INTRODUCTION

This sample uses the caret package in R to perform classification of the bank note dataset
avaiable on http://archive.ics.uci.edu/ml/. The dataset is read directly from the website into R.

The algorithms investigated here are naive Bayes, k nearest neighbour, Random Forest and Decion Tree (C 4.5 leveraging J48 algorithm).

The core is to perform different cross validation methods on the dataset and then compare their performances leveraging receiver operating characteristic curve (ROC) as the performance metric.

Now let's get started.

(Data Set Selection and Visualization):

Data Set Information:

The dataset used for the classification tasks was obtained from the Center for Machine Learning and Intelligent Systems [1]. Data were extracted from images that were taken for the evaluation of an authentication procedure for banknotes.
Data were extracted from images that were taken from genuine and forged banknote like specimens. For digitization, an industrial camera usually used for print inspection was used. The final images have 400x 400 pixels. Due to the object lens and distance to the investigated object gray-scale pictures with a resolution of about 660 dpi were gained. Wavelet Transform tool were used to extract features from images.

Attribute Information:

1. variance of Wavelet Transformed image (continuous)
2. skewness of Wavelet Transformed image (continuous)
3. curtosis of Wavelet Transformed image (continuous)
4. entropy of image (continuous)
5. class (integer) [1 = genuine and 2 = forged banknote]

Let's load and check the structure of the dataset:

```
> link = "http://archive.ics.uci.edu/ml/machine-learning-databases/00267/data_banknote_aut
hentication.txt"
> banknote <- read.table(link, header = FALSE, sep = ",")
> colnames(banknote) <- c("variance", "skewness", "curtosis", "entropy", "class")
> str(banknote)
'data.frame':	1372 obs. of  5 variables:
 $ variance: num  3.622 4.546 3.866 3.457 0.329 ...
 $ skewness: num  8.67 8.17 -2.64 9.52 -4.46 ...
 $ curtosis: num  -2.81 -2.46 1.92 -4.01 4.57 ...
 $ entropy : num  -0.447 -1.462 0.106 -3.594 -0.989 ...
 $ class   : int  0 0 0 0 0 0 0 0 0 0 ...
>
```

The class variable is shown as integer, we'll convert it to factor:

```
> banknote$class<- as.factor(banknote$class)
> str(banknote)
'data.frame':	1372 obs. of  5 variables:
 $ variance: num  3.622 4.546 3.866 3.457 0.329 ...
 $ skewness: num  8.67 8.17 -2.64 9.52 -4.46 ...
 $ curtosis: num  -2.81 -2.46 1.92 -4.01 4.57 ...
 $ entropy : num  -0.447 -1.462 0.106 -3.594 -0.989 ...
 $ class   : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
>
```

Let’s check the summary of the dataset:

```
> summary(banknote)
    variance          skewness          curtosis          entropy        class
 Min.   :-7.0421   Min.   :-13.773   Min.   :-5.2861   Min.   :-8.5482   0:762
 1st Qu.:-1.7730   1st Qu.: -1.708   1st Qu.:-1.5750   1st Qu.:-2.4135   1:610
 Median : 0.4962   Median :  2.320   Median : 0.6166   Median :-0.5867
 Mean   : 0.4337   Mean   :  1.922   Mean   : 1.3976   Mean   :-1.1917
 3rd Qu.: 2.8215   3rd Qu.:  6.815   3rd Qu.: 3.1793   3rd Qu.: 0.3948
 Max.   : 6.8248   Max.   : 12.952   Max.   :17.9274   Max.   : 2.4495
>
```

We have 762 forged and 610 genuine banknote. The numerical variables are in different scales or units so normalization to [0, 1] will be required and this will be applied latter on to help the models learn better. Moreover, normalization will not hurt our models.

Next we’ll check if there is any NA:

```
> anyNA(banknote)
[1] FALSE
>
```

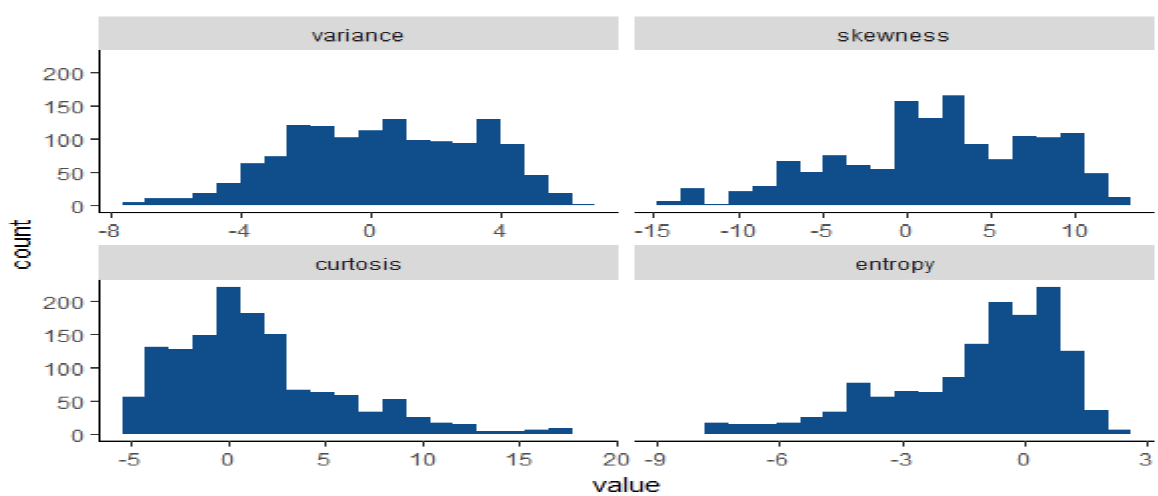
There is no missing data in this dataset.

Next we’ll visualize the banknote dataset to get more insights:

VISUALIZE UNIVARIATE VARIABLES

Next is to plot the histogram of all the numerical variables:

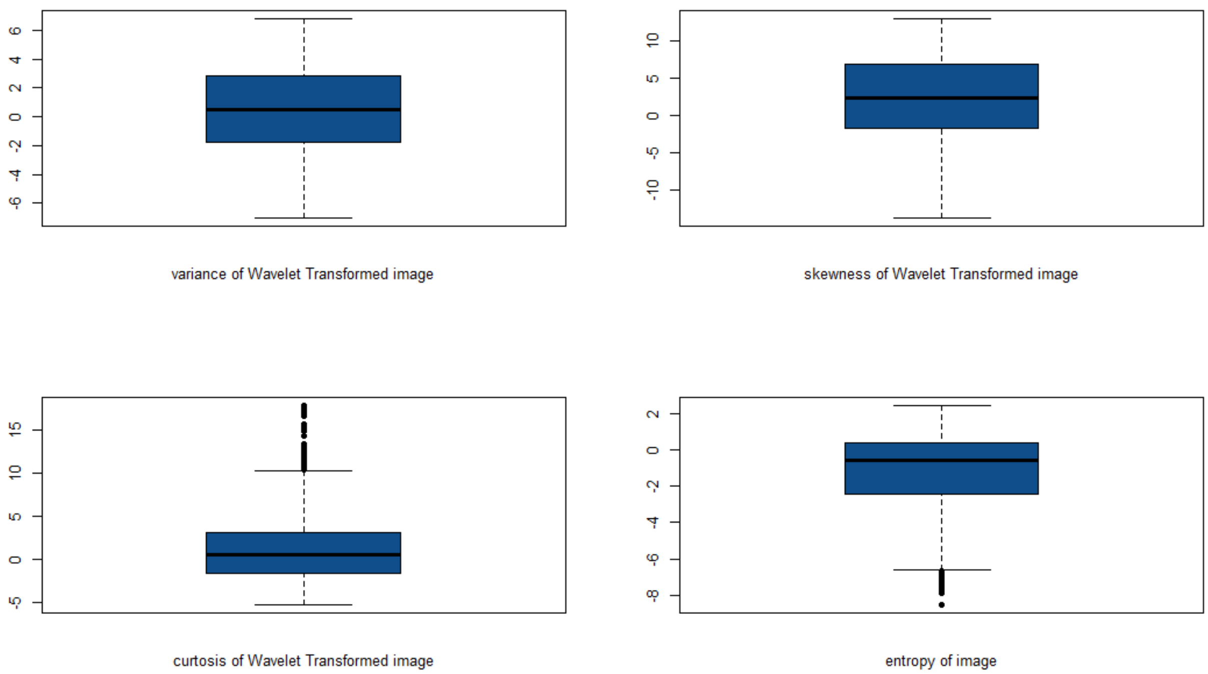
```
> library(ggplot2)
> library(magrittr)
>
> ggplot(data = reshape2::melt(banknote[, -5]), mapping = aes(x = value)) +
+   geom_histogram(bins = 20, fill = "dodgerblue4") + facet_wrap(~variable, scales = 'free
_x')+
+   theme(panel.grid.major = element_blank(),
+         panel.grid.minor = element_blank(),
+         panel.background = element_blank(),
+         axis.line = element_line(colour = "black"))
No id variables; using all as measure variables
>
```



Curtosis appears to be skewed to the right with some outliers, while entropy appears to be skewed to the left with some outliers. Variance and skewness are both skewed to the right but with no pronounced outlier.

Let's plot boxplot to complement the histogram:

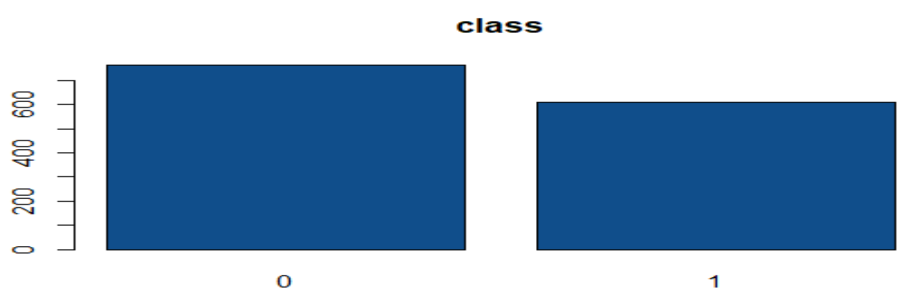
```
> par( mfrow = c( 2, 2 ) )
> boxplot(banknote$variance, col= "dodgerblue4", pch=19)
> mtext("variance of Wavelet Transformed image", cex=0.8, side=1, line=2)
> boxplot(banknote$skewness, col= "dodgerblue4", pch=19)
> mtext("skewness of Wavelet Transformed image", cex=0.8, side=1, line=2)
> boxplot(banknote$curtosis, col= "dodgerblue4", pch=19)
> mtext("curtosis of Wavelet Transformed image", cex=0.8, side=1, line=2)
> boxplot(banknote$entropy, col= "dodgerblue4", pch=19)
> mtext("entropy of image", cex=0.8, side=1, line=2)
```



Curtosis has some outliers on the higher side, entropy has some outliers on the lower side. variance and skewness not to have outlier. This affirm the result of the histogram.

Next let's plot the bar plot of the class variables:

```
> barplot(table(banknote$class), col = "dodgerblue4", main = "class")
> 
```



The class variable contain more forged than genuine banknotes. The class is reasonably balanced.

Next let's get more insight into the class variable:

```
> table(banknote$class)

  0   1
762 610
> paste0(round(100*prop.table(table(banknote$class)), 2), "%")
[1] "55.54%" "44.46%"
>
```

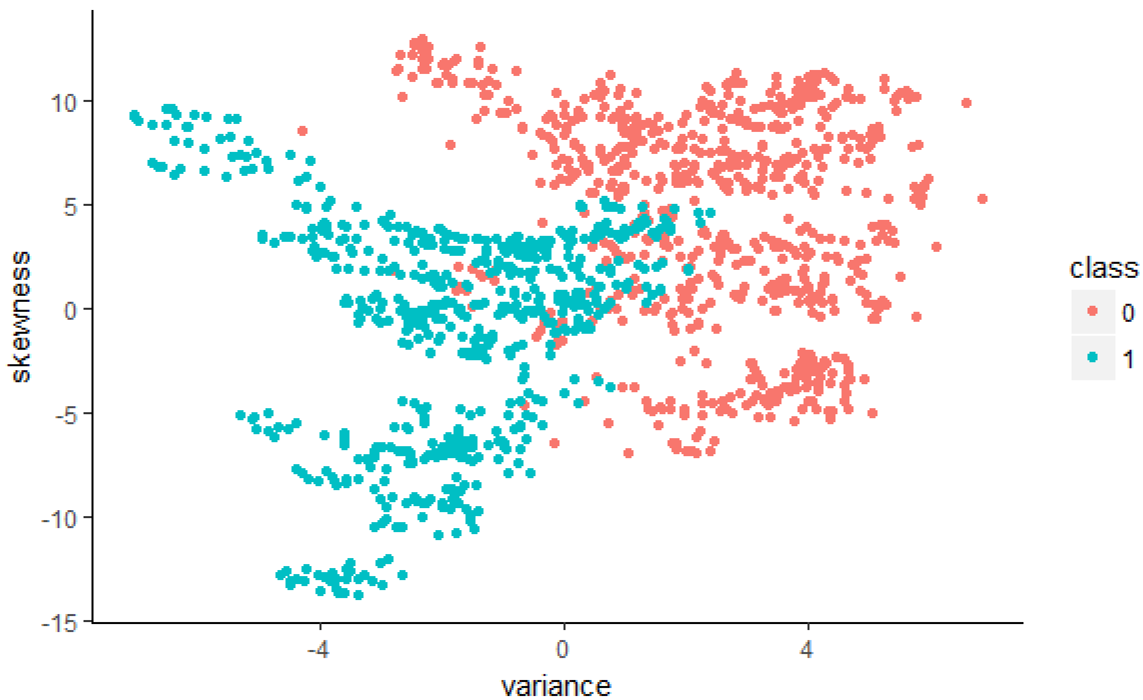
There are 762 (55.54%) forged notes and 610 (44.46%) genuine banknotes.

