.887  97.649 68.497 61.168
CJ            8.847 15.346 16.707 11.748 97.600  8.8466  8.847  19.763  8.847 15.346
p.r_interval 97.428 18.190  3.334 19.421  2.104  0.9387 19.069   2.643 92.439 97.428
AA            2.276  9.416 12.151 36.994  2.362 44.7602  0.000  97.281  0.000  9.416
q.t_interval 38.922  9.356  6.841 11.877 33.779 90.8962 59.136  94.857 39.514 38.922
AM           30.564  4.539 18.955  5.533 71.026 38.1252  4.539  93.093 36.994 30.564
BN           52.848 52.848 52.848 52.848 52.848 52.8481 52.848  52.848 90.549 46.638
JM           74.026 10.660 37.956 90.399 28.893 23.3638 12.768  10.660 50.855 74.026
AJ           55.767 53.555 53.555 53.555 53.555 53.5554 53.555  53.555 89.919 55.767
BB           65.925 15.147 15.147 15.147 88.985 28.0957 15.147  15.147 52.115 65.925
BZ           86.884 11.920 14.855 20.021 50.666 26.6041 11.920  45.995 43.294 86.884
JP           17.217 17.217 17.217 86.199 30.179 38.0224 23.672  53.489 24.392 12.820
IG           85.470 23.757 29.433  5.143 41.537 22.5923  5.143   6.831 77.318 85.470
AU           59.882 14.317 18.985 40.380 85.299 13.6171 13.617  44.966 76.688 59.882
EF           17.320 17.320 17.320 84.227 18.478 31.9532 17.320  65.686 23.132 11.663
DM           33.831 33.831 33.831 83.113 33.831 33.8305 33.831  43.129 33.831 23.757
IT           41.494 35.232 35.232 79.598 35.232 35.2321 35.232  45.333 65.347 41.494
DD           62.801 62.801 62.801 75.740 62.801 62.8006 62.801  62.801 76.688 39.437
>
```

Fig: Top 20 Important predictors of Mixture Discriminant Analysis

