CYO Project - Titanic

Osman CALISIR

1. Introduction

I choose Titanic as part of my CYO for Data Science Program. The dataset comes from Kaggle as part of introduction to Kaggle prediction competitions. Titanic sank on April 15, 1912 after colliding with iceberg killing 1502 out of 2224 passengers and crew. In this challenge, I will show you the details of the passengers who died and who don't.

In this challenge, almost all the structure is done as much simpler as it can be and it will be easier to follow by others for peer grading since most people are quite new with machine learning.

Please click here (https://www.kaggle.com/c/titanic/data) for details of the definitions.

The dataset contains 9 features as per below to predict label ***survival*** :

- pclass
- sex
- Age
- sibsp
- parch
- ticket
- fare
- cabin
- embarked

In the project, these will be shown:

- Ggplot2
- Summarising with Dplyr
- Caret Package
- Random Forest
- Cross Validation
- Confusion Matrix
- Data Wrangling
- RMarkdown

2.Dataset

The dataset for this exercise comes from Kaggle and attached for the peer grading. The file is:

- trainData - for training the model

```
#setwd("/Titanic")
download <- read.csv("trainData.csv")
```

A check on the download data showing the classes of each feature and number of observations and variables.

```
str(download)
```

```
## 'data.frame':    891 obs. of  12 variables:
##  $ PassengerId: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Survived   : int  0 1 1 1 0 0 0 0 1 1 ...
##  $ Pclass     : int  3 1 3 1 3 3 1 3 3 2 ...
##  $ Name       : Factor w/ 891 levels "Abbing, Mr. Anthony",..: 109 191 358 277 16 559 520
629 417 581 ...
##  $ Sex        : Factor w/ 2 levels "female","male": 2 1 1 1 2 2 2 2 1 1 ...
##  $ Age        : num  22 38 26 35 35 NA 54 2 27 14 ...
##  $ SibSp      : int  1 1 0 1 0 0 0 3 0 1 ...
##  $ Parch      : int  0 0 0 0 0 0 0 1 2 0 ...
##  $ Ticket     : Factor w/ 681 levels "110152","110413",..: 524 597 670 50 473 276 86 396 3
45 133 ...
##  $ Fare       : num  7.25 71.28 7.92 53.1 8.05 ...
##  $ Cabin      : Factor w/ 148 levels "","A10","A14",..: 1 83 1 57 1 1 131 1 1 1 ...
##  $ Embarked   : Factor w/ 4 levels "","C","Q","S": 4 2 4 4 4 3 4 4 4 2 ...
```

2.1 Reviewing Data from Kaggle

After importing the data, what I see is data wrangling required.

- Survived and Pclass are currently integer unless converted to factors
- The Age feature has 177 NAs as can seen below.

```
summary(download$Age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    0.42   20.12   28.00   29.70   38.00   80.00     177
```

- Fare has some outliers and will be explained later on in this report.

```
summary(download$Fare)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    7.91   14.45   32.20   31.00  512.33
```

- There are two missing embarkation points that need to be fixed.

```
summary(download$Embarked)
```

```
##       C   Q   S
##   2 168  77 644
```

2.2 Converting to Factors

Both of these features (Survived and Pclass) have been converted to factors as per below code.

```
# convert train dataset
download$Survived <- factor(download$Survived)
download$Pclass <- factor(download$Pclass)
```

Quick check on both features showing they are both factors.

```
str(download$Survived)
```

```
##  Factor w/ 2 levels "0","1": 1 2 2 2 1 1 1 1 2 2 ...
```

```
str(download$Pclass)
```

```
##  Factor w/ 3 levels "1","2","3": 3 1 3 1 3 3 1 3 3 2 ...
```

2.3 Removing NAs

Both Age and Fare contain NAs values that have been replaced with mean values.

```
# fix NAs for Age
avg <- mean(download$Age, na.rm = TRUE)
download$Age <- replace(download$Age,is.na(download$Age),avg)

# fix NAs for Fare
avg <- mean(download$Fare, na.rm = TRUE)
download$Fare <- replace(download$Fare,is.na(download$Fare),avg)
```

