CS 513 - KNOWLEDGE DISCOVERY & DATA MINING

Prediction of Patient Survival After one Year

Group 6:

Ruobing Liu, Wencheng Qiu, Jiayin Huang, Zimeng Zhao

Date: 04/22/2023

PROBLEM STATEMENT & OBJECTIVE

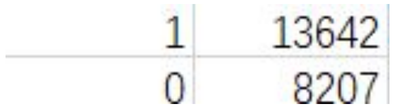
Problem Statement:

- A hospital in New Jersey has been trying to improve its care conditions by looking at the historic survival of the patients. There are many factors that can affect a patient's survival, such as age, Treated_with_drugs, mental health,Residence, life style, etc.
- The accurately predicting patient survival can aid in treatment decisions, personalized therapy, and the management of patient expectations. Therefore, develop accurate models for predicting patient survival can significantly impact patient care and ultimately lead to better clinical outcomes.

Objective:

- Understand the predictor variables which have a larger influence on patient survival across the board.
- Predict the Patient's survival after One Year of Treatment, based on different predictor variables.

DATASET:



1	13642
0	8207

- Cleaned data : remove the blanks and normalize the variables.
- The dataset comprises 16 features in the form of columns (plus one tag for prediction). We use PCA and correlation for feature reduction.
- Training data: 15 columns and 15,294 training rows of data.
- Test data: 15 columns and 6,555 rows.

Dataset Variables Attributes:

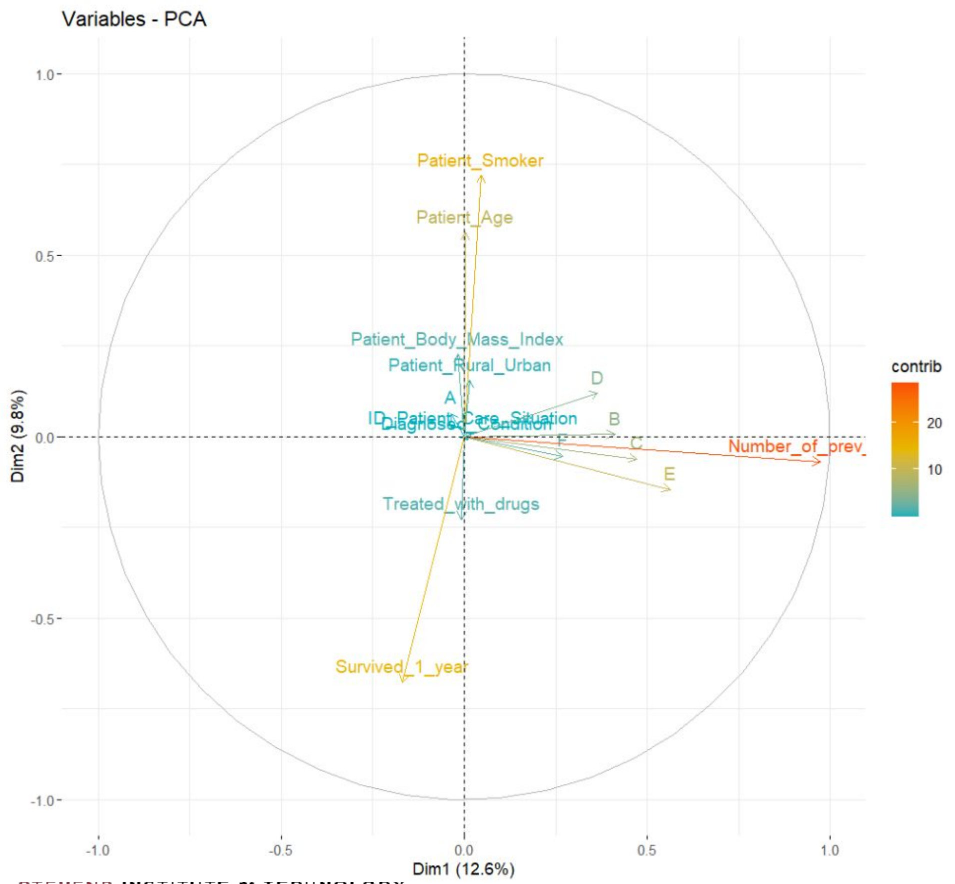
1. ID_Patient_Care_Situation: Care situation of a patient during treatment
2. Diagnosed_Condition: The diagnosed condition of the patient
3. ID_Patient: Patient identifier number
4. Treatment_with_drugs: Class of drugs used during treatment
5. Survived_1_year: If the patient survived after one year (0 means did not survive; 1 means survived)
6. Patient_Age: Age of the patient
7. Patient_Body_Mass_Index: A calculated value based on the patient’s weight, height, etc.
8. Patient_Smoker: If the patient was a smoker or not
9. Patient_Rural_Urban: If the patient stayed in Rural or Urban part of the country
10. Previous_Condition: Condition of the patient before the start of the treatment.