VISUALIZE BIVARIATE VARIABLES

Next we'll visualize bivariate variables:

```
> banknote %>%
+   ggplot(aes(x=variance, y=skewness, color=class)) +
+   geom_point()+
+   theme(panel.grid.major = element_blank(),
+         panel.grid.minor = element_blank(),
+         panel.background = element_blank(),
+         axis.line = element_line(colour = "black"))
>
```

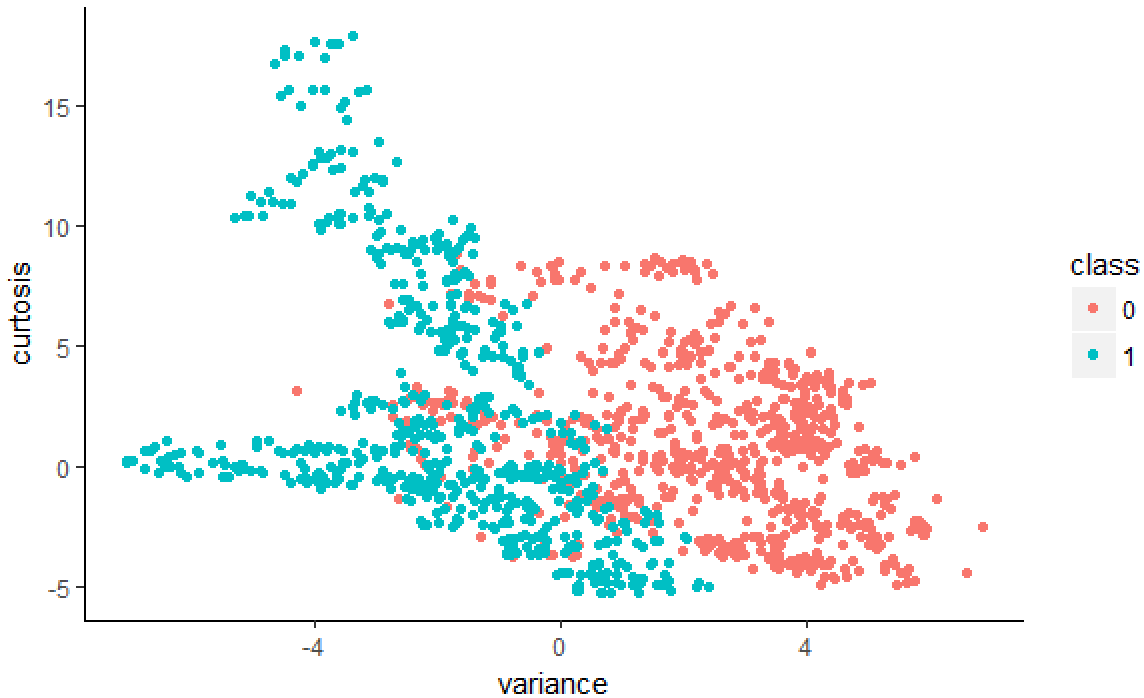
variance vs skewness



The plot of skewness against variance shows that the data is fairly classified, there are few collapse between classes.

variance vs curtosis

```
> banknote %>%
+   ggplot(aes(x=variance, y=curtosis, color=class)) +
+   geom_point()+
+   theme(panel.grid.major = element_blank(),
+         panel.grid.minor = element_blank(),
+         panel.background = element_blank(),
+         axis.line = element_line(colour = "black"))
>
```



The plot of variance against curtosis shows that the data is fairly classified, there are few collapse between classes.

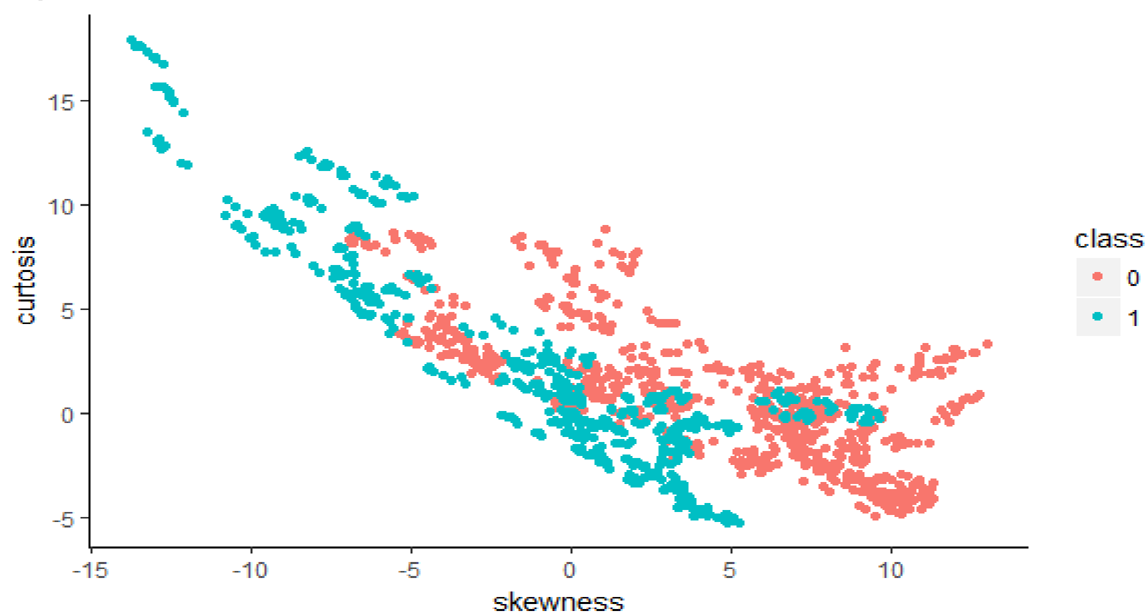
variance vs entropy

```
> banknote %>%
+   ggplot(aes(x=variance, y=entropy, color=class)) +
+   geom_point()+
+   theme(panel.grid.major = element_blank(),
+         panel.grid.minor = element_blank(),
+         panel.background = element_blank(),
+         axis.line = element_line(colour = "black"))
>
```

The plot of variance against entropy shows that the data is fairly classified, there are few collapse between classes.

skewness vs curtosis

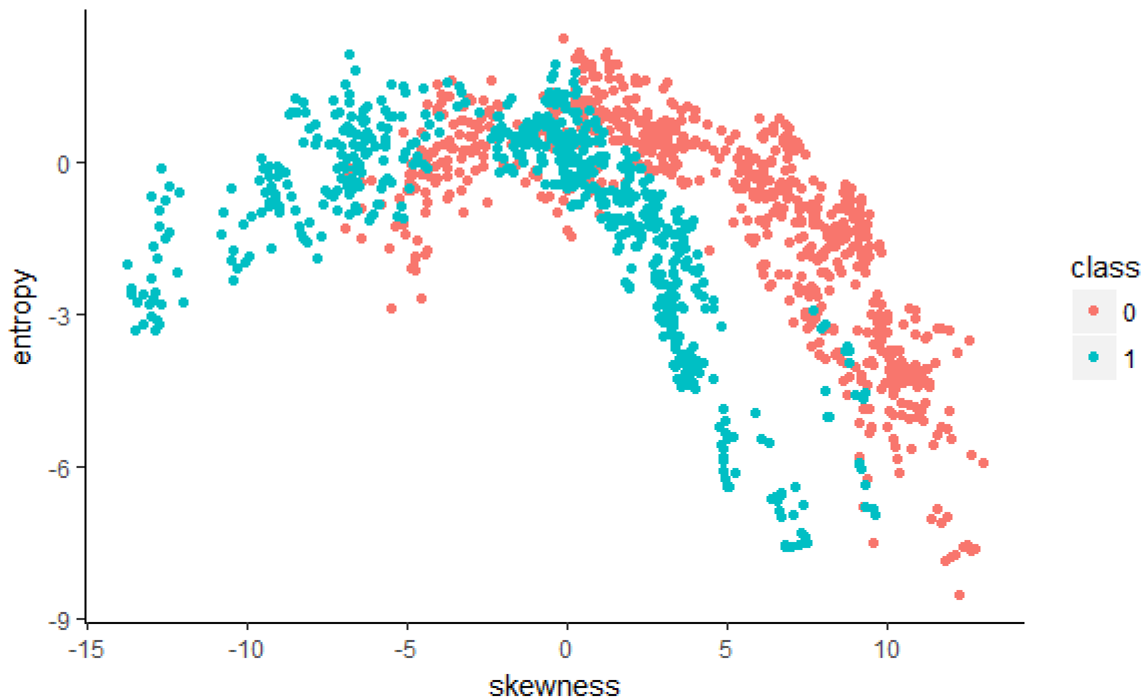
```
> banknote %>%
+   ggplot(aes(x=skewness, y=curtosis, color=class)) +
+   geom_point()+
+   theme(panel.grid.major = element_blank(),
+         panel.grid.minor = element_blank(),
+         panel.background = element_blank(),
+         axis.line = element_line(colour = "black"))
>
```



The plot of skewness against curtosis shows that the data is fairly classified, there are few collapse between classes.

curtosis vs entropy

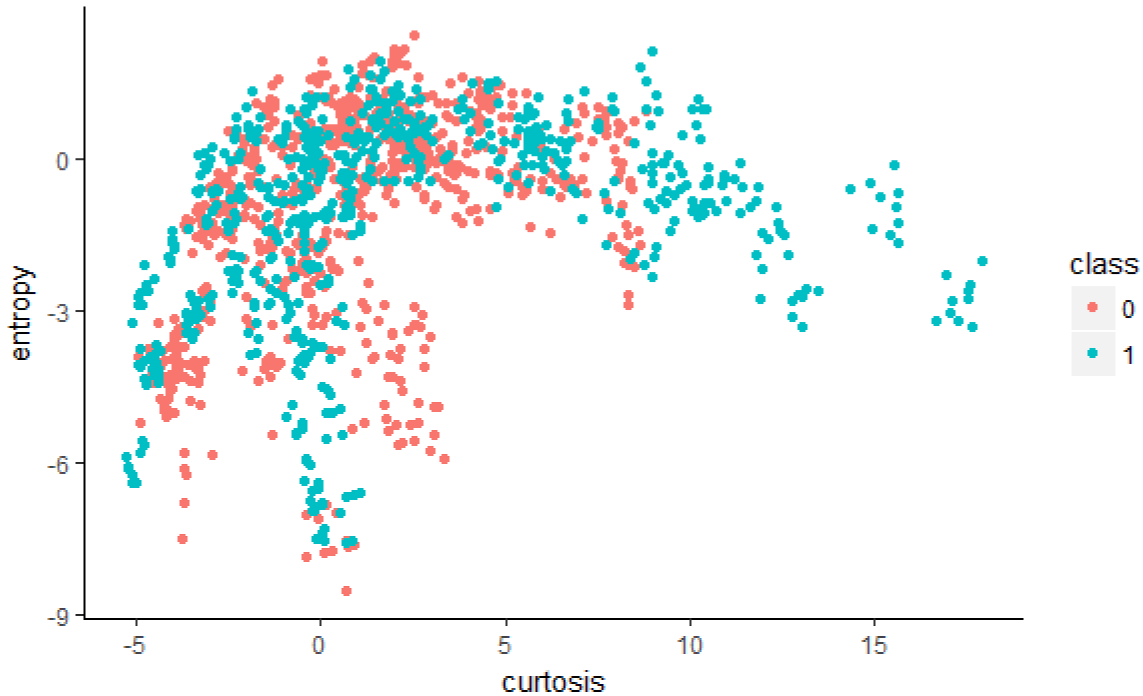
```
> banknote %>%
+   ggplot(aes(x=skewness, y=entropy, color=class)) +
+   geom_point()+
+   theme(panel.grid.major = element_blank(),
+         panel.grid.minor = element_blank(),
+         panel.background = element_blank(),
+         axis.line = element_line(colour = "black"))
>
```



The plot of curtosis against entropy shows that the data is fairly classified, there are some collapse between classes.

curtosis vs entropy

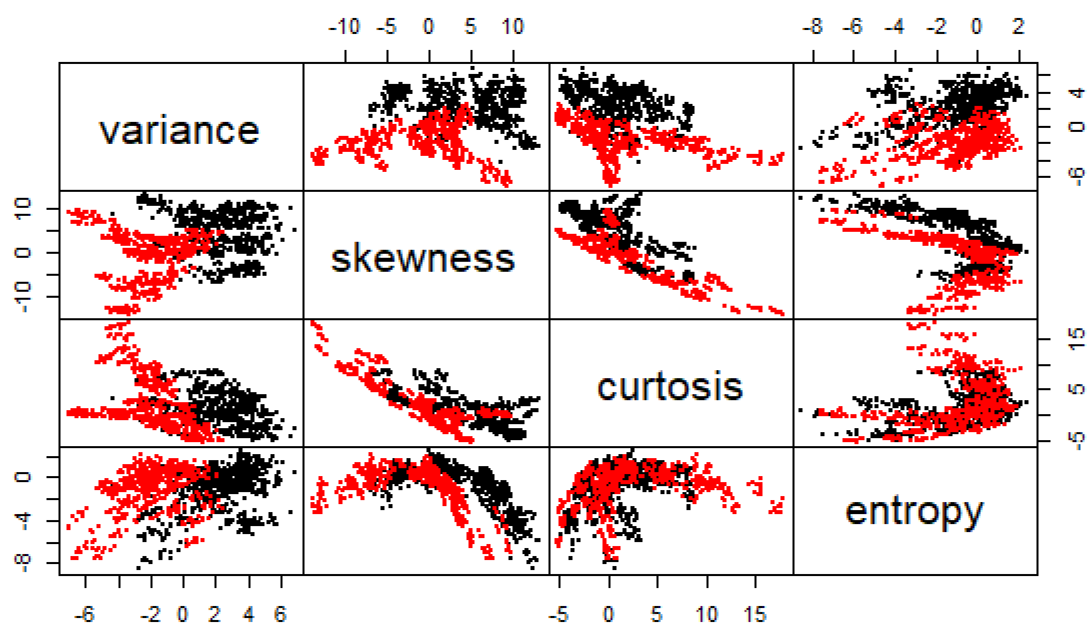
```
> banknote %>%
+   ggplot(aes(x=curtosis, y=entropy, color=class)) +
+   geom_point()+
+   theme(panel.grid.major = element_blank(),
+         panel.grid.minor = element_blank(),
+         panel.background = element_blank(),
+         axis.line = element_line(colour = "black"))
>
```



The plot of entropy against curtosis shows that the data is not well separated, there is a lot of collapse between classes.

Let's put all together using pairwise scatterplot:

```
> pairs(banknote[, -5], gap=0, pch=19, cex=0.4, col=banknote[,5])
>
```



Next we'll calculate the correlation between all numerical variables:

```
> round(cor(banknote[, -5], method = "pearson"), 2)
          variance skewness curtosis entropy
variance      1.00     0.26    -0.38    0.28
skewness      0.26     1.00    -0.79   -0.53
curtosis     -0.38    -0.79     1.00    0.32
entropy       0.28    -0.53     0.32    1.00
>
```

Curtosis has a strong negative relationship with skewness, entropy has a fairly strong negative relationship with skewness. The class with high correlation are highlighted in colors. I have considered correlation of 40% and above as high.