Running summary function for both features showing no more NAs.

```
summary(download$Age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.42   22.00   29.70   29.70   35.00   80.00
```

```
summary(download$Fare)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    7.91   14.45   32.20   31.00  512.33
```

2.4 Replacing Missing Embarked

Because most people from Southampton, I will now convert the two rows to default value as S

```
download$Embarked[!(download$Embarked %in% c('C','Q','S'))] <- 'S'
summary(download$Embarked)
```

```
##       C   Q   S
##   0 168  77 646
```

2.5 Partitioning Data

Now, time to split the data to 80% training and 20% test. The below dim function confirms our dataset has now been partitioned.

```
# create partition on test data
set.seed(1)
index <- createDataPartition(download$Survived,times = 1,p=0.8,list=FALSE)
train <- download[index,]
test <- download[-index,]

# check the dimesions
dim(download)
```

```
## [1] 891  12
```

```
dim(train)
```

```
## [1] 714  12
```

```
dim(test)
```

```
## [1] 177  12
```

3. Visualisation

We know a lot of people did not survive this tragedy and we can use table function to see the actual numbers in our dataset. In actual dataset, 0 means died and 1 means survived.

```
#train %>% group_by(Survived) %>% summarise(Count=n()) %>% knitr::kable()
table(train$Survived)
```

```
##
##   0   1
## 440 274
```

However, we need to get more more insights on what sort of people survived this tragedy. ggplot will be using to drill down further on different features to get more insights.

3.1 Sex

There is a quote in the Kaggle that says women and kids were given priority to lifeboats. I will show the statistics from the dataset to prove this statement.

```
num <- table(train$Sex,train$Survived)
pct <- round(prop.table(table(train$Sex,train$Survived),1),3) * 100
tbl <- cbind(num,pct)
colnames(tbl) <- c('Died','Survived','Died (%)','Survived (%)')
tbl
```

```
##        Died Survived Died (%) Survived (%)
## female   60      188     24.2         75.8
## male    380       86     81.5         18.5
```

As you can see above, only 18.8% of men survived compared to 75.3% of women. If you happened to be in Titanic at that time as a male, your chance is not that great!