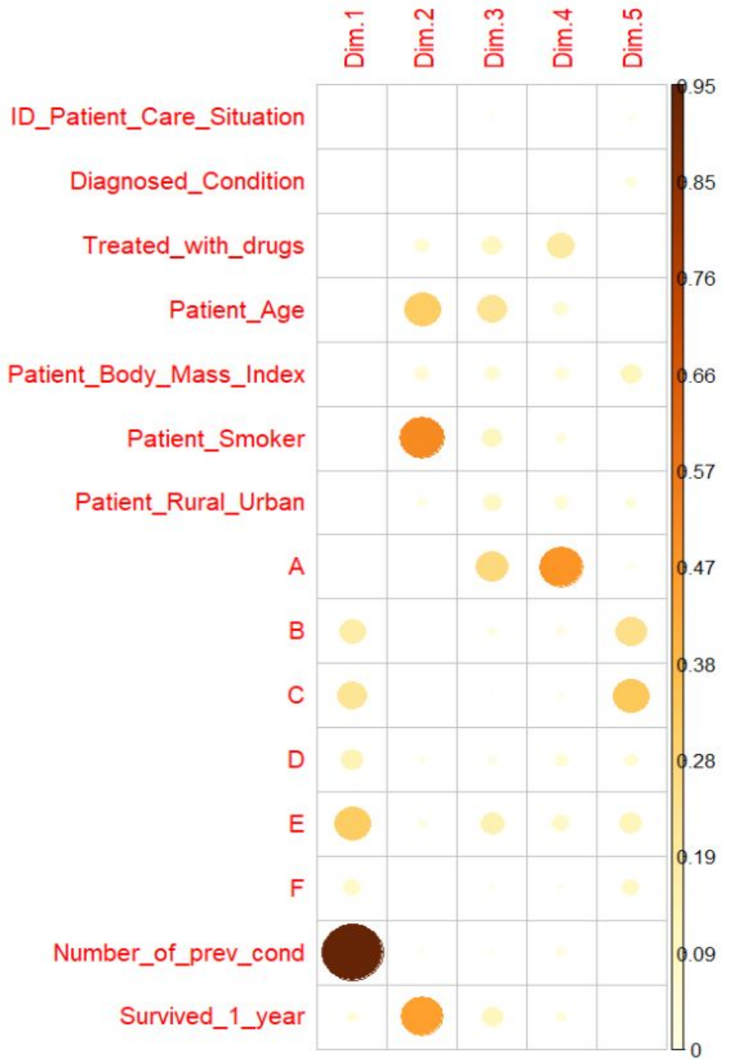
CLASSIFICATION TECHNIQUES USED

Methods

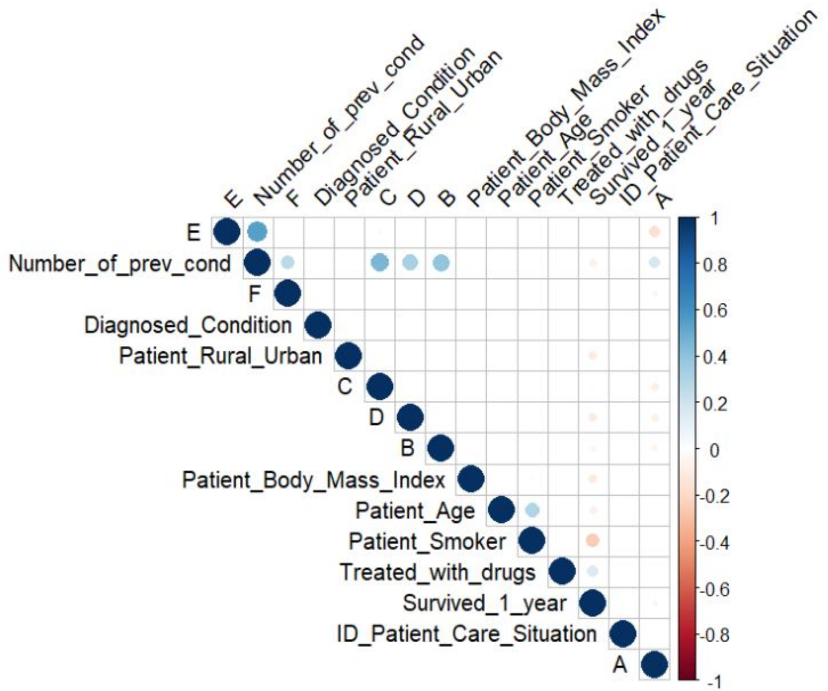
- KNN
- Naive Bayes
- CART Decision Tree
- Random Forest
- Logistic Regression
- ANN
- SVM
- XGBOOST

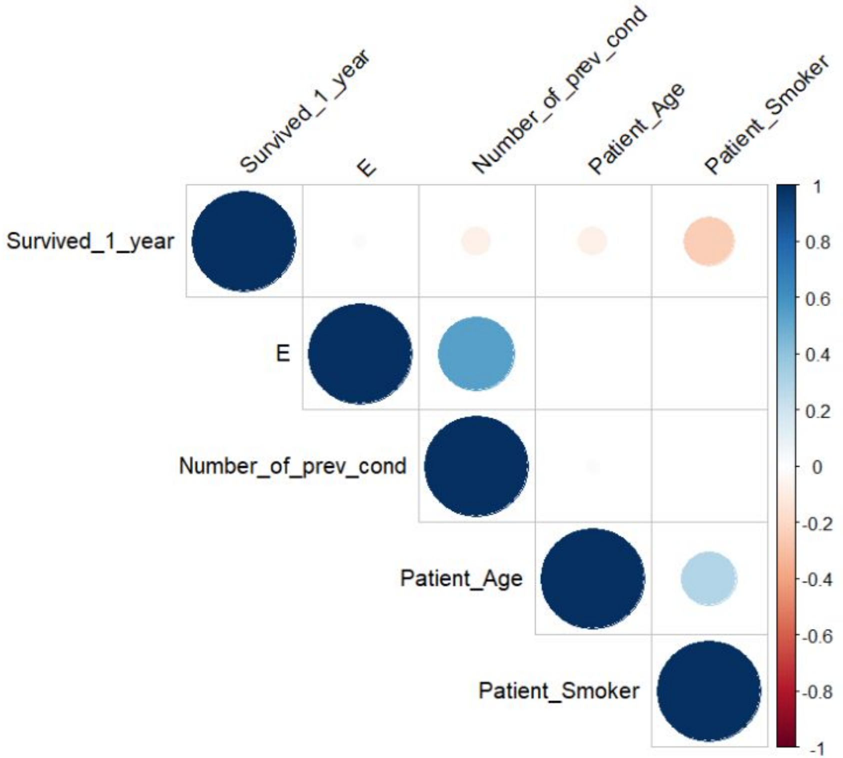
PCA


Variables - PCA
Patient_Smoker
Patient_Age
Patient_Body_Mass_Index
Patient_Rural_Urban
ID_Patient_Care_Situation
Diagnosed_Condition
A
B
C
D
E
F
Number_of_prev_
Treated_with_drugs
Survived_1_year
contrib
20
10
Dim1 (12.6%)
Dim2 (9.8%)
-1.0
-0.5
0.0
0.5
1.0


Dim.1
Dim.2
Dim.3
Dim.4
Dim.5
ID_Patient_Care_Situation
Diagnosed_Condition
Treated_with_drugs
Patient_Age
Patient_Body_Mass_Index
Patient_Smoker
Patient_Rural_Urban
A
B
C
D
E
F
Number_of_prev_cond
Survived_1_year
0.95
0.85
0.76
0.66
0.57
0.47
0.38
0.28
0.19
0.09
0