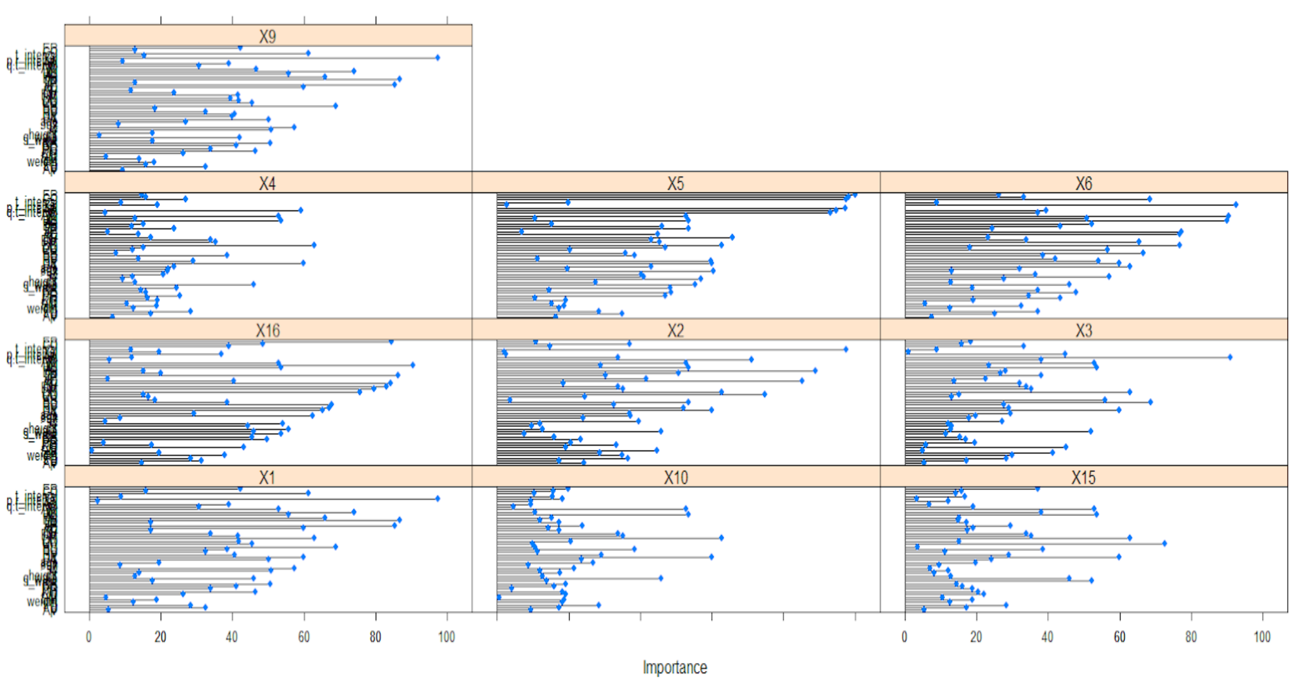


Fig: Plot of top 20 important predictors of Mixture Discriminant Analysis

7. Summary

From the above tables, we can say that the best model found during the analysis is Penalized model followed by mixture discriminant analysis which is used predict the presence of cardiac arrhythmia in a person with the accuracy rates of 0.6772093 , 0.6247636 and kappa values of 0.5985, 0.5138 respectively.

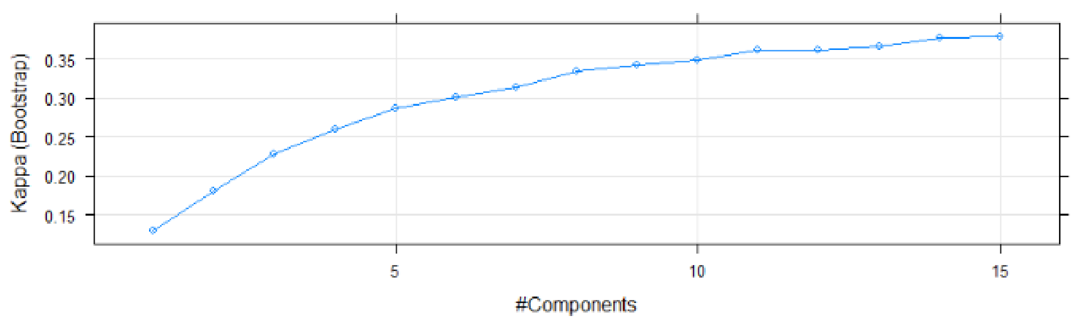
Appendix 1: Tuning parameter plot for linear classification models

a. Linear Discriminant Analysis

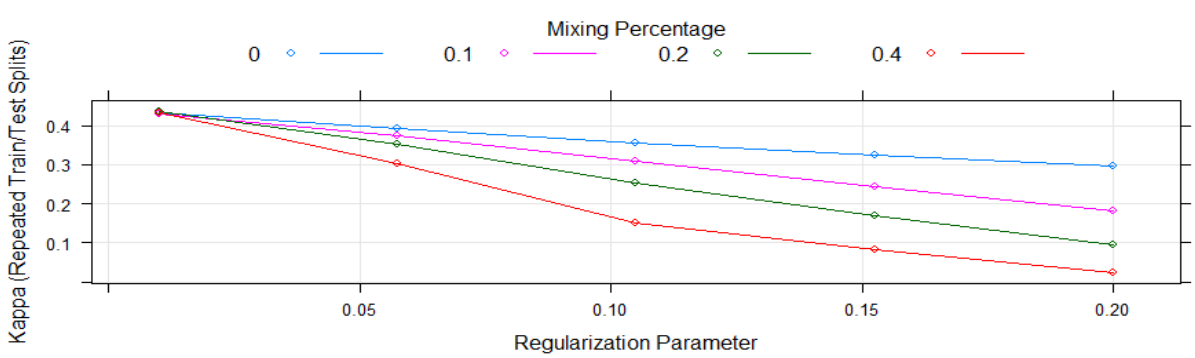
NO PLOT

(There are no tuning parameters for this model)

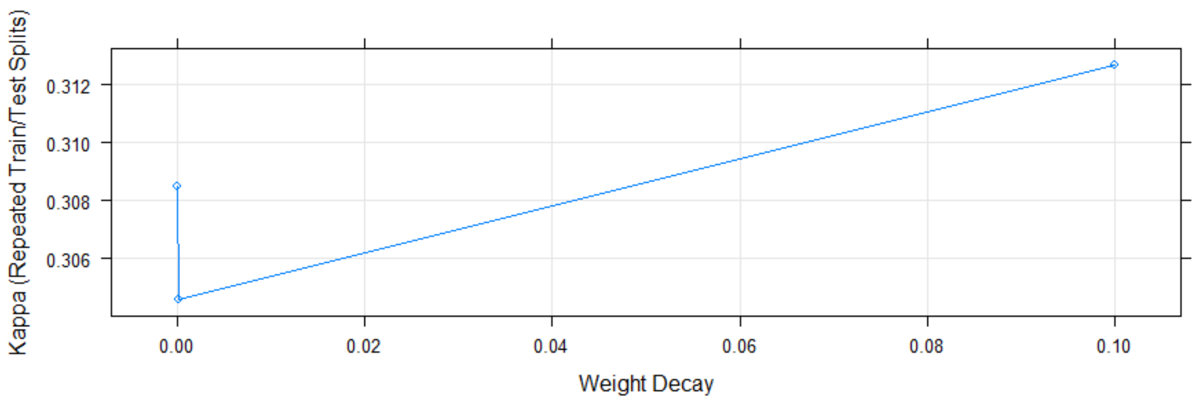
b. Partial Least square discriminant analysis (PLSDA) : The optimal model uses ncomp=15