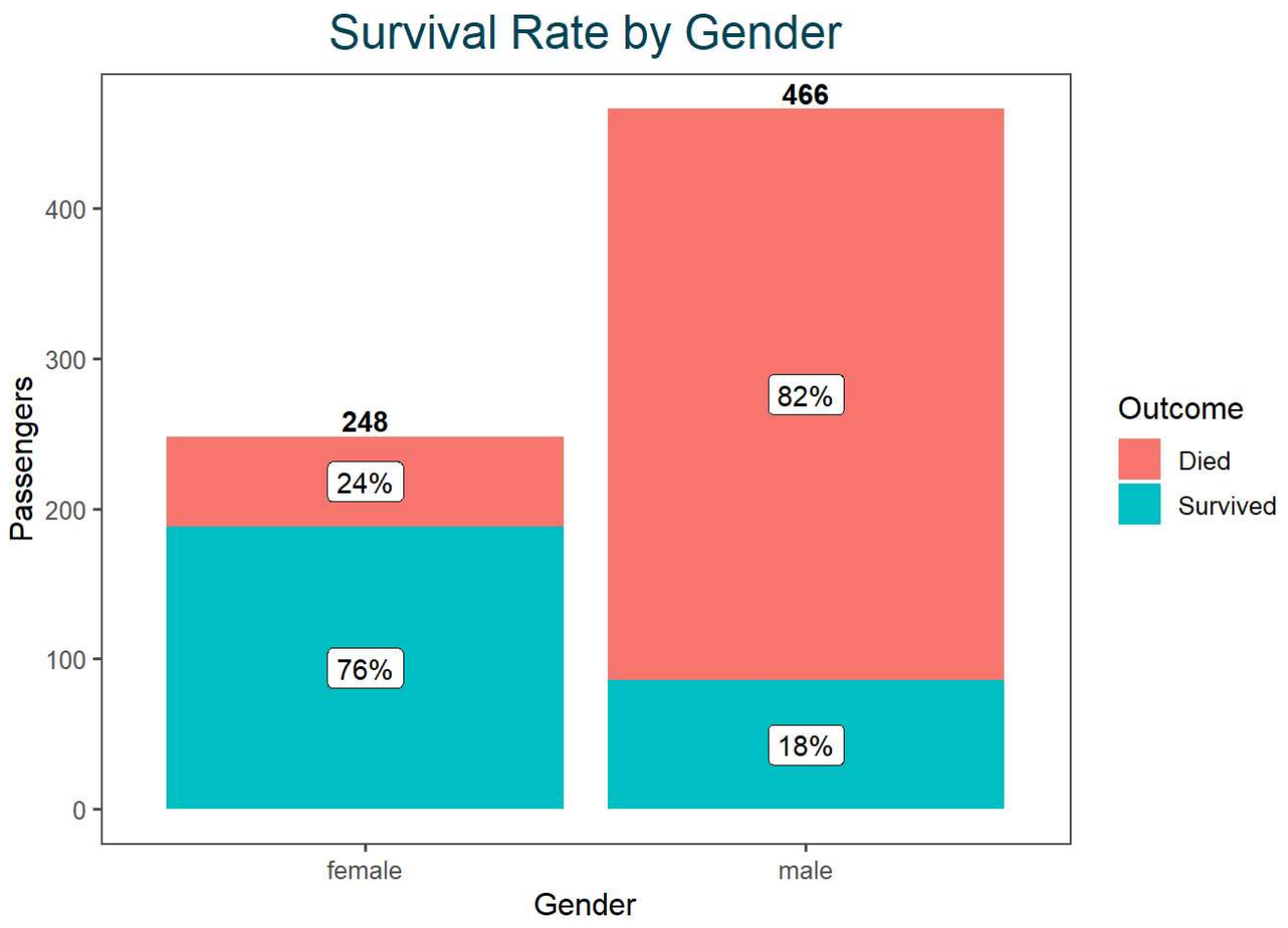
Let's see if we can drill down to kids category just to prove the above assumption. I make assumption kids as everyone under 15 years.

```
kids <- train %>% filter(Age<=15)
num <- table(kids$Sex,kids$Survived)
pct <- round(prop.table(table(kids$Sex,kids$Survived),1),3) * 100
tbl <- cbind(num,pct)
colnames(tbl) <- c('Died','Survived','Died (%)','Survived (%)')
tbl
```

```
##        Died Survived Died (%) Survived (%)
## female   11       23     32.4         67.6
## male     16       18     47.1         52.9
```

We can see male kids had 54.8% chance to survive compared to overall male of 18.8%. It does prove the assumption that women and kids were given priority to life support.