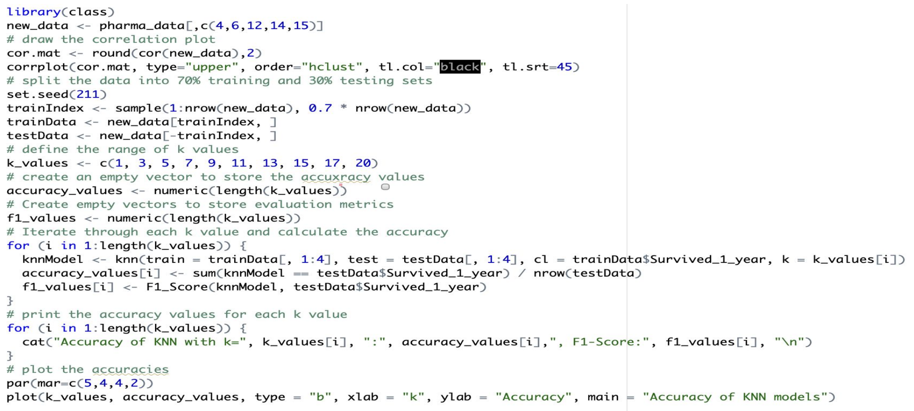
VARIABLE CORRELATION



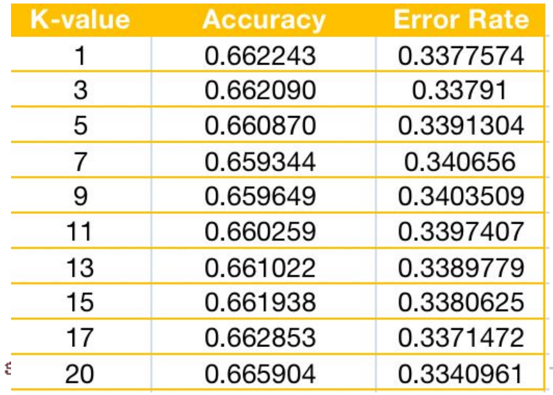


KNN METHOD

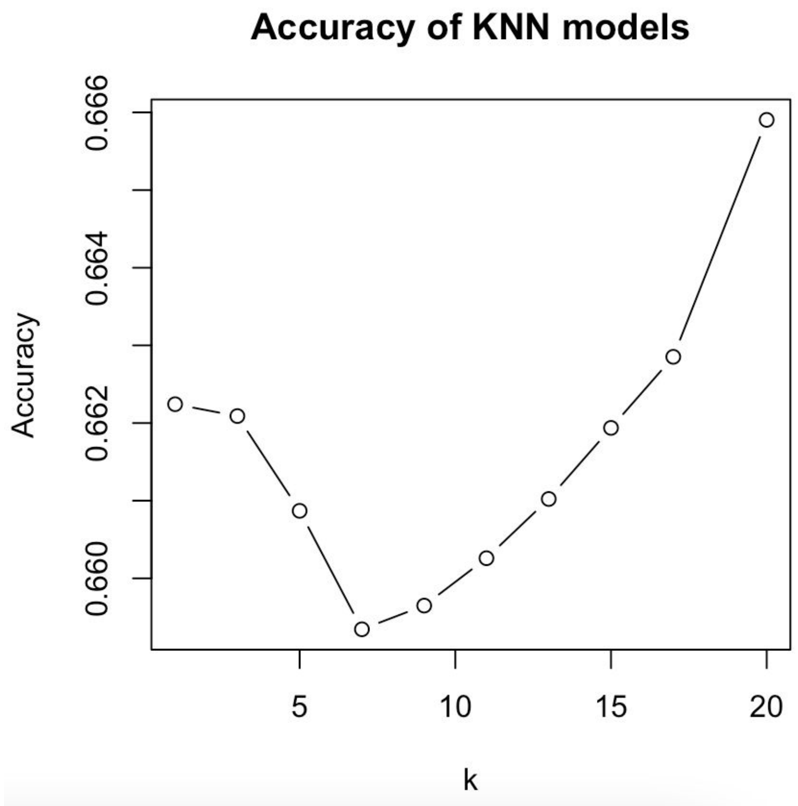
Accuracy: ~65.65%



```
library(class)
new_data <- pharma_data[,c(4,6,12,14,15)]
# draw the correlation plot
cor.mat <- round(cor(new_data),2)
corrplot(cor.mat, type="upper", order="hclust", tl.col="black", tl.srt=45)
# split the data into 70% training and 30% testing sets
set.seed(211)
trainIndex <- sample(1:nrow(new_data), 0.7 * nrow(new_data))
trainData <- new_data[trainIndex, ]
testData <- new_data[-trainIndex, ]
# define the range of k values
k_values <- c(1, 3, 5, 7, 9, 11, 13, 15, 17, 20)
# create an empty vector to store the accuxracy values
accuracy_values <- numeric(length(k_values))
# Create empty vectors to store evaluation metrics
f1_values <- numeric(length(k_values))
# Iterate through each k value and calculate the accuracy
for (i in 1:length(k_values)) {
  knnModel <- knn(train = trainData[, 1:4], test = testData[, 1:4], cl = trainData$Survived_1_year, k = k_values[i])
  accuracy_values[i] <- sum(knnModel == testData$Survived_1_year) / nrow(testData)
  f1_values[i] <- F1_Score(knnModel, testData$Survived_1_year)
}
# print the accuracy values for each k value
for (i in 1:length(k_values)) {
  cat("Accuracy of KNN with k=", k_values[i], ":", accuracy_values[i],", F1-Score:", f1_values[i], "\n")
}
# plot the accuracies
par(mar=c(5,4,4,2))
plot(k_values, accuracy_values, type = "b", xlab = "k", ylab = "Accuracy", main = "Accuracy of KNN models")
```

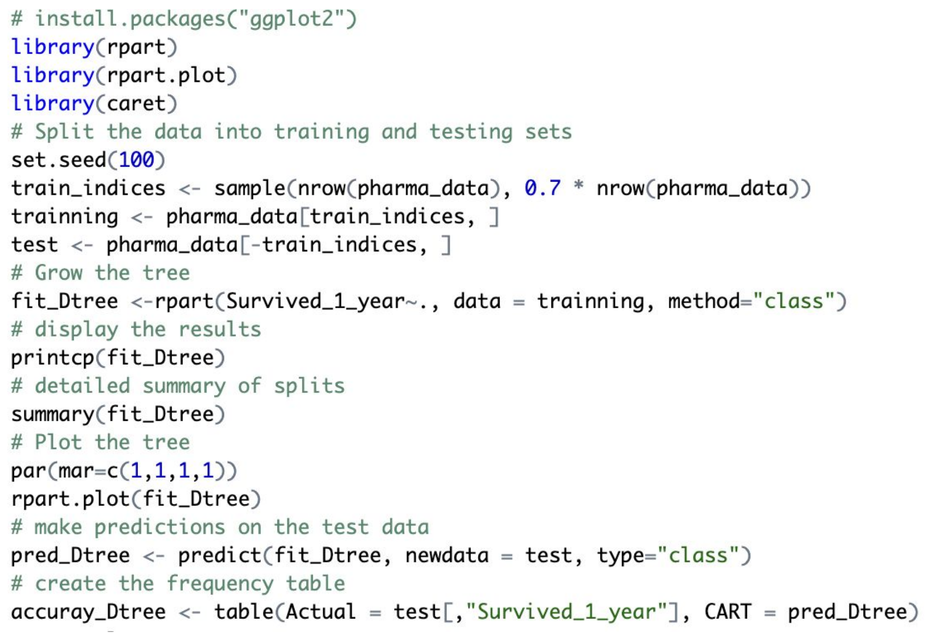


K-value	Accuracy	Error Rate
1	0.662243	0.3377574
3	0.662090	0.33791
5	0.660870	0.3391304
7	0.659344	0.340656
9	0.659649	0.3403509
11	0.660259	0.3397407
13	0.661022	0.3389779
15	0.661938	0.3380625
17	0.662853	0.3371472
20	0.665904	0.3340961