Normalize the banknote data:

```
> normalize <- function(x) {
+   num <- x - min(x)
+   denom <- max(x) - min(x)
+   return (num/denom)
+ }
> banknote_norm <- as.data.frame(lapply(banknote[1:4], normalize))
> banknote_norm <- cbind(banknote_norm, banknote[5])
> summary(banknote_norm)
    variance         skewness         curtosis          entropy        class
 Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   0:762
 1st Qu.:0.3800   1st Qu.:0.4515   1st Qu.:0.1599   1st Qu.:0.5578   1:610
 Median :0.5436   Median :0.6022   Median :0.2543   Median :0.7239
 Mean   :0.5391   Mean   :0.5873   Mean   :0.2879   Mean   :0.6689
 3rd Qu.:0.7113   3rd Qu.:0.7704   3rd Qu.:0.3647   3rd Qu.:0.8132
 Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
>
```

We can see that all the numerical variables are meber of [0,1].

(Formation of Training and Test Sets):

Let's load the required libraries for this task:

```
> library(caret)
> library(klaR)
Loading required package: MASS
Warning message:
package 'klaR' was built under R version 3.3.2
> library(e1071)
Warning message:
package 'e1071' was built under R version 3.3.2
>
```

Let's set seed for reproducibility of our work and then split the dataset into training and test set. The test set will later be used to evaluate the performance of the models:

```
> set.seed(3211)
> split=0.75
> trainIndex <- createDataPartition(banknote_norm$class, p=split, list=FALSE)
> data_train <- banknote_norm[ trainIndex,]
> data_test <- banknote_norm[-trainIndex,]
> dim(data_train) ; dim(data_test)
[1] 1030    5
[1] 342   5
```

We have 1030 training set with 5 variables and we have and 342 test set with 5 variables.

Decision Tree (C 4.5)

The decision tree (C 4.5) requires a RWeka library [2]:

C 4.5: Holdout method

```
> library(RWeka)
Warning message:
package 'RWeka' was built under R version 3.3.3
> model_J48_HO_1 <- J48(class~., data=data_train)
> print(model_J48_HO_1)
J48 pruned tree
------------------

variance <= 0.53077
|   skewness <= 0.797521
|   |   variance <= 0.379039: 1 (217.0)
|   |   variance > 0.379039
|   |   |   curtosis <= 0.52233
|   |   |   |   skewness <= 0.662099: 1 (152.0/1.0)
|   |   |   |   skewness > 0.662099
|   |   |   |   |   curtosis <= 0.11461: 1 (4.0)
|   |   |   |   |   curtosis > 0.11461: 0 (6.0)
|   |   |   curtosis > 0.52233
|   |   |   |   skewness <= 0.307547: 1 (13.0)
|   |   |   |   skewness > 0.307547: 0 (13.0)
|   skewness > 0.797521
|   |   variance <= 0.15886: 1 (15.0)
|   |   variance > 0.15886: 0 (66.0)
variance > 0.53077
|   curtosis <= 0.03868
|   |   skewness <= 0.784435: 1 (24.0)
|   |   skewness > 0.784435: 0 (8.0)
|   curtosis > 0.03868
|   |   variance <= 0.633494
|   |   |   curtosis <= 0.129373
|   |   |   |   skewness <= 0.727061: 1 (21.0)
|   |   |   |   skewness > 0.727061: 0 (2.0)
|   |   |   curtosis > 0.129373
|   |   |   |   entropy <= 0.78895: 0 (98.0/1.0)
|   |   |   |   entropy > 0.78895
|   |   |   |   |   curtosis <= 0.236673: 1 (9.0)
|   |   |   |   |   curtosis > 0.236673
|   |   |   |   |   |   skewness <= 0.487727: 1 (3.0/1.0)
|   |   |   |   |   |   skewness > 0.487727: 0 (14.0)
|   |   variance > 0.633494: 0 (365.0/1.0)

Number of Leaves  :     17

Size of the tree :      33
```

Let's now predict the model on the test data to see the performance:

```
> predict_J48_HO_1 <- predict(model_J48_HO_1, newdata = data_test)
> confusionMatrix(predict_J48_HO_1, data_test$class )
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 187   3
         1   3 149

               Accuracy : 0.9825
                 95% CI : (0.9622, 0.9935)
    No Information Rate : 0.5556
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9645
 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9842
            Specificity : 0.9803
         Pos Pred Value : 0.9842
         Neg Pred Value : 0.9803
             Prevalence : 0.5556
         Detection Rate : 0.5468
   Detection Prevalence : 0.5556
      Balanced Accuracy : 0.9822

       'Positive' Class : 0

> |
```

The hold out method is approximately 98.25% accurate.

Let's see how we can build the same hold out method using caret package:

Alternative method

```
> train_control_HO <- trainControl(method="none")
> model_J48_HO_2 <- train(class~., data=data_train, method="J48", trControl=train_control_
HO)
> print(model_J48_HO_2)
C4.5-like Trees

1030 samples
   4 predictor
   2 classes: '0', '1'

No pre-processing
Resampling: None
> |
```

As we can see, the resampling used here is none.

Let's predict the model on the test data:

```
> predict_J48_HO_2 <- predict(model_J48_HO_2, newdata = data_test)
> confusionMatrix(predict_J48_HO_2, data_test$class )
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 187   3
         1   3 149

               Accuracy : 0.9825
                 95% CI : (0.9622, 0.9935)
    No Information Rate : 0.5556
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9645
 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9842
            Specificity : 0.9803
         Pos Pred Value : 0.9842
         Neg Pred Value : 0.9803
             Prevalence : 0.5556
         Detection Rate : 0.5468
   Detection Prevalence : 0.5556
      Balanced Accuracy : 0.9822

       'Positive' Class : 0

>
```

The result is exactly the same as the one obtained from the first holdout method.

C 4.5: 10 fold cross validation method

```
> train_control_CV <- trainControl(method="cv", number=10)
> model_J48_CV <- train(class~., data=data_train, trControl=train_control_CV, method="J48"
)
> print(model_J48_CV)
C4.5-like Trees

1030 samples
   4 predictor
   2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 926, 927, 927, 926, 927, 927, ...
Resampling results across tuning parameters:

  C      M  Accuracy   Kappa
  0.010  1  0.9815626  0.9626498
  0.010  2  0.9815626  0.9626498
  0.010  3  0.9805728  0.9605487
  0.255  1  0.9805917  0.9606816
  0.255  2  0.9815626  0.9626498
  0.255  3  0.9825146  0.9645438
  0.500  1  0.9805917  0.9606816
  0.500  2  0.9815626  0.9626498
  0.500  3  0.9825146  0.9645438

Accuracy was used to select the optimal model using the largest value.
The final values used for the model were C = 0.255 and M = 3.
>
```

This model select the optimal model using the highest accuracy. The model accuracy used is 98.25%.

C 4.5: Leave-one-out cross-validation method

```
> train_control_LOOCV <- trainControl(method="LOOCV")
> model_J48_LOOCV <- train(class~., data=data_train, trControl=train_control_LOOCV, method
="J48")
> print(model_J48_LOOCV)
C4.5-like Trees

1030 samples
   4 predictor
   2 classes: '0', '1'

No pre-processing
Resampling: Leave-One-Out Cross-Validation
Summary of sample sizes: 1029, 1029, 1029, 1029, 1029, 1029, ...
Resampling results across tuning parameters:

  C      M  Accuracy   Kappa
  0.010  1  0.9805825  0.9607005
  0.010  2  0.9805825  0.9607005
  0.010  3  0.9834951  0.9666027
  0.255  1  0.9796117  0.9587266
  0.255  2  0.9805825  0.9607005
  0.255  3  0.9834951  0.9666027
  0.500  1  0.9825243  0.9646458
  0.500  2  0.9834951  0.9666172
  0.500  3  0.9834951  0.9666027

Accuracy was used to select the optimal model using the largest value.
The final values used for the model were C = 0.01 and M = 3.
> 
```

This model select the optimal model using the highest accuracy. The model accuracy used is 98.35%.

Decision Tree (Random Forest)

Random Forest: Holdout method

The decision tree (randomForest) requires a randomForest library:

```
> library(randomForest)
> model_RF_HO_1 <- randomForest(class~., data=data_train)
> print(model_RF_HO_1)

Call:
 randomForest(formula = class ~ ., data = data_train)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 2

        OOB estimate of  error rate: 0.87%
Confusion matrix:
    0   1 class.error
0 566   6 0.010489510
1   3 455 0.006550218
> 
```

Let’s now predict the model on the test data to see the performance:

```
> predict_RF_HO_1 <- predict(model_RF_HO_1, newdata = data_test)
> confusionMatrix(predict_RF_HO_1, data_test$class )
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 187   0
         1   3 152

               Accuracy : 0.9912
                 95% CI : (0.9746, 0.9982)
    No Information Rate : 0.5556
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9823
 Mcnemar's Test P-Value : 0.2482

            Sensitivity : 0.9842
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9806
             Prevalence : 0.5556
         Detection Rate : 0.5468
   Detection Prevalence : 0.5468
      Balanced Accuracy : 0.9921

       'Positive' Class : 0

> |
```

The hold out method is approximately 99.12% accurate.

Let’s see how we can build the same hold out method using caret package:

Alternative method

```
> train_control_HO <- trainControl(method="none")
> model_RF_HO_2 <- train(class~., data=data_train, method="rf", trControl=train_control_HO
)
> print(model_RF_HO_2)
Random Forest

1030 samples
   4 predictor
   2 classes: '0', '1'

No pre-processing
Resampling: None
>
```

As we can see, the resampling used here is none.

```
> predict_RF_HO_2 <- predict(model_RF_HO_2, newdata = data_test)
> confusionMatrix(predict_RF_HO_2, data_test$class )
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 187   0
         1   3 152

               Accuracy : 0.9912
                 95% CI : (0.9746, 0.9982)
    No Information Rate : 0.5556
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9823
 Mcnemar's Test P-Value : 0.2482

            Sensitivity : 0.9842
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9806
             Prevalence : 0.5556
         Detection Rate : 0.5468
   Detection Prevalence : 0.5468
      Balanced Accuracy : 0.9921

       'Positive' Class : 0

> 
```

The result is exactly the same as the one obtained from the first holdout method.

Random Forest: 10 fold cross validation method

```
> train_control_CV <- trainControl(method="cv", number=10)
> model_RF_CV <- train(class ~ ., data=data_train, trControl=train_control_CV, method="rf"
)
> print(model_RF_CV)
Random Forest

1030 samples
   4 predictor
   2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 926, 927, 926, 927, 927, 928, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
  2     0.9912429  0.9822855
  3     0.9912429  0.9822855
  4     0.9873781  0.9744426

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 2.
> 
```

This model select the optimal model using the highest accuracy. The model accuracy used is 99.12%.

Random Forest: Leave-one-out cross-validation

```
> train_control_LOOCV <- trainControl(method="LOOCV")
> model_RF_LOOCV <- train(class~., data=data_train, trControl=train_control_LOOCV, method=
"rf")
> print(model_RF_LOOCV)
Random Forest

1030 samples
   4 predictor
   2 classes: '0', '1'

No pre-processing
Resampling: Leave-One-Out Cross-Validation
Summary of sample sizes: 1029, 1029, 1029, 1029, 1029, 1029, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
  2     0.9922330  0.9842802
  3     0.9932039  0.9862422
  4     0.9902913  0.9803332

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 3.
>
```

This model select the optimal model using the highest accuracy. The model accuracy used is 99.32%.

The Naïve Bayes

Naïve Bayes: Holdout method

```
> model_NB_HO_1 <- NaiveBayes(class~., data=data_train)
> print(model_NB_HO_1)
$apriori
grouping
        0         1
0.5553398 0.4446602

$tables
$tables$variance
       [,1]      [,2]
0 0.6752704 0.1436518
1 0.3727814 0.1374920

$tables$skewness
       [,1]      [,2]
0 0.6803027 0.1942577
1 0.4819830 0.2043575

$tables$curtosis
       [,1]      [,2]
0 0.2578294 0.1383986
1 0.3194533 0.2237655

$tables$entropy
       [,1]      [,2]
0 0.6666103 0.1940068
1 0.6574805 0.1978024


$levels
[1] "0" "1"

$call
NaiveBayes.default(x = X, grouping = Y)

$x
         variance    skewness    curtosis     entropy
1     0.7690038870 0.839642728 0.106782691 0.73662766
```

Let’s now predict the model on the test data to see the performance:

```
> predict_NB_HO_1 <- predict(model_NB_HO_1, newdata = data_test)
> confusionMatrix(predict_NB_HO_1$class, data_test$class )
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 160  30
         1  30 122

               Accuracy : 0.8246
                 95% CI : (0.78, 0.8634)
    No Information Rate : 0.5556
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.6447
 Mcnemar's Test P-Value : 1

            Sensitivity : 0.8421
            Specificity : 0.8026
         Pos Pred Value : 0.8421
         Neg Pred Value : 0.8026
             Prevalence : 0.5556
         Detection Rate : 0.4678
   Detection Prevalence : 0.5556
      Balanced Accuracy : 0.8224

       'Positive' Class : 0

> |
```

The hold out method is approximately 82.46% accurate.

Naïve Bayes: 10 fold cross validation method

```
> train_control_CV <- trainControl(method="cv", number=10)
> model_NB_CV <- train(class~., data=data_train, trControl=train_control_CV, method="nb")
> print(model_NB_CV)
Naive Bayes

1030 samples
   4 predictor
   2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 927, 927, 926, 927, 927, 927, ...
Resampling results across tuning parameters:

  usekernel  Accuracy   Kappa
  FALSE      0.8359910  0.6661872
   TRUE      0.9271815  0.8518122

Tuning parameter 'fL' was held constant at a value of 0
Tuning parameter 'adjust' was held
 constant at a value of 1
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were fL = 0, usekernel = TRUE and adjust = 1.
> |
```

This model select the optimal model using the highest accuracy. The model accuracy used is 92.72%.

