Titanic Machine Learning with R

Lucas Torres Valiente

Contents

The following notebook is intended to solve the problem Titanic - Machine Learning from Disaster by Kaggle.

The main objective is Predict survival on the Titanic and get familiar with ML basics.

Note: Algorithms definitions are from Wikipedia

Introduction

```
#Necessary Packages

library(caret)
library(rpart)
library(rpart.plot)
library(ROCR)
library(randomForest)
library(e1071)
library(hrbrthemes)
library(ggplot2)
library(dplyr)
library(class)
library(kernlab)
library(extrafont)

#Seed

set.seed(2021)
```

Data importation

```
train <- read.csv("C:/Users/PC/Desktop/PROG/Machine Learning/Titanic/train.csv",
                  na.strings = "")
test <- read.csv("C:/Users/PC/Desktop/PROG/Machine Learning/Titanic/test.csv",
                 na.strings = "")
gender_submission <-
  read.csv("C:/Users/PC/Desktop/PROG/Machine Learning/Titanic/gender_submission.csv")
```

```
library(knitr)
opts_chunk$set(tidy.opts=list(width.cutoff = 45),tidy = TRUE)
```

Exploratory Data Analysis and Data transformation

```
head(train)
```

```
##   PassengerId Survived Pclass
## 1           1        0      3
## 2           2        1      1
## 3           3        1      3
## 4           4        1      1
## 5           5        0      3
## 6           6        0      3
##                                                 Name    Sex Age SibSp Parch
## 1                            Braund, Mr. Owen Harris   male  22     1     0
## 2 Cumings, Mrs. John Bradley (Florence Briggs Thayer) female  38     1     0
## 3                             Heikkinen, Miss. Laina female  26     0     0
## 4       Futrelle, Mrs. Jacques Heath (Lily May Peel) female  35     1     0
## 5                           Allen, Mr. William Henry   male  35     0     0
## 6                                   Moran, Mr. James   male  NA     0     0
##             Ticket    Fare Cabin Embarked
## 1        A/5 21171  7.2500  <NA>        S
```

```
## 2         PC 17599 71.2833   C85        C
## 3 STON/O2. 3101282  7.9250  <NA>        S
## 4           113803 53.1000  C123        S
## 5           373450  8.0500  <NA>        S
## 6           330877  8.4583  <NA>        Q
```

Table 1: Data Dictionary

Variable	Definition	Key
survival	Survival	0 = No, 1 = Yes
pclass	Ticket class	1 = 1st, 2 = 2nd, 3 = 3rd
sex	Sex	
Age	Age in years	
sibsp	# of siblings / spouses aboard the Titanic	
parch	# of parents / children aboard the Titanic	
ticket	Ticket number	
fare	Passenger fare	
cabin	Cabin number	
embarked	Port of Embarkation	C = Cherbourg, Q = Queenstown, S = Southampton

Variable Notes:

- **pclass**: A proxy for socio-economic status (SES)
  1st = Upper
  2nd = Middle
  3rd = Lower
- **age**: Age is fractional if less than 1. If the age is estimated, is it in the form of xx.5
- **sibsp**: The dataset defines family relations in this way...
  Sibling = brother, sister, stepbrother, stepsister
  Spouse = husband, wife (mistresses and fiancés were ignored)
- **parch**: The dataset defines family relations in this way...
  Parent = mother, father
  Child = daughter, son, stepdaughter, stepson
  Some children travelled only with a childcarer, therefore parch=0 for them.

Let's create new variables from the existing ones and edit some variables

```
train$Child <- ifelse(train$Age < 18,1,0)
train$Child <- factor(train$Child)

train$family_size <- train$SibSp + train$Parch + 1

train$Title <- substring(train$Name,regexpr(",",train$Name)+2,regexpr("\\.",train$Name)-1)

train$Title[train$Title %in% c("Capt","Don","Major","Col",
                               "Rev","Dr","Sir","Mr","Jonkheer")] <- "man"
train$Title[train$Title %in% c("Dona","the Countess","Mme","Mlle",
                               "Ms","Miss","Lady","Mrs")] <- "woman"
train$Title[train$Title %in% c("Master")] <- "boy"




test$Child <- ifelse(test$Age < 18,1,0)
test$Child <- factor(test$Child)
```

```
test$family_size <- test$SibSp + test$Parch + 1

test$Title <- substring(test$Name,regexpr(",",test$Name)+2,regexpr("\\.",test$Name)-1)
test$Title[test$Title %in% c("Capt","Don","Major","Col",
                             "Rev","Dr","Sir","Mr","Jonkheer")] <- "man"
test$Title[test$Title %in% c("Dona","the Countess","Mme","Mlle",
                             "Ms","Miss","Lady","Mrs")] <- "woman"
test$Title[test$Title %in% c("Master")] <- "boy"

train$Title <- factor(train$Title)
test$Title <- factor(test$Title)

test$Survived <- 0

train$Survived <- factor(train$Survived)
test$Survived <- factor(test$Survived)
train$Pclass <- factor(train$Pclass)
test$Pclass <- factor(test$Pclass)

test$Sex <- factor(test$Sex)
train$Sex <- factor(train$Sex)

test$Embarked <- factor(test$Embarked)
train$Embarked <- factor(train$Embarked)
```

The new variables are:

- *Child:* Indicates if the passenger is a child or not
- *Family_size:* Indicates the family size, it's the sum of the number of siblings or spouses aboard the Titanic plus the number of parents and children
- *Title:* Indicates the title of the passenger

Let's see how many missings are in the global data set

```
merged_data <- merge(train,test,all = T)
colSums(is.na(merged_data))
```

```
## PassengerId    Survived      Pclass        Name         Sex         Age
##           0           0           0           0           0         263
##       SibSp       Parch      Ticket        Fare       Cabin    Embarked
##           0           0           0           1        1014           2
##       Child family_size       Title
##         263           0           0
```

As we can see, there are a lot off missings in Cabin, Survived and Age, and to lesser extent in fare and embarked.

The Survived missings are are expected, since the column "Survived" does not exist in the test dataset.

- Embarked Variable:

```
table(merged_data$Embarked)
```

```
##
##   C   Q   S
## 270 123 914
```

Since many passengers embarked at Southampton, we assign them the value S.

```
which(is.na(merged_data$Embarked), arr.ind = TRUE)
```

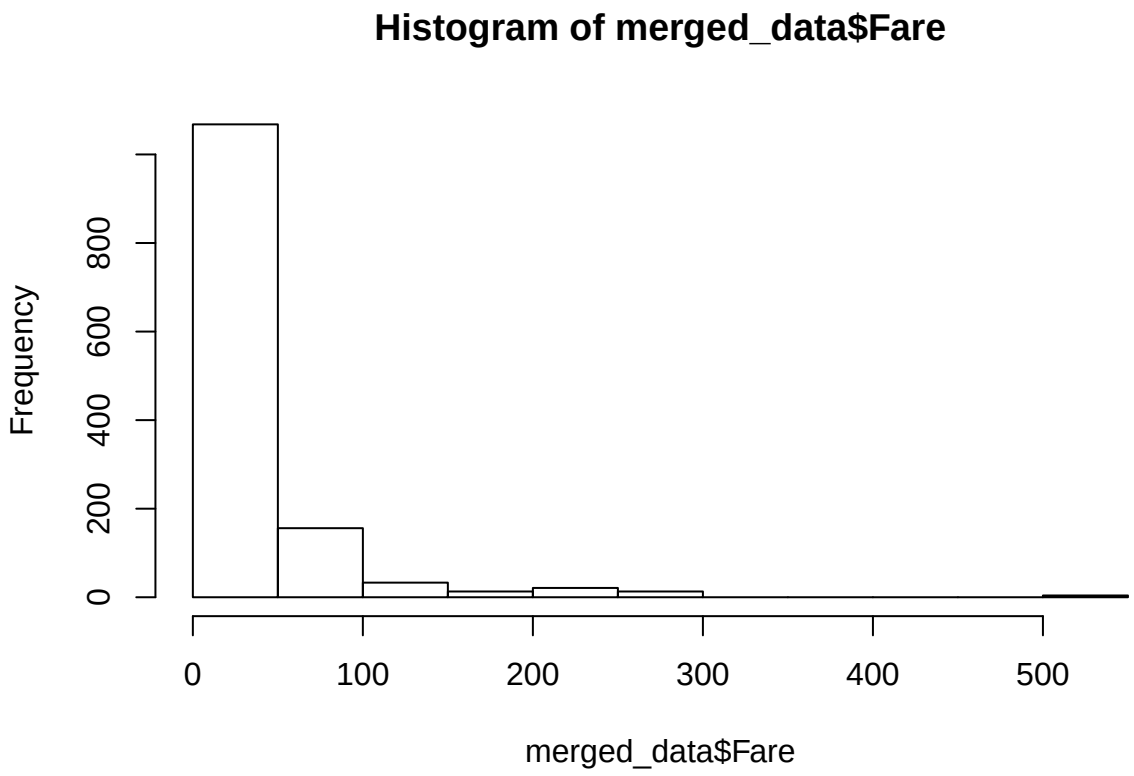
```
## [1]  62 830
```

```
merged_data$Embarked[c(62,830)] <- "S"
```

- Fare Variable:

Let's see the Fare distribution using a simple histogram:

```
hist(merged_data$Fare)
```