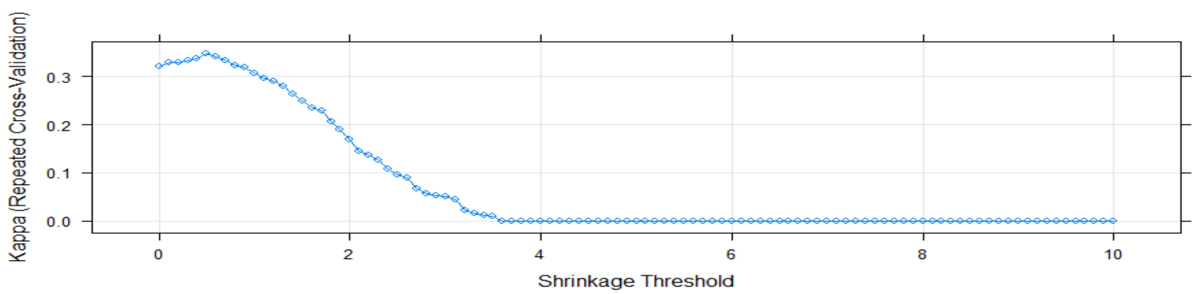
c. Penalized Models: The optimal model uses alpha = 0.2 , lambda = 0.01



d. Logistic Regression: The optimal model has decay of 1e-01

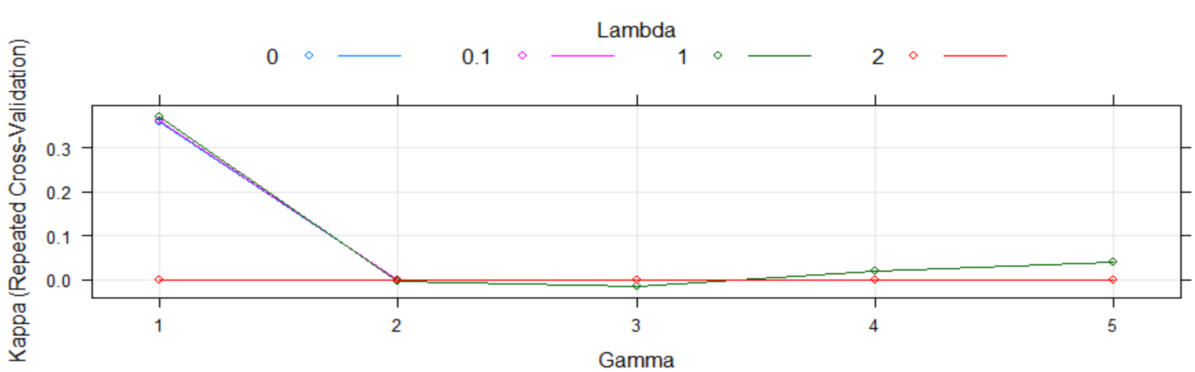


e. Nearest Shrunken Centroid: The optimal model has threshold of 0.5

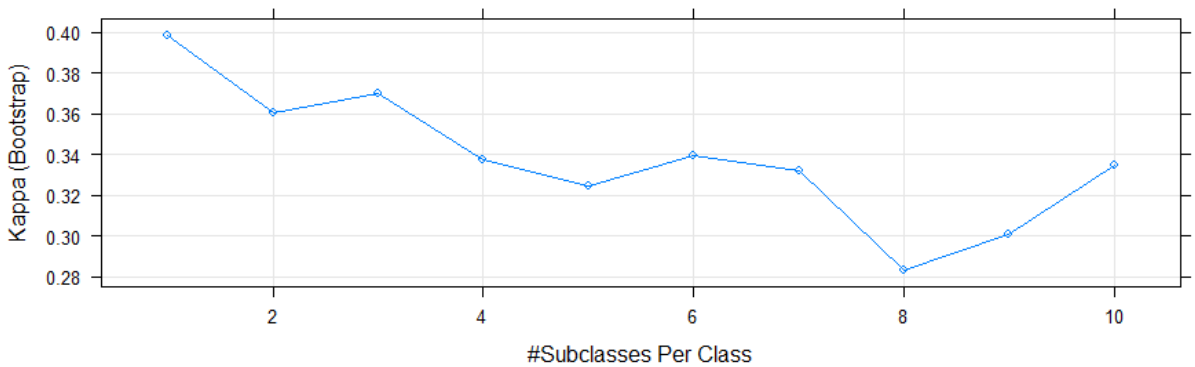


Appendix 2: Tuning Parameter plot for Non-Linear classification model

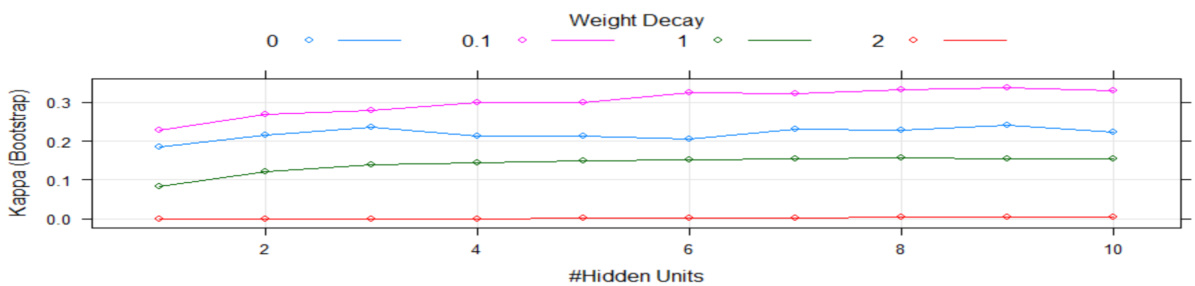
a. Regularized Discriminant Analysis: The optimal model uses gamma=1, lambda=1



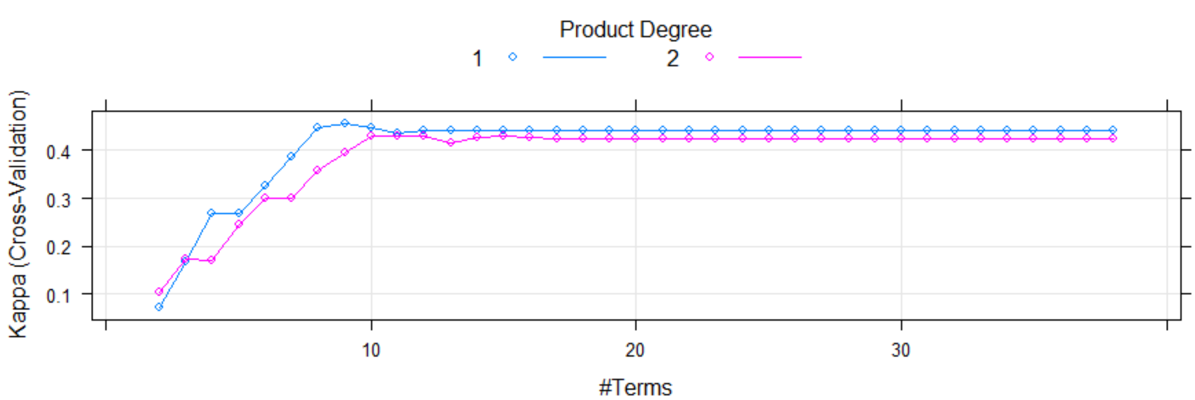
b. Mixture Discriminant Analysis: The optimal model uses subclass of 1



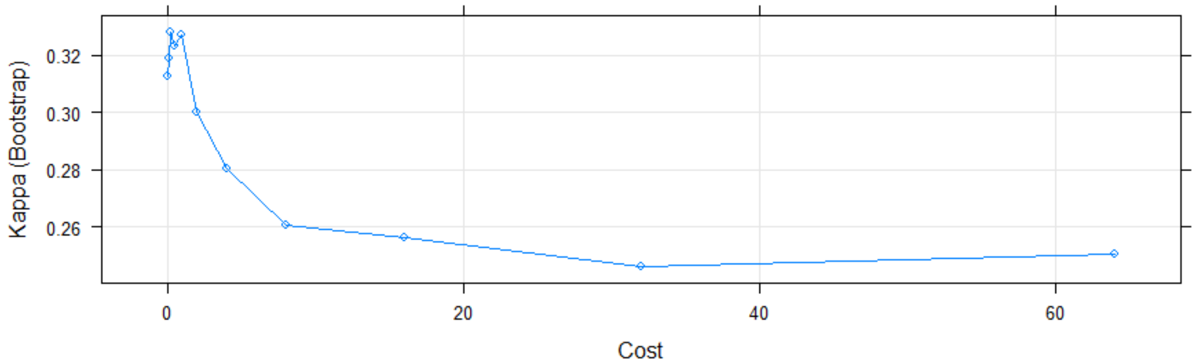
c. Neural Networks: The optimal model uses size=9 , decay=0.1