Naïve Bayes: Leave-one-out cross-validation

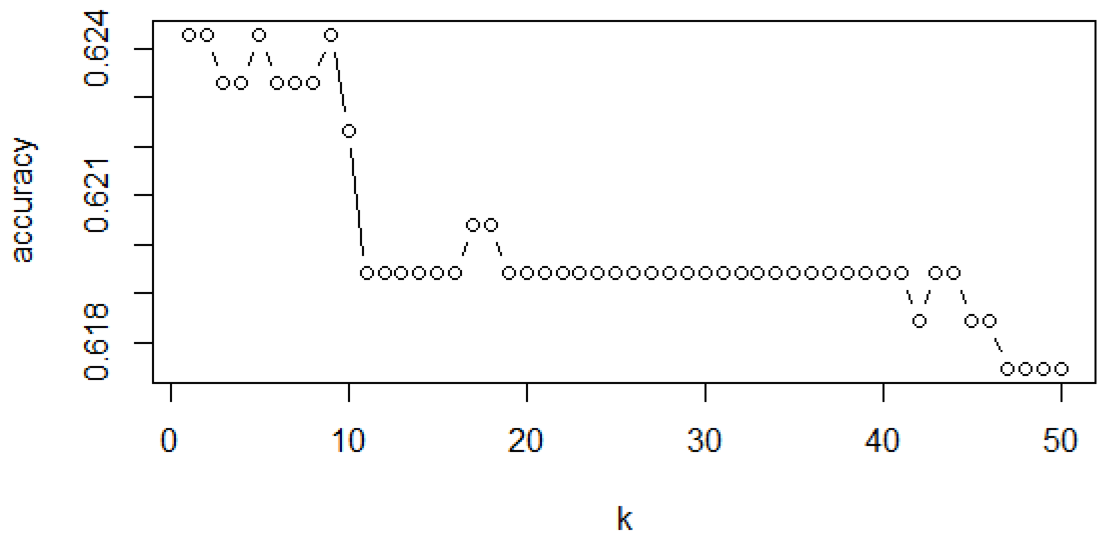
```
> train_control_LOOCV <- trainControl(method="LOOCV")
> model_NB_LOOCV <- train(class~., data=data_train, trControl=train_control_LOOCV, method=
"nb")
> print(model_NB_LOOCV)
Naive Bayes

1030 samples
   4 predictor
   2 classes: '0', '1'

No pre-processing
Resampling: Leave-One-Out Cross-Validation
Summary of sample sizes: 1029, 1029, 1029, 1029, 1029, 1029, ...
Resampling results across tuning parameters:

  usekernel  Accuracy   Kappa
  FALSE      0.8398058  0.6735786
   TRUE      0.9223301  0.8419082

Tuning parameter 'fL' was held constant at a value of 0
Tuning parameter 'adjust' was held
 constant at a value of 1
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were fL = 0, usekernel = TRUE and adjust = 1.
>
```

This model select the optimal model using the highest accuracy. The model accuracy used is 92.23%.

The knn

Let's plot classification accuracy as a function of k (k = 1, ... , 50) to determine heuristically the possible 'best' number of k.

```
> library(class)
> accuracy <- rep(0, 50)
> k <- 1:50
> for(x in k){
+   prediction <- knn(train = data_train[,1:4], test = data_test[,1:4], cl = data_train$cl
ass, k = x)
+   accuracy[x] <- mean(prediction == data_train$class)
+
+ }
> plot(k, accuracy, type = 'b')
>
```



From the plot, we can see that k = 1, 2, 5, and 9 will gives us the best accuracy for class prediction. Since it is better to use an odd k for knn in order to avoid voting problem, hence, we'll consider k = 5 for the holdout method.

Knn: Holdout method

```
> model_KNN_HO_1 <- knn(train = data_train[,1:4], test = data_test[,1:4], cl = data_train$
class, k = 5)
> confusionMatrix(model_KNN_HO_1, data_test$class )
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 189   0
         1   1 152

               Accuracy : 0.9971
                 95% CI : (0.9838, 0.9999)
    No Information Rate : 0.5556
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9941
 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9947
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9935
             Prevalence : 0.5556
         Detection Rate : 0.5526
   Detection Prevalence : 0.5526
      Balanced Accuracy : 0.9974

       'Positive' Class : 0
```

The hold out method is approximately 99.71% accurate.

Knn: 10 fold cross validation method

```
> train_control_CV <- trainControl(method="cv", number=10)
> model_KNN_CV <- train(data_train[,1:4], data_train$class, method = "knn", trControl = tr
ain_control_CV)
> print(model_KNN_CV)
k-Nearest Neighbors

1030 samples
   4 predictor
   2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 927, 927, 926, 927, 927, 928, ...
Resampling results across tuning parameters:

  k  Accuracy   Kappa
  5  0.9990385  0.9980553
  7  0.9970967  0.9941435
  9  0.9941651  0.9882161

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 5.
>
```

This model select the optimal model using the highest accuracy. The model accuracy used is 99.90%.

Knn: Leave-one-out cross-validation

```
> train_control_LOOCV <- trainControl(method="LOOCV")
> model_KNN_LOOCV <- train(data_train[,1:4], data_train$class, method = "knn", trControl =
 train_control_LOOCV)
> print(model_KNN_LOOCV)
k-Nearest Neighbors

1030 samples
   4 predictor
   2 classes: '0', '1'

No pre-processing
Resampling: Leave-One-Out Cross-Validation
Summary of sample sizes: 1029, 1029, 1029, 1029, 1029, 1029, ...
Resampling results across tuning parameters:

  k  Accuracy   Kappa
  5  0.9990291  0.9980346
  7  0.9990291  0.9980346
  9  0.9941748  0.9882204

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 7.
>
```

This model select the optimal model using the highest accuracy. The model accuracy used is 99.9%.

Decision Tree (C 4.5) [predict on test set]

C 4.5: Hold out method

```
> predict_J48_HO_1 <- predict(model_J48_HO_1, newdata = data_test)
> confusionMatrix(predict_J48_HO_1, data_test$class)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 187   3
         1   3 149

               Accuracy : 0.9825
                 95% CI : (0.9622, 0.9935)
    No Information Rate : 0.5556
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9645
 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9842
            Specificity : 0.9803
         Pos Pred Value : 0.9842
         Neg Pred Value : 0.9803
             Prevalence : 0.5556
         Detection Rate : 0.5468
   Detection Prevalence : 0.5556
      Balanced Accuracy : 0.9822

       'Positive' Class : 0

> |
```

The hold out predict 98.25% of the new test data accurately.

C 4.5: 10 fold cross validation method

```
> predict_J48_CV <- predict(model_J48_CV, newdata = data_test)
> confusionMatrix(predict_J48_CV, data_test$class)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 187   3
         1   3 149

               Accuracy : 0.9825
                 95% CI : (0.9622, 0.9935)
    No Information Rate : 0.5556
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9645
 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9842
            Specificity : 0.9803
         Pos Pred Value : 0.9842
         Neg Pred Value : 0.9803
             Prevalence : 0.5556
         Detection Rate : 0.5468
   Detection Prevalence : 0.5556
      Balanced Accuracy : 0.9822

       'Positive' Class : 0

> |
```

The 10 fold cross validation method predicts 98.25% of the new test data accurately.

C 4.5: Leave One Out Cross Validation

```
> predict_J48_LOOCV <- predict(model_J48_LOOCV, newdata = data_test)
> confusionMatrix(predict_J48_LOOCV, data_test$class)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 187   3
         1   3 149

               Accuracy : 0.9825
                 95% CI : (0.9622, 0.9935)
    No Information Rate : 0.5556
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9645
 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9842
            Specificity : 0.9803
         Pos Pred Value : 0.9842
         Neg Pred Value : 0.9803
             Prevalence : 0.5556
         Detection Rate : 0.5468
   Detection Prevalence : 0.5556
      Balanced Accuracy : 0.9822

       'Positive' Class : 0

> 
```

The Leave One Out Cross Validation predict 98.25% of the new test data accurately.

Decision Tree (Random Forest) [predict on test set]

Random Forest: Hold out method

```
> predict_RF_HO_1 <- predict(model_RF_HO_1, newdata = data_test)
> confusionMatrix(predict_RF_HO_1, data_test$class)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 187   0
         1   3 152

               Accuracy : 0.9912
                 95% CI : (0.9746, 0.9982)
    No Information Rate : 0.5556
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9823
 Mcnemar's Test P-Value : 0.2482

            Sensitivity : 0.9842
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9806
             Prevalence : 0.5556
         Detection Rate : 0.5468
   Detection Prevalence : 0.5468
      Balanced Accuracy : 0.9921

       'Positive' Class : 0

> 
```

The hold out predict 99.12% of the new test data accurately.

Random Forest: 10 fold cross validation method

```
> predict_RF_CV <- predict(model_RF_CV, newdata = data_test)
> confusionMatrix(predict_RF_CV, data_test$class)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 187   0
         1   3 152

               Accuracy : 0.9912
                 95% CI : (0.9746, 0.9982)
    No Information Rate : 0.5556
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9823
 Mcnemar's Test P-Value : 0.2482

            Sensitivity : 0.9842
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9806
             Prevalence : 0.5556
         Detection Rate : 0.5468
   Detection Prevalence : 0.5468
      Balanced Accuracy : 0.9921

       'Positive' Class : 0

>
```

The 10 fold cross validation method predict 99.12% of the new test data accurately.

Random Forest: Leave One Out Cross Validation

```
> predict_RF_LOOCV <- predict(model_RF_LOOCV, newdata = data_test)
> confusionMatrix(predict_RF_LOOCV, data_test$class)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 186   1
         1   4 151

               Accuracy : 0.9854
                 95% CI : (0.9662, 0.9952)
    No Information Rate : 0.5556
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9705
 Mcnemar's Test P-Value : 0.3711

            Sensitivity : 0.9789
            Specificity : 0.9934
         Pos Pred Value : 0.9947
         Neg Pred Value : 0.9742
             Prevalence : 0.5556
         Detection Rate : 0.5439
   Detection Prevalence : 0.5468
      Balanced Accuracy : 0.9862

       'Positive' Class : 0

>
```

The Leave One Out Cross Validation predict 98.54% of the new test data accurately.

NAÏVE BAYES [predict on test set]

NAÏVE BAYES: Hold out method

```
> predict_NB_HO_1 <- predict(model_NB_HO_1, newdata = data_test)
> confusionMatrix(predict_NB_HO_1$class, data_test$class )
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 160  30
         1  30 122

               Accuracy : 0.8246
                 95% CI : (0.78, 0.8634)
    No Information Rate : 0.5556
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.6447
 Mcnemar's Test P-Value : 1

            Sensitivity : 0.8421
            Specificity : 0.8026
         Pos Pred Value : 0.8421
         Neg Pred Value : 0.8026
             Prevalence : 0.5556
         Detection Rate : 0.4678
   Detection Prevalence : 0.5556
      Balanced Accuracy : 0.8224

       'Positive' Class : 0

> 
```

The hold out predict 82.46% of the new test data accurately.

NAÏVE BAYES: 10 fold cross validation method

```
> predict_NB_CV <- predict(model_NB_CV, newdata = data_test)
> confusionMatrix(predict_NB_CV, data_test$class)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 171  16
         1  19 136

               Accuracy : 0.8977
                 95% CI : (0.8606, 0.9277)
    No Information Rate : 0.5556
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.7932
 Mcnemar's Test P-Value : 0.7353

            Sensitivity : 0.9000
            Specificity : 0.8947
         Pos Pred Value : 0.9144
         Neg Pred Value : 0.8774
             Prevalence : 0.5556
         Detection Rate : 0.5000
   Detection Prevalence : 0.5468
      Balanced Accuracy : 0.8974

       'Positive' Class : 0

> 
```

The 10 fold cross validation method predict 89.77% of the new test data accurately.

NAÏVE BAYES: Leave One Out Cross Validation

```
> predict_NB_LOOCV <- predict(model_NB_LOOCV, newdata = data_test)
> confusionMatrix(predict_NB_LOOCV, data_test$class)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 171  16
         1  19 136

               Accuracy : 0.8977
                 95% CI : (0.8606, 0.9277)
    No Information Rate : 0.5556
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.7932
 Mcnemar's Test P-Value : 0.7353

            Sensitivity : 0.9000
            Specificity : 0.8947
         Pos Pred Value : 0.9144
         Neg Pred Value : 0.8774
             Prevalence : 0.5556
         Detection Rate : 0.5000
   Detection Prevalence : 0.5468
      Balanced Accuracy : 0.8974

       'Positive' Class : 0

> |
```

The : Leave One Out Cross Validation predict 89.77% of the new test data accurately.

KNN [predict on test set]

KNN: Hold out method

```
> confusionMatrix(model_KNN_HO_1, data_test$class )
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 189   0
         1   1 152

               Accuracy : 0.9971
                 95% CI : (0.9838, 0.9999)
    No Information Rate : 0.5556
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9941
 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9947
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9935
             Prevalence : 0.5556
         Detection Rate : 0.5526
   Detection Prevalence : 0.5526
      Balanced Accuracy : 0.9974

       'Positive' Class : 0

> |
```

The hold out predict 99.71% of the new test data accurately.

KNN: 10 fold cross validation method

```
> predict_KNN_CV <- predict(model_KNN_CV, newdata = data_test)
> confusionMatrix(predict_KNN_CV, data_test$class)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 189   0
         1   1 152

               Accuracy : 0.9971
                 95% CI : (0.9838, 0.9999)
    No Information Rate : 0.5556
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9941
 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9947
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9935
             Prevalence : 0.5556
         Detection Rate : 0.5526
   Detection Prevalence : 0.5526
      Balanced Accuracy : 0.9974

       'Positive' Class : 0

> 
```

The 10 fold cross validation method predict 99.71% of the new test data accurately.

KNN: Leave One Out Cross Validation

```
> predict_KNN_LOOCV <- predict(model_KNN_LOOCV, newdata = data_test)
> confusionMatrix(predict_KNN_LOOCV, data_test$class)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 188   0
         1   2 152

               Accuracy : 0.9942
                 95% CI : (0.979, 0.9993)
    No Information Rate : 0.5556
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9882
 Mcnemar's Test P-Value : 0.4795

            Sensitivity : 0.9895
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9870
             Prevalence : 0.5556
         Detection Rate : 0.5497
   Detection Prevalence : 0.5497
      Balanced Accuracy : 0.9947

       'Positive' Class : 0

> 
```

The Leave One Out Cross Validation predict 99.42% of the new test data accurately.

Summary

	Decision Tree (C 4.5)			Random Forest			Naïve Bayes			knn		
	HO	CV	LOOCV	HO	CV	LOOCV	HO	CV	LOOCV	HO	CV	LOOCV
Resampling Accuracy		98.25%	98.35%		99.12%	99.32%		92.72%	92.23%		99.90%	99.90%
Accuracy on test set	98.25%	98.25%	98.25%	99.12%	99.12%	98.54%	82.46%	89.77%	89.77%	99.71%	99.71%	99.42%

The table shows the resampling accuracy of 2nd Task and estimated accuracy when predicted on test set of (3rd Task to 5th Task). KNN is on top of the list, followed by Random forest, then C 4.5 and lastly Naïve Bayes. There is no resampling accuracy for the Holdout methods since we evaluates the built model on test set. The CV and LOOCV predicts well with KNN, Random Forest and C 4.5. Naïve Bayes' accuracy is though lower than others during resampling, the accuracy when predicted on test set suggests a little bit of overfitting during cross validation. Also, the CV seems to outperform the LOOCV in all except for Naïve Bayes where there was a tie. The possibility of luck was exposed on holdout method using Naïve Bayes, where holdout method performed less than both CV and LOOCV whereas, we were very lucky with the prediction with HO on all other methods.