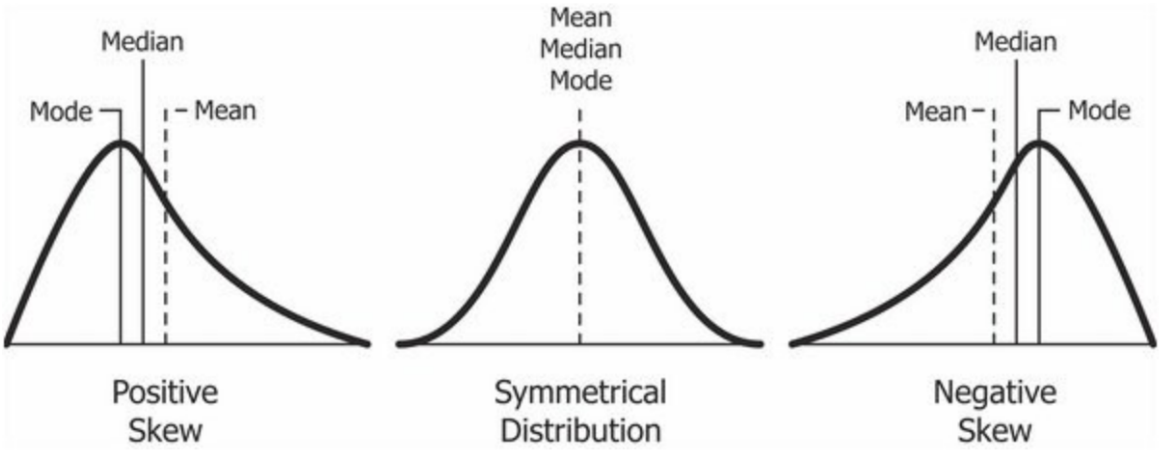
Judging the histogram, the Fare distribution is totally skewed, so the median will be a better center position statistic than the mean.

With Fisher's coefficient of skewness, we can see that the Fare distribution is positive skewed:

$$g_1 = \frac{m_3}{s^3} = \frac{\mathbb{E}[(X-\mu)^3]}{s^3} = \frac{\frac{1}{n}\sum_{i=i}^{n}(x_i - \bar{x})^3}{\frac{1}{n-1}[\sum_{i=i}^{n}(x_i - \bar{x})^2]^{3/2}}$$

```
skewness(merged_data$Fare, na.rm = T, type = 3)
```

```
## [1] 4.357697
```




```
which(is.na(merged_data$Fare), arr.ind = TRUE)
```

```
## [1] 1044
```

```
merged_data$Fare[1044] <- median(merged_data$Fare, na.rm = T)
```

- Age Variable

There are a lot of Age's missing values. We can make a prediction of a passenger's Age using the other variables and a decision tree model. The method will be "anova" since we are predicting a continuous variable.

```
predicted_age <- rpart(Age ~ Pclass + Sex + SibSp + Parch + Fare + Embarked + Title +
                         family_size + Child,
                         data = merged_data[!is.na(merged_data$Age),], method = "anova")


merged_data$Age[is.na(merged_data$Age)] <-
  predict(predicted_age, merged_data[is.na(merged_data$Age),])
```

Splitting the data back into a train set and a test set

```
train <- merged_data[1:891,]
test <- merged_data[892:1309,]
```

Visualization

Visualization is an important tool to provide insight. Nevertheless, it is rare to get the data in the exact format which is required. Often you'll need to create some new variables or summaries, or maybe you just want to rename the variables or reorder the observations in order to make the data a little easier to work with.

```
table(train$Survived)
```

```
##
##   0   1
## 549 342
```

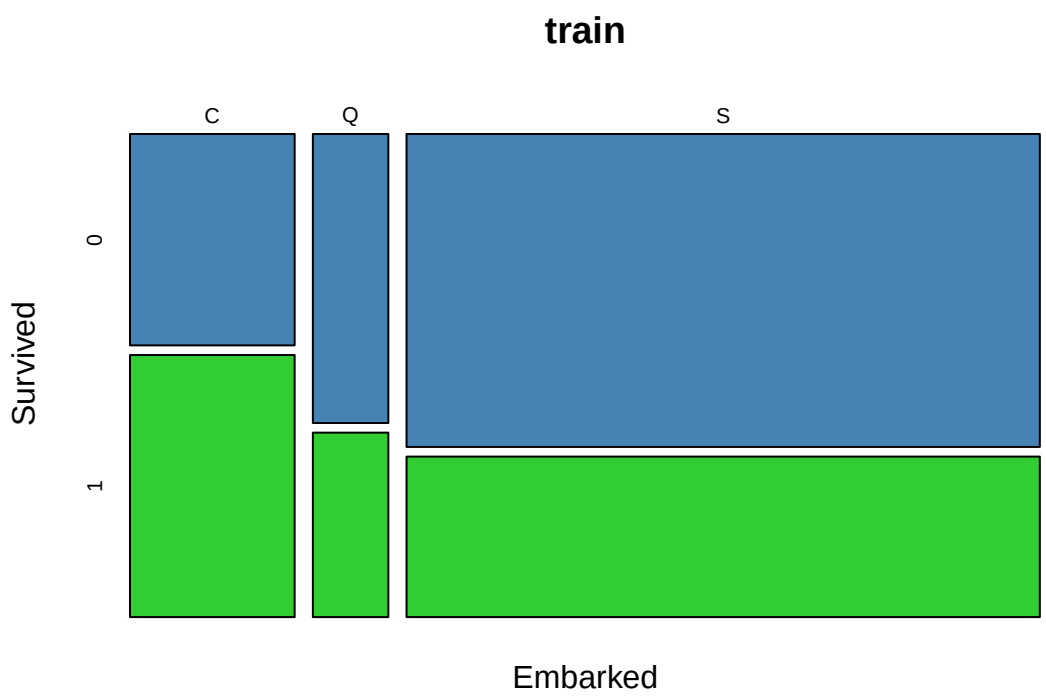
```
round(prop.table(table(train$Survived)),3)
```

```
##
##     0     1
## 0.616 0.384
```

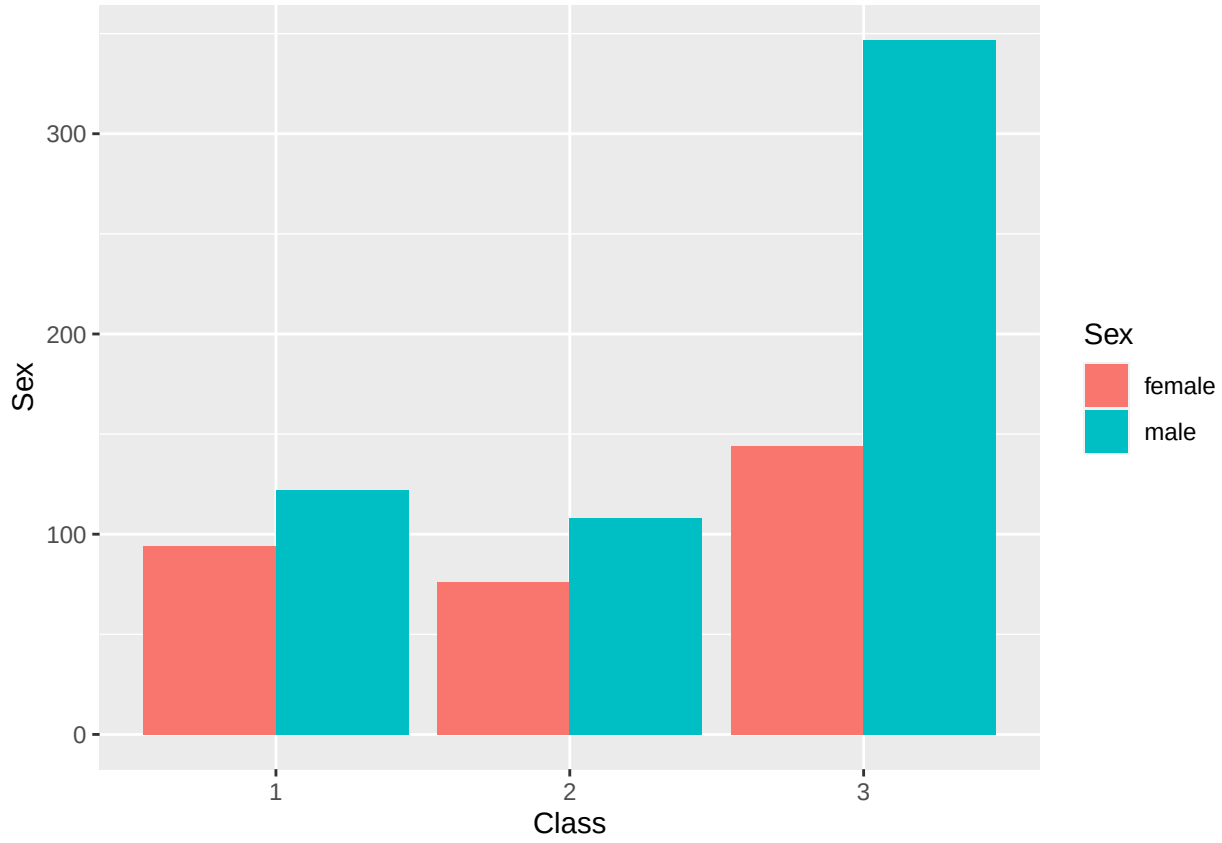
```
table(train$Sex, train$Survived)
```

```
##
##             0   1
##   female   81 233
##   male    468 109
```