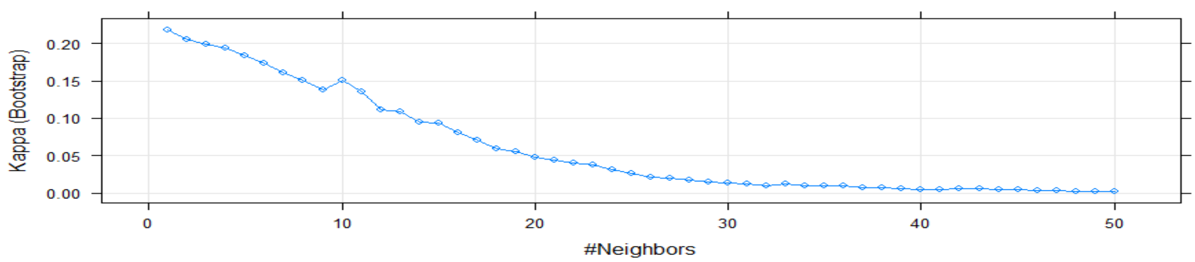
d. Flexible Discriminant Analysis: The optimal model uses degree=1 and nprune=9



e. Support Vector Machines: The optimal model uses sigma=0.006724775, C= 0.25



f. K-Nearest Neighbors: The optimal model uses k=1



g. Naive Bayes: The optimal model has tuning parameter "fL" which was held constant at value of 2

NO PLOT

(There are no tuning parameters for this model)

8. References:

- https://www.health.harvard.edu/a_to_z/cardiac-arrhythmias-a-to-z
- https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9152186/
- https://ieeexplore.ieee.org/abstract/document/1020538
- https://link.springer.com/article/10.1007/BF02344702
- https://www.sciencedirect.com/science/article/pii/S0957417408005745

9. R-CODE

