Titanic. First steps in ML. ver. 1.1

Andrey Korotkiy

May 23, 2019

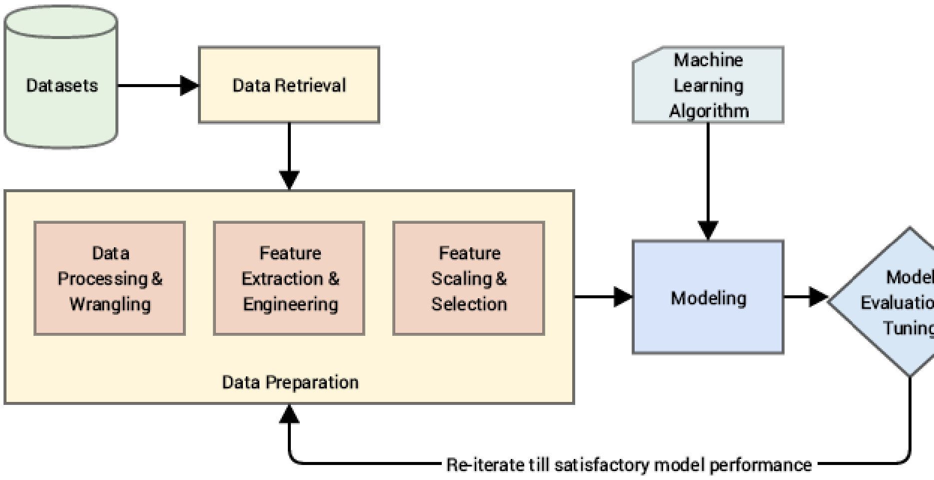
```
library(tidyverse)
library(scales)
library(knitr)
library(corrplot)
library(alluvial)
library(vcd)
library(data.table)
library(DT)

library(caret)
library(rpart)
library(rpart.plot)
library(e1071)
library(ROCR)
```

I would like to share my first steps in ML on the example of a training dataset. To solve the classification problem, I studied and applied the following approaches:

1) logistic regression

2) decision tree

3) catboost



I follow next steps in data analysis process:

1. Data Processing & Wrangling

Let's take a look at our data structure

```
train_df <- read_csv('train.csv')
```

```
## Parsed with column specification:
```

```
## cols(
##   PassengerId = col_double(),
##   Survived = col_double(),
##   Pclass = col_double(),
##   Name = col_character(),
##   Sex = col_character(),
##   Age = col_double(),
##   SibSp = col_double(),
##   Parch = col_double(),
##   Ticket = col_character(),
##   Fare = col_double(),
##   Cabin = col_character(),
##   Embarked = col_character()
## )
```

```
test_df  <- read_csv('test.csv')
```

```
## Parsed with column specification:
## cols(
##   PassengerId = col_double(),
##   Pclass = col_double(),
##   Name = col_character(),
##   Sex = col_character(),
##   Age = col_double(),
##   SibSp = col_double(),
##   Parch = col_double(),
##   Ticket = col_character(),
##   Fare = col_double(),
##   Cabin = col_character(),
##   Embarked = col_character()
## )
```

Let's mark our test and train sets and merge them into one

```
train_df$set <- "train"
test_df$set  <- "test"
test_df$Survived <- NA
```

Now let's analyze the full date frame.

```
full_df <- rbind(train_df, test_df)
str(full_df)
```

```
## Classes 'spec_tbl_df', 'tbl_df', 'tbl' and 'data.frame': 1309 obs. of  13 variables:
##  $ PassengerId: num  1 2 3 4 5 6 7 8 9 10 ...
##  $ Survived   : num  0 1 1 1 0 0 0 0 1 1 ...
##  $ Pclass     : num  3 1 3 1 3 3 1 3 3 2 ...
##  $ Name       : chr  "Braund, Mr. Owen Harris" "Cumings, Mrs. John Bradley (Florence Briggs Thayer)"
##  $ Sex        : chr  "male" "female" "female" "female" ...
##  $ Age        : num  22 38 26 35 35 NA 54 2 27 14 ...
##  $ SibSp      : num  1 1 0 1 0 0 0 3 0 1 ...
##  $ Parch      : num  0 0 0 0 0 0 0 1 2 0 ...
##  $ Ticket     : chr  "A/5 21171" "PC 17599" "STON/O2. 3101282" "113803" ...
##  $ Fare       : num  7.25 71.28 7.92 53.1 8.05 ...
##  $ Cabin      : chr  NA "C85" NA "C123" ...
##  $ Embarked   : chr  "S" "C" "S" "S" ...
##  $ set        : chr  "train" "train" "train" "train" ...
```

```
##  - attr(*, "spec")=
##   .. cols(
##   ..   PassengerId = col_double(),
##   ..   Survived = col_double(),
##   ..   Pclass = col_double(),
##   ..   Name = col_character(),
##   ..   Sex = col_character(),
##   ..   Age = col_double(),
##   ..   SibSp = col_double(),
##   ..   Parch = col_double(),
##   ..   Ticket = col_character(),
##   ..   Fare = col_double(),
##   ..   Cabin = col_character(),
##   ..   Embarked = col_character()
##   .. )
```

First of all I would like to work with NA values. Lets see NA distribution by our feature.

```r
na_values <- full_df %>%
    gather(key = "key", value = "val") %>%
    mutate(is.missing = is.na(val)) %>%
    group_by(key, is.missing) %>%
    summarise(num.missing = n()) %>%
    filter(is.missing==T) %>%
    select(-is.missing) %>%
    arrange(desc(num.missing))

na_values
```

```
## # A tibble: 5 x 2
## # Groups:   key [5]
##   key      num.missing
##   <chr>          <int>
## 1 Cabin           1014
## 2 Survived         418
## 3 Age              263
## 4 Embarked           2
## 5 Fare               1
```

Now we can deal with our NA values.
In the quantitative variable Age, we can simply replace them with average values

```r
full_df$Age[is.na(full_df$Age)] <- mean(full_df$Age,na.rm = T)
```

For Embarked feature we use the most common code

```r
full_df$Embarked <- replace(full_df$Embarked, which(is.na(full_df$Embarked)), 'S')
```

I'm not going to work with the Cabin and Fare variables, leave them as they are.

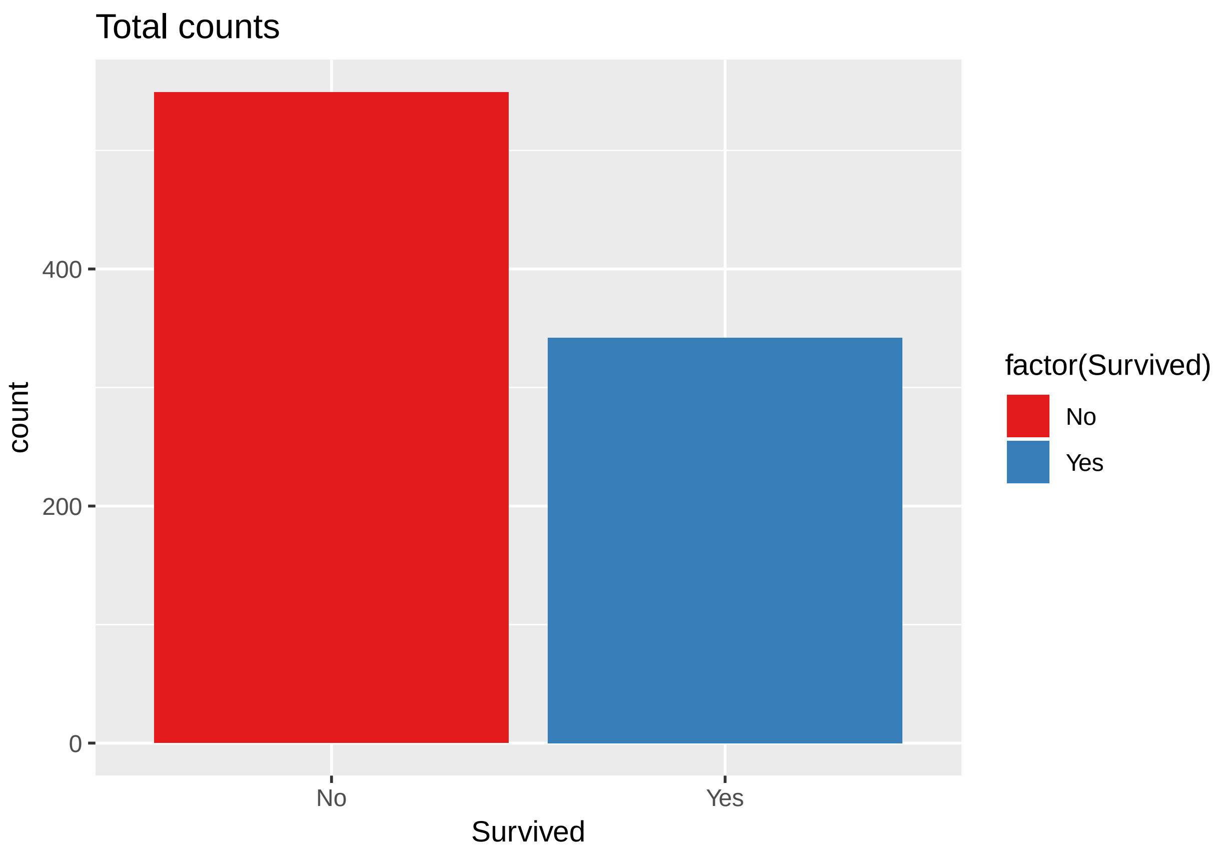
Let's find out the percentage of survivors after the disaster…

```r
full_df <- full_df %>%
  mutate(Survived = case_when(Survived==1 ~ "Yes", Survived==0 ~ "No"))

full_df$Survived <- factor(full_df$Survived, levels=c('No', 'Yes'))
table(full_df$Survived)
```

```
##
##  No Yes
## 549 342
```