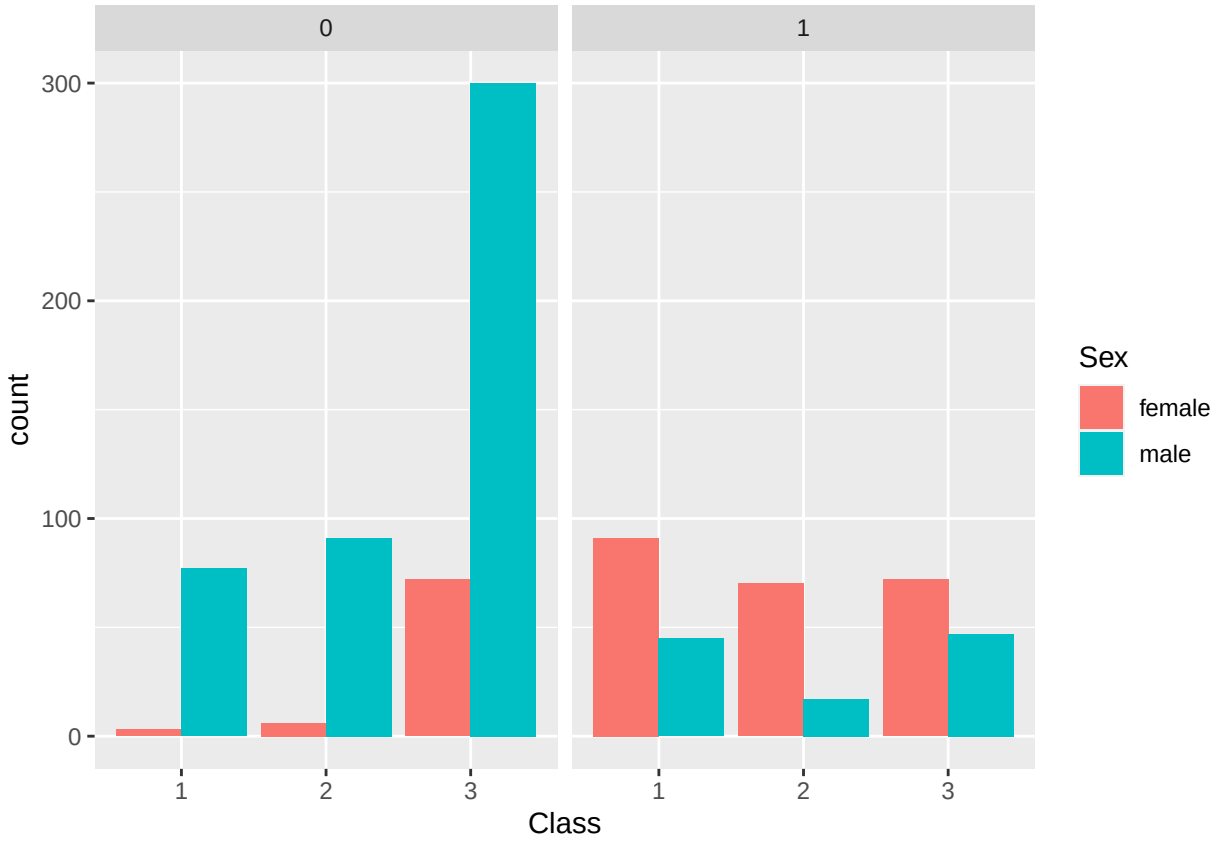
```
mosaicplot(Embarked ~ Survived, data = train, col = c("steelblue","limegreen"))
```



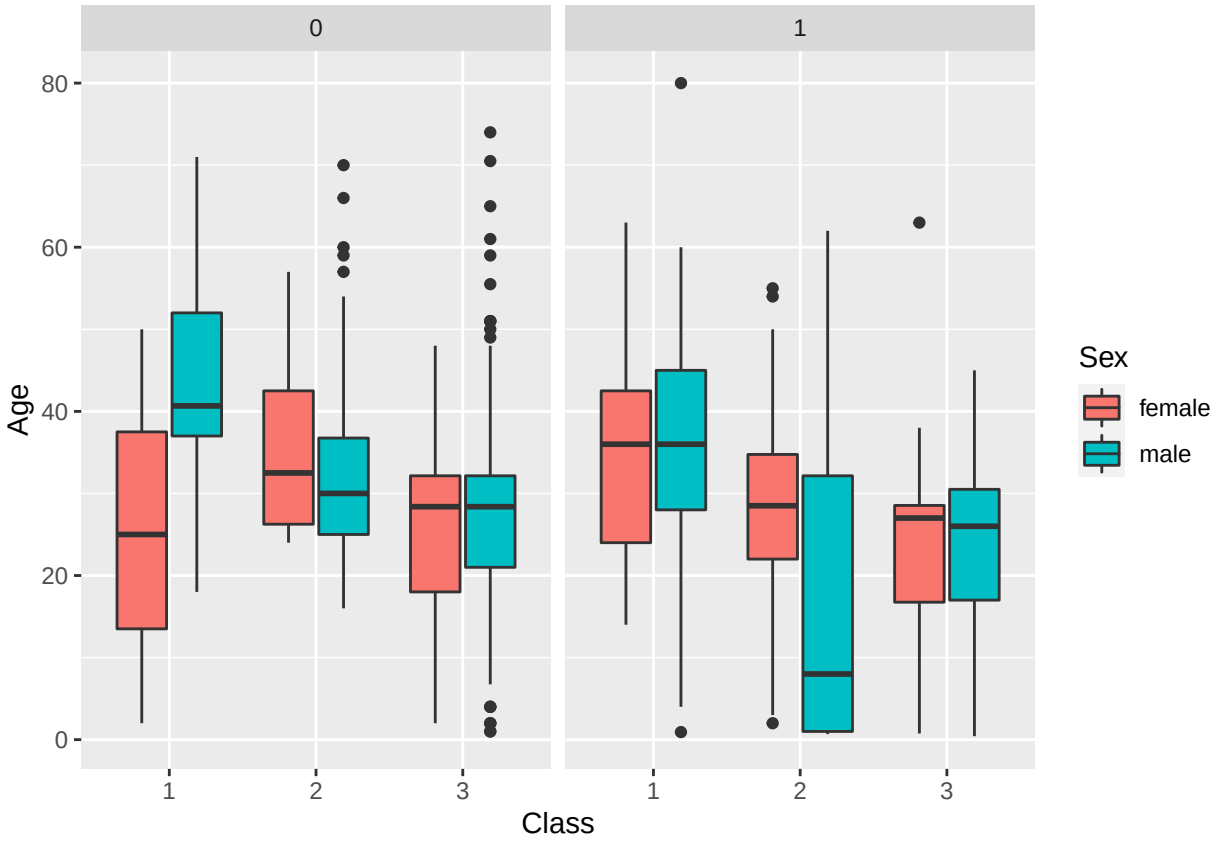
```
ggplot(train,aes(x=factor(Pclass),fill=factor(Sex)))+
  geom_bar(position="dodge") +
  labs(x = "Class", y = "Sex", fill = "Sex")
```



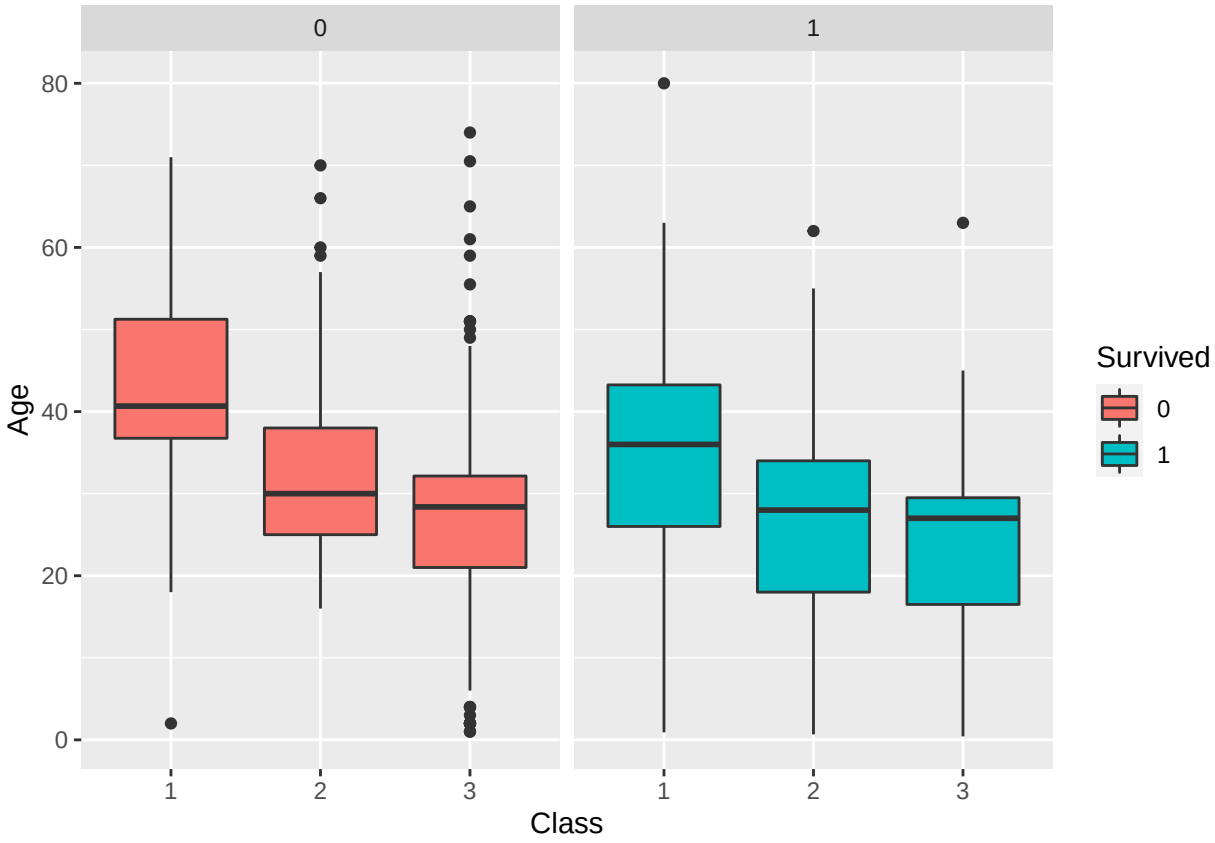
```
ggplot(train,aes(x=factor(Pclass),fill=factor(Sex)))+
  geom_bar(position="dodge")+
  facet_grid(". ~ Survived") +
  labs(x = "Class",fill = "Sex")
```