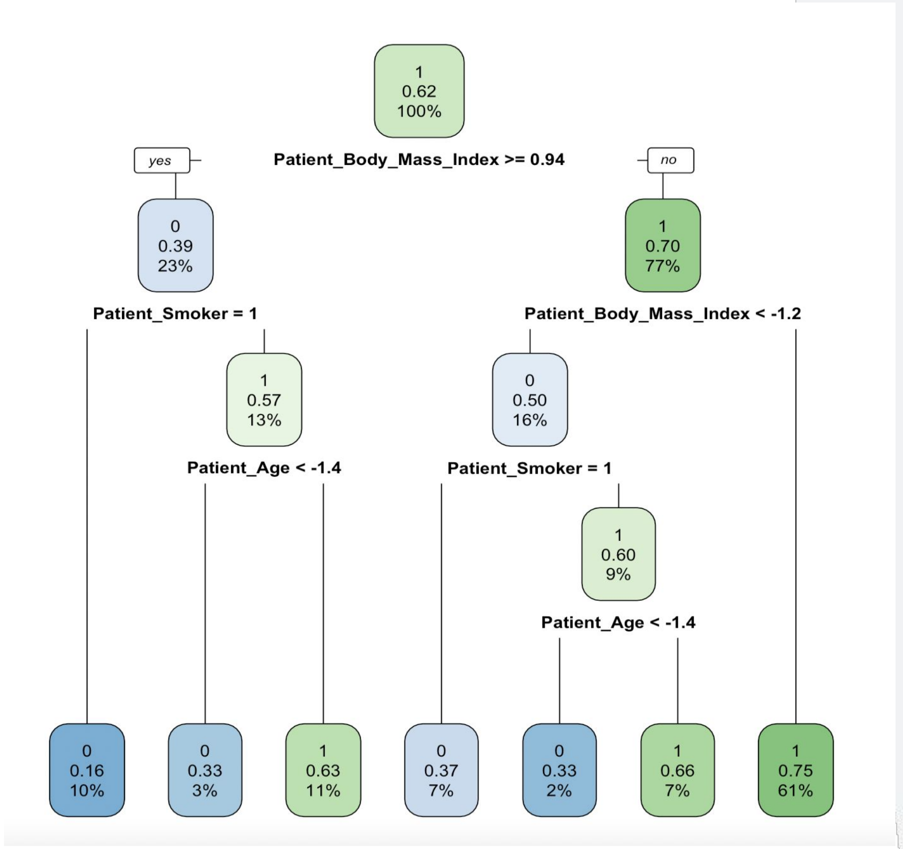


CART DECISION TREE

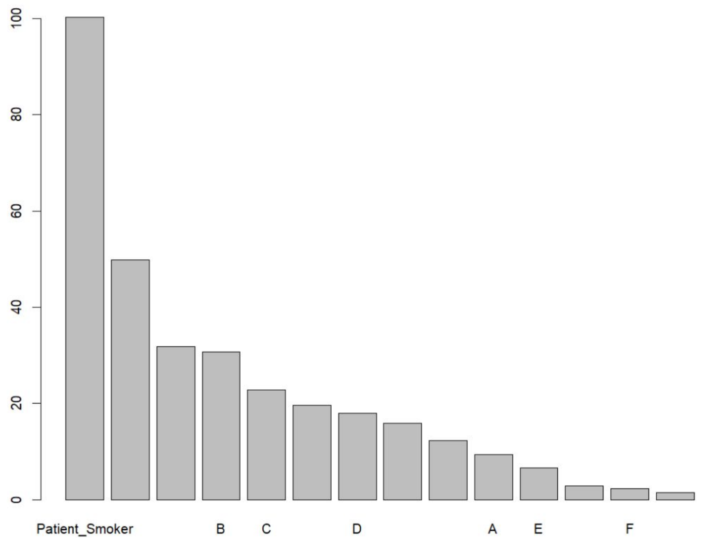
Accuracy: ~71.1%

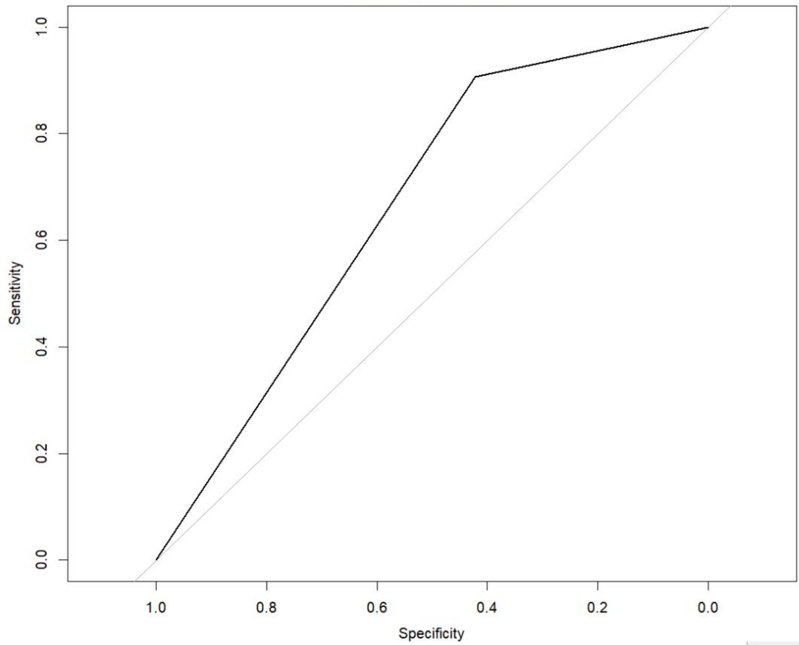


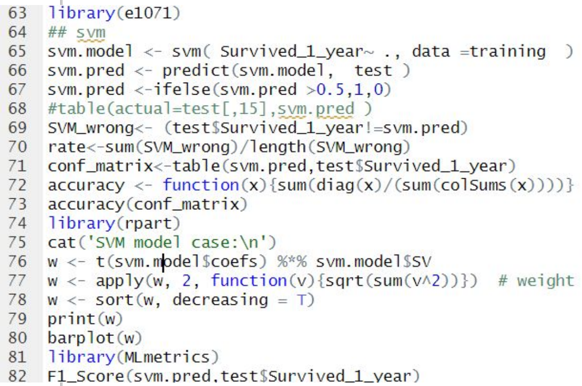
```
# install.packages("ggplot2")
library(rpart)
library(rpart.plot)
library(caret)
# Split the data into training and testing sets
set.seed(100)
train_indices <- sample(nrow(pharma_data), 0.7 * nrow(pharma_data))
trainning <- pharma_data[train_indices, ]
test <- pharma_data[-train_indices, ]
# Grow the tree
fit_Dtree <-rpart(Survived_1_year~., data = trainning, method="class")
# display the results
printcp(fit_Dtree)
# detailed summary of splits
summary(fit_Dtree)
# Plot the tree
par(mar=c(1,1,1,1))
rpart.plot(fit_Dtree)
# make predictions on the test data
pred_Dtree <- predict(fit_Dtree, newdata = test, type="class")
# create the frequency table
accuray_Dtree <- table(Actual = test[,"Survived_1_year"], CART = pred_Dtree)
```



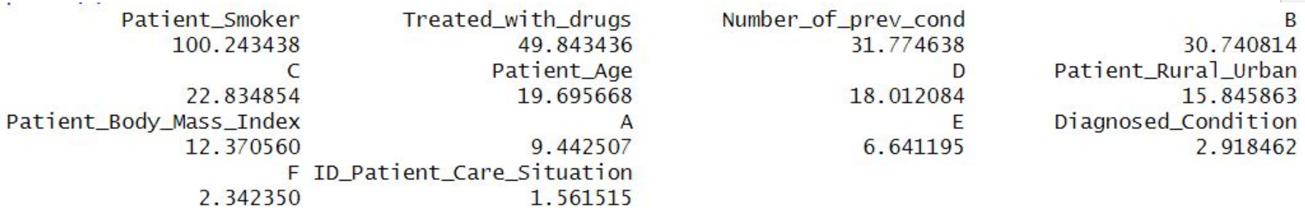
SVM







```r
library(e1071)
## svm
svm.model <- svm( Survived_1_year~ ., data =training  )
svm.pred <- predict(svm.model,  test )
svm.pred <-ifelse(svm.pred >0.5,1,0)
#table(actual=test[,15],svm.pred )
SVM_wrong<- (test$Survived_1_year!=svm.pred)
rate<-sum(SVM_wrong)/length(SVM_wrong)
conf_matrix<-table(svm.pred,test$Survived_1_year)
accuracy <- function(x){sum(diag(x)/(sum(colSums(x))))}