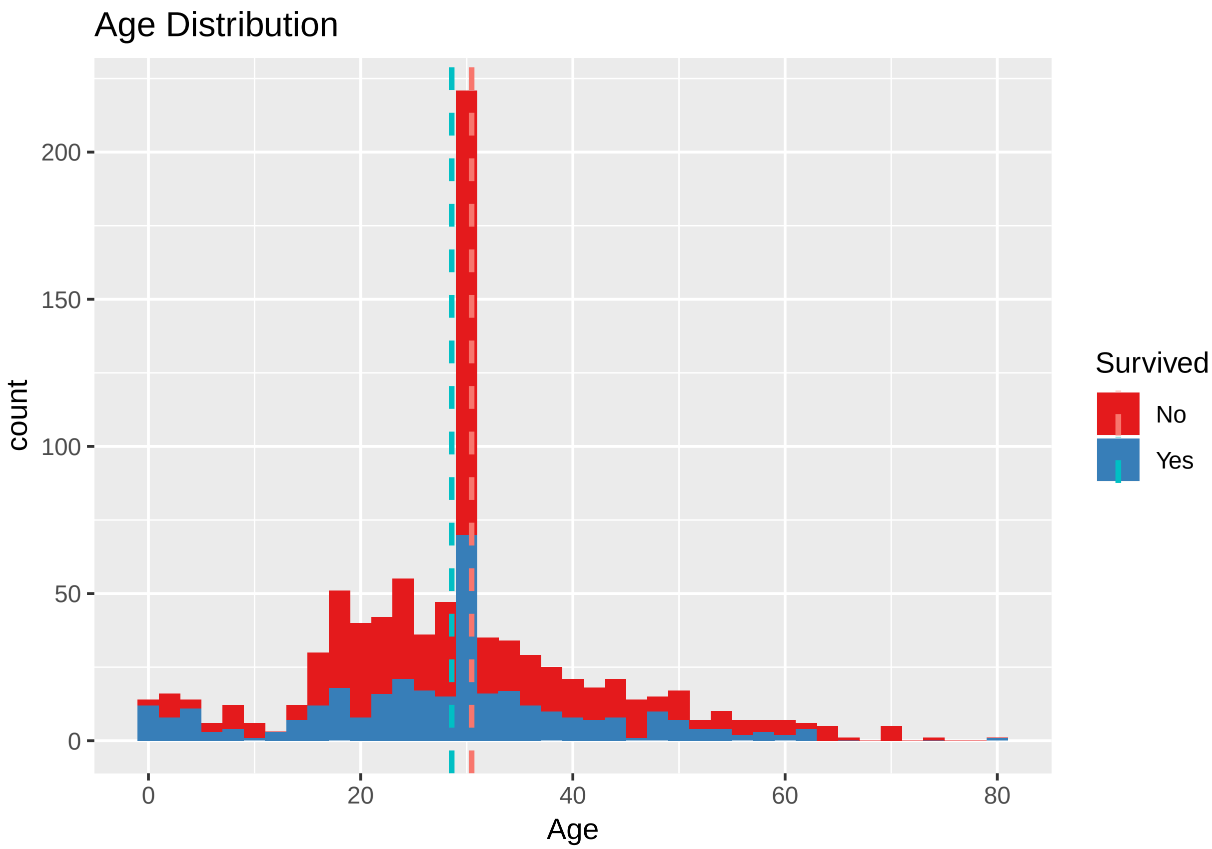
Survival rate is 0.3838384

```
ggplot(full_df %>% filter(set=="train"), aes(Survived, fill=factor(Survived))) +
  geom_bar() +
  scale_fill_brewer(palette="Set1") +
  ggtitle("Total counts")
```



Let's find out how the Titanic passengers were distributed by age.

```
age_df <- full_df %>%
  filter(set=="train") %>%
  select(Age, Survived) %>%
  group_by(Survived) %>%
  summarise(mean_age = mean(Age, na.rm=TRUE))

ggplot(full_df %>% filter(set=="train"), aes(Age, fill=Survived)) +
  geom_histogram(aes(y=..count..),binwidth = 2) +
  geom_vline(data=age_df, aes(xintercept=mean_age, colour=Survived), lty=2, size=1) +
  scale_fill_brewer(palette="Set1") +
  ggtitle("Age Distribution")
```



We see roughly the same distribution pattern, but the mean values of age in the groups survived/not survived are slightly different

```
age_df
```

```
## # A tibble: 2 x 2
##   Survived mean_age
##   <fct>       <dbl>
## 1 No           30.5
## 2 Yes          28.6
```

let's look at distribution by class histograms

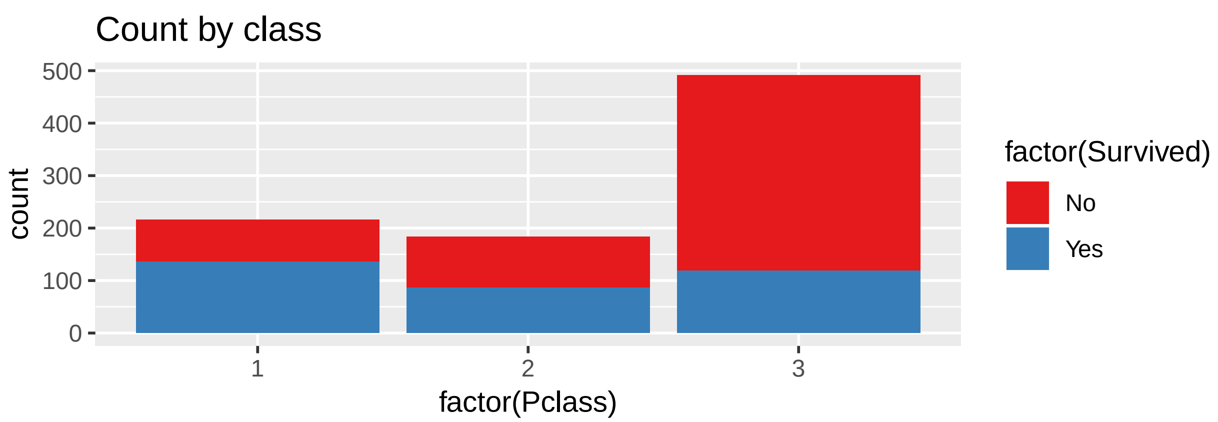
```
plot_a <- ggplot(full_df %>% filter(set=="train"), aes(factor(Pclass))) +
  geom_bar(aes(fill = factor(Survived))) +
  scale_fill_brewer(palette="Set1") +
  ggtitle("Count by class")

plot_b <- ggplot(full_df %>% filter(set=="train"), aes(Age, stat(density))) +
  geom_histogram(binwidth = 4) +
  facet_grid(. ~ Pclass) +
  scale_y_continuous(labels = percent, name = "Percent")+
  ggtitle("Age distribution by class")

require(gridExtra)
```

```
## Loading required package: gridExtra
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

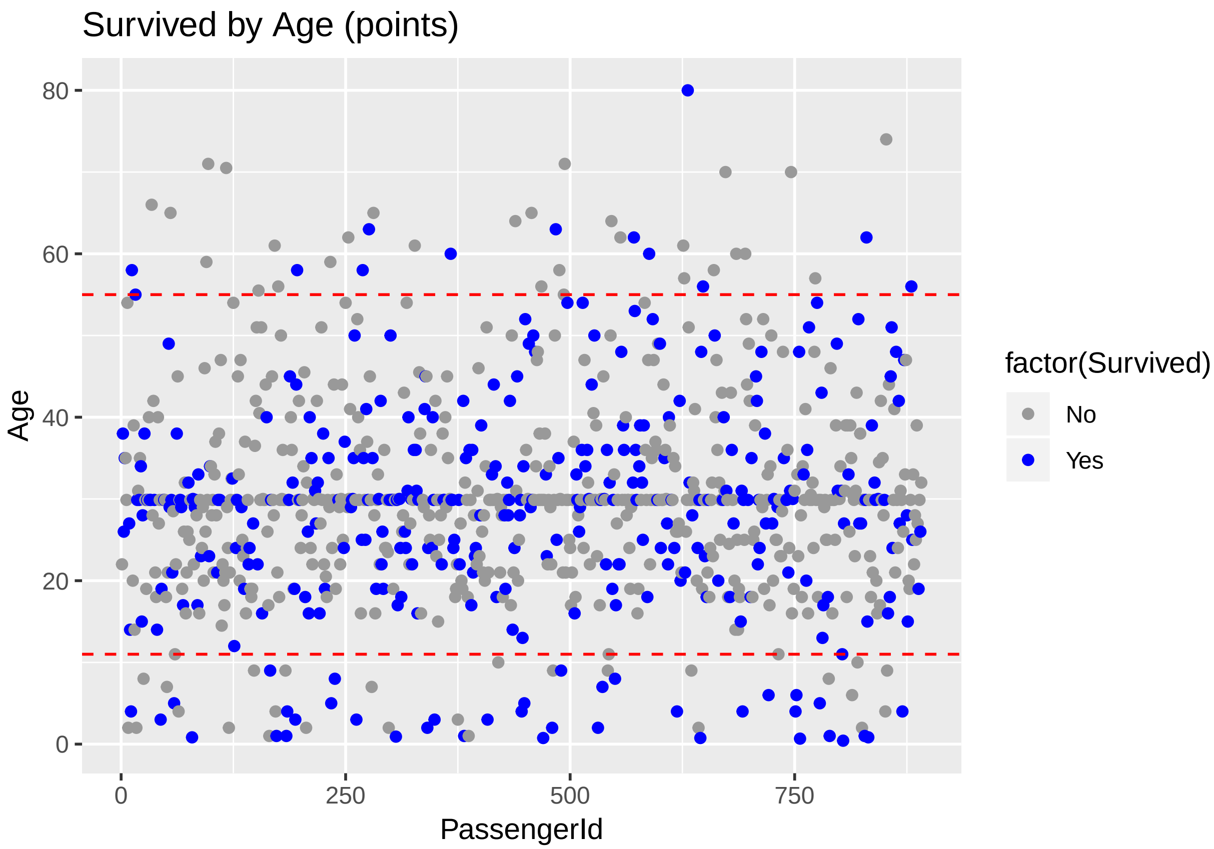
```r
grid.arrange(plot_b, plot_a, ncol=1)
```



we see that people from 20 to 40 years old prevailed in the third grade, while distributions in other classes are more even

Now we can try to divide our Age feature into several groups Lets look on pint plot