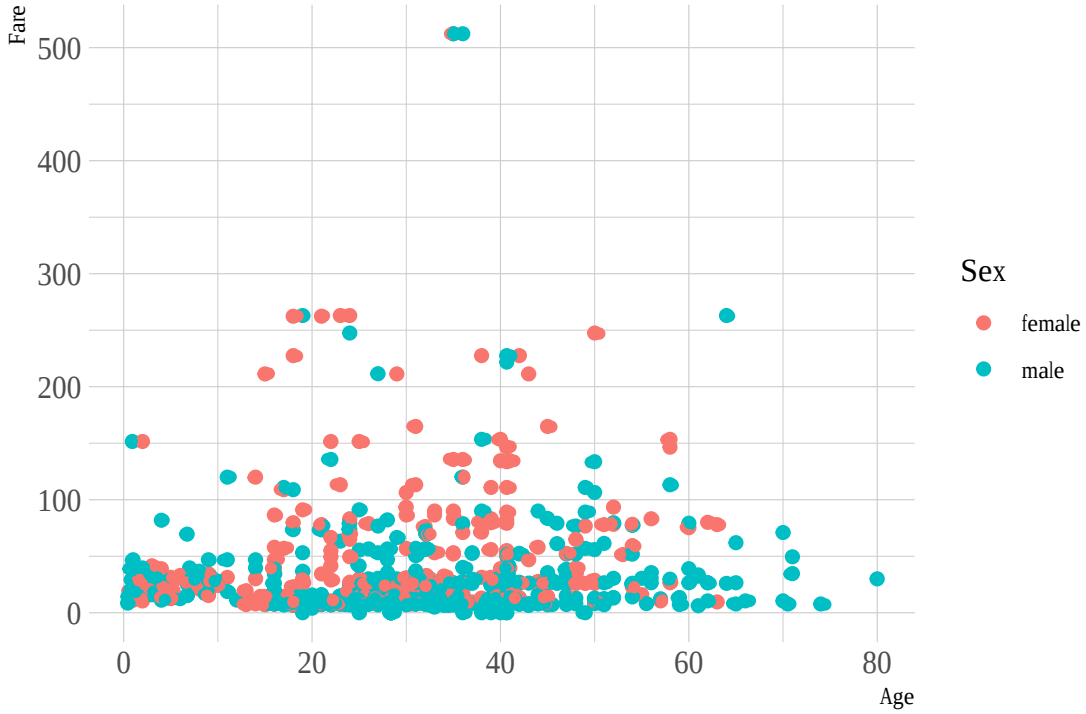
```
ggplot(train,aes(x=factor(Pclass),y=Age, fill = factor(Sex)))+
  geom_boxplot() +
  facet_wrap(~Survived) +
  labs(x = "Class",fill = "Sex")
```



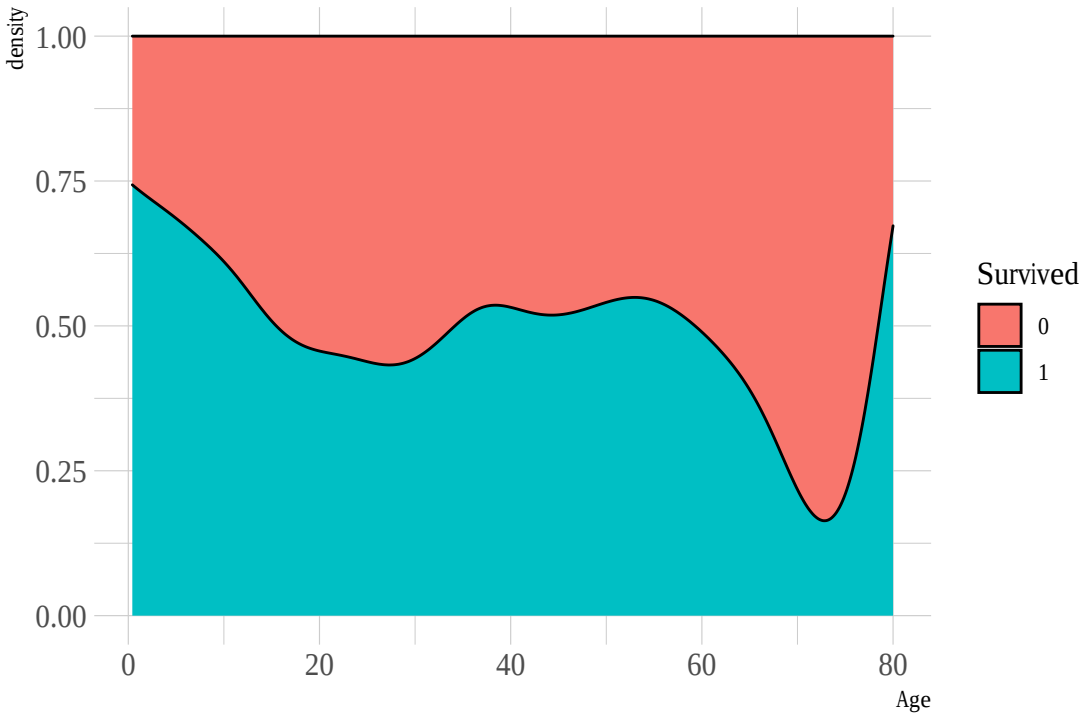
```
ggplot(train, aes(x=factor(Pclass), y=Age, fill= factor(Survived))) +
  geom_boxplot() +
  facet_wrap(~Survived)+
  labs(x = "Class", y = "Age", fill = "Survived")
```



```
loadfonts()
ggplot(train, aes(x=Age, y=Fare, color=Sex)) +
  geom_point(size=2) +
  theme_ipsum() +
  geom_jitter(width = 0.5, height = 0.5)
```



```
ggplot(data=train, aes(x=Age, group=Survived, fill= factor(Survived))) +
  geom_density(adjust=1.5, position="fill") +
  theme_ipsum()+
  labs(fill = "Survived")
```



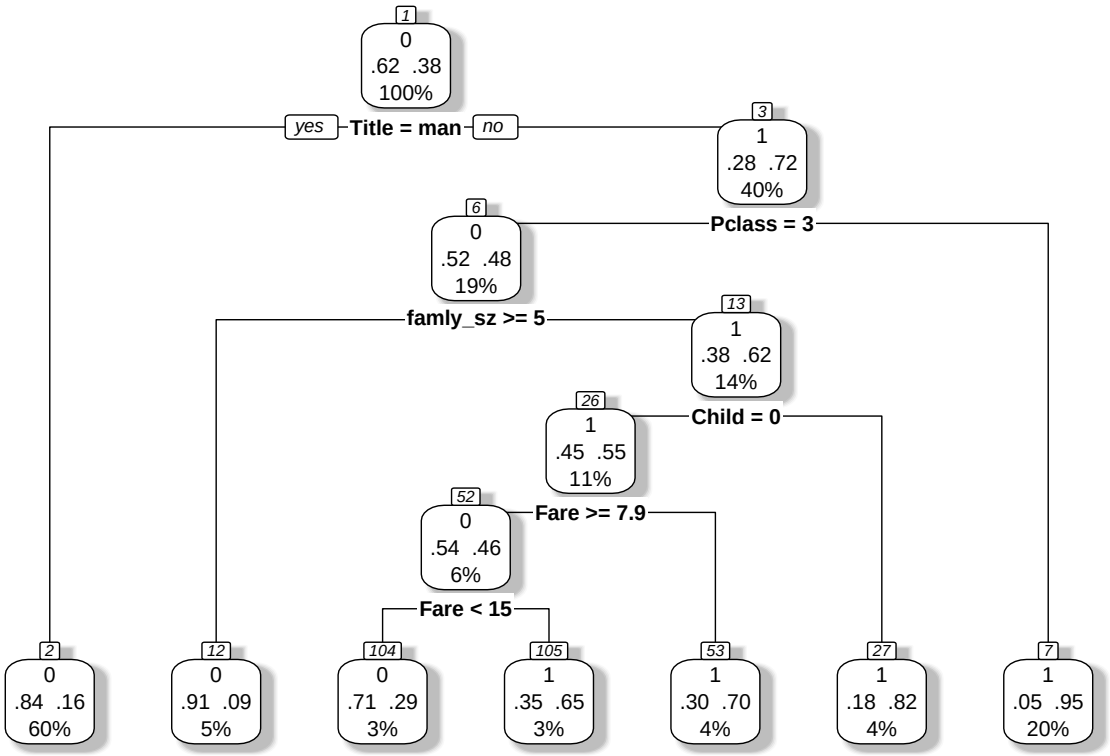
Decision Tree

Decision tree learning is one of the predictive modelling approaches used in statistics, data mining and machine learning. It uses a decision tree (as a predictive model) to go from observations about an item (represented in the branches) to reach conclusions about the item's target value (represented in the leaves). Tree models where the target variable can take a discrete set of values are called classification trees; in these tree structures, leaves represent class labels and branches represent sets of features that lead to aforementioned class labels.

```
DecisionTree <- rpart(Survived  ~ Pclass + Sex + Age + SibSp + Parch + Fare +
                        Embarked + Child + family_size + Title, data = train,
             method = "class",
             control = rpart.control(minsplit = 20,cp=0.01))

DecisionTree
```

```
## n= 891
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##   1) root 891 342 0 (0.61616162 0.38383838)
##     2) Title=man 538  87 0 (0.83828996 0.16171004) *
##     3) Title=boy,woman 353  98 1 (0.27762040 0.72237960)
##       6) Pclass=3 172  83 0 (0.51744186 0.48255814)
##        12) family_size>=4.5 45   4 0 (0.91111111 0.08888889) *
##        13) family_size< 4.5 127  48 1 (0.37795276 0.62204724)
##          26) Child=0 94  42 1 (0.44680851 0.55319149)
##            52) Fare>=7.8875 57  26 0 (0.54385965 0.45614035)
##             104) Fare< 14.8729 31   9 0 (0.70967742 0.29032258) *
##             105) Fare>=14.8729 26   9 1 (0.34615385 0.65384615) *
##            53) Fare< 7.8875 37  11 1 (0.29729730 0.70270270) *
##          27) Child=1 33   6 1 (0.18181818 0.81818182) *
##       7) Pclass=1,2 181   9 1 (0.04972376 0.95027624) *
```

```
prp(DecisionTree, type = 2, extra = 104, nn = TRUE,
    fallen.leaves = TRUE, faclen = 4, varlen = 8,
    shadow.col = "gray")
```