accuracy(conf_matrix)
library(rpart)
cat('SVM model case:\n')
w <- t(svm.model$coefs) %*% svm.model$SV
w <- apply(w, 2, function(v){sqrt(sum(v^2))})  # weight
w <- sort(w, decreasing = T)
print(w)
barplot(w)
library(MLmetrics)
F1_Score(svm.pred,test$Survived_1_year)
```



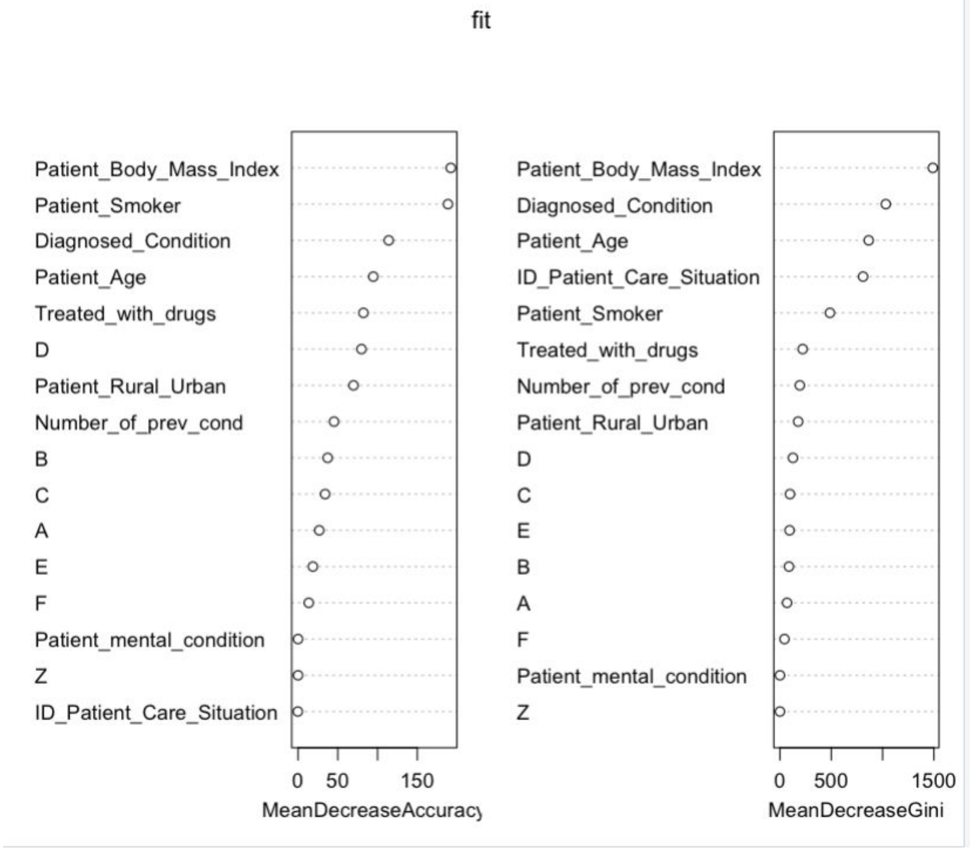
```
           Patient_Smoker        Treated_with_drugs        Number_of_prev_cond                         B
               100.243438                 49.843436                  31.774638                 30.740814
                        C               Patient_Age                          D        Patient_Rural_Urban
                22.834854                 19.695668                  18.012084                 15.845863
  Patient_Body_Mass_Index                         A                          E        Diagnosed_Condition
                12.370560                  9.442507                   6.641195                  2.918462
                        F ID_Patient_Care_Situation
                 2.342350                  1.561515
```

Random Forest

Accuracy: ~78.01%

- PaMent_Body_Mass_Index: This variable has a high MeanDecreaseAccuracy, indicating that it is an important predictor of survival outcome. Its MeanDecreaseGini value is also the highest among all variables, suggesting that it is a very effective at reducing impurity in the decision tree.