```r
ggplot(full_df %>% filter(set=="train"), aes(PassengerId, Age)) +
  geom_point(aes(colour = factor(Survived))) +
  geom_hline(yintercept = 11,linetype="dashed", color = "red") +
  geom_hline(yintercept = 55,linetype="dashed", color = "red") +
  scale_color_manual(values=c("#999999", "blue", "blue")) +
  scale_fill_manual(values=c("#999999", "blue", "blue")) +
  ggtitle("Survived by Age (points)")
```



I

choose 3 age groups

```
full_df <- full_df %>%
  mutate(`Age Group` =
           case_when(Age <= 11 ~ "Children",
                Age > 11 & Age < 55 ~ "Adult",
                Age >= 50 ~ "Old"))
```

Now let's work on the name variable Extract an individual's title from the Name feature.

```
names <- full_df$Name
title <- gsub("^.*, (.*?)\\..*$", "\\1", names)
full_df$title <- title
table(title)
```

```
## title
##         Capt          Col          Don         Dona           Dr
##            1            4            1            1            8
##     Jonkheer         Lady        Major       Master         Miss
##            1            1            2           61          260
##         Mlle          Mme           Mr          Mrs           Ms
##            2            1          757          197            2
##          Rev          Sir the Countess
##            8            1            1
```

Mr, Mrs and miss are most popular

```
full_df$title[full_df$title == 'Mlle'] <- 'Miss'
full_df$title[full_df$title == 'Ms']   <- 'Miss'
full_df$title[full_df$title == 'Mme']  <- 'Mrs'
```

```r
full_df$title[full_df$title == 'Lady'] <- 'Mrs'
full_df$title[full_df$title == 'Dona'] <- 'Miss'
```

Put others in one class Officer

```r
full_df$title[full_df$title == 'Capt'] <- 'Officer'
full_df$title[full_df$title == 'Col'] <- 'Officer'
full_df$title[full_df$title == 'Major'] <- 'Officer'
full_df$title[full_df$title == 'Dr'] <- 'Officer'
full_df$title[full_df$title == 'Rev'] <- 'Officer'
full_df$title[full_df$title == 'Don'] <- 'Officer'
full_df$title[full_df$title == 'Sir'] <- 'Officer'
full_df$title[full_df$title == 'the Countess'] <- 'Officer'
full_df$title[full_df$title == 'Jonkheer'] <- 'Officer'
```

Also we can add discretized feature based on family member count.

```r
full_df$FamilySize <- full_df$SibSp + full_df$Parch + 1

full_df$FamilySized[full_df$FamilySize == 1] <- 'Single'
full_df$FamilySized[full_df$FamilySize < 5 & full_df$FamilySize >= 2] <- 'Small'
full_df$FamilySized[full_df$FamilySize >= 5] <- 'Big'

full_df$FamilySized=as.factor(full_df$FamilySized)
```

Engineer features based on all the passengers with the same ticket.

```r
ticket.unique <- rep(0, nrow(full_df))
tickets <- unique(full_df$Ticket)

for (i in 1:length(tickets)) {
  current.ticket <- tickets[i]
  party.indexes <- which(full_df$Ticket == current.ticket)
  for (k in 1:length(party.indexes)) {
    ticket.unique[party.indexes[k]] <- length(party.indexes)
  }
}

full_df$ticket.unique <- ticket.unique

full_df$ticket.size[full_df$ticket.unique == 1]   <- 'Single'
```

```
## Warning: Unknown or uninitialised column: 'ticket.size'.
```

```r
full_df$ticket.size[full_df$ticket.unique < 5 & full_df$ticket.unique>= 2]   <- 'Small'
full_df$ticket.size[full_df$ticket.unique >= 5]   <- 'Big'
```

In total:

```r
str(full_df)
```

```
## Classes 'spec_tbl_df', 'tbl_df', 'tbl' and 'data.frame': 1309 obs. of  19 variables:
##  $ PassengerId  : num  1 2 3 4 5 6 7 8 9 10 ...
##  $ Survived     : Factor w/ 2 levels "No","Yes": 1 2 2 2 1 1 1 1 2 2 ...
##  $ Pclass       : num  3 1 3 1 3 3 1 3 3 2 ...
##  $ Name         : chr  "Braund, Mr. Owen Harris" "Cumings, Mrs. John Bradley (Florence Briggs Thayer)
##  $ Sex          : chr  "male" "female" "female" "female" ...
```

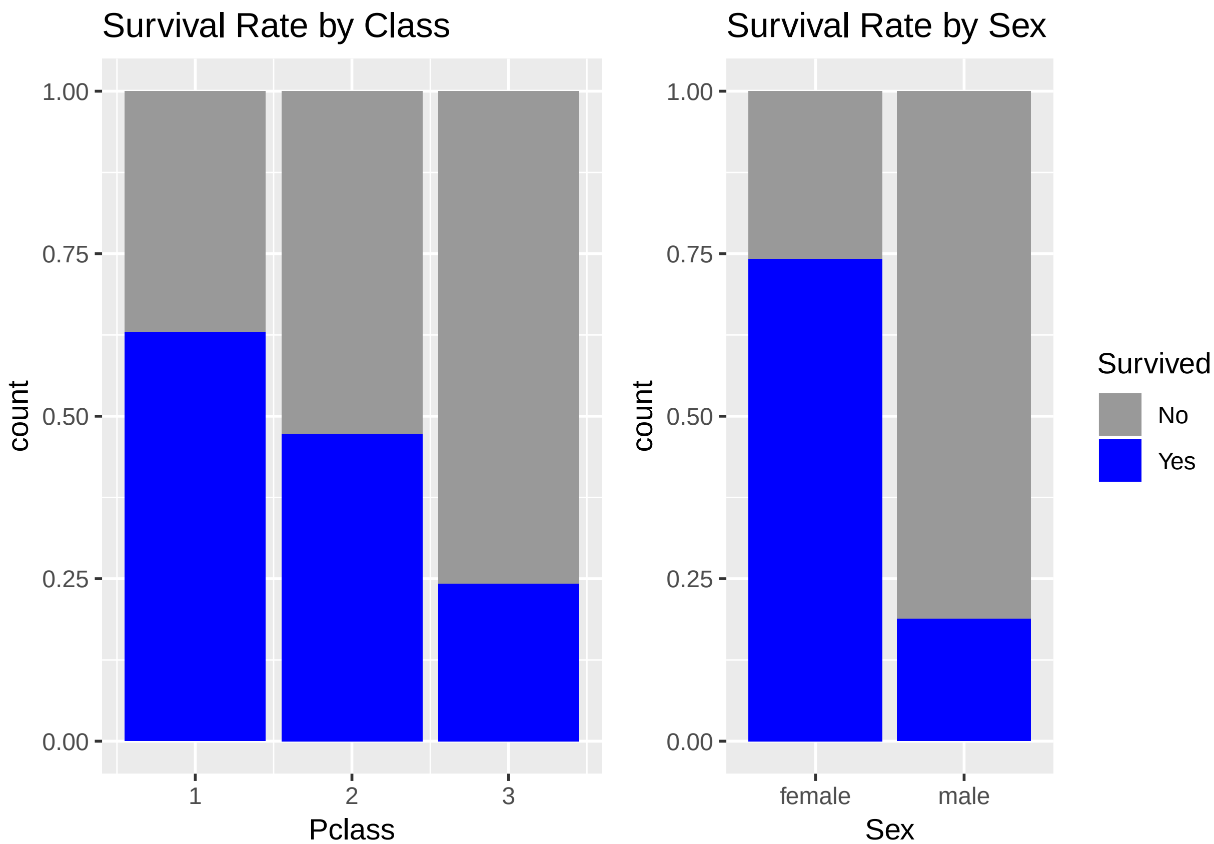
```
##  $ Age          : num  22 38 26 35 35 ...
##  $ SibSp        : num  1 1 0 1 0 0 0 3 0 1 ...
##  $ Parch        : num  0 0 0 0 0 0 0 1 2 0 ...
##  $ Ticket       : chr  "A/5 21171" "PC 17599" "STON/O2. 3101282" "113803" ...
##  $ Fare         : num  7.25 71.28 7.92 53.1 8.05 ...
##  $ Cabin        : chr  NA "C85" NA "C123" ...
##  $ Embarked     : chr  "S" "C" "S" "S" ...
##  $ set          : chr  "train" "train" "train" "train" ...
##  $ Age Group    : chr  "Adult" "Adult" "Adult" "Adult" ...
##  $ title        : chr  "Mr" "Mrs" "Miss" "Mrs" ...
##  $ FamilySize   : num  2 2 1 2 1 1 1 5 3 2 ...
##  $ FamilySized  : Factor w/ 3 levels "Big","Single",..: 3 3 2 3 2 2 2 1 3 3 ...
##  $ ticket.unique: num  1 2 1 2 1 1 2 5 3 2 ...
##  $ ticket.size  : chr  "Single" "Small" "Single" "Small" ...
```

The independent variable, Survived, is labeled as a Bernoulli trial where a passenger or crew member survive (1) or not (0)

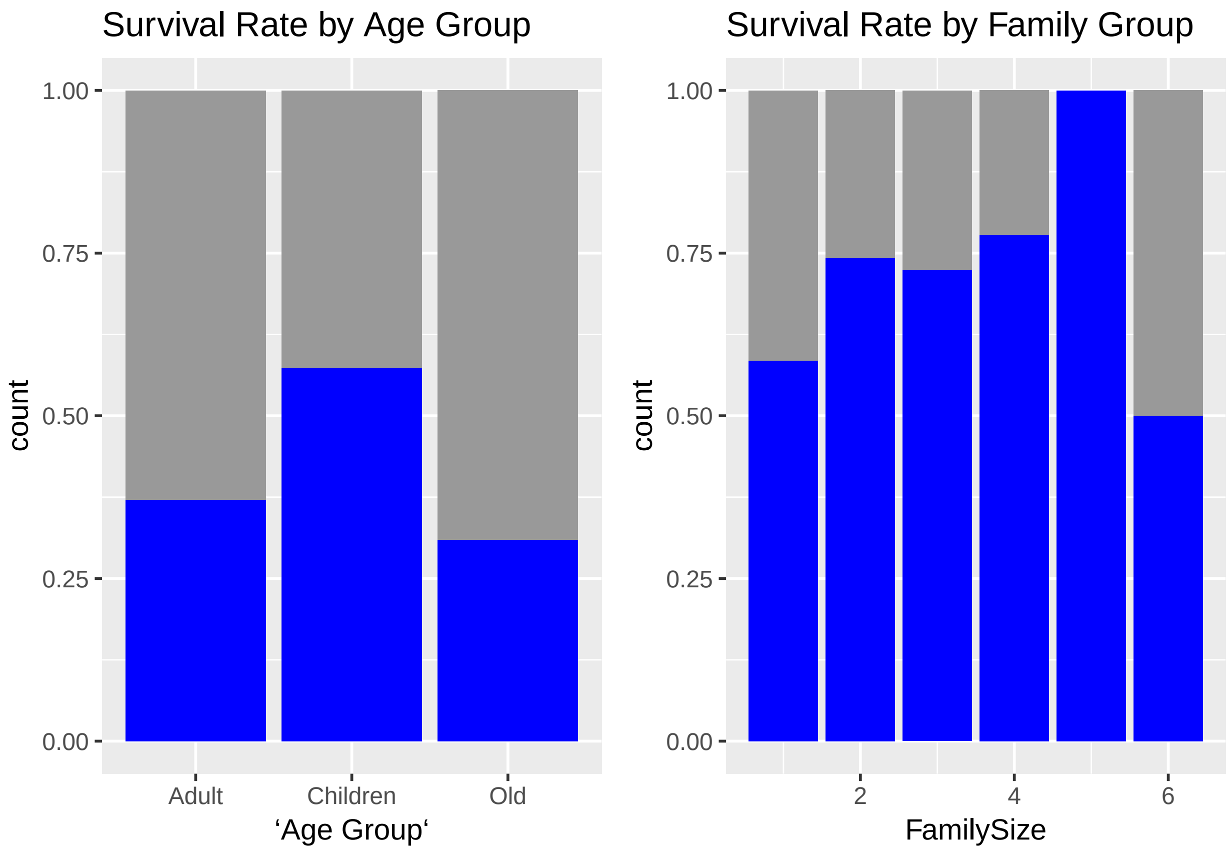
Relationship Between Dependent and Independent Variables

```
plot1 <- ggplot(full_df %>% filter(set=="train"), aes(Pclass, fill=Survived)) +
  geom_bar(position = "fill")  +
  scale_color_manual(values=c("#999999", "#E69F00", "#56B4E9")) +
  scale_fill_manual(values=c("#999999", "blue", "##00b159")) +
  ggtitle("Survival Rate by Class") +
  scale_color_brewer(palette="Dark2") +
  theme(legend.position = "none")

plot2 <- ggplot(full_df %>% filter(set=="train"), aes(Sex, fill=Survived)) +
  scale_color_manual(values=c("#999999", "#E69F00", "#56B4E9")) +
  scale_fill_manual(values=c("#999999", "blue", "##00b159")) +
  geom_bar(position = "fill") +
  ggtitle("Survival Rate by Sex")+
  theme()

plot3 <- ggplot(full_df %>% filter(set=="train" & !is.na(Age)), aes(`Age Group`, fill=Survived)) +
  geom_bar(position = "fill") +
  scale_color_manual(values=c("#999999", "#E69F00", "#56B4E9")) +
  scale_fill_manual(values=c("#999999", "blue", "##00b159")) +
  ggtitle("Survival Rate by Age Group") +
  theme(legend.position = "none")

plot4 <- ggplot(full_df %>% filter(set=="train") %>% na.omit, aes(`FamilySize`, fill=Survived)) +
  geom_bar(position="fill") +
  scale_color_manual(values=c("#999999", "#E69F00", "#56B4E9")) +
  scale_fill_manual(values=c("#999999", "blue", "##00b159")) +
  ggtitle("Survival Rate by Family Group") +
  theme(legend.position = "none")

plot5 <- ggplot(full_df %>% filter(set=="train") %>% na.omit, aes(title, fill=Survived)) +
  geom_bar(position="fill") +
  scale_color_manual(values=c("#999999", "#E69F00", "#56B4E9")) +
  scale_fill_manual(values=c("#999999", "blue", "##00b159")) +
```

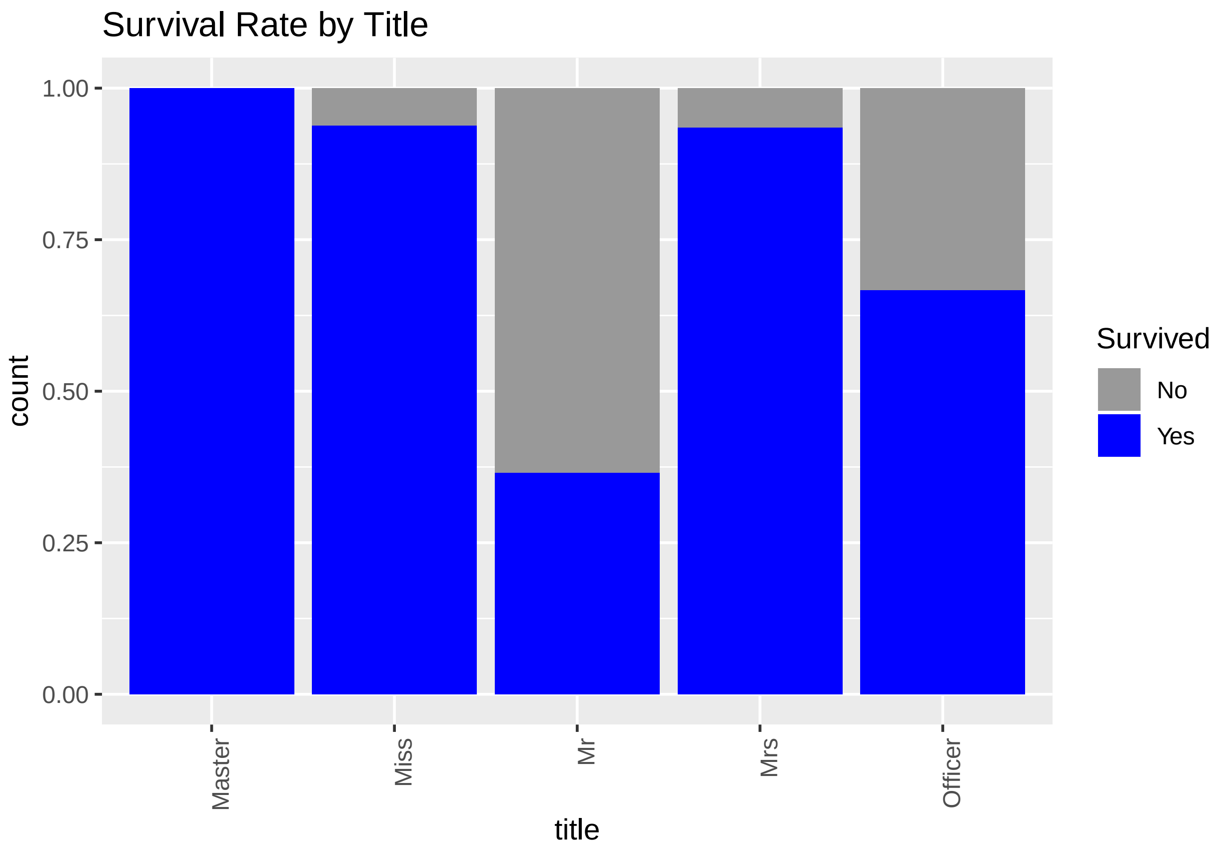
```
  ggtitle("Survival Rate by Title") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))