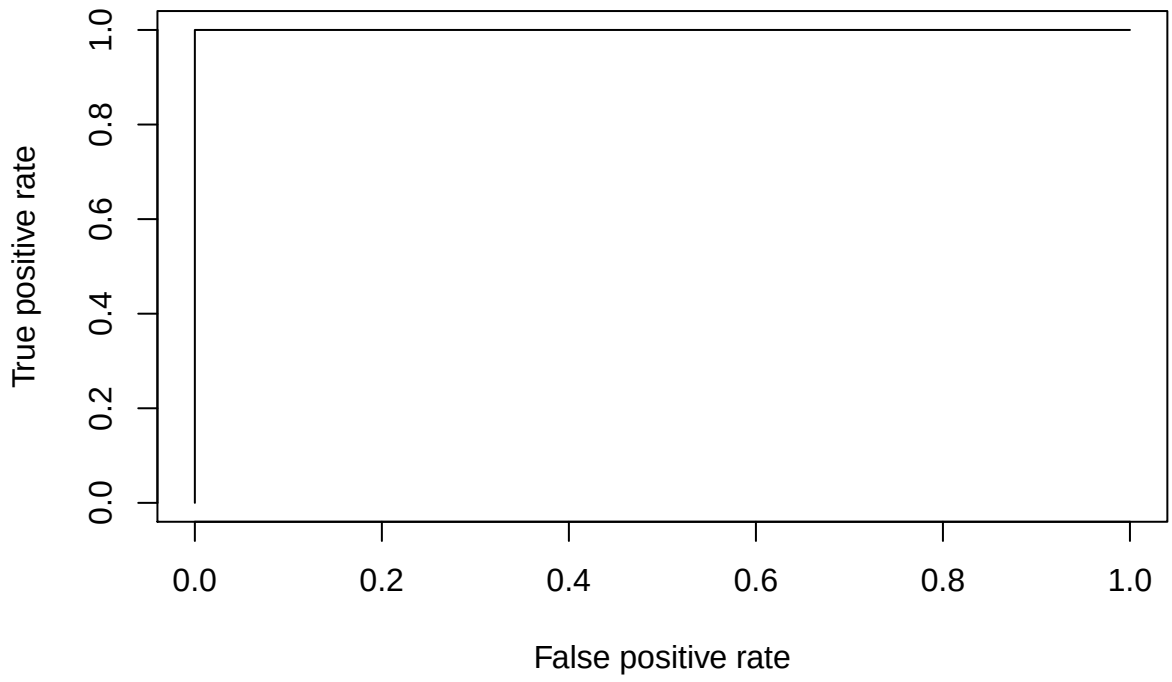

1
0
.62 .38
100%
yes
Title = man
no
3
1
.28 .72
40%
6
0
.52 .48
19%
Pclass = 3
famly_sz >= 5
13
1
.38 .62
14%
26
1
.45 .55
11%
Child = 0
52
0
.54 .46
6%
Fare >= 7.9
Fare < 15
2
0
.84 .16
60%
12
0
.91 .09
5%
104
0
.71 .29
3%
105
1
.35 .65
3%
53
1
.30 .70
4%
27
1
.18 .82
4%
7
1
.05 .95
20%

```
prediction <- predict(DecisionTree, test,
                      type = "class")

prediction_prob <- predict(DecisionTree, test,
                           type = "prob")
```

```
table(gender_submission$Survived, prediction, dnn = c("Actual", "Predicted"))
```

```
##        Predicted
## Actual   0   1
##      0 250  16
##      1  23 129
```

```
pred <- prediction(prediction_prob[,2],prediction)

perf <- performance(pred, "tpr", "fpr")
plot(perf)
```



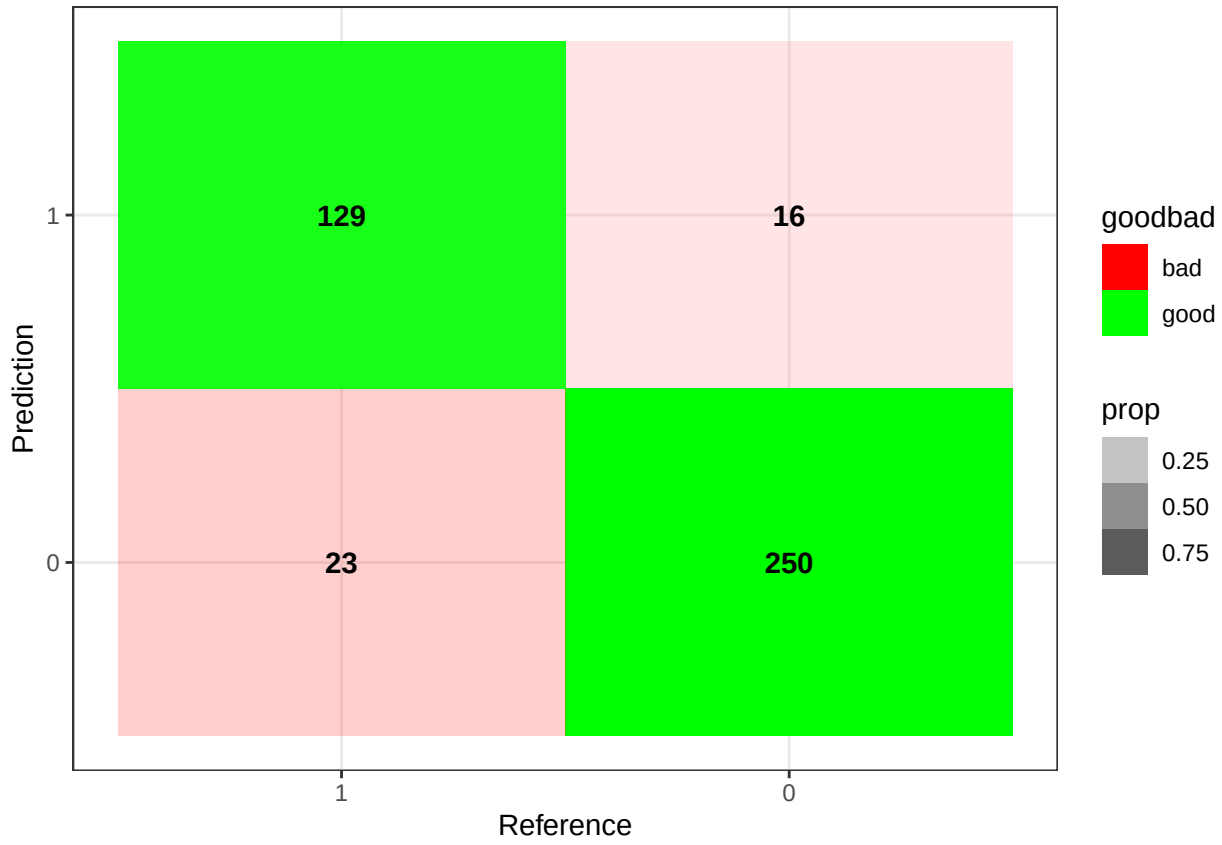
The confusion Matrix is:

```
confusionMatrix(prediction, as.factor(gender_submission$Survived))
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction   0   1
##          0 250  23
##          1  16 129
```

```
##
##                Accuracy : 0.9067
##                  95% CI : (0.8747, 0.9328)
##     No Information Rate : 0.6364
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7964
##
##  Mcnemar's Test P-Value : 0.3367
##
##             Sensitivity : 0.9398
##             Specificity : 0.8487
##          Pos Pred Value : 0.9158
##          Neg Pred Value : 0.8897
##              Prevalence : 0.6364
##          Detection Rate : 0.5981
##    Detection Prevalence : 0.6531
##       Balanced Accuracy : 0.8943
##
##        'Positive' Class : 0
##
```

```
table <- data.frame(confusionMatrix(prediction, as.factor(gender_submission$Survived))$table)
plotTable <- table %>%
  mutate(goodbad = ifelse(table$Prediction == table$Reference, "good", "bad")) %>%
  group_by(Reference) %>%
  mutate(prop = Freq/sum(Freq))

ggplot(data = plotTable, mapping = aes(x = Reference, y = Prediction, fill = goodbad,
                                       alpha = prop)) +
  geom_tile() +
  geom_text(aes(label = Freq), vjust = .5, fontface  = "bold", alpha = 1) +
  scale_fill_manual(values = c(good = "green", bad = "red")) +
  theme_bw() +
  xlim(rev(levels(table$Reference)))
```



Solution With Decision Tree

```
solution1 <- data.frame(PassengerId = test$PassengerId, Survived = prediction)

write.csv(solution1, file = "solution1.csv", row.names = F)
```