"Diagnosed_CondiMon" has a high MeanDecreaseAccuracy value, indicating that it is an important predictor of survival outcome. Its MeanDecreaseGini is also relatively high compared to some other variables, suggesting that it is effective at reducing impurity in the decision tree.

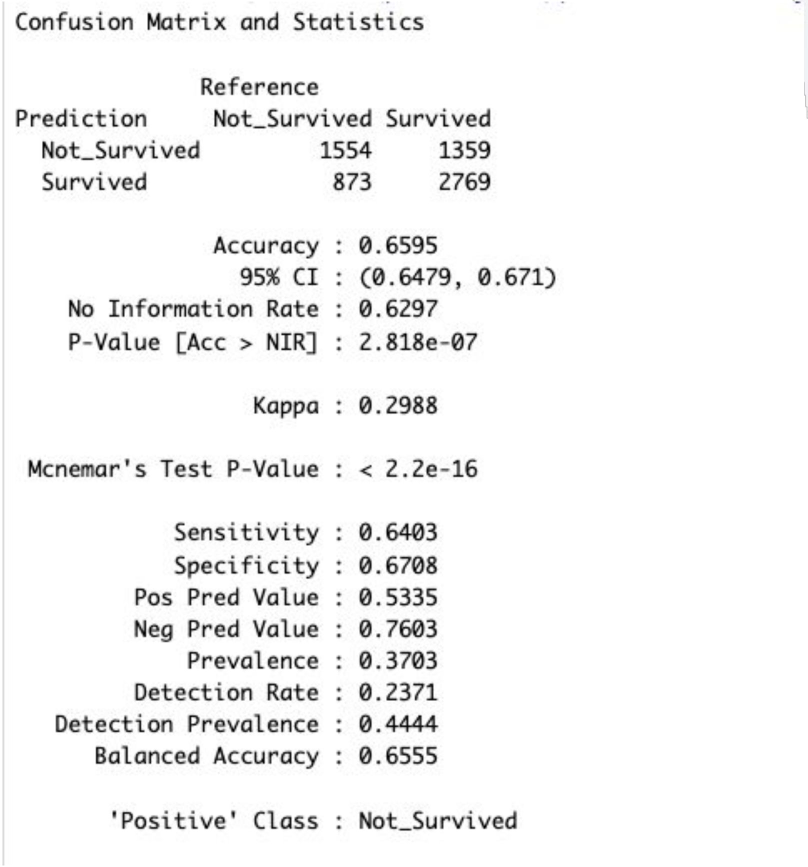


Logistic Regression

Accuracy: ~65.9%

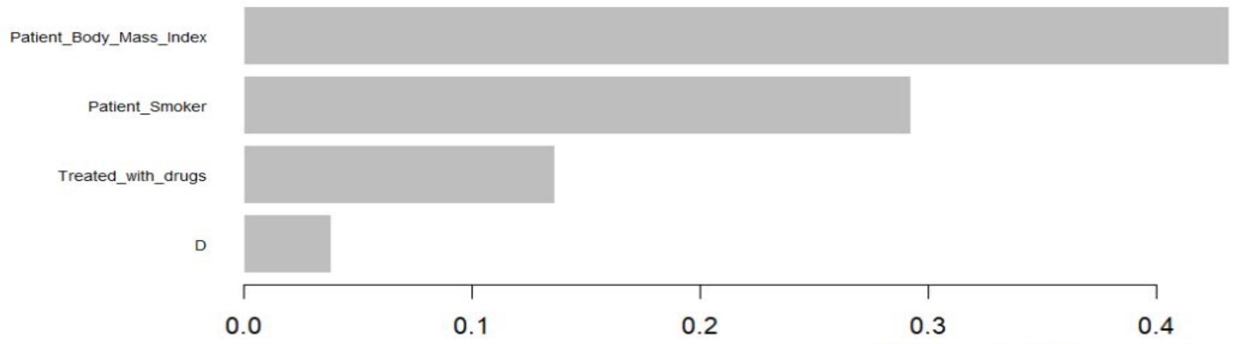
1. Accuracy: The proportion of correct predictions (both true positives and true negatives) out of the total predictions. In this case, the accuracy is 0.6595, meaning the model correctly predicted the survival outcome for about 65.95% of the passengers.

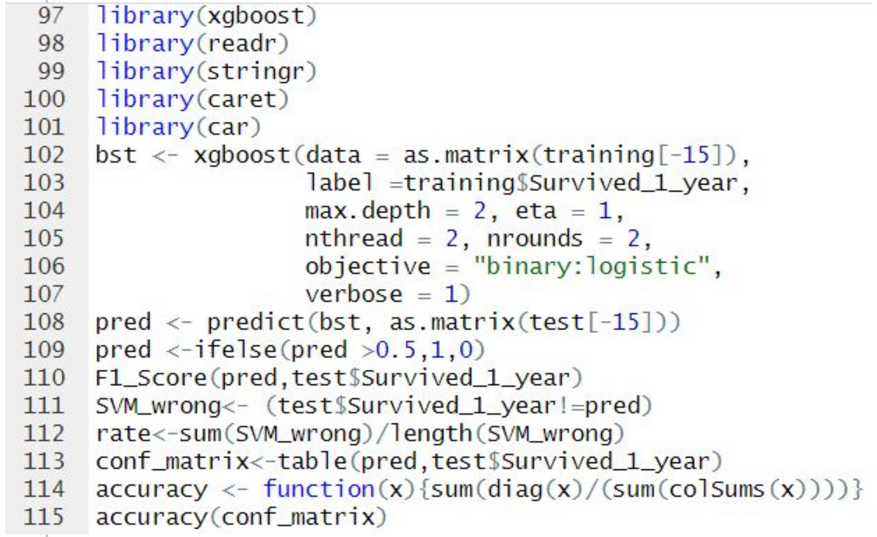
2. A p-value of 2.818e-07 suggests that the model is significantly better than just predicting the majority class.