```
library(ggplot2)
library(dplyr)
library(reshape2)

#read
data_arrhythmia <-
read.csv("data_arrhythmia.csv",sep=';',header=TRUE,na.strings=c("?"))
data_arrhythmia
#View(data_arrhythmia)

# Check for null values in the dataset
null_values <- sapply(data_arrhythmia, function(x) sum(is.na(x)))
na_counts <- null_values[null_values > 0]

# Print the null values if there are any
if (length(na_counts) > 0) {
  print(na_counts)
} else {
  print("No null values found")
}

#barplot representation of null values
```

```
ggplot(data = data.frame(Column = names(na_counts), NAs = na_counts), aes(x = 
Column, y = NAs)) +
  geom_bar(stat = 'identity') +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(x = 'Columns', y = 'Number of NA Values', title = 'NA Values in Each 
Column')

#Handling Missing Values
# Heart rate
heart_rate_mean <- mean(data_arrhythmia$heart_rate, na.rm = TRUE)
data_arrhythmia$heart_rate[is.na(data_arrhythmia$heart_rate)] <- 
heart_rate_mean
sum(is.na(data_arrhythmia$heart_rate))

# dropping J Column

data_arrhythmia$J<-NULL
View(data_arrhythmia)

#P Column (KNN Imputation)
library(VIM)

# Perform KNN imputation
imputed_data_p <- kNN(data_arrhythmia, variable='P', k=5)

# Display the head of the imputed dataframe
data_arrhythmia <- imputed_data_p
is.na(data_arrhythmia$P)

#QRST
median_QRST <- median(data_arrhythmia$QRST, na.rm = TRUE)
data_arrhythmia$QRST[is.na(data_arrhythmia$QRST)] <- median_QRST
is.na(data_arrhythmia$QRST)

#T column
```

```
imputed_data_t <- kNN(data_arrhythmia, variable='T', k=5)
data_arrhythmia <- imputed_data_t
is.na(data_arrhythmia$T)

#Outlier Examination
library(GGally)
subset_data <- data_arrhythmia %>% select(age, sex, height, weight)
ggpairs(subset_data,aes(alpha=0.8))


#Boxplot
par(mfrow=c(1,2))
boxplot(data_arrhythmia$height, main='Boxplot for Height')
boxplot(data_arrhythmia$weight, main='Boxplot for Weight')

#Height column
median_height <- median(data_arrhythmia[data_arrhythmia$height < 600,
'height'], na.rm=TRUE)
data_arrhythmia[data_arrhythmia$height > 600, 'height'] <- median_height
data_arrhythmia[1:6, ]

#Data Exploration
par(mfrow=c(1,2))
# Plotting the distributions

# Age distribution
ggplot(subset_data, aes(x=age)) +
  geom_histogram(bins=30, fill="lightblue", alpha=0.7) +
  labs(title="Distribution of Age") +
  theme_minimal()

# Sex count plot
ggplot(subset_data, aes(x=factor(sex))) +
  geom_bar(fill="red", alpha=0.7) +
  labs(title="Count of Sex", x="Sex") +
  theme_minimal()
```

```
# Height distribution
ggplot(subset_data, aes(x=height)) +
  geom_histogram(bins=30, fill="blue", alpha=0.7) +
  labs(title="Distribution of Height") +
  theme_minimal()

# Weight distribution
ggplot(subset_data, aes(x=weight)) +
  geom_histogram(bins=30, fill="yellow", alpha=0.7) +
  labs(title="Distribution of Weight") +
  theme_minimal()

# Heart rate distribution
ggplot(subset_data, aes(x=heart_rate)) +
  geom_histogram(bins=30, fill="gray", alpha=0.7) +
  labs(title="Distribution of Age") +
  theme_minimal()


library(ggplot2)

#Heatmap after handling missing values
corrplot::corrplot(cor(data_arrhythmia[,1:50]),height="500")

NZVfingerprints <- nearZeroVar(data_arrhythmia)
noNZVfingerprints <- data_arrhythmia[,-NZVfingerprints]
print(str(noNZVfingerprints))

high_Cor<-findCorrelation(cor(noNZVfingerprints),cutoff = 0.70)
noNZVfingerprints<-noNZVfingerprints[,-high_Cor]
dim(noNZVfingerprints)

# stratified random sample splitting with 70% training and 30% testing

filtered_data$diagnosis <-make.names(filtered_data$diagnosis,unique =
FALSE,allow_ = FALSE)
```

```
filtered_data$diagnosis<-as.factor(filtered_data$diagnosis)
#class(noNZVfingerprints$diagnosis)
set.seed(1234)
folds <- createFolds(filtered_data$diagnosis, k = 5, list = TRUE, returnTrain =
FALSE)