require(gridExtra)

grid.arrange(plot1, plot2, ncol=2)
```




```
grid.arrange(plot3, plot4, ncol=2)
```

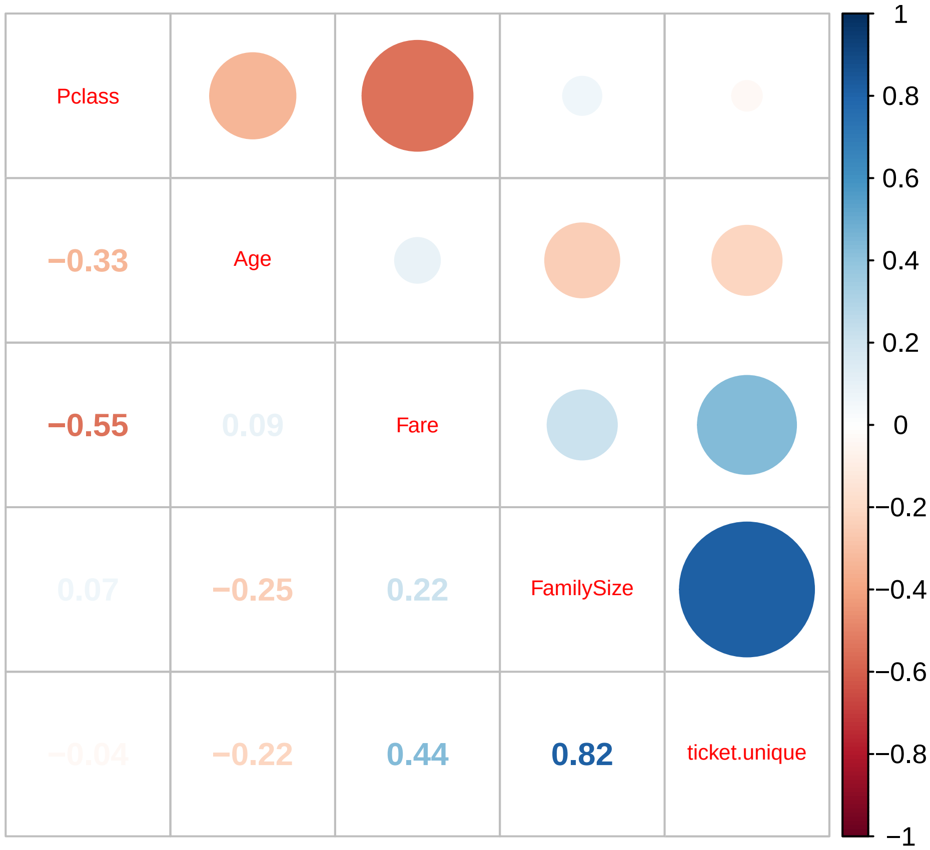

Survival Rate by Age Group
Survival Rate by Family Group
1.00
0.75
0.50
0.25
0.00
count
Adult
Children
Old
‘Age Group‘
2
4
6
FamilySize