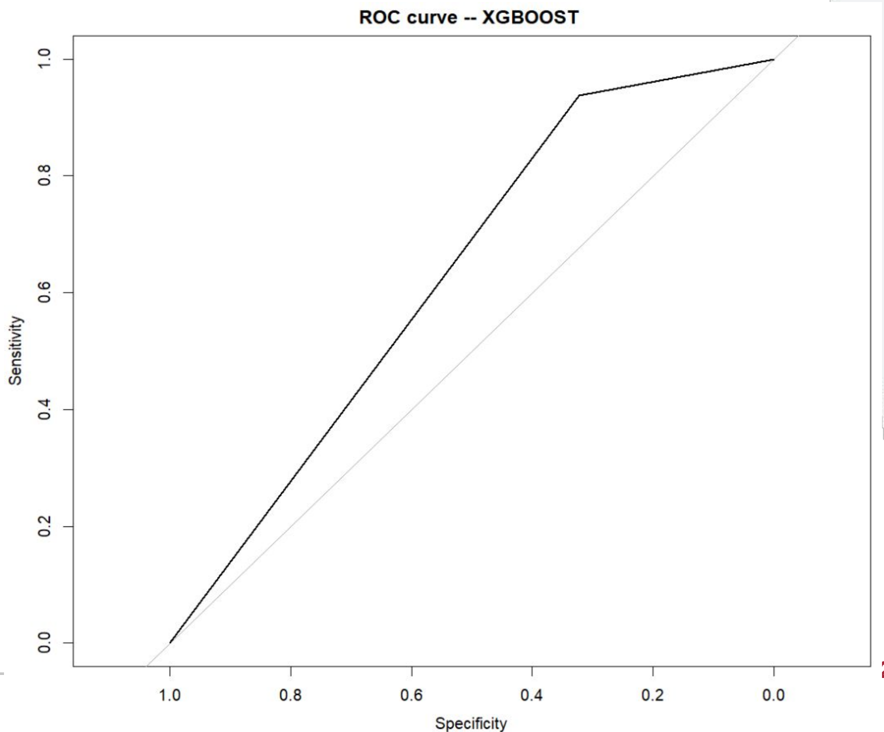
```
Confusion Matrix and Statistics

              Reference
Prediction     Not_Survived Survived
  Not_Survived         1554     1359
  Survived              873     2769

               Accuracy : 0.6595
                 95% CI : (0.6479, 0.671)
    No Information Rate : 0.6297
    P-Value [Acc > NIR] : 2.818e-07

                  Kappa : 0.2988

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.6403
            Specificity : 0.6708
         Pos Pred Value : 0.5335
         Neg Pred Value : 0.7603
             Prevalence : 0.3703
         Detection Rate : 0.2371
   Detection Prevalence : 0.4444
      Balanced Accuracy : 0.6555

       'Positive' Class : Not_Survived
```

XGBOOST



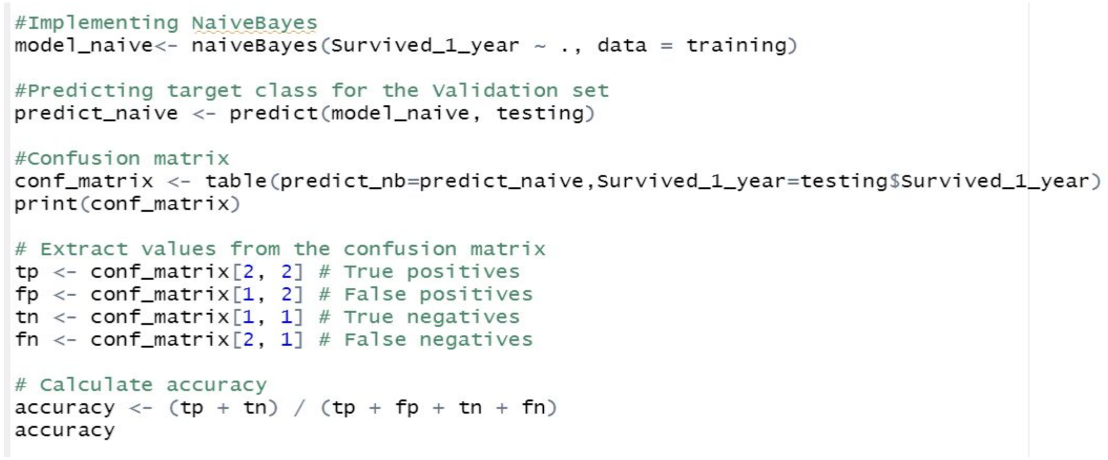


```
library(xgboost)
library(readr)
library(stringr)
library(caret)
library(car)
bst <- xgboost(data = as.matrix(training[-15]),
               label =training$Survived_1_year,
               max.depth = 2, eta = 1,
               nthread = 2, nrounds = 2,
               objective = "binary:logistic",
               verbose = 1)
pred <- predict(bst, as.matrix(test[-15]))
pred <-ifelse(pred >0.5,1,0)
F1_Score(pred,test$Survived_1_year)
SVM_wrong<- (test$Survived_1_year!=pred)
rate<-sum(SVM_wrong)/length(SVM_wrong)
conf_matrix<-table(pred,test$Survived_1_year)
accuracy <- function(x){sum(diag(x)/(sum(colSums(x))))}
accuracy(conf_matrix)
```

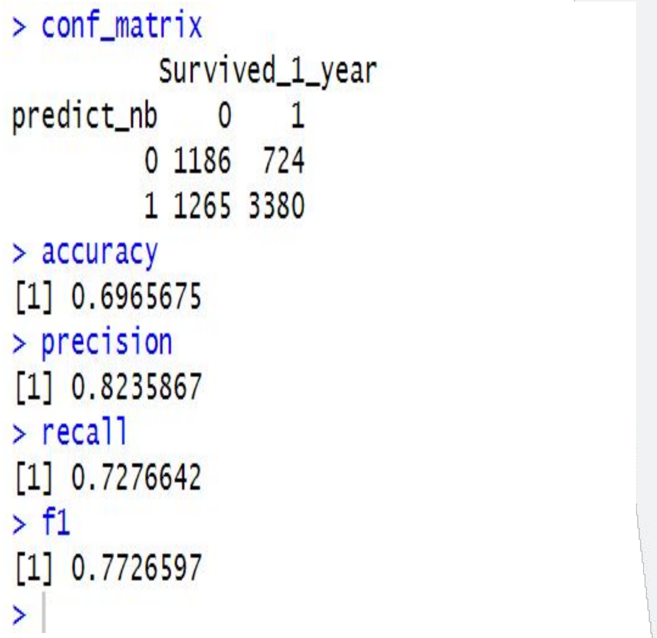


Naïve Bayes

Accuracy: ~69.66%



```
#Implementing NaiveBayes
model_naive<- naiveBayes(Survived_1_year ~ ., data = training)

#Predicting target class for the Validation set
predict_naive <- predict(model_naive, testing)

#Confusion matrix
conf_matrix <- table(predict_nb=predict_naive,Survived_1_year=testing$Survived_1_year)
print(conf_matrix)

# Extract values from the confusion matrix
tp <- conf_matrix[2, 2] # True positives
fp <- conf_matrix[1, 2] # False positives
tn <- conf_matrix[1, 1] # True negatives
fn <- conf_matrix[2, 1] # False negatives

# Calculate accuracy
accuracy <- (tp + tn) / (tp + fp + tn + fn)
accuracy
```