# Perform cross-validation
for (fold in seq_along(folds)) {
  # Extract indices for the current fold
  test_indices <- unlist(folds[[fold]])
}

#train_Rows = createDataPartition(data_arrhythmia, p = .7, list= FALSE,strata =
as.factor(data_arrhythmia$diagnosis))
train_Fprints <- noNZVfingerprints[-test_indices,]
train_Permeability <- factor(filtered_data[-test_indices,280])
test_Fprints <- noNZVfingerprints[test_indices,]
test_Permeability <- factor(filtered_data[test_indices,280])
ctrl <- trainControl(method = "repeatedcv", repeats=3, number = 3)

# PLS Model
permeabiltyPLS <- train(x = train_Fprints , y = train_Permeability,
                preProcess = c("center","scale"), method = "pls",
                metric="Kappa",
                tuneGrid = expand.grid(ncomp = 1:15), trControl = ctrl)

print(permeabiltyPLS)
plot(permeabiltyPLS,main = "PLS Tuning Parameter for Permeability Data")

predictPLS <- predict(permeabiltyPLS,newdata=test_Fprints)
confusionMatrix(predictPLS,test_Permeability)


#Logistic Regression
set.seed(124)
ctrl <- trainControl(method = "LGOCV",
                  summaryFunction = defaultSummary,
```

```
                  classProbs = TRUE,
                  savePredictions = TRUE)

logModel <- train(x=train_Fprints,y=train_Permeability,method = "multinom",
                metric = "Kappa", trControl = ctrl)

print(logModel)
confMatrixLog <- confusionMatrix(logModel$pred$pred, logModel$pred$obs)
print(confMatrixLog)
plot(logModel)
summary(logModel)

#LDA
ldaTune<-train(x=train_Fprints,y=train_Permeability,method = "lda",
             metric = "Kappa",trControl = ctrl)
ldaTune
plot(ldaTune)
ldaPred<-predict(ldaTune,newdata = test_Fprints)
confusionMatrix(ldaPred,test_Permeability)

#Penalised model
glmnGrid <- expand.grid(.alpha=c(0,.1,.2,.4),.lambda=seq(.01,0.2,length=5))
set.seed(369)
glmn_Tune<-train(x=train_Fprints,y=train_Permeability,method =
"glmnet",tuneGrid =
                    glmnGrid,
                  metric = "Kappa",trControl = ctrl)

glmn_Tune
plot(glmn_Tune)
glmPred<-predict(glmn_Tune,newdata = test_Fprints)
confusionMatrix(glmPred,test_Permeability)
varImp(glmn_Tune)
plot(varImp(glmn_Tune,top=5))
#Nearest Shruken Centroid

library(pamr)
```

```
nsc_GridBio<-data.frame(.threshold=seq(0,10, by=0.1))
set.seed(951)
nsc_TuneBio<-train(x=train_Fprints,y=train_Permeability,method = "pam",
             tuneGrid = nsc_GridBio,
             metric = "Kappa",trControl = ctrl)
nsc_TuneBio

pred_nscBio<-predict(nsc_TuneBio,newdata=test_Fprints)
pred_nscBio
levels(train_Permeability)
plot(nsc_TuneBio)
summary(nsc_TuneBio)
confusionMatrix(data = pred_nscBio,reference = test_Permeability)


####NON Linear
#mda
ctrl <- trainControl(summaryFunction = defaultSummary,
              classProbs = TRUE)
set.seed(476)
mdaFit <- train(x = train_Fprints,
            y = train_Permeability,
            method = "mda",
            metric = "Kappa",
            tuneGrid = expand.grid(.subclasses = 1:10),
            trControl = ctrl)
mdaFit
plot(mdaFit)
mdaPred<-predict(mdaFit,newdata = test_Fprints)
print(confusionMatrix(mdaPred,test_Permeability))
varImp(mdaFit)
plot(varImp(mdaFit,top=5))
#qda
qdaFit <- train(x = train_Fprints,
            y = train_Permeability,
            method = "qda",
            metric = "Kappa",
```

```
               tuneGrid = expand.grid(.subclasses = 1:10),
               trControl = ctrl)
qdaFit