```
plot5
```



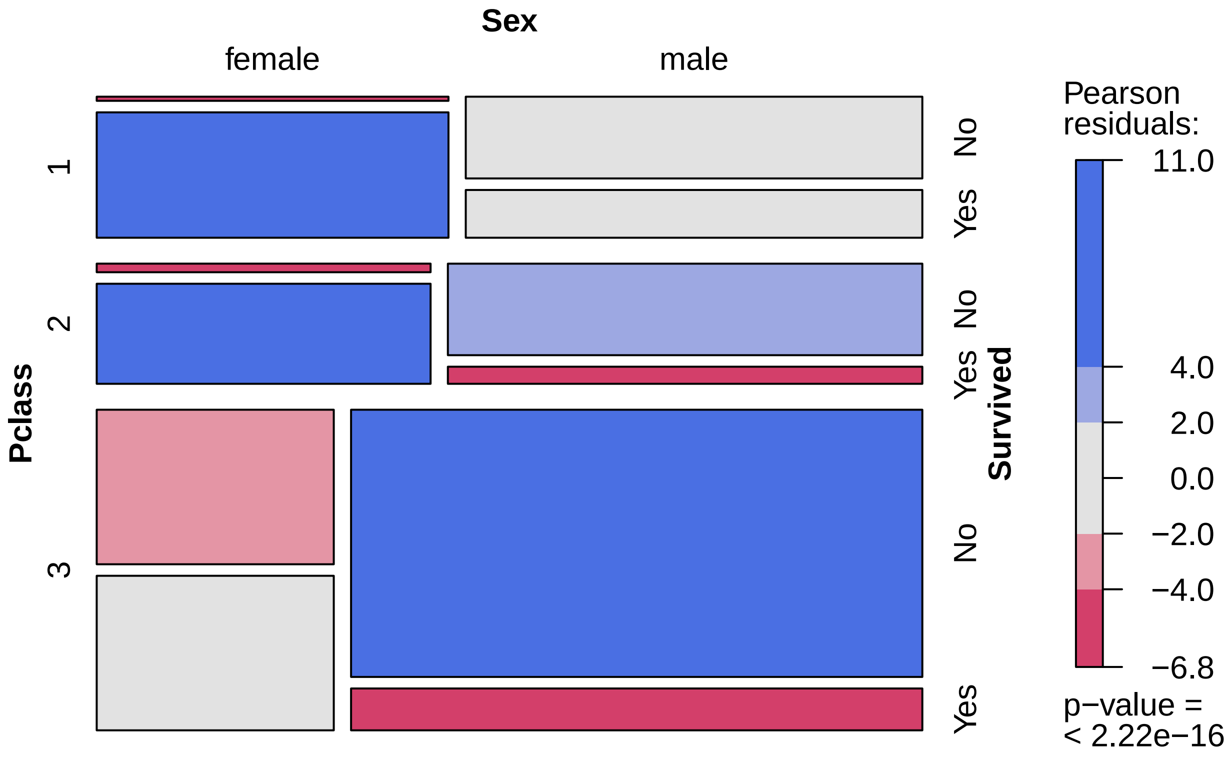
Correlation Plot

```
tbl_corr <- full_df %>%
  filter(set=="train") %>%
  select(-PassengerId, -SibSp, -Parch) %>%
  select_if(is.numeric) %>%
  cor(use="complete.obs") %>%
  corrplot.mixed(tl.cex=0.7)
```



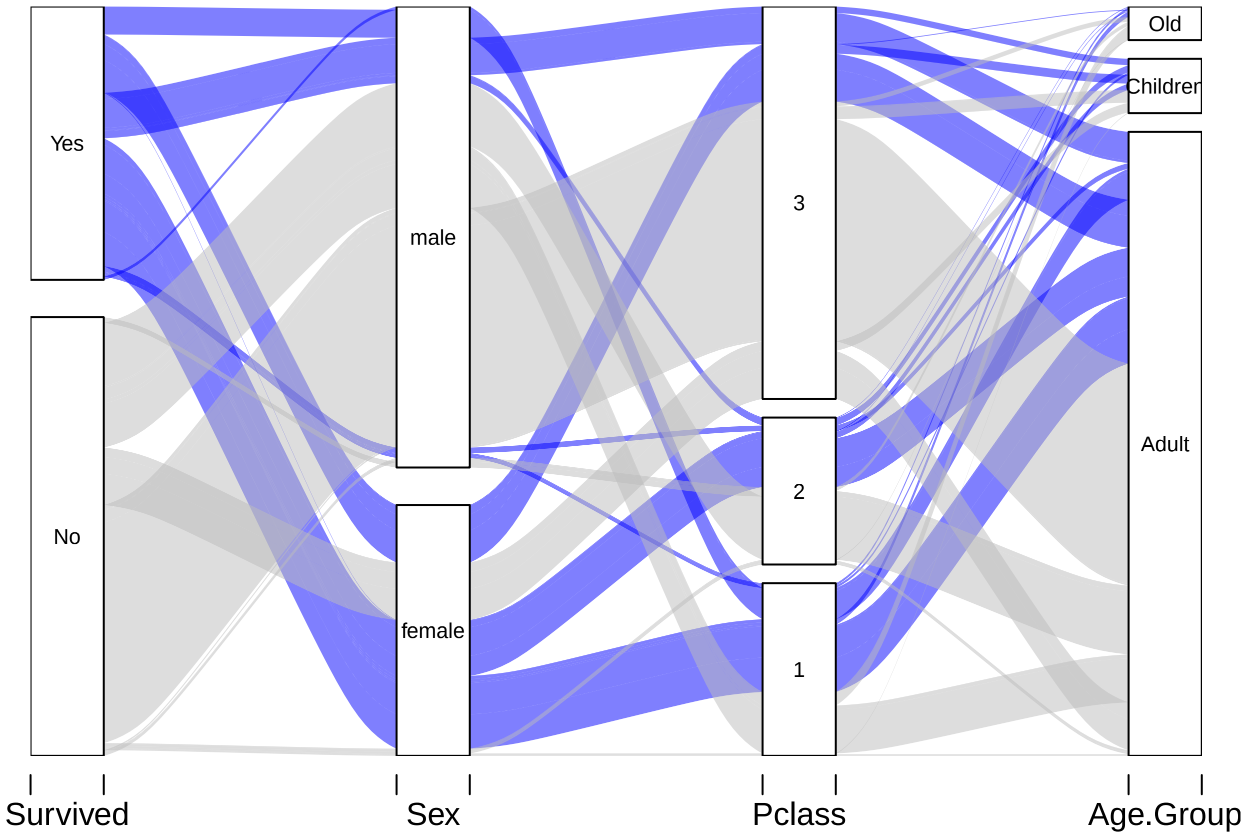
Mosaic Plot

```
tbl_mosaic <- full_df %>%
  filter(set=="train") %>%
  select(Survived, Pclass, Sex, AgeGroup=`Age Group`, title, Embarked, `FamilySize`) %>%
  mutate_all(as.factor)

mosaic(~Pclass+Sex+Survived, data=tbl_mosaic,shade = T, colorise = T, legend =T)
```



Alluvial Diagram

```
tbl_summary <- full_df %>%
  filter(set=="train") %>%
  group_by(Survived, Sex, Pclass, `Age Group`, title) %>%
  summarise(N = n()) %>%
  ungroup %>%
  na.omit

alluvial(tbl_summary[, c(1:4)],
        freq=tbl_summary$N, border=NA,
         col=ifelse(tbl_summary$Survived == "Yes", "blue", "Grey"),
         cex=0.65,
         ordering = list(
           order(tbl_summary$Survived, tbl_summary$Pclass==1),
           order(tbl_summary$Sex, tbl_summary$Pclass==1),
           NULL,
           NULL))
```



2. Machine learning algorithm

Prepare and keep data set.

Lets prepare and keep data in the proper format

```
full_df$Pclass <- as.factor(full_df$Pclass)

feature1 <- full_df[1:891, c("Pclass","Sex","Age Group","title", "ticket.size")]

response <- as.factor(train_df$Survived)
feature1$Survived=as.factor(train_df$Survived)
feature1$Survived=as.factor(train_df$Survived)
```

For Cross validation purpose will keep 20% of data aside from orginal train set This is just to check how well my data works for unseen data

```
set.seed(100)

ind= createDataPartition(feature1$Survived,times=1,p=0.8,list=FALSE)
train_val <- feature1[ind,]
test_val <- feature1[-ind,]

train_val$Sex <- as.factor(train_val$Sex)
test_val$Sex <- as.factor(test_val$Sex)
```

check the proprtion of Survival rate in orginal training data, current traing and testing data

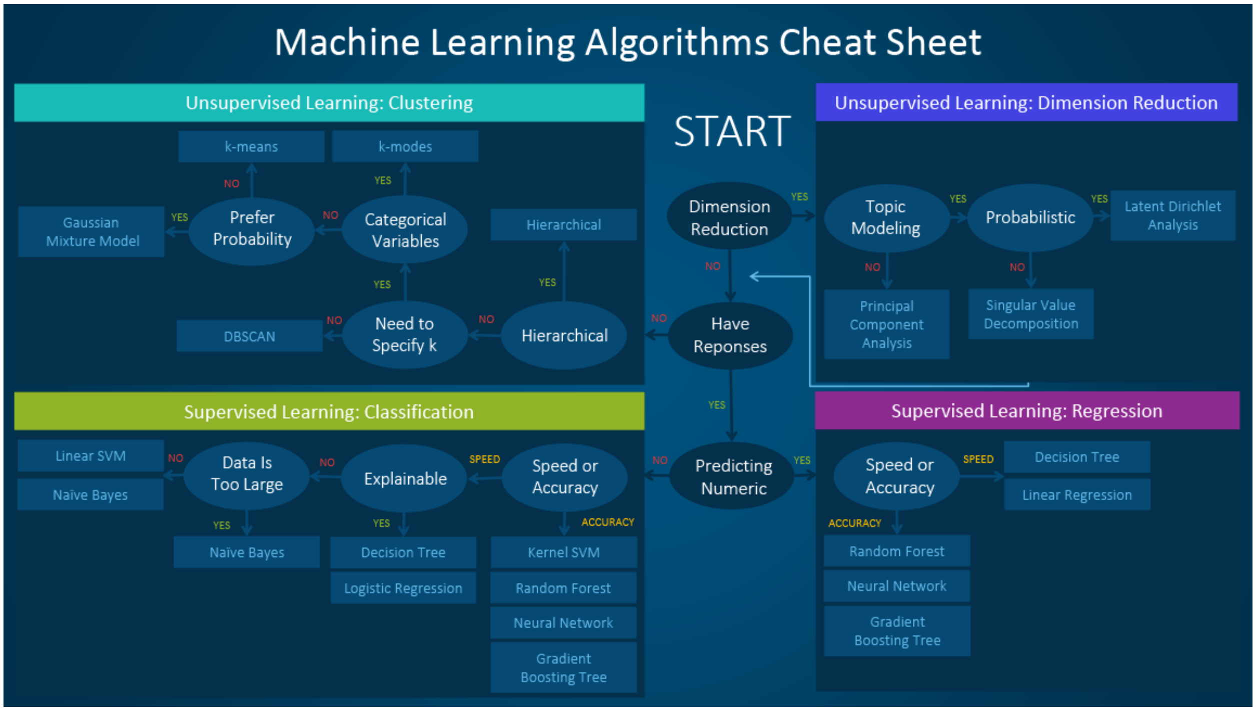


Figure 1: the scheme I followed in analyzing the data

```
round(prop.table(table(train_df$Survived)*100),digits = 1)
```

```
##
##   0   1
## 0.6 0.4
```

```
round(prop.table(table(train_val$Survived)*100),digits = 1)
```

```
##
##   0   1
## 0.6 0.4
```

```
round(prop.table(table(test_val$Survived)*100),digits = 1)
```

```
##
##   0   1
## 0.6 0.4
```

2.1 logistic regression

```
contrasts(train_val$Sex)
```

```
##        male
## female    0
## male      1
```

```
contrasts(train_val$Pclass)
```

```
##   2 3
```

```
## 1 0 0
## 2 1 0
## 3 0 1
```

Lets run Logistic regression model

```
log.mod <- glm(Survived ~ ., family = binomial(link=logit), data = train_val)
summary(log.mod)
```

```
##
## Call:
## glm(formula = Survived ~ ., family = binomial(link = logit),
##     data = train_val)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3888  -0.6013  -0.4202   0.5940   2.9956
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)           16.5632   623.9700   0.027   0.9788
## Pclass2               -1.3928     0.3269  -4.260 2.04e-05 ***
## Pclass3               -2.2153     0.2864  -7.735 1.03e-14 ***
## Sexmale              -15.4646   623.9696  -0.025   0.9802
## `Age Group`Children   0.3837     0.4851   0.791   0.4290
## `Age Group`Old       -1.0493     0.5243  -2.001   0.0454 *
## titleMiss            -15.8624   623.9699  -0.025   0.9797
## titleMr               -3.3587     0.6378  -5.266 1.39e-07 ***
## titleMrs             -15.5592   623.9700  -0.025   0.9801
## titleOfficer          -4.0610     0.9222  -4.404 1.06e-05 ***
## ticket.sizeSingle      2.0930     0.4410   4.746 2.08e-06 ***
## ticket.sizeSmall       2.0343     0.4081   4.984 6.22e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 950.86  on 713  degrees of freedom
## Residual deviance: 595.05  on 702  degrees of freedom
## AIC: 619.05
##
## Number of Fisher Scoring iterations: 13
```

```
confint(log.mod)
```

```
## Waiting for profiling to be done...