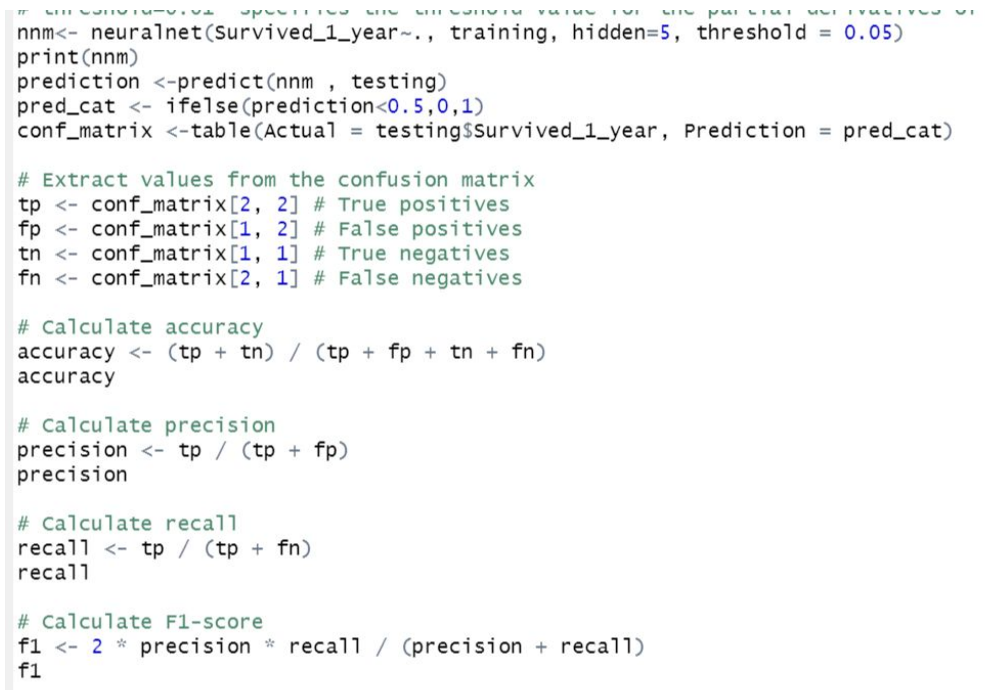


```
> conf_matrix
          Survived_1_year
predict_nb    0    1
         0 1186  724
         1 1265 3380
> accuracy
[1] 0.6965675
> precision
[1] 0.8235867
> recall
[1] 0.7276642
> f1
[1] 0.7726597
>
```

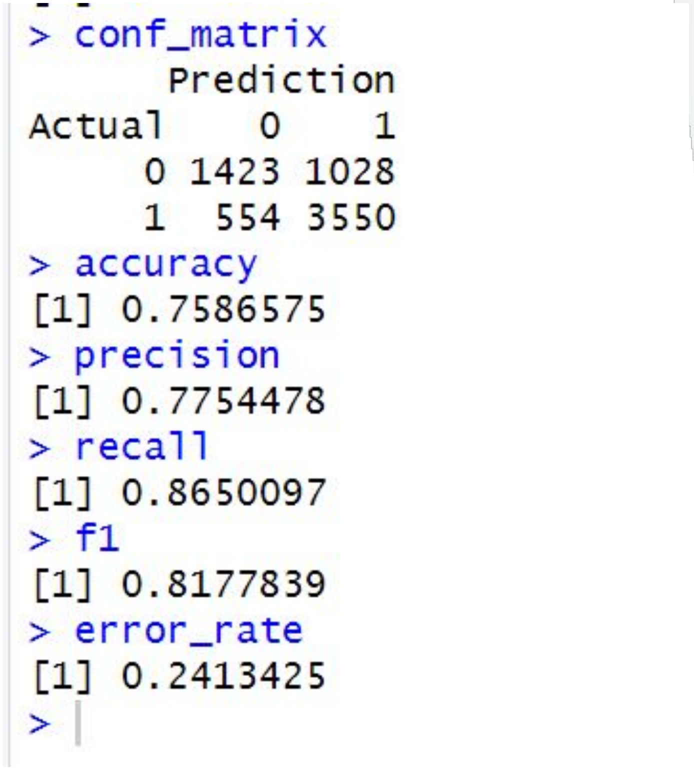
Independent Fast

ANN

Accuracy: ~75.86%



```
# threshold=0.01" specifies the threshold value for the partial derivatives of
nnm<- neuralnet(Survived_1_year~., training, hidden=5, threshold = 0.05)
print(nnm)
prediction <-predict(nnm , testing)
pred_cat <- ifelse(prediction<0.5,0,1)
conf_matrix <-table(Actual = testing$Survived_1_year, Prediction = pred_cat)

# Extract values from the confusion matrix
tp <- conf_matrix[2, 2] # True positives
fp <- conf_matrix[1, 2] # False positives
tn <- conf_matrix[1, 1] # True negatives
fn <- conf_matrix[2, 1] # False negatives

# Calculate accuracy
accuracy <- (tp + tn) / (tp + fp + tn + fn)
accuracy

# Calculate precision
precision <- tp / (tp + fp)
precision

# Calculate recall
recall <- tp / (tp + fn)
recall

# Calculate F1-score
f1 <- 2 * precision * recall / (precision + recall)
f1
```



```
> conf_matrix
      Prediction
Actual    0    1
     0 1423 1028
     1  554 3550
> accuracy
[1] 0.7586575
> precision
[1] 0.7754478
> recall
[1] 0.8650097
> f1
[1] 0.8177839
> error_rate
[1] 0.2413425
>
```

Results and Evaluation

Model	F1 SCORE	ERROR RATE	ACCURACY
SVM	0.5500382	26.96%	73.04%
XGBOOST	0.4560408	29.26%	70.74%
KNN	0.4475277	33.41%	66.59%
CART Decision Tree	0.5044433	28.92%	71.08%
Naïve Bayes	0.7726597	30.34%	69.66%
ANN	0.8177839	24.13%	75.87%
Random Forest	0.834577	21.98%	78.01%
Logistic Regression	0.7098613	34.05%	65.95%

Conclusion

- Patient Survived_1_year is most closely linked to prediction of patient survival after one year .
- The main attributes affecting whether a patient survives after one year are:Patient_Smoker,Number_of_previous_conditions,Patient_Age and E
- The Random Forest method yielded the highest accuracy for the prediction of patient survival after one year followed closely by the ANN and SVM. This may relate to their ability to resist noises.
- Algorithms faster than others（Based on this database）:Logistic Regression, Naïve Bayes, CART Decision Tree, KNN and XGBoost.

STEVENS
INSTITUTE OF TECHNOLOGY
1870

THANK YOU

Stevens Institute of Technology
1 Castle Point Terrace, Hoboken, NJ 07030