Random Forest

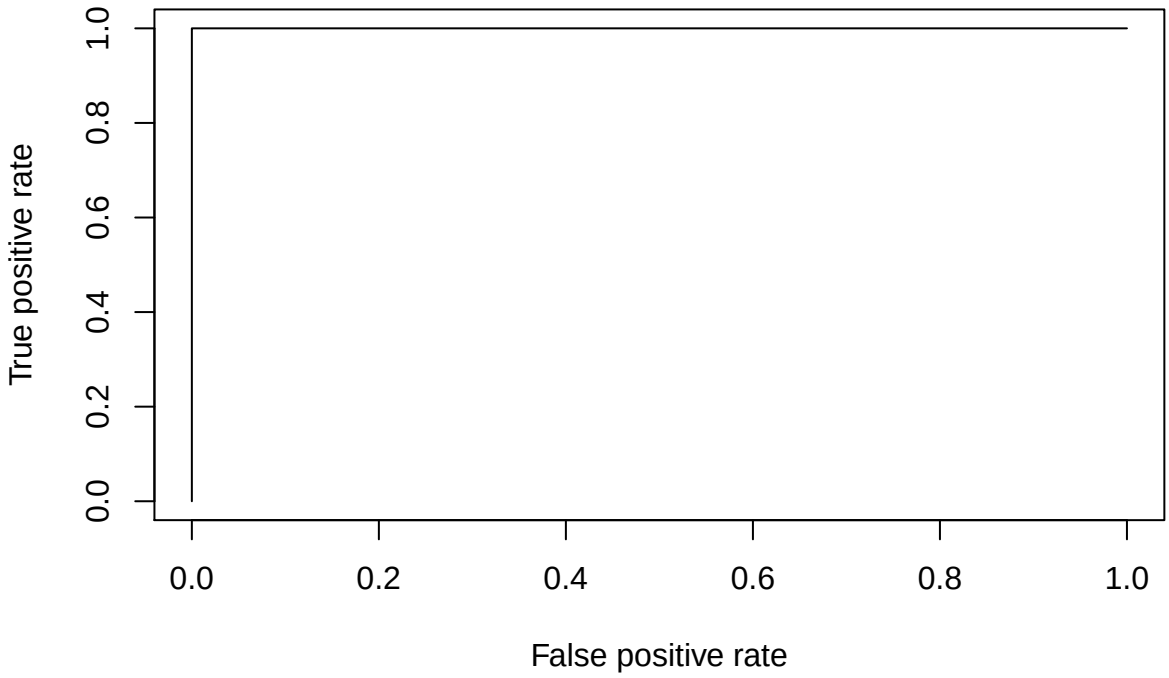
Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean/average prediction (regression) of the individual trees.

```
randomForest <- randomForest(as.factor(Survived) ~ Pclass + Sex +
                    Age + SibSp + Parch + Fare + Embarked  + family_size  + Title,
```

```
prediction <- predict(randomForest, test)
prediction_prob <- predict(randomForest, test, type = "prob")

prediction_prob <- as.data.frame(prediction_prob)

head(prediction_prob)
```

```
##         0     1
## 892 0.979 0.021
## 893 0.699 0.301
## 894 0.849 0.151
## 895 0.825 0.175
## 896 0.534 0.466
## 897 0.937 0.063
```

```
pred <- prediction(prediction_prob[,2],prediction)

perf <- performance(pred,"tpr","fpr")
plot(perf)
```




```
table(gender_submission$Survived, prediction, dnn = c("Actual", "Predicted"))
```

```
##       Predicted
## Actual   0   1
##      0 245  21
##      1  31 121
```

Solution with Random Forest

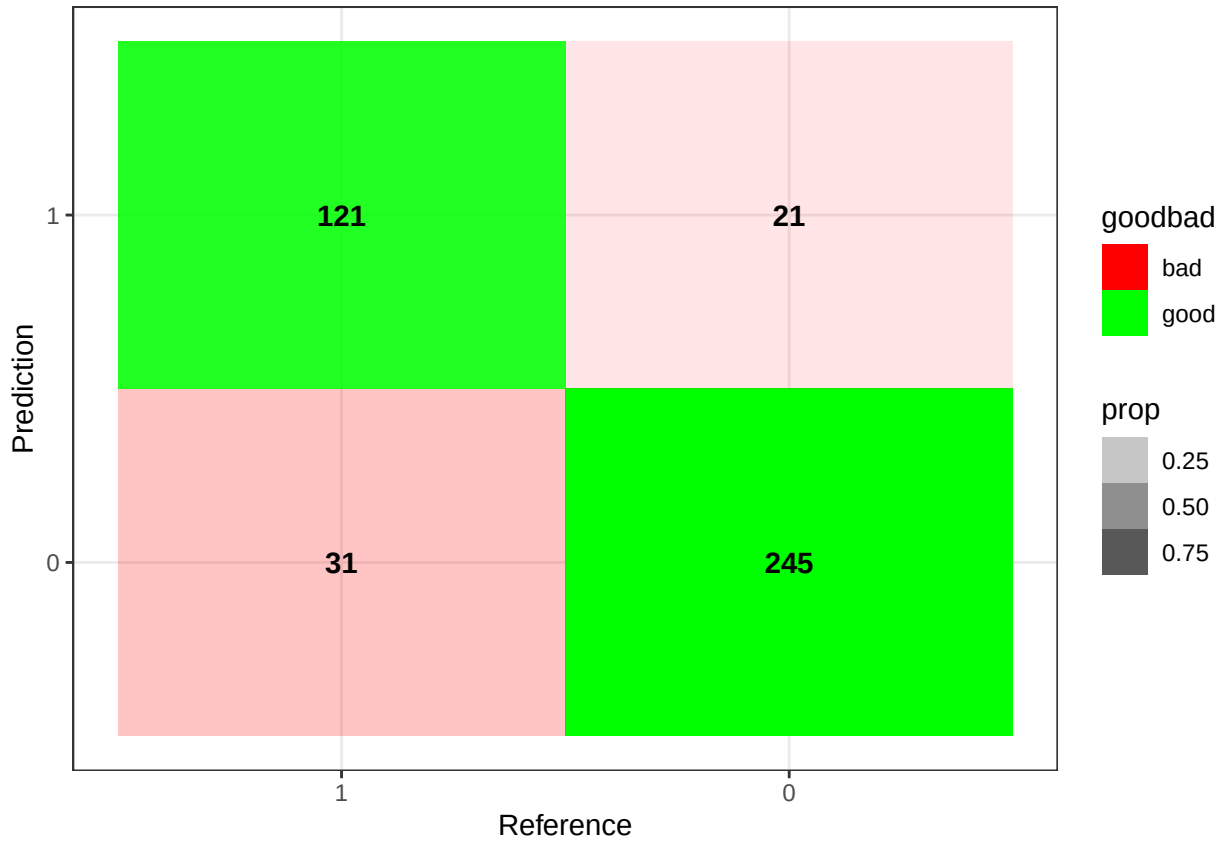
```
solution2 <- data.frame(PassengerId = test$PassengerId,Survived=prediction)
write.csv(solution2, file = "solution2.csv", row.names = F)
```

```
confusionMatrix(prediction, as.factor(gender_submission$Survived))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 245  31
##          1  21 121
##
##                Accuracy : 0.8756
##                  95% CI : (0.8401, 0.9057)
##     No Information Rate : 0.6364
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7274
##
```

```
##  Mcnemar's Test P-Value : 0.212
##
##             Sensitivity : 0.9211
##             Specificity : 0.7961
##          Pos Pred Value : 0.8877
##          Neg Pred Value : 0.8521
##              Prevalence : 0.6364
##          Detection Rate : 0.5861
##    Detection Prevalence : 0.6603
##       Balanced Accuracy : 0.8586
##
##        'Positive' Class : 0
##
```

```
table <- data.frame(confusionMatrix(prediction, as.factor(gender_submission$Survived))$table)
plotTable <- table %>%
  mutate(goodbad = ifelse(table$Prediction == table$Reference, "good", "bad")) %>%
  group_by(Reference) %>%
  mutate(prop = Freq/sum(Freq))


ggplot(data = plotTable, mapping = aes(x = Reference, y = Prediction, fill = goodbad,
                                       alpha = prop)) +
  geom_tile() +
  geom_text(aes(label = Freq), vjust = .5, fontface  = "bold", alpha = 1) +
  scale_fill_manual(values = c(good = "green", bad = "red")) +
  theme_bw() +
  xlim(rev(levels(table$Reference)))
```


121
21
31
245
1
0
Prediction
1
0
Reference
goodbad
bad
good
prop
0.25
0.50
0.75

Support Vector Machine

Support-vector machines are supervised learning models with associated learning algorithms that analyze data for classification and regression analysis.

Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier (although methods such as Platt scaling exist to use SVM in a probabilistic classification setting).

```
SVM <- svm(Survived ~ Pclass + Sex + Age + SibSp + Parch +
             Fare + Embarked  + family_size  + Title, data=train)
```

```
table(train$Survived, fitted(SVM), dnn = c("Actual", "Predicted"))
```

```
##       Predicted
## Actual   0   1
##      0 492  57
##      1  90 252
```

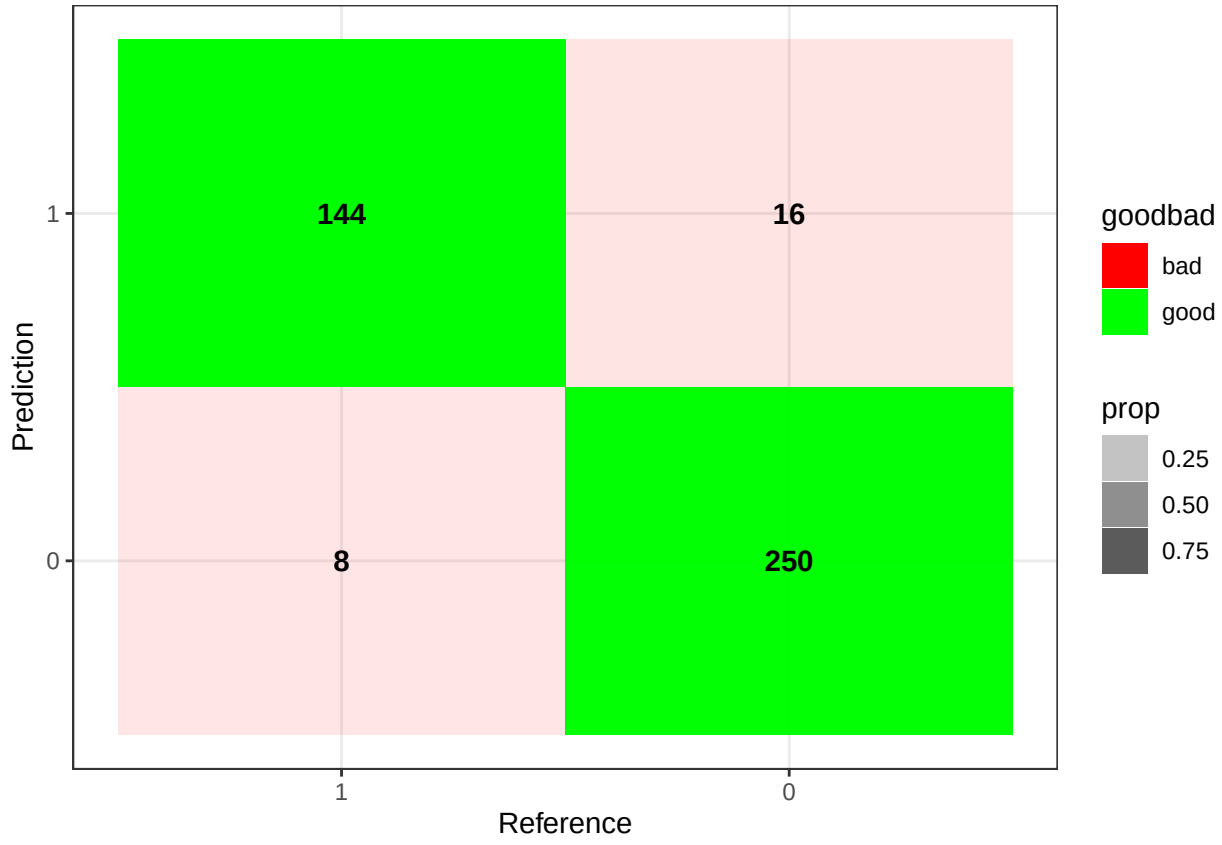
```
prediction <- predict(SVM, test, type="class", na.action = na.pass)
table(as.factor(gender_submission$Survived), prediction, dnn = c("Actual", "Predicted"))
```

```
##       Predicted
## Actual   0   1
##      0 250  16
##      1   8 144
```

```
confusionMatrix(prediction, as.factor(gender_submission$Survived))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 250   8
##          1  16 144
##
##                Accuracy : 0.9426
##                  95% CI : (0.9158, 0.9629)
##     No Information Rate : 0.6364
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8773
##
##  Mcnemar's Test P-Value : 0.153
##
##             Sensitivity : 0.9398
##             Specificity : 0.9474
##          Pos Pred Value : 0.9690
##          Neg Pred Value : 0.9000
##              Prevalence : 0.6364
##          Detection Rate : 0.5981
##    Detection Prevalence : 0.6172
##       Balanced Accuracy : 0.9436
##
##        'Positive' Class : 0
##
```

```
table <- data.frame(confusionMatrix(prediction, as.factor(gender_submission$Survived))$table)
plotTable <- table %>%
  mutate(goodbad = ifelse(table$Prediction == table$Reference, "good", "bad")) %>%
  group_by(Reference) %>%
  mutate(prop = Freq/sum(Freq))

ggplot(data = plotTable, mapping = aes(x = Reference, y = Prediction, fill = goodbad,
                                       alpha = prop)) +
  geom_tile() +
  geom_text(aes(label = Freq), vjust = .5, fontface  = "bold", alpha = 1) +
  scale_fill_manual(values = c(good = "green", bad = "red")) +
  theme_bw() +
  xlim(rev(levels(table$Reference)))
```



Solution with SVM:

```
solution3 <- data.frame(PassengerId = test$PassengerId, Survived = prediction)
write.csv(solution3, file = "solution3.csv", row.names = F)
```

K-Nearest Neighbors

The k-nearest neighbors algorithm (k-NN) is a non-parametric classification method. It is used for classification and regression. In both cases, the input consists of the k closest training examples in data set. The output depends on whether k-NN is used for classification or regression.

- In k-NN classification, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the most common class among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor.
- In k-NN regression, the output is the property value for the object. This value is the average of the values of k nearest neighbors.

For this algorithm, it's necessary to normalize the values of each variable to the range 0:1 so that no variable's range has an unduly large impact on the distance measurement.

Therefore, it will be necessary to apply the following formula:

$$z = \frac{x - \max(x)}{\max(x) - \min(x)}$$

Briefly:

- Normalization makes training less sensitive to the scale of features, so we can better solve for coefficients
- The use of a normalization method will improve analysis from multiple models.
  - Additionally, if we were to use any algorithms on this data set before we normalized, it would be hard (potentially not possible) to converge the vectors because of the scaling issues. Normalization makes the data better conditioned for convergence.
- Normalizing will ensure that a convergence problem does not have a massive variance, making optimization feasible.

```
train_z <- train
test_z <- test


train_z[,c("Age","SibSp","Parch","Fare","family_size")] <- scale(train[,c("Age","SibSp","Parch",
                                                                          "Fare","family_size")])
test_z[,c("Age","SibSp","Parch","Fare","family_size")] <- scale(test[,c("Age","SibSp","Parch",
                                                                       "Fare","family_size")])
```

To apply correctly the algorithm, we must transform those variables treated as a factor into numeric variables.

```
train_z$Sex <- as.numeric(train_z$Sex)
train_z$Embarked <- as.numeric(train_z$Embarked)
train_z$Title <- as.numeric(train_z$Title)

test_z$Sex <- as.numeric(test_z$Sex)
test_z$Embarked <- as.numeric(test_z$Embarked)
test_z$Title <- as.numeric(test_z$Title)
```

With $k = 1$

```
pred1 <- knn(train_z[,c("Pclass","Sex","Age","SibSp","Parch","Fare",
                        "Embarked","family_size","Title")],
```

```
              test_z[,c("Pclass","Sex","Age","SibSp","Parch","Fare",
                        "Embarked","family_size","Title")],
              train_z$Survived,
              k = 1)
```

The error matrix:

```
errmat1 <- table(test_z$Survived, pred1, dnn = c("Actual","Predicted"))
errmat1
```

```
##       Predicted
## Actual   0   1
##      0 257 161
##      1   0   0
```

Finding the best value for k:

```
trcontrol <- trainControl(method = "repeatedcv",
                          number = 10,
                          repeats = 3)

k_value <- train(Survived ~ Pclass + Sex + Age + SibSp + Parch + Fare +
                   Embarked  + family_size  + Title,
                       data = train_z,
                       method = "knn",
                       trControl = trcontrol,
                       preProcess = c("center", "scale"),
                       tuneLength = 10)
k_value
```

```
## k-Nearest Neighbors
##
## 891 samples
##   9 predictor
##   2 classes: '0', '1'
##
## Pre-processing: centered (10), scaled (10)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 803, 802, 802, 802, 802, 802, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    5  0.8155652  0.6027019
##    7  0.8166971  0.6039450
##    9  0.8178290  0.6038277
##   11  0.8181744  0.6030849
##   13  0.8178080  0.6016586
##   15  0.8114325  0.5857368
##   17  0.8065384  0.5748142
##   19  0.8035630  0.5683531
##   21  0.8039506  0.5673149
##   23  0.8035632  0.5643040
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 11.
```

```
pred <- knn(train_z[,c("Pclass","Sex","Age","SibSp","Parch","Fare",
                       "Embarked","family_size","Title")],
            test_z[,c("Pclass","Sex","Age","SibSp","Parch","Fare",
                      "Embarked","family_size","Title")],
            train_z$Survived,
            k = 11)
```

```
solution4 <- data.frame(PassengerId = test$PassengerId, Survived = pred)

write.csv(solution4, file = "solution4.csv", row.names = F)
```