##                          2.5 %      97.5 %
## (Intercept)         -80.397766          NA
## Pclass2              -2.046239 -0.76235768
## Pclass3              -2.788149 -1.66348884
## Sexmale                     NA 82.05767291
## `Age Group`Children  -0.553509  1.35888345
## `Age Group`Old       -2.122140 -0.05783063
## titleMiss                   NA 81.24853765
## titleMr              -4.643307 -2.12678755
```

```
## titleMrs                          NA 81.40687990
## titleOfficer          -5.992963 -2.32911519
## ticket.sizeSingle      1.254879  2.98859246
## ticket.sizeSmall       1.255321  2.86076478
```

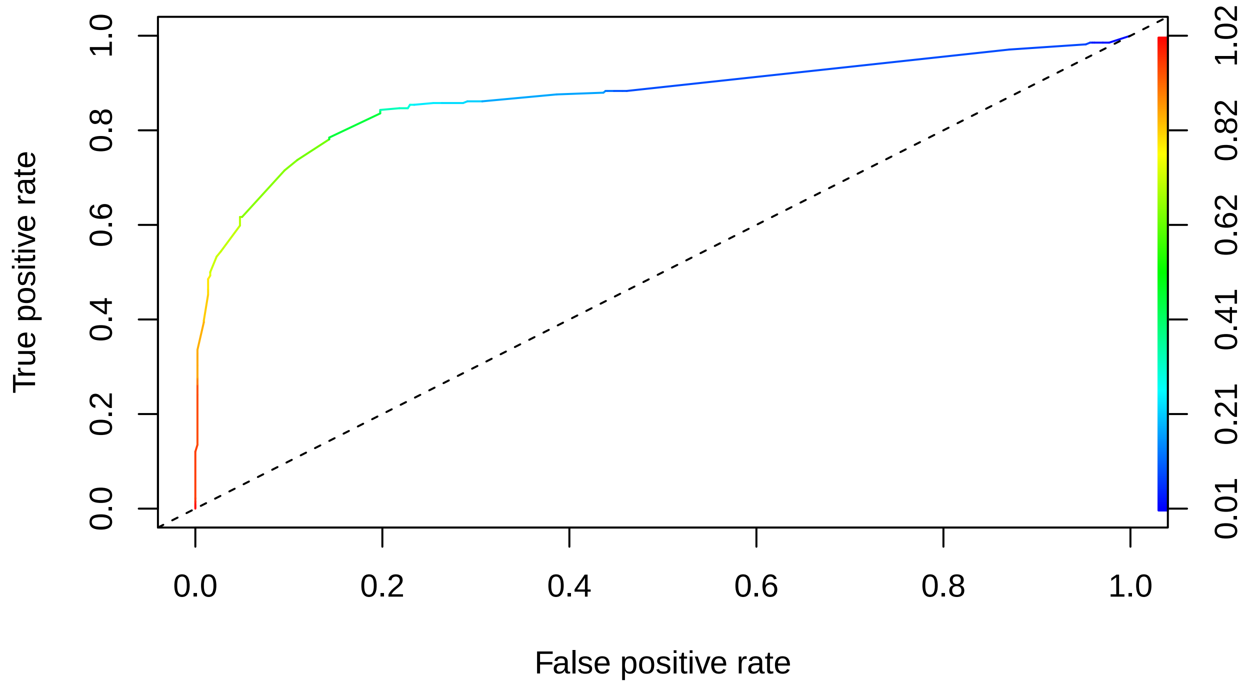
```
logreg_prediction <- predict(log.mod, data=train_val,type =  "response")
table(train_val$Survived, logreg_prediction > 0.5)
```

```
##
##     FALSE TRUE
##   0   392   48
##   1    72  202
```

```
(392+202)/(392+202+72+48)
```

```
## [1] 0.8319328
```

```
pred_fit1 <- prediction(logreg_prediction, train_val$Survived)
perf_fit1 <- performance(pred_fit1,"tpr","fpr")

plot(perf_fit1, colorize=T ,lwd=1)
par(new=TRUE)
abline(a=0, b=1,lty=2)
```




```
auc  <- performance(pred_fit1, measure = "auc")
auc
```

```
## An object of class "performance"
## Slot "x.name":
## [1] "None"
##
## Slot "y.name":
## [1] "Area under the ROC curve"
##
## Slot "alpha.name":
```

```
## [1] "none"
##
## Slot "x.values":
## list()
##
## Slot "y.values":
## [[1]]
## [1] 0.869679
##
##
## Slot "alpha.values":
## list()
```

Lets check it in test

```r
logreg_prediction <- predict(log.mod, newdata=test_val,type =  "response")

table(test_val$Survived,logreg_prediction > 0.5)
```

```
##
##     FALSE TRUE
##   0   100    9
##   1    20   48
```

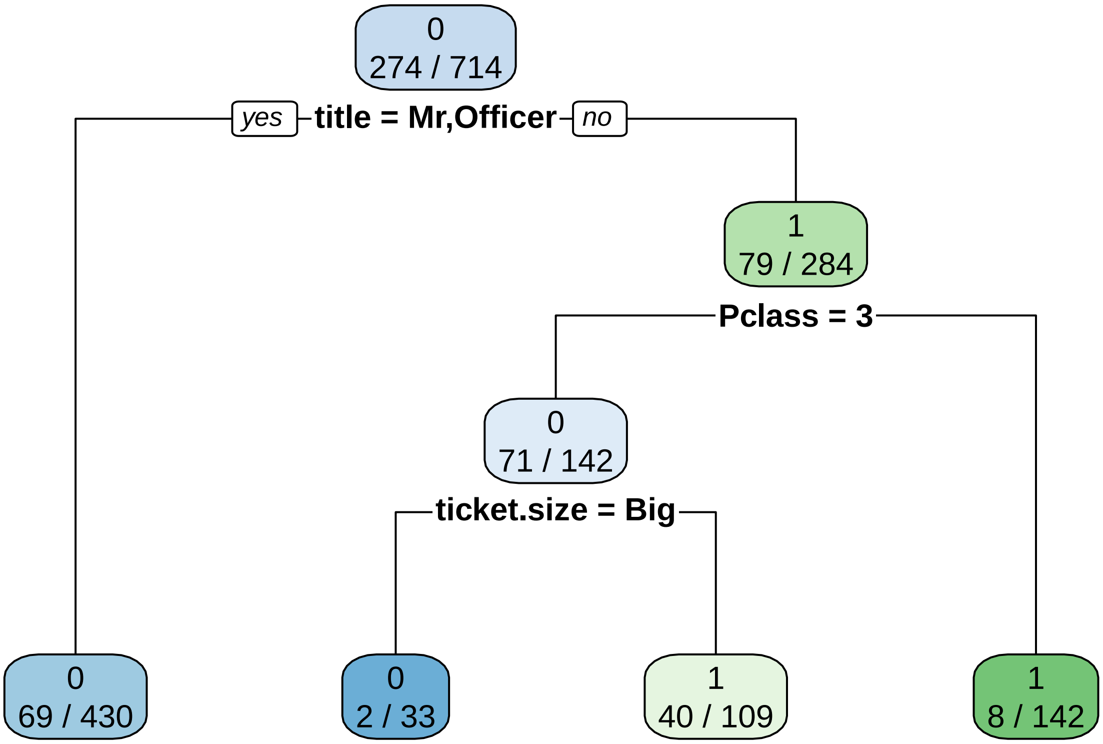
```r
logreg_result <- (100+48)/(100+48+20+9)
logreg_result
```

```
## [1] 0.8361582
```

Accuracy rate of test data is 0.83 Let's remove non significant variables and and make the model again

2.2 Decision tree

```r
set.seed(123)

decision_tree <- rpart(Survived~., data = train_val, method="class")
rpart.plot(decision_tree,extra = 3, fallen.leaves = T)
```



Lets Predict train data and check the accuracy of single tree

```
pred_dt <-  predict(decision_tree, data = train_val, type="class")

confusionMatrix(pred_dt, train_val$Survived)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 392  71
##          1  48 203
##
##                Accuracy : 0.8333
##                  95% CI : (0.8039, 0.8599)
##     No Information Rate : 0.6162
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.6419
##
##  Mcnemar's Test P-Value : 0.04372
##
##             Sensitivity : 0.8909
##             Specificity : 0.7409
##          Pos Pred Value : 0.8467
##          Neg Pred Value : 0.8088
##              Prevalence : 0.6162
##          Detection Rate : 0.5490
##    Detection Prevalence : 0.6485
##       Balanced Accuracy : 0.8159
##
```

```
##        'Positive' Class : 0
##
```

```
pred_dt_test <- predict(decision_tree, newdata = test_val, type="class")

confusionMatrix(pred_dt_test,test_val$Survived)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 100  20
##          1   9  48
##
##                Accuracy : 0.8362
##                  95% CI : (0.7732, 0.8874)
##     No Information Rate : 0.6158
##     P-Value [Acc > NIR] : 1.38e-10
##
##                   Kappa : 0.6429
##
##  Mcnemar's Test P-Value : 0.06332
##
##             Sensitivity : 0.9174
##             Specificity : 0.7059
##          Pos Pred Value : 0.8333
##          Neg Pred Value : 0.8421
##              Prevalence : 0.6158
##          Detection Rate : 0.5650
##    Detection Prevalence : 0.6780
##       Balanced Accuracy : 0.8117
##
##        'Positive' Class : 0
##
```