C 4.5: Hold out method [Measure Performance] [3]

Confusion matrix

```
> library(ROCR)
> confusionMatrix(predict_J48_HO_1, data_test$class)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 187   3
         1   3 149

               Accuracy : 0.9825
                 95% CI : (0.9622, 0.9935)
    No Information Rate : 0.5556
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9645
 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9842
            Specificity : 0.9803
         Pos Pred Value : 0.9842
         Neg Pred Value : 0.9803
             Prevalence : 0.5556
         Detection Rate : 0.5468
   Detection Prevalence : 0.5556
      Balanced Accuracy : 0.9822

       'Positive' Class : 0

> 
```

The hold out predicts 187 corretly as 0, misclassified 3 (0s) as 1. It classified 149 correctly as 1 but misclassified 3 1s as 0s. The accuracy is 98.25% on the new test data accurately.

Precision vs. Recall estimation

```
> pred_J48_HO <- prediction(as.numeric(predict_J48_HO_1), as.numeric(data_test$class))
> perf_J48_HO_preRecall <- performance(pred_J48_HO, measure="prec", x.measure="rec")
> perf_J48_HO_preRecall@x.values[[1]] # Recall values
[1] 0.0000000 0.9802632 1.0000000
> perf_J48_HO_preRecall@y.values[[1]] # Precision values
[1]       NaN 0.9802632 0.4444444
> paste0(round(100*perf_J48_HO_preRecall@x.values[[1]][2], 2), "%") # Recall values
[1] "98.03%"
> paste0(round(100*perf_J48_HO_preRecall@y.values[[1]][2], 2), "%") # Precision values
[1] "98.03%"
>
```

We are interested in the middle result, The precision is 98.03% while recall is 98.03%.

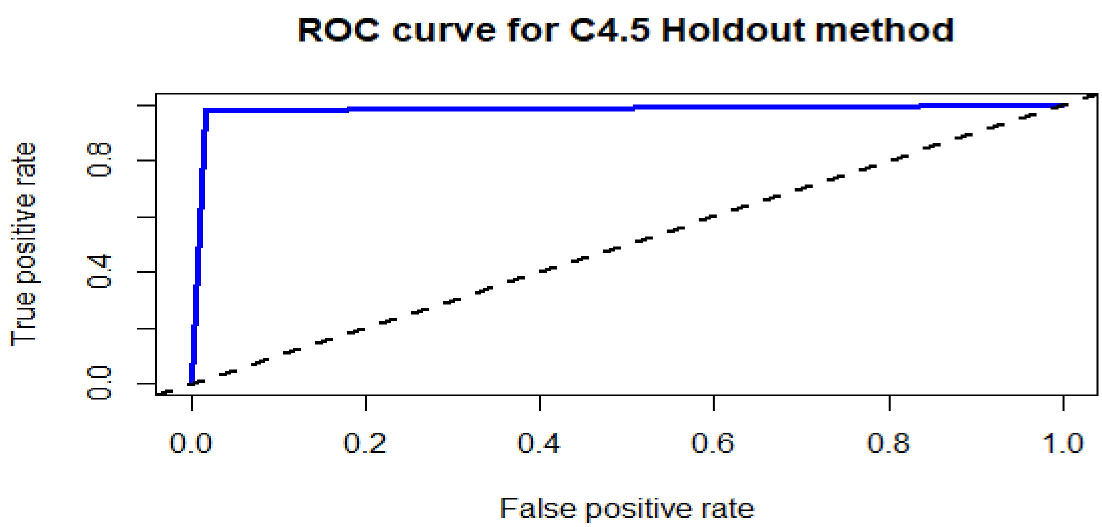
Accuracy estimation

```
> perf_J48_HO_acc <- performance(pred_J48_HO, "acc")
> perf_J48_HO_acc@y.values[[1]] #Accuracy
[1] 0.5555556 0.9824561 0.4444444
> paste0(round(100*perf_J48_HO_acc@y.values[[1]][2], 2), "%") #Accuracy
[1] "98.25%"
>
```

The accuracy is 98.25%.

ROC(receiver operating characteristic curve)

```
> perf_J48_HO_ROC <- performance(pred_J48_HO,  measure = "tpr", x.measure = "fpr")
> plot(perf_J48_HO_ROC, main = "ROC curve for C4.5 Holdout method", col = "blue", lwd = 3)
> abline(a = 0, b = 1, lwd = 2, lty = 2)
>
```



RAUC (receiver under the curve area)

```
> perf_J48_HO_AUC <- performance(pred_J48_HO, measure = "auc")
> perf_J48_HO_AUC@y.values[[1]] #AUC
[1] 0.9822368
> paste0(round(100*perf_J48_HO_AUC@y.values[[1]], 2), "%") #AUC
[1] "98.22%"
>
```

The area under the curve is 98.22% which imply that we have an excellent model.

C 4.5: 10 fold cross validation method [Measure Performance]

```
> confusionMatrix(predict_J48_CV, data_test$class)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 187   3
         1   3 149

               Accuracy : 0.9825
                 95% CI : (0.9622, 0.9935)
    No Information Rate : 0.5556
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9645
 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9842
            Specificity : 0.9803
         Pos Pred Value : 0.9842
         Neg Pred Value : 0.9803
             Prevalence : 0.5556
         Detection Rate : 0.5468
   Detection Prevalence : 0.5556
      Balanced Accuracy : 0.9822

       'Positive' Class : 0

> |
```

The cross validation method predicts 187 corretly as 0, misclassified 3 (0s) as 1. It classified 149 correctly as 1 but misclassified 3 1s as 0s. The accuracy is 98.25% on the new test data accurately.

Precision vs. Recall estimation

```
> pred_J48_CV <- prediction(as.numeric(predict_J48_CV), as.numeric(data_test$class))
> perf_J48_CV_preRecall <- performance(pred_J48_CV, measure="prec", x.measure="rec")
> perf_J48_CV_preRecall@x.values[[1]] # Recall values
[1] 0.0000000 0.9802632 1.0000000
> perf_J48_CV_preRecall@y.values[[1]] # Precision values
[1]       NaN 0.9802632 0.4444444
> paste0(round(100*perf_J48_CV_preRecall@x.values[[1]][2], 2), "%") # Recall values
[1] "98.03%"
> paste0(round(100*perf_J48_CV_preRecall@y.values[[1]][2], 2), "%") # Precision values
[1] "98.03%"
> |
```

We are interested in the middle result, The precision is 98.03% while recall is 98.03%.

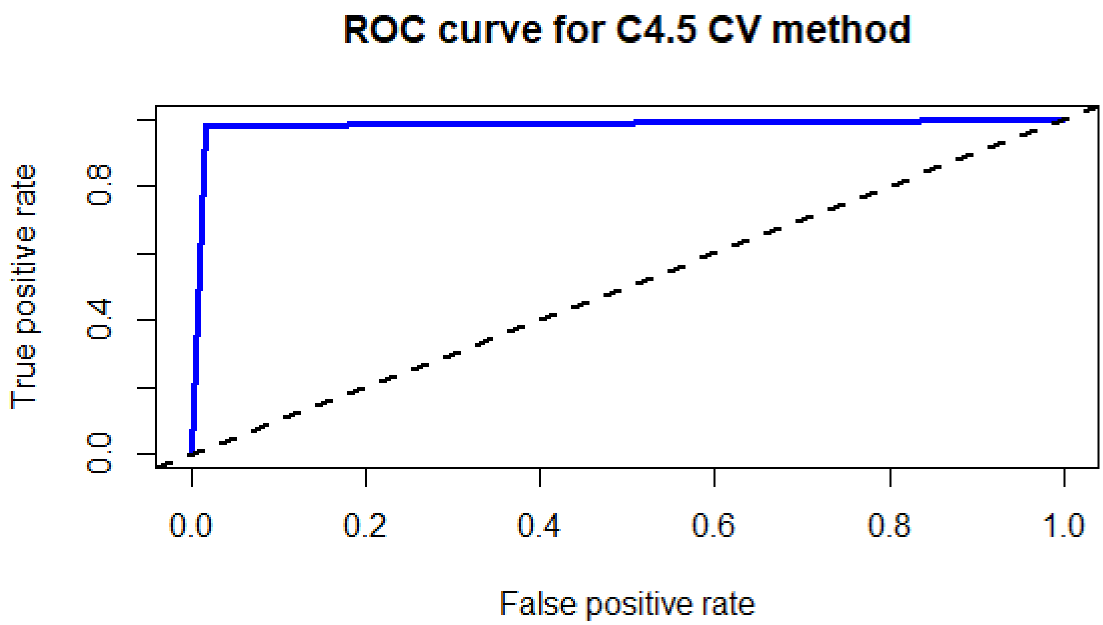
Accuracy estimation

```
> perf_J48_CV_acc <- performance(pred_J48_CV, "acc")
> perf_J48_CV_acc@y.values[[1]] #Accuracy
[1] 0.5555556 0.9824561 0.4444444
> paste0(round(100*perf_J48_CV_acc@y.values[[1]][2], 2), "%") #Accuracy
[1] "98.25%"
> |
```

The accuracy is 98.25%.

ROC(receiver operating characteristic curve)

```
> perf_J48_CV_ROC <- performance(pred_J48_CV,  measure = "tpr", x.measure = "fpr")
> plot(perf_J48_CV_ROC, main = "ROC curve for C4.5 CV method", col = "blue", lwd = 3)
> abline(a = 0, b = 1, lwd = 2, lty = 2)
> |
```




```
> perf_J48_CV_AUC <- performance(pred_J48_CV, measure = "auc")
> perf_J48_CV_AUC@y.values[[1]] #AUC
[1] 0.9822368
> paste0(round(100*perf_J48_CV_AUC@y.values[[1]], 2), "%") #AUC
[1] "98.22%"
> 
```

The area under the curve is 98.22% which imply that we have an excellent model.

C 4.5: Leave One Out Cross Validation [Measure Performance]

```
> confusionMatrix(predict_J48_LOOCV, data_test$class)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 187   3
         1   3 149
                                          
               Accuracy : 0.9825          
                 95% CI : (0.9622, 0.9935)
    No Information Rate : 0.5556          
    P-Value [Acc > NIR] : <2e-16          
                                          
                  Kappa : 0.9645          
 Mcnemar's Test P-Value : 1               
                                          
            Sensitivity : 0.9842          
            Specificity : 0.9803          
         Pos Pred Value : 0.9842          
         Neg Pred Value : 0.9803          
             Prevalence : 0.5556          
         Detection Rate : 0.5468          
   Detection Prevalence : 0.5556          
      Balanced Accuracy : 0.9822          
                                          
       'Positive' Class : 0               

> 
```

The leave one out method predicts 187 corretly as 0, misclassified 3 (0s) as 1. It classified 149 correctly as 1 but misclassified 3 1s as 0s. The accuracy is 98.25% on the new test data accurately.

Precision vs. Recall estimation

```
> pred_J48_LOOCV <- prediction(as.numeric(predict_J48_LOOCV), as.numeric(data_test$class))
> perf_J48_LOOCV_preRecall <- performance(pred_J48_LOOCV, measure="prec", x.measure="rec")
> perf_J48_LOOCV_preRecall@x.values[[1]] # Recall values
[1] 0.0000000 0.9802632 1.0000000
> perf_J48_LOOCV_preRecall@y.values[[1]] # Precision values
[1]       NaN 0.9802632 0.4444444
> paste0(round(100*perf_J48_LOOCV_preRecall@x.values[[1]][2], 2), "%") # Recall values
[1] "98.03%"
> paste0(round(100*perf_J48_LOOCV_preRecall@y.values[[1]][2], 2), "%") # Precision values
[1] "98.03%"
>
```

We are interested in the middle result, The precision is 98.03% while recall is 98.03%.

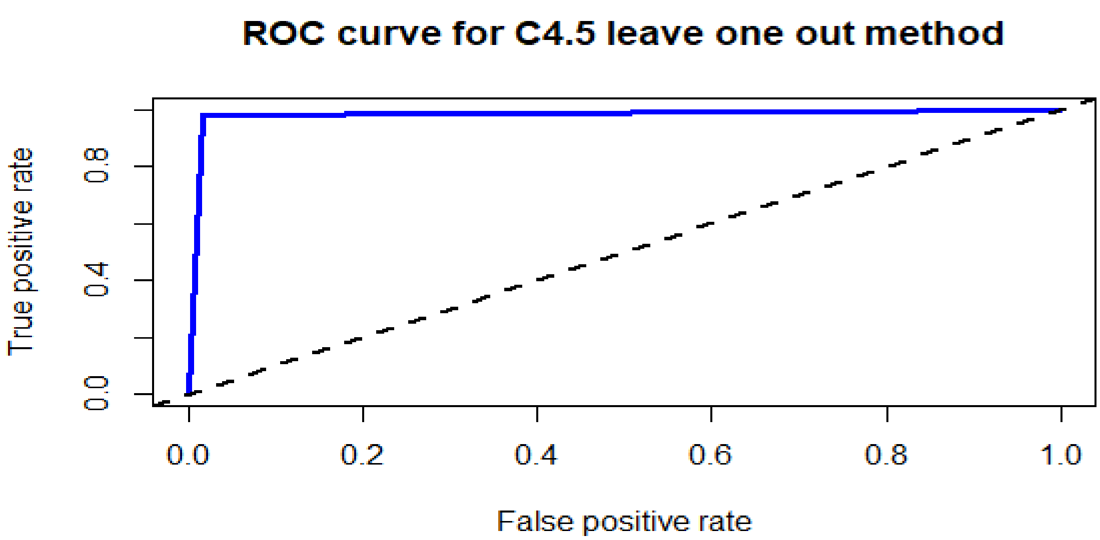
Accuracy estimation

```
> perf_J48_LOOCV_acc <- performance(pred_J48_LOOCV, "acc")
> perf_J48_LOOCV_acc@y.values[[1]] #Accuracy
[1] 0.5555556 0.9824561 0.4444444
> paste0(round(100*perf_J48_LOOCV_acc@y.values[[1]][2], 2), "%")
[1] "98.25%"
>
```

The accuracy is 98.25%.

ROC(receiver operating characteristic curve)

```
> perf_J48_LOOCV_ROC <- performance(pred_J48_LOOCV,  measure = "tpr", x.measure = "fpr")
> plot(perf_J48_LOOCV_ROC, main = "ROC curve for C4.5 leave one out method", col = "blue",
 lwd = 3)
> abline(a = 0, b = 1, lwd = 2, lty = 2)
>
```




```
> perf_J48_LOOCV_AUC <- performance(pred_J48_LOOCV, measure = "auc")
> perf_J48_LOOCV_AUC@y.values[[1]] #AUC
[1] 0.9822368
> paste0(round(100*perf_J48_LOOCV_AUC@y.values[[1]], 2), "%") #AUC
[1] "98.22%"
>
```

The area under the curve is 98.22% which imply that we have an excellent model.