Logistic Regression

The logistic model is used to model the probability of a certain class or event existing such as pass/fail, win/lose, alive/dead or healthy/sick. This can be extended to model several classes of events such as determining whether an image contains a cat, dog, lion, etc. Each object being detected in the image would be assigned a probability between 0 and 1, with a sum of one.

```
LogisticRegression <- glm(Survived ~ Pclass + Sex + Age + SibSp + Parch + Fare + Embarked, family=binom
summary(LogisticRegression)
```

```
##
## Call:
## glm(formula = Survived ~ Pclass + Sex + Age + SibSp + Parch +
##     Fare + Embarked, family = binomial(link = "logit"), data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7144  -0.6038  -0.4074   0.6138   2.4886
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  4.414782   0.501865   8.797  < 2e-16 ***
## Pclass2     -1.081225   0.306067  -3.533 0.000411 ***
## Pclass3     -2.332380   0.309419  -7.538 4.77e-14 ***
## Sexmale     -2.741465   0.202718 -13.524  < 2e-16 ***
## Age         -0.044136   0.008138  -5.423 5.86e-08 ***
## SibSp       -0.338068   0.111637  -3.028 0.002460 **
## Parch       -0.106743   0.120555  -0.885 0.375927
## Fare         0.002012   0.002469   0.815 0.415072
## EmbarkedQ   -0.016163   0.383543  -0.042 0.966386
## EmbarkedS   -0.421356   0.240556  -1.752 0.079844 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1186.66  on 890  degrees of freedom
## Residual deviance:  778.57  on 881  degrees of freedom
## AIC: 798.57
##
## Number of Fisher Scoring iterations: 5
```

```
result <- predict(LogisticRegression,newdata=test,type='response')
result <- ifelse(result > 0.5,1,0)

confusionMatrix(data = as.factor(result), reference=test$Survived)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 263   0
##          1 155   0
##
##                Accuracy : 0.6292
##                  95% CI : (0.5809, 0.6756)
```

```
##     No Information Rate : 1
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.6292
##             Specificity :     NA
##          Pos Pred Value :     NA
##          Neg Pred Value :     NA
##              Prevalence : 1.0000
##          Detection Rate : 0.6292
##    Detection Prevalence : 0.6292
##       Balanced Accuracy :     NA
##
##        'Positive' Class : 0
##
```

```
solution5 <- data.frame(PassengerId = test$PassengerId, Survived = result)

write.csv(solution5, file = "solution5.csv", row.names = F)
```

Accuracy Comparison

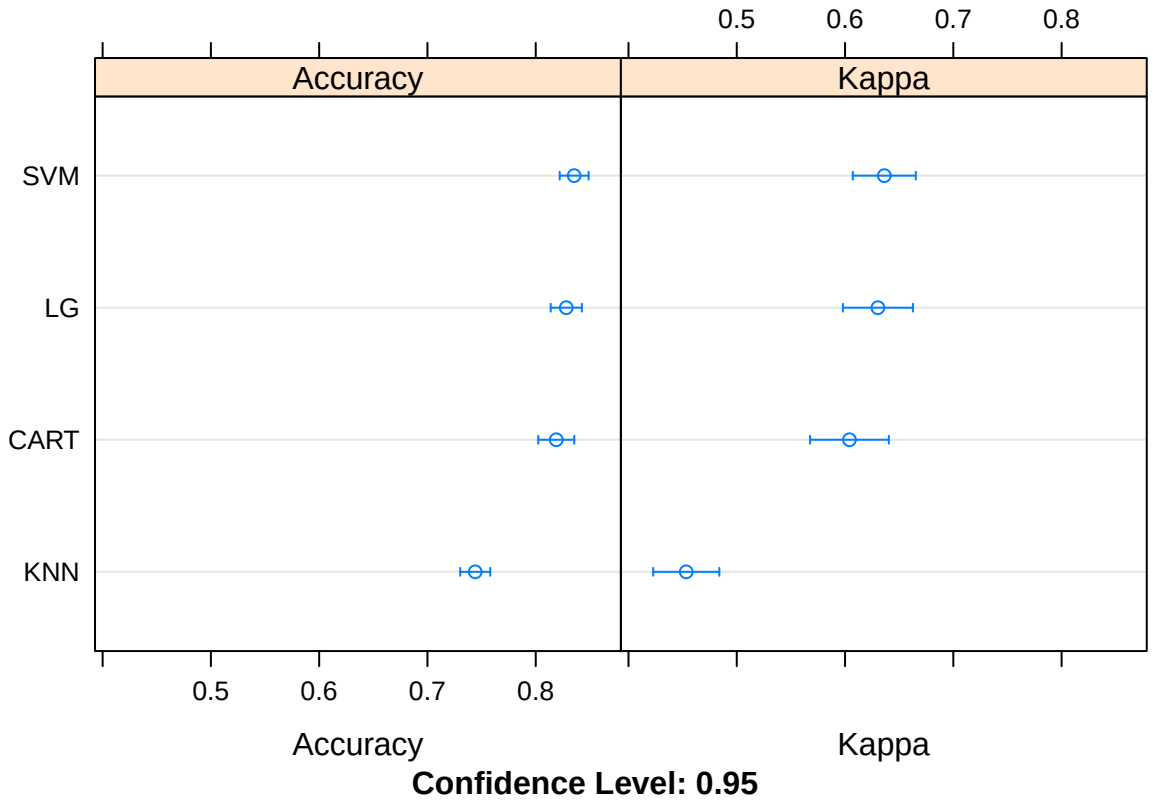
```
datasetTrain <- train[,c(-1, -4, -9, -11,-13)]

summary(datasetTrain)
```

```
##  Survived Pclass      Sex           Age            SibSp           Parch
##  0:549    1:216   female:314   Min.   : 0.42   Min.   :0.000   Min.   :0.0000
##  1:342    2:184   male  :577   1st Qu.:22.00   1st Qu.:0.000   1st Qu.:0.0000
##           3:491                Median :28.39   Median :0.000   Median :0.0000
##                                Mean   :29.94   Mean   :0.523   Mean   :0.3816
##                                3rd Qu.:37.00   3rd Qu.:1.000   3rd Qu.:0.0000
##                                Max.   :80.00   Max.   :8.000   Max.   :6.0000
##       Fare        Embarked  family_size       Title
##  Min.   :  0.00   C:168    Min.   : 1.000   boy  : 40
##  1st Qu.:  7.91   Q: 77    1st Qu.: 1.000   man  :538
##  Median : 14.45   S:646    Median : 1.000   woman:313
##  Mean   : 32.20            Mean   : 1.905
##  3rd Qu.: 31.00            3rd Qu.: 2.000
##  Max.   :512.33            Max.   :11.000
```

```
trainControl <- trainControl(method="repeatedcv", number=10, repeats=3)
metric <- "Accuracy"

fit.glm <- train(Survived~., data=datasetTrain, method="glm", metric=metric, trControl=trainControl)
fit.knn <- train(Survived~., data=datasetTrain, method="knn", metric=metric, trControl=trainControl)
fit.cart <- train(Survived~., data=datasetTrain, method="rpart", metric=metric,
                  trControl=trainControl)
fit.svm <- train(Survived~., data=datasetTrain, method="svmRadial", metric=metric,
                 trControl=trainControl)
results <- resamples(list(LG=fit.glm, KNN=fit.knn,
                          CART=fit.cart, SVM=fit.svm))
summary(results)
```

```
##
## Call:
## summary.resamples(object = results)
##
## Models: LG, KNN, CART, SVM
## Number of resamples: 30
##
## Accuracy
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## LG   0.7500000 0.8000000 0.8314607 0.8282179 0.8627653 0.8988764    0
## KNN  0.6741573 0.7198814 0.7415730 0.7441199 0.7724719 0.8202247    0
## CART 0.7191011 0.7871099 0.8202247 0.8189729 0.8551498 0.8876404    0
## SVM  0.7444444 0.8111111 0.8314607 0.8354855 0.8616162 0.9101124    0
##
## Kappa
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## LG   0.4453258 0.5683664 0.6368849 0.6303480 0.7011851 0.7870247    0
## KNN  0.2899587 0.3889735 0.4560612 0.4533488 0.5247006 0.6059768    0
## CART 0.3948871 0.5252795 0.6012887 0.6040658 0.6892891 0.7593294    0
## SVM  0.4595300 0.5810778 0.6239970 0.6362566 0.6969149 0.8029884    0
```

```
dotplot(results)
```



The SVM algorithm has the highest accuracy

Bibliography

- Gomila Salas, J., 2018. *La técnica del Random Trees en Rstudio* [online] Youtube. Available at: https://www.youtube.com/watch?v=oA0bHBEdMuw&list=PLLhrRW5wp33XkT-y4GXZrTSgx1Z1V3EA7&index=2 [Accessed February 2021].
- Gomila Salas, J., 2018. *La técnica del Random Forest en Rstudio* [online] Youtube. Available at: https://www.youtube.com/watch?v=HJB6XFkmezM&list=PLLhrRW5wp33XkT-y4GXZrTSgx1Z1V3EA7&index=3 [Accessed February 2021].
- Salas, J., 2018. *Support Vector Machines en Rstudio.* [online] Youtube. Available at: https://www.youtube.com/watch?v=_JdK4FMzd28&list=PLLhrRW5wp33XkT-y4GXZrTSgx1Z1V3EA7&index=4 [Accessed February 2021].
- Salas, J., 2018. *K Nearest Neighbors en RStudio.* [online] Youtube. Available at: https://www.youtube.com/watch?v=9C6HI_CyRG4&list=PLLhrRW5wp33XkT-y4GXZrTSgx1Z1V3EA7&index=5 [Accessed February 2021].
- Berrendero, J., 2016. *RPubs - Introducción al paquete Caret.* [online] Rpubs.com. Available at: https://rpubs.com/joser/caret [Accessed February 2021].
- Ononse Bisong, E., 2016. *RPubs - Titanic Machine Learning from Disaster: How to evaluate Algorithms.* [online] Rpubs.com. Available at: https://rpubs.com/dvdbisong/titanic [Accessed February 2021].
- Stack Overflow. 2019. *Plot confusion matrix in R using ggplot.* [online] Available at: https://stackoverflow.com/questions/37897252/plot-confusion-matrix-in-r-using-ggplot/37897416 [Accessed February 2021].
- DeFilippi, R., 2018. *Standardize or Normalize? — Examples in Python.* [online] Medium. Available at: https://medium.com/@rrfd/standardize-or-normalize-examples-in-python-e3f174b65dfc [Accessed February 2021].
- Wickham, H., 2016. *R for Data Science: Visualize, Model, Transform, Tidy, and Import Data.* O'Reilly Media.