```
dt_result <- (100+48)/(100+48+20+9)
dt_result
```

```
## [1] 0.8361582
```

2.3 Catbost

CatBoost is an algorithm for gradient boosting on decision trees.

Prepare a dataset using the catboost.load_pool function:

```
library(catboost)

feature1$Survived <- train_df$Survived
feature1[c("Pclass","Sex","Age Group","title", "ticket.size")] <-
  lapply(feature1[c("Pclass","Sex","Age Group","title", "ticket.size")], factor)

train_pool <- catboost.load_pool(data = feature1[,-6], label = unlist(feature1[,6]))
```

Train the model using the catboost.train function:

```r
catboost_model <- catboost.train(train_pool,
    params = list(loss_function = 'Logloss', iterations = 100, metric_period=10))
```

```
## Learning rate set to 0.126165
## 0:   learn: 0.6482775    total: 63.3ms   remaining: 6.27s
## 10:  learn: 0.4379171    total: 192ms    remaining: 1.55s
## 20:  learn: 0.4040429    total: 295ms    remaining: 1.11s
## 30:  learn: 0.3899650    total: 379ms    remaining: 844ms
## 40:  learn: 0.3837949    total: 446ms    remaining: 642ms
## 50:  learn: 0.3802491    total: 509ms    remaining: 489ms
## 60:  learn: 0.3764078    total: 597ms    remaining: 381ms
## 70:  learn: 0.3653554    total: 719ms    remaining: 294ms
## 80:  learn: 0.3588287    total: 842ms    remaining: 197ms
## 90:  learn: 0.3524785    total: 971ms    remaining: 96ms
## 99:  learn: 0.3461828    total: 1.09s    remaining: 0us
## Dataset is provided, but PredictionValuesChange feature importance don't use it, since non-empty Lea
```

Apply the trained model using the catboost.predict function:

```r
test_val <- feature1[-ind,]

real_pool <- catboost.load_pool(data = test_val[,-6], label = unlist(test_val[,6]))

catboost_prediction <- catboost.predict(catboost_model, real_pool, prediction_type = 'Probability')

print(catboost_prediction)
```

```
##   [1] 0.07346243 0.58784675 0.91700313 0.14508186 0.68133064 0.14508186
##   [7] 0.58784675 0.17666426 0.10840645 0.58784675 0.07346243 0.91285331
##  [13] 0.07346243 0.98964141 0.68133064 0.93912214 0.10840645 0.34158844
##  [19] 0.10840645 0.18552612 0.13105168 0.58784675 0.34158844 0.10840645
##  [25] 0.34158844 0.34158844 0.10840645 0.14508186 0.13105168 0.11823775
##  [31] 0.07346243 0.96118047 0.17666426 0.20146317 0.07346243 0.09669188
##  [37] 0.10840645 0.34417166 0.96541544 0.10840645 0.19012962 0.15070593
##  [43] 0.14508186 0.14508186 0.68133064 0.09669188 0.95501856 0.34417166
##  [49] 0.34158844 0.13105168 0.11823775 0.10840645 0.98972990 0.10840645
##  [55] 0.34417166 0.46827432 0.98973164 0.98973164 0.13105168 0.93912214
##  [61] 0.13105168 0.95501856 0.10840645 0.95501856 0.93908220 0.34417166
##  [67] 0.98973164 0.98964141 0.91285331 0.58784675 0.10840645 0.96473528
##  [73] 0.34158844 0.10840645 0.58784675 0.93912214 0.10840645 0.10840645
##  [79] 0.11723286 0.10840645 0.10840645 0.09669188 0.10840645 0.10840645
##  [85] 0.95501856 0.10840645 0.96541544 0.34417166 0.95501856 0.10840645
##  [91] 0.95501856 0.10840645 0.10840645 0.81173101 0.40650332 0.10840645
##  [97] 0.10840645 0.10840645 0.10840645 0.10840645 0.95501856 0.34158844
## [103] 0.98973164 0.10840645 0.09669188 0.58784675 0.95501856 0.34417166
## [109] 0.10840645 0.09669188 0.10840645 0.98973164 0.10840645 0.98972990
## [115] 0.95501856 0.58784675 0.09669188 0.34417166 0.58784675 0.91285331
## [121] 0.10840645 0.13105168 0.10840645 0.14508186 0.10840645 0.58784675
## [127] 0.34417166 0.69294379 0.09669188 0.18552612 0.18552612 0.10840645
## [133] 0.40650332 0.34417166 0.17666426 0.96541544 0.11723286 0.10840645
## [139] 0.10840645 0.14508186 0.34158844 0.14750786 0.58784675 0.10840645
## [145] 0.34417166 0.34417166 0.88160960 0.34417166 0.98638053 0.14508186
## [151] 0.16105411 0.14508186 0.17666426 0.10840645 0.98964141 0.14508186
## [157] 0.10840645 0.98973164 0.95501856 0.10840645 0.10840645 0.34417166
## [163] 0.10840645 0.13105168 0.34417166 0.68133064 0.95501856 0.10840645
```

```
## [169] 0.13105168 0.07346243 0.98973164 0.20146317 0.98188557 0.14508186
## [175] 0.10840645 0.10840645 0.14400546
```

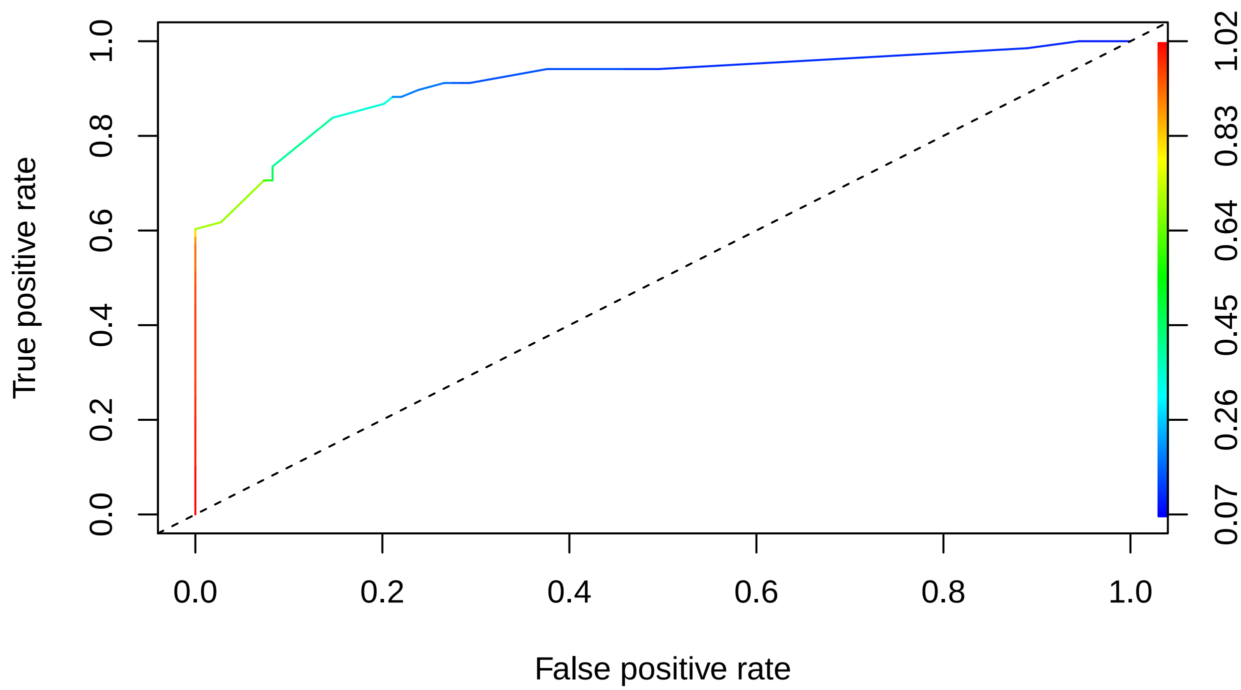
```r
table(test_val$Survived, catboost_prediction > 0.5)
```

```
##
##     FALSE TRUE
##   0   101    8
##   1    20   48
```

```r
catboost_result <- (101+48)/(101+48+20+8)
catboost_result
```

```
## [1] 0.8418079
```

```r
pred_fit3 <- prediction(catboost_prediction, test_val$Survived)
perf_fit3 <- performance(pred_fit3,"tpr","fpr")

plot(perf_fit3, colorize=T ,lwd=1)
par(new=TRUE)
abline(a=0, b=1,lty=2)
```



Result:

```r
Models <- c("Catboost","Logistic Regression","Decision Tree")
Performance <- c(catboost_result, logreg_result, dt_result)
Result <- data.frame(Models, Performance)
Result
```

```
##                Models Performance
## 1            Catboost   0.8418079
## 2 Logistic Regression   0.8361582
## 3       Decision Tree   0.8361582
```

Catboost is the best model!