RANDOM FOREST: Hold out method [Measure Performance]

Confusion matrix

```
> confusionMatrix(predict_RF_HO_1, data_test$class)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 187   0
         1   3 152

               Accuracy : 0.9912
                 95% CI : (0.9746, 0.9982)
    No Information Rate : 0.5556
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9823
 Mcnemar's Test P-Value : 0.2482

            Sensitivity : 0.9842
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9806
             Prevalence : 0.5556
         Detection Rate : 0.5468
   Detection Prevalence : 0.5468
      Balanced Accuracy : 0.9921

       'Positive' Class : 0

>
```

The hold out predicts 187 corretly as 0, misclassified no (0s) as 1. It classified 152 correctly as 1 but misclassified 3 (1s) as 0s. The accuracy is 99.12% on the new test data accurately.

Precision vs. Recall estimation

```
> pred_RF_HO <- prediction(as.numeric(predict_RF_HO_1), as.numeric(data_test$class))
> perf_RF_HO_preRecall <- performance(pred_RF_HO, measure="prec", x.measure="rec")
> perf_RF_HO_preRecall@x.values[[1]] # Recall values
[1] 0 1 1
> perf_RF_HO_preRecall@y.values[[1]] # Precision values
[1]       NaN 0.9806452 0.4444444
> paste0(round(100*perf_RF_HO_preRecall@x.values[[1]][2], 2), "%") # Recall values
[1] "100%"
> paste0(round(100*perf_RF_HO_preRecall@y.values[[1]][2], 2), "%") # Precision values
[1] "98.06%"
>
```

We are interested in the middle result, The precision is 98.06% while recall is 100%.

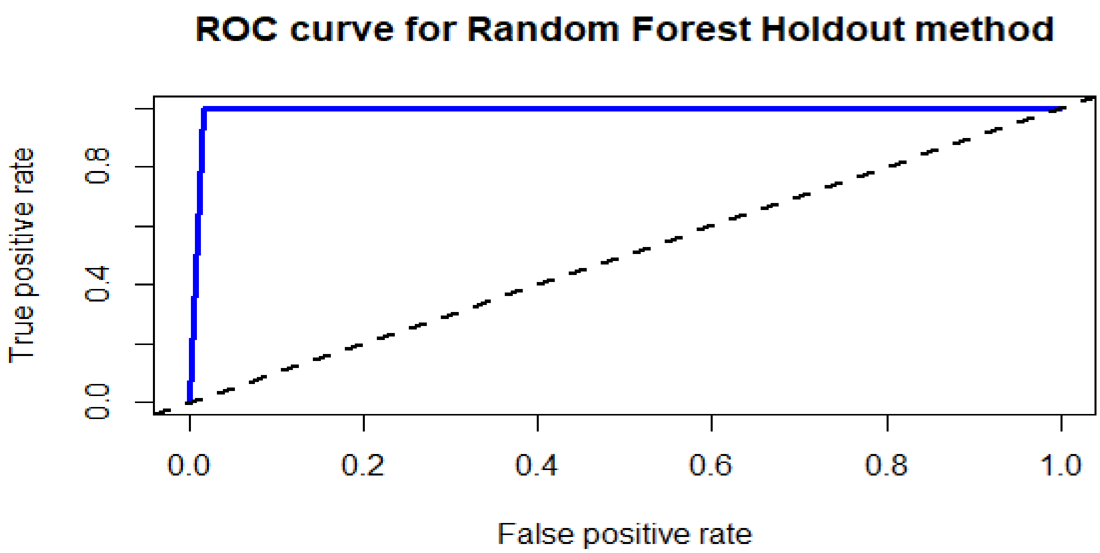
Accuracy estimation

```
> perf_RF_HO_acc <- performance(pred_RF_HO, "acc")
> perf_RF_HO_acc@y.values[[1]] #Accuracy
[1] 0.5555556 0.9912281 0.4444444
> paste0(round(100*perf_RF_HO_acc@y.values[[1]][2], 2), "%") #Accuracy
[1] "99.12%"
>
```

The accuracy is 99.12%.

ROC(receiver operating characteristic curve)

```
> perf_RF_HO_ROC <- performance(pred_RF_HO,  measure = "tpr", x.measure = "fpr")
> plot(perf_RF_HO_ROC, main = "ROC curve for Random Forest Holdout method", col = "blue",
lwd = 3)
> abline(a = 0, b = 1, lwd = 2, lty = 2)
>
```



RAUC (receiver under the curve area)

```
> perf_RF_HO_AUC <- performance(pred_RF_HO, measure = "auc")
> perf_RF_HO_AUC@y.values[[1]] #AUC
[1] 0.9921053
> paste0(round(100*perf_RF_HO_AUC@y.values[[1]], 2), "%") #AUC
[1] "99.21%"
>
```

The area under the curve is 99.21% which imply that we have an excellent model.

RANDOM FOREST: 10 fold cross validation method [Measure Performance]

```
> confusionMatrix(predict_RF_CV, data_test$class)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 187   0
         1   3 152

               Accuracy : 0.9912
                 95% CI : (0.9746, 0.9982)
    No Information Rate : 0.5556
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9823
 Mcnemar's Test P-Value : 0.2482

            Sensitivity : 0.9842
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9806
             Prevalence : 0.5556
         Detection Rate : 0.5468
   Detection Prevalence : 0.5468
      Balanced Accuracy : 0.9921

       'Positive' Class : 0

>
```

The cross validation method predicts 187 corretly as 0, misclassified no (0s) as 1. It classified 152 correctly as 1 but misclassified 3 1s as 0s. The accuracy is 99.12% on the new test data accurately.

Precision vs. Recall estimation

```
> pred_RF_CV <- prediction(as.numeric(predict_RF_CV), as.numeric(data_test$class))
> perf_RF_CV_preRecall <- performance(pred_RF_CV, measure="prec", x.measure="rec")
> perf_RF_CV_preRecall@x.values[[1]] # Recall values
[1] 0 1 1
> perf_RF_CV_preRecall@y.values[[1]] # Precision values
[1]       NaN 0.9806452 0.4444444
> paste0(round(100*perf_RF_CV_preRecall@x.values[[1]][2], 2), "%") # Recall values
[1] "100%"
> paste0(round(100*perf_RF_CV_preRecall@y.values[[1]][2], 2), "%") # Precision values
[1] "98.06%"
> 
```

We are interested in the middle result, The precision is 98.06% while recall is 100%.

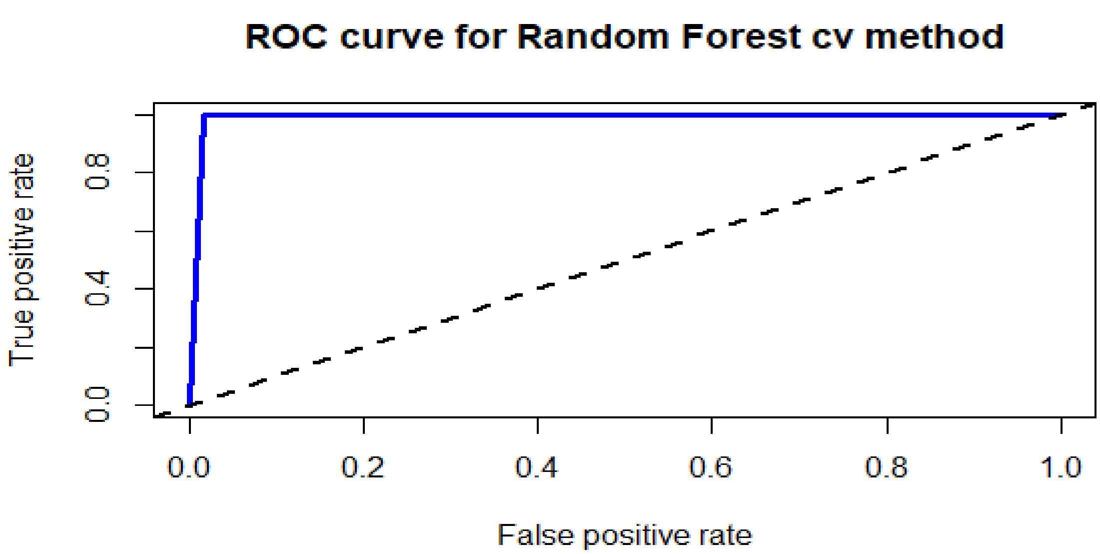
Accuracy estimation

```
> perf_RF_CV_acc <- performance(pred_RF_CV, "acc")
> perf_RF_CV_acc@y.values[[1]] #Accuracy
[1] 0.5555556 0.9912281 0.4444444
> paste0(round(100*perf_RF_CV_acc@y.values[[1]][2], 2), "%") #Accuracy
[1] "99.12%"
> 
```

The accuracy is 99.12%.

ROC(receiver operating characteristic curve)

```
> perf_RF_CV_ROC <- performance(pred_RF_CV,  measure = "tpr", x.measure = "fpr")
> plot(perf_RF_CV_ROC, main = "ROC curve for Random Forest cv method", col = "blue", lwd =
 3)
> abline(a = 0, b = 1, lwd = 2, lty = 2)
> 
```




```
> perf_RF_CV_AUC <- performance(pred_RF_CV, measure = "auc")
> perf_RF_CV_AUC@y.values[[1]] #AUC
[1] 0.9921053
> paste0(round(100*perf_RF_CV_AUC@y.values[[1]], 2), "%") #AUC
[1] "99.21%"
> 
```

The area under the curve is 99.21% which imply that we have an excellent model.

RANDOM FOREST: Leave One Out Cross Validation [Measure Performance]

```
> confusionMatrix(predict_RF_LOOCV, data_test$class)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 186   1
         1   4 151

               Accuracy : 0.9854
                 95% CI : (0.9662, 0.9952)
    No Information Rate : 0.5556
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9705
 Mcnemar's Test P-Value : 0.3711

            Sensitivity : 0.9789
            Specificity : 0.9934
         Pos Pred Value : 0.9947
         Neg Pred Value : 0.9742
             Prevalence : 0.5556
         Detection Rate : 0.5439
   Detection Prevalence : 0.5468
      Balanced Accuracy : 0.9862

       'Positive' Class : 0

>
```

The leave one out method predicts 186 corretly as 0, misclassified 1 (0s) as 1. It classified 151 correctly as 1 but misclassified 4 1s as 0s. The accuracy is 98.54% on the new test data accurately.

Precision vs. Recall estimation

```
> pred_RF_LOOCV <- prediction(as.numeric(predict_RF_LOOCV), as.numeric(data_test$class))
> perf_RF_LOOCV_preRecall <- performance(pred_RF_LOOCV, measure="prec", x.measure="rec")
> perf_RF_LOOCV_preRecall@x.values[[1]] # Recall values
[1] 0.0000000 0.9934211 1.0000000
> perf_RF_LOOCV_preRecall@y.values[[1]] # Precision values
[1]       NaN 0.9741935 0.4444444
> paste0(round(100*perf_RF_LOOCV_preRecall@x.values[[1]][2], 2), "%") # Recall values
[1] "99.34%"
> paste0(round(100*perf_RF_LOOCV_preRecall@y.values[[1]][2], 2), "%") # Precision values
[1] "97.42%"
>
```

We are interested in the middle result, The precision is 97.42% while recall is 99.34%.

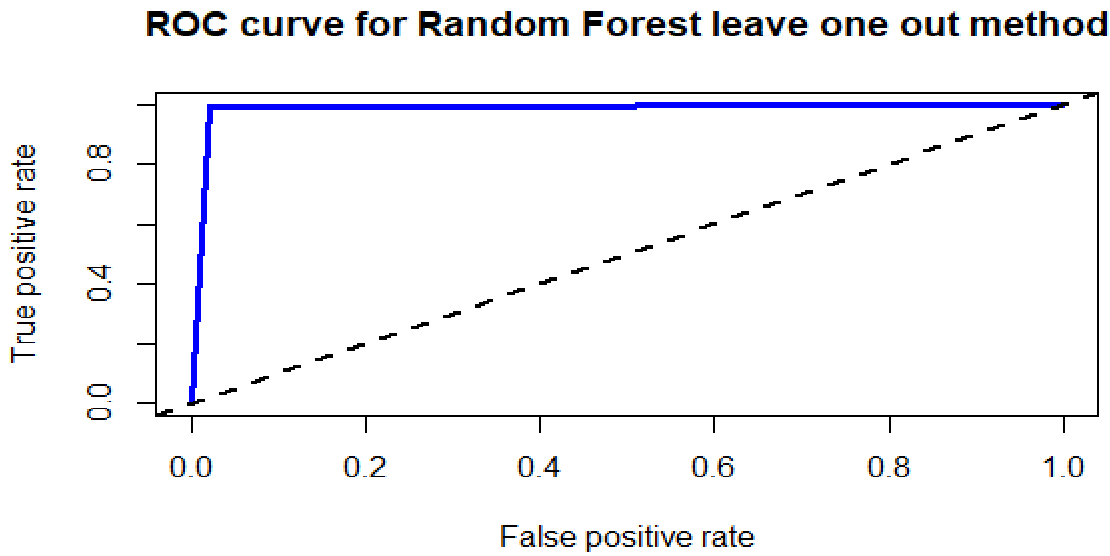
Accuracy estimation

```
> perf_RF_LOOCV_acc <- performance(pred_RF_LOOCV, "acc")
> perf_RF_LOOCV_acc@y.values[[1]] #Accuracy
[1] 0.5555556 0.9853801 0.4444444
> paste0(round(100*perf_RF_LOOCV_acc@y.values[[1]][2], 2), "%") #Accuracy
[1] "98.54%"
>
```

The accuracy is 98.54%.

ROC(receiver operating characteristic curve)

```
> perf_RF_LOOCV_ROC <- performance(pred_RF_LOOCV,  measure = "tpr", x.measure = "fpr")
> plot(perf_RF_LOOCV_ROC, main = "ROC curve for Random Forest leave one out method", col =
 "blue", lwd = 3)
> abline(a = 0, b = 1, lwd = 2, lty = 2)
>
```




```
> perf_RF_LOOCV_AUC <- performance(pred_RF_LOOCV, measure = "auc")
> perf_RF_LOOCV_AUC@y.values[[1]] #AUC
[1] 0.9861842
> paste0(round(100*perf_RF_LOOCV_AUC@y.values[[1]], 2), "%") #AUC
[1] "98.62%"
>
```

The area under the curve is 98.62% which imply that we have an excellent model.

NAÏVE BAYES: Hold out method [Measure Performance]

Confusion matrix

```
> confusionMatrix(predict_NB_HO_1$class, data_test$class )
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 160  30
         1  30 122

               Accuracy : 0.8246
                 95% CI : (0.78, 0.8634)
    No Information Rate : 0.5556
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.6447
 Mcnemar's Test P-Value : 1

            Sensitivity : 0.8421
            Specificity : 0.8026
         Pos Pred Value : 0.8421
         Neg Pred Value : 0.8026
             Prevalence : 0.5556
         Detection Rate : 0.4678
   Detection Prevalence : 0.5556
      Balanced Accuracy : 0.8224

       'Positive' Class : 0

>
```

The hold out predicts 160 corretly as 0, misclassified 30 (0s) as 1. It classified 122 correctly as 1 but misclassified 30 1s as 0s. The accuracy is 82.46% on the new test data accurately.