#rda
rdaGrid <- expand.grid(.gamma= 1:5, .lambda = c(0, .1, 1, 2))
rdaFit <- train(x = train_Fprints,
               y = train_Permeability,
               method = "rda",
               metric = "Kappa",
               tuneGrid = rdaGrid,
               trControl = ctrl)
rdaFit
plot(rdaFit)
rdaPred<-predict(rdaFit,newdata = test_Fprints)
print(confusionMatrix(rdaPred,test_Permeability))
#nnet
nnetGrid <- expand.grid(.size = 1:10, .decay = c(0, .1, 1, 2))
maxSize <- max(nnetGrid$.size)
numWts <- (maxSize * (96 + 1) + (maxSize+1)*2)
ctrl <- trainControl(summaryFunction = defaultSummary,
                    classProbs = TRUE)
set.seed(359)

nnet_Fit <- train(x = train_Fprints,
                 y = train_Permeability,
                 method = "nnet",
                 metric = "Kappa",
                 preProc = c("center", "scale", "spatialSign"),
                 tuneGrid = nnetGrid,
                 trace = FALSE,
                 maxit = 2000,
                 MaxNWts = numWts,
                 trControl = ctrl)
nnet_Fit
plot(nnet_Fit)
summary(nnet_Fit)
```

```
nnet_Pred <- predict(nnet_Fit,newdata = test_Fprints)
confusionMatrix(nnet_Pred,test_Permeability)

#FDA
marsGrid <- expand.grid(.degree = 1:2, .nprune = 2:38)

fda_Fit <- train(x = train_Fprints,
                 y = train_Permeability,
                 method = "fda",
                 metric = "Kappa",
                 tuneGrid = marsGrid,
                 trControl = trainControl(method = "cv"))

fda_Fit
plot(fda_Fit)
summary(fda_Fit)
fda_Pred <- predict(fda_Fit,newdata = test_Fprints)
confusionMatrix(fda_Pred,test_Permeability)

library(kernlab)
#SVM
ctrl <- trainControl(summaryFunction = defaultSummary,
                     classProbs = TRUE)
sigmaRangeReduced <- sigest(as.matrix(train_Fprints))
svmRGridReduced <- expand.grid(.sigma = sigmaRangeReduced[1],
                               .C = 2^(seq(-4, 6)))
set.seed(125)
svmR_Model <- train(x = train_Fprints,
                    y = train_Permeability,
                    method = "svmRadial",
                    metric = "Kappa",
                    preProc = c("center", "scale"),
                    tuneGrid = svmRGridReduced,
                    fit = FALSE,
                    trControl = ctrl)
svmR_Model
plot(svmR_Model)
```

```
summary(svmR_Model)
svm_Pred <- predict(svmR_Model,newdata = test_Fprints)
confusionMatrix(svm_Pred,test_Permeability)


#Knn
ctrl <- trainControl(summaryFunction = defaultSummary,
                  classProbs = TRUE)
set.seed(852)
knn_Fit <- train(x = train_Fprints,
               y = train_Permeability,
               method = "knn",
               metric = "Kappa",
               preProc = c("center", "scale"),
               ##tuneGrid = data.frame(.k = c(4*(0:5)+1, 20*(1:5)+1, 50*(2:9)+1)), ##
21 is the best
               tuneGrid = data.frame(.k = 1:50),
               trControl = ctrl)
knn_Fit
plot(knn_Fit)
summary(knn_Fit)
knn_Pred <- predict(knn_Fit,newdata = test_Fprints)
confusionMatrix(knn_Pred,test_Permeability)

#Naive Bayes
install.packages("klaR")
library(klaR)
set.seed(476)
nbFit <- train( x = train_Fprints,
              y = train_Permeability,
              method = "nb",
              metric = "Kappa",
              ## preProc = c("center", "scale"),
              ##tuneGrid = data.frame(.k = c(4*(0:5)+1, 20*(1:5)+1, 50*(2:9)+1)), ##
21 is the best
              tuneGrid = data.frame(.fL = 2,.usekernel = TRUE,.adjust = TRUE),
              trControl = ctrl)
```

```
nbFit
plot(nbFit)
summary(nbFit)
nb_Pred <- predict(nbFit,newdata = test_Fprints)
confusionMatrix(knn_Pred,test_Permeability)
```