Precision vs. Recall estimation

```
> pred_NB_HO <- prediction(as.numeric(predict_NB_HO_1$class), as.numeric(data_test$class))
> perf_NB_HO_preRecall <- performance(pred_NB_HO, measure="prec", x.measure="rec")
> perf_NB_HO_preRecall@x.values[[1]] # Recall values
[1] 0.0000000 0.8026316 1.0000000
> perf_NB_HO_preRecall@y.values[[1]] # Precision values
[1]       NaN 0.8026316 0.4444444
> paste0(round(100*perf_NB_HO_preRecall@x.values[[1]][2], 2), "%") # Recall values
[1] "80.26%"
> paste0(round(100*perf_NB_HO_preRecall@y.values[[1]][2], 2), "%") # Precision values
[1] "80.26%"
>
```

We are interested in the middle result, The precision is 80.26% while recall is 80.26%.

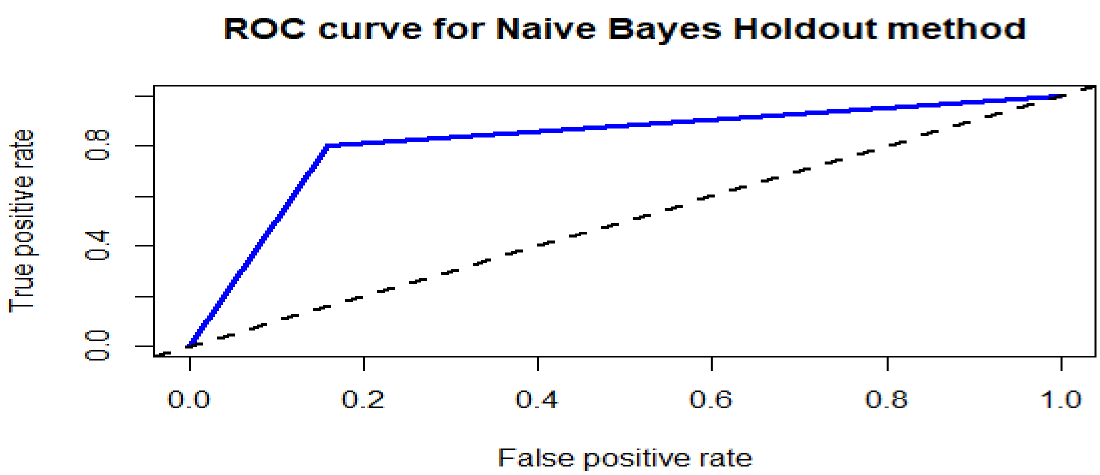
Accuracy estimation

```
> perf_NB_HO_acc <- performance(pred_NB_HO, "acc")
> perf_NB_HO_acc@y.values[[1]] #Accuracy
[1] 0.5555556 0.8245614 0.4444444
> paste0(round(100*perf_NB_HO_acc@y.values[[1]][2], 2), "%") #Accuracy
[1] "82.46%"
>
```

The accuracy is 82.46%.

ROC(receiver operating characteristic curve)

```
> perf_NB_HO_ROC <- performance(pred_NB_HO,  measure = "tpr", x.measure = "fpr")
> plot(perf_NB_HO_ROC, main = "ROC curve for Naive Bayes Holdout method", col = "blue", lw
d = 3)
> abline(a = 0, b = 1, lwd = 2, lty = 2)
>
```



RAUC (receiver under the curve area)

```
> perf_NB_HO_AUC <- performance(pred_NB_HO, measure = "auc")
> perf_NB_HO_AUC@y.values[[1]] #AUC
[1] 0.8223684
> paste0(round(100*perf_NB_HO_AUC@y.values[[1]], 2), "%") #AUC
[1] "82.24%"
>
```

The area under the curve is 82.24% which imply that we have a good model.

NAÏVE BAYES: 10 fold cross validation method [Measure Performance]

```
> confusionMatrix(predict_NB_CV, data_test$class)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 171  16
         1  19 136

               Accuracy : 0.8977
                 95% CI : (0.8606, 0.9277)
    No Information Rate : 0.5556
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.7932
 Mcnemar's Test P-Value : 0.7353

            Sensitivity : 0.9000
            Specificity : 0.8947
         Pos Pred Value : 0.9144
         Neg Pred Value : 0.8774
             Prevalence : 0.5556
         Detection Rate : 0.5000
   Detection Prevalence : 0.5468
      Balanced Accuracy : 0.8974

       'Positive' Class : 0

>
```

The cross validation method predicts 171 corretly as 0, misclassified 16 (0s) as 1. It classified 136 correctly as 1 but misclassified 19 1s as 0s. The accuracy is 89.77% on the new test data accurately.

Precision vs. Recall estimation

```
> pred_NB_CV <- prediction(as.numeric(predict_NB_CV), as.numeric(data_test$class))
> perf_NB_CV_preRecall <- performance(pred_NB_CV, measure="prec", x.measure="rec")
> perf_NB_CV_preRecall@x.values[[1]] # Recall values
[1] 0.0000000 0.8947368 1.0000000
> perf_NB_CV_preRecall@y.values[[1]] # Precision values
[1]       NaN 0.8774194 0.4444444
> paste0(round(100*perf_NB_CV_preRecall@x.values[[1]][2], 2), "%") # Recall values
[1] "89.47%"
> paste0(round(100*perf_NB_CV_preRecall@y.values[[1]][2], 2), "%") # Precision values
[1] "87.74%"
>
```

We are interested in the middle result, The precision is 87.74% while recall is 89.47%.

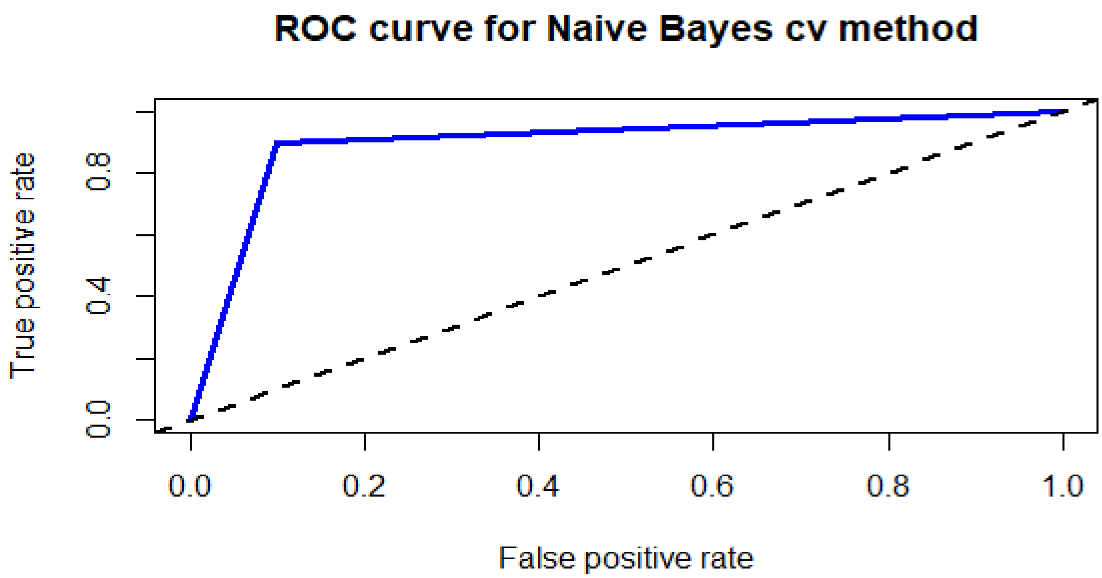
Accuracy estimation

```
> perf_NB_CV_acc <- performance(pred_NB_CV, "acc")
> perf_NB_CV_acc@y.values[[1]] #Accuracy
[1] 0.5555556 0.8976608 0.4444444
> paste0(round(100*perf_NB_CV_acc@y.values[[1]][2], 2), "%") #Accuracy
[1] "89.77%"
>
```

The accuracy is 89.77%.

ROC(receiver operating characteristic curve)

```
> perf_NB_CV_ROC <- performance(pred_NB_CV,  measure = "tpr", x.measure = "fpr")
> plot(perf_NB_CV_ROC, main = "ROC curve for Naive Bayes cv method", col = "blue", lwd = 3
)
> abline(a = 0, b = 1, lwd = 2, lty = 2)
>
```




```
> perf_NB_CV_AUC <- performance(pred_NB_CV, measure = "auc")
> perf_NB_CV_AUC@y.values[[1]] #AUC
[1] 0.8973684
> paste0(round(100*perf_NB_CV_AUC@y.values[[1]], 2), "%") #AUC
[1] "89.74%"
>
```

The area under the curve is 89.74% which imply that we have a good model.

NAÏVE BAYES: Leave One Out Cross Validation [Measure Performance]

```
> confusionMatrix(predict_NB_LOOCV, data_test$class)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 171  16
         1  19 136

               Accuracy : 0.8977
                 95% CI : (0.8606, 0.9277)
    No Information Rate : 0.5556
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.7932
 Mcnemar's Test P-Value : 0.7353

            Sensitivity : 0.9000
            Specificity : 0.8947
         Pos Pred Value : 0.9144
         Neg Pred Value : 0.8774
             Prevalence : 0.5556
         Detection Rate : 0.5000
   Detection Prevalence : 0.5468
      Balanced Accuracy : 0.8974

       'Positive' Class : 0

>
```

The cross validation method predicts 171 corretly as 0, misclassified 16 (0s) as 1. It classified 136 correctly as 1 but misclassified 19 1s as 0s. The accuracy is 89.77% on the new test data accurately.

Precision vs. Recall estimation

```
> pred_NB_LOOCV <- prediction(as.numeric(predict_NB_LOOCV), as.numeric(data_test$class))
> perf_NB_LOOCV_preRecall <- performance(pred_NB_LOOCV, measure="prec", x.measure="rec")
> perf_NB_LOOCV_preRecall@x.values[[1]] # Recall values
[1] 0.0000000 0.8947368 1.0000000
> perf_NB_LOOCV_preRecall@y.values[[1]] # Precision values
[1]       NaN 0.8774194 0.4444444
> paste0(round(100*perf_NB_LOOCV_preRecall@x.values[[1]][2], 2), "%") # Recall values
[1] "89.47%"
> paste0(round(100*perf_NB_LOOCV_preRecall@y.values[[1]][2], 2), "%") # Precision values
[1] "87.74%"
>
```

We are interested in the middle result, The precision is 87.74% while recall is 89.47%.

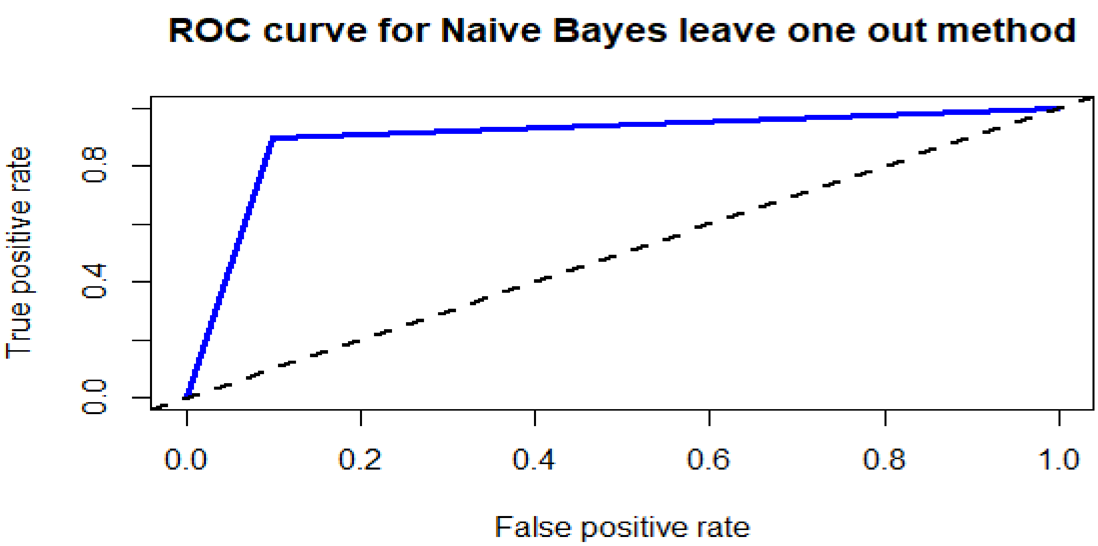
Accuracy estimation

```
> perf_NB_LOOCV_acc <- performance(pred_NB_LOOCV, "acc")
> perf_NB_LOOCV_acc@y.values[[1]] #Accuracy
[1] 0.5555556 0.8976608 0.4444444
> paste0(round(100*perf_NB_LOOCV_acc@y.values[[1]][2], 2), "%") #Accuracy
[1] "89.77%"
>
```

The accuracy is 89.77%.

ROC(receiver operating characteristic curve)

```
> perf_NB_LOOCV_ROC <- performance(pred_NB_LOOCV,  measure = "tpr", x.measure = "fpr")
> plot(perf_NB_LOOCV_ROC, main = "ROC curve for Naive Bayes leave one out method", col = "
blue", lwd = 3)
> abline(a = 0, b = 1, lwd = 2, lty = 2)
>
```




```
> perf_NB_LOOCV_AUC <- performance(pred_NB_LOOCV, measure = "auc")
> perf_NB_LOOCV_AUC@y.values[[1]] #AUC
[1] 0.8973684
> paste0(round(100*perf_NB_LOOCV_AUC@y.values[[1]], 2), "%") #AUC
[1] "89.74%"
>
```

The area under the curve is 89.74% which imply that we have a good model.

KNN: Hold out method [Measure Performance]

Confusion matrix

```
> confusionMatrix(model_KNN_HO_1, data_test$class )
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 189   0
         1   1 152

               Accuracy : 0.9971
                 95% CI : (0.9838, 0.9999)
    No Information Rate : 0.5556
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9941
 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9947
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9935
             Prevalence : 0.5556
         Detection Rate : 0.5526
   Detection Prevalence : 0.5526
      Balanced Accuracy : 0.9974

       'Positive' Class : 0

> 
```

The hold out predicts 189 corretly as 0, misclassified no (0s) as 1. It classified 152 correctly as 1 but misclassified 1 (1s) as 0s. The accuracy is 99.71% on the new test data accurately.

Precision vs. Recall estimation

```
> pred_KNN_HO <- prediction(as.numeric(model_KNN_HO_1), as.numeric(data_test$class))
> perf_KNN_HO_preRecall <- performance(pred_KNN_HO, measure="prec", x.measure="rec")
> perf_KNN_HO_preRecall@x.values[[1]] # Recall values
[1] 0 1 1
> perf_KNN_HO_preRecall@y.values[[1]] # Precision values
[1]       NaN 0.9934641 0.4444444
> paste0(round(100*perf_KNN_HO_preRecall@x.values[[1]][2], 2), "%") # Recall values
[1] "100%"
> paste0(round(100*perf_KNN_HO_preRecall@y.values[[1]][2], 2), "%") # Precision values
[1] "99.35%"
> 
```

We are interested in the middle result, The precision is 99.35% while recall is 100%.

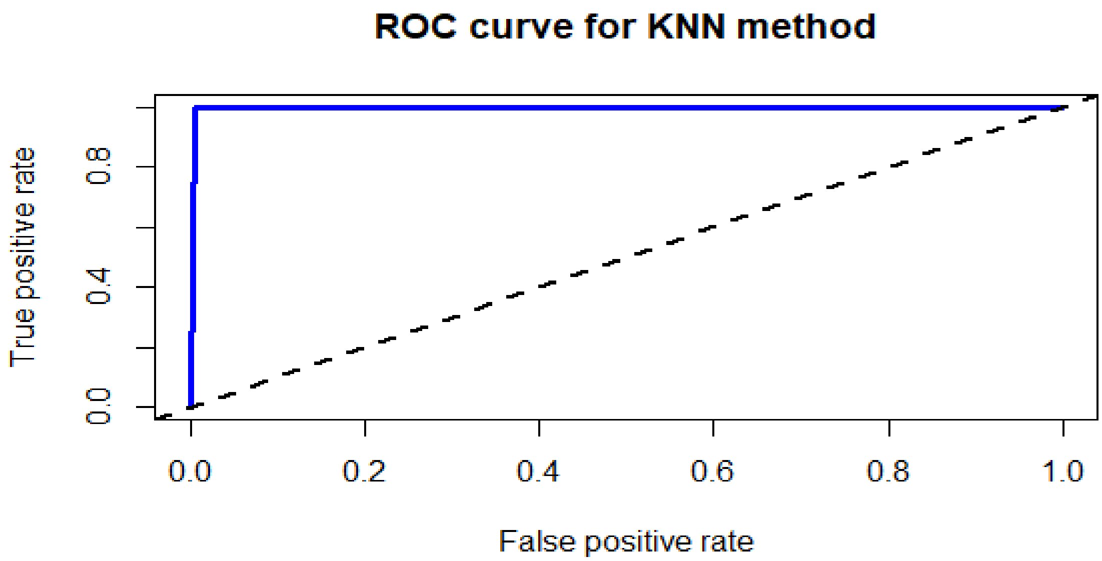
Accuracy estimation

```
> perf_KNN_HO_acc <- performance(pred_KNN_HO, "acc")
> perf_KNN_HO_acc@y.values[[1]] #Accuracy
[1] 0.5555556 0.9970760 0.4444444
> paste0(round(100*perf_KNN_HO_acc@y.values[[1]][2], 2), "%") #Accuracy
[1] "99.71%"
> 
```

The accuracy is 99.71%.

ROC(receiver operating characteristic curve)

```
> perf_KNN_HO_ROC <- performance(pred_KNN_HO,  measure = "tpr", x.measure = "fpr")
> plot(perf_KNN_HO_ROC, main = "ROC curve for KNN method", col = "blue", lwd = 3)
> abline(a = 0, b = 1, lwd = 2, lty = 2)
> 
```



RAUC (receiver under the curve area)

```
> perf_KNN_HO_AUC <- performance(pred_KNN_HO, measure = "auc")
> perf_KNN_HO_AUC@y.values[[1]] #AUC
[1] 0.9973684
> paste0(round(100*perf_KNN_HO_AUC@y.values[[1]], 2), "%") #AUC
[1] "99.74%"
> 
```

The area under the curve is 99.74% which imply that we have an excellent model.

KNN: 10 fold cross validation method [Measure Performance]

```
> confusionMatrix(predict_KNN_CV, data_test$class)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 189   0
         1   1 152

               Accuracy : 0.9971
                 95% CI : (0.9838, 0.9999)
    No Information Rate : 0.5556
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9941
 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9947
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9935
             Prevalence : 0.5556
         Detection Rate : 0.5526
   Detection Prevalence : 0.5526
      Balanced Accuracy : 0.9974

       'Positive' Class : 0

> 
```

The cross validation method predicts 189 corretly as 0, misclassified no (0s) as 1. It classified 152 correctly as 1 but misclassified 1 (1s) as 0s. The accuracy is 99.71% on the new test data accurately.

Precision vs. Recall estimation

```
> pred_KNN_CV <- prediction(as.numeric(predict_KNN_CV), as.numeric(data_test$class))
> perf_KNN_CV_preRecall <- performance(pred_KNN_CV, measure="prec", x.measure="rec")
> perf_KNN_CV_preRecall@x.values[[1]] # Recall values
[1] 0 1 1
> perf_KNN_CV_preRecall@y.values[[1]] # Precision values
[1]       NaN 0.9934641 0.4444444
> paste0(round(100*perf_KNN_CV_preRecall@x.values[[1]][2], 2), "%") # Recall values
[1] "100%"
> paste0(round(100*perf_KNN_CV_preRecall@y.values[[1]][2], 2), "%") # Precision values
[1] "99.35%"
>
```

We are interested in the middle result, The precision is 99.35% while recall is 100%.

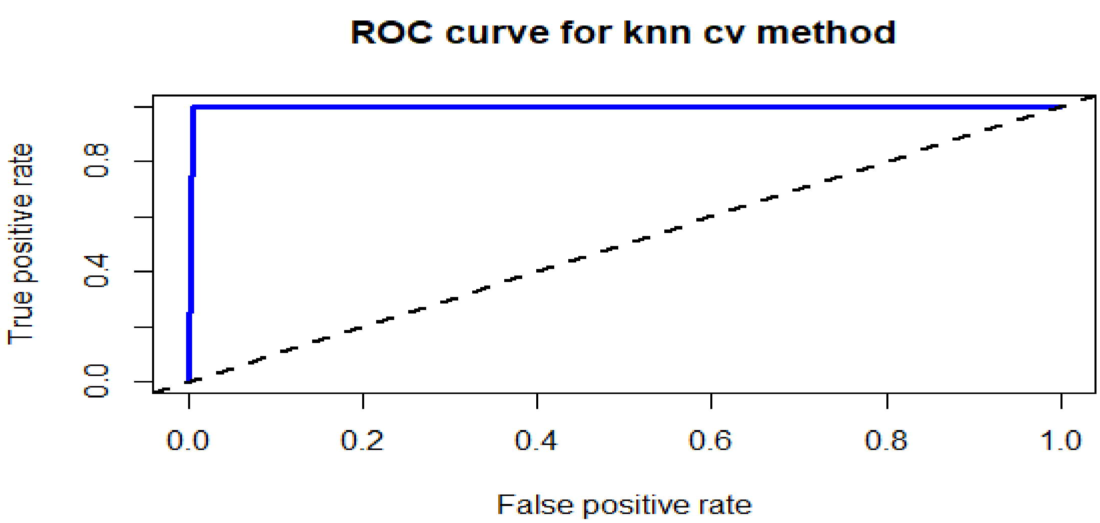
Accuracy estimation

```
> perf_KNN_CV_acc <- performance(pred_KNN_CV, "acc")
> perf_KNN_CV_acc@y.values[[1]] #Accuracy
[1] 0.5555556 0.9970760 0.4444444
> paste0(round(100*perf_KNN_CV_acc@y.values[[1]][2], 2), "%") #Accuracy
[1] "99.71%"
>
```

The accuracy is 99.71%.

ROC(receiver operating characteristic curve)

```
> perf_KNN_CV_ROC <- performance(pred_KNN_CV,  measure = "tpr", x.measure = "fpr")
> plot(perf_KNN_CV_ROC, main = "ROC curve for knn cv method", col = "blue", lwd = 3)
> abline(a = 0, b = 1, lwd = 2, lty = 2)
>
```




```
> perf_KNN_CV_AUC <- performance(pred_KNN_CV, measure = "auc")
> perf_KNN_CV_AUC@y.values[[1]] #AUC
[1] 0.9973684
> paste0(round(100*perf_KNN_CV_AUC@y.values[[1]], 2), "%") #AUC
[1] "99.74%"
>
```

The area under the curve is 99.74% which imply that we have an excellent model.

KNN: Leave One Out Cross Validation [Measure Performance]

```
> confusionMatrix(predict_KNN_LOOCV, data_test$class)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 188   0
         1   2 152

               Accuracy : 0.9942
                 95% CI : (0.979, 0.9993)
    No Information Rate : 0.5556
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9882
 Mcnemar's Test P-Value : 0.4795

            Sensitivity : 0.9895
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9870
             Prevalence : 0.5556
         Detection Rate : 0.5497
   Detection Prevalence : 0.5497
      Balanced Accuracy : 0.9947

       'Positive' Class : 0

>
```

The leave one out method predicts 188 corretly as 0, misclassified no (0s) as 1. It classified 152 correctly as 1 but misclassified 2 (1s) as 0s. The accuracy is 99.71% on the new test data accurately.

Precision vs. Recall estimation

```
> pred_KNN_LOOCV <- prediction(as.numeric(predict_KNN_LOOCV), as.numeric(data_test$class))
> perf_KNN_LOOCV_preRecall <- performance(pred_KNN_LOOCV, measure="prec", x.measure="rec")
> perf_KNN_LOOCV_preRecall@x.values[[1]] # Recall values
[1] 0 1 1
> perf_KNN_LOOCV_preRecall@y.values[[1]] # Precision values
[1]       NaN 0.9870130 0.4444444
> paste0(round(100*perf_KNN_LOOCV_preRecall@x.values[[1]][2], 2), "%") # Recall values
[1] "100%"
> paste0(round(100*perf_KNN_LOOCV_preRecall@y.values[[1]][2], 2), "%") # Precision values
[1] "98.7%"
>
```

We are interested in the middle result, The precision is 98.7% while recall is 100%.

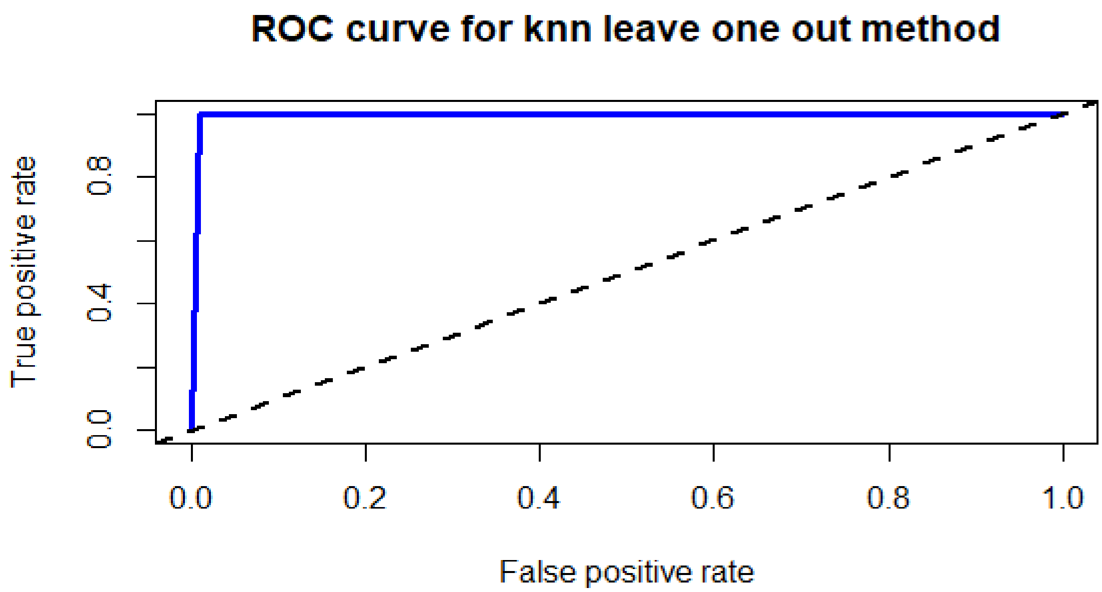
Accuracy estimation

```
> perf_KNN_LOOCV_acc <- performance(pred_KNN_LOOCV, "acc")
> perf_KNN_LOOCV_acc@y.values[[1]] #Accuracy
[1] 0.5555556 0.9941520 0.4444444
> paste0(round(100*perf_KNN_LOOCV_acc@y.values[[1]][2], 2), "%") #Accuracy
[1] "99.42%"
>
```

The accuracy is 99.42%.

ROC(receiver operating characteristic curve)

```
> perf_KNN_LOOCV_ROC <- performance(pred_KNN_LOOCV,  measure = "tpr", x.measure = "fpr")
> plot(perf_KNN_LOOCV_ROC, main = "ROC curve for knn leave one out method", col = "blue",
lwd = 3)
> abline(a = 0, b = 1, lwd = 2, lty = 2)
>
```




```
> perf_KNN_LOOCV_AUC <- performance(pred_KNN_LOOCV, measure = "auc")
> perf_KNN_LOOCV_AUC@y.values[[1]] #AUC
[1] 0.9947368
> paste0(round(100*perf_KNN_LOOCV_AUC@y.values[[1]], 2), "%") #AUC
[1] "99.47%"
>
```

The area under the curve is 99.47% which imply that we have an excellent.

<u>Summary</u>

	Decision Tree (C 4.5)			Random Forest			Naïve Bayes			knn		
	HO	CV	LOOCV	HO	CV	LOOCV	HO	CV	LOOCV	HO	CV	LOOCV
Precision	98.03%	98.03%	98.03%	98.06%	98.06%	97.42%	80.26%	87.74%	87.74%	99.35%	99.35%	98.70%
Recall	98.03%	98.03%	98.03%	100.00%	100.00%	99.34%	80.26%	89.47%	89.47%	100.00%	100.00%	100.00%
Accuracy Estimate	98.25%	98.25%	98.25%	99.12%	99.12%	98.54%	82.46%	89.77%	89.77%	99.71%	99.71%	99.42%
AUC	98.22%	98.22%	98.22%	99.21%	99.21%	98.62%	82.24%	89.74%	89.74%	99.74%	99.74%	99.74%

The table shows the performance measure of all the models. KNN, Random forest, and C 4.5 produced excellent models whereas, Naïve Bayes produced a good model based on all of AUC. Accuracy, Precision and Recall measured from the model prediction on the test set.

References:

[1] Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.
[2] https://cran.r-project.org/web/packages/caret/caret.pdf
[3] https://cran.r-project.org/web/packages/ROCR/ROCR.pdf