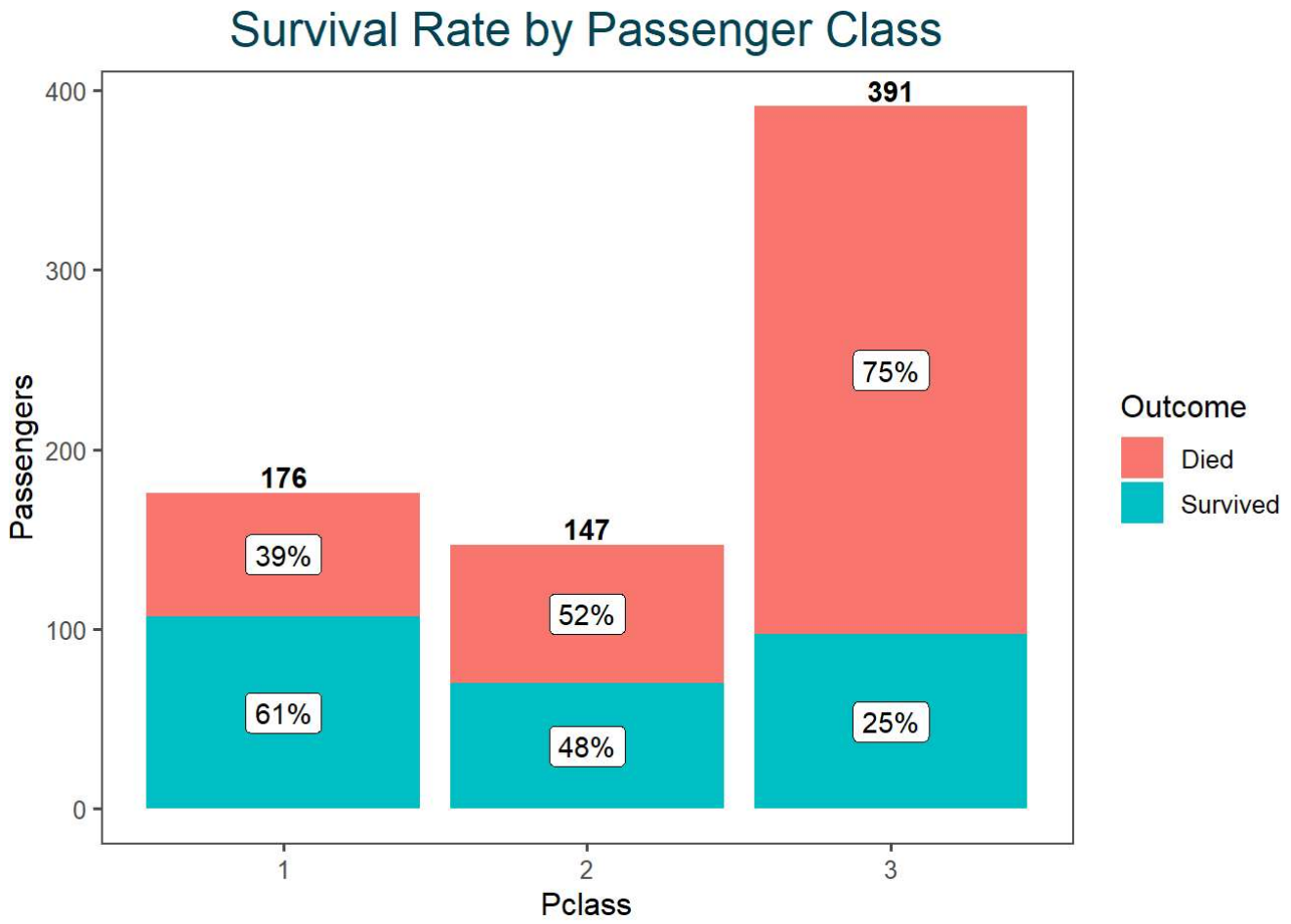
Let's go back analysing the feature sex and the below plot helps to visualise the the different statistics in one plot.



The above plot gives us enough statistic to what we have discussed above.

3.2 Pclass

Pclass is passenger class containing of 3 factors : 1,2 and 3. Passengers in class-1 pay more fees than class-2 and class-3. I also expect higher classes to have better access to life boats and less crowded than lower classes.



From the above plot, we can see the following :

- 63% of passengers in class-1 survived
- 47% of passengers in class-2 survived
- 25% of passengers in class-3 survived

It does validate the assumptions as we can see if you happened to be in Titanic at that time and bought a ticket in passenger class 3, then you were most likely not going to survive.

3.3 Family

In Titanic dataset, there are two features that are quite similar :

- Parch - travelling with parents or children
- Sibsp - travelling with sibbling or spouses

I am going to create a new feature called ***FamilySize*** which is the sum of ***Parch*** and ***SibSp***. I also added 1 in the end which is basically that person itself otherwise you just count your other family members excluding you.

```
# group to determine family/single
train$FamilySize <- train$Parch + train$SibSp + 1

# work out the composition of the data
a <- prop.table(table(train$FamilySize)) * 100
b <- table(train$FamilySize)
rbind(Count = b,Pctg = round(a,1))
```

```
##           1     2    3    4    5    6   7   8  11
## Count 429.0 127.0 85.0 23.0 11.0 18.0 9.0 6.0 6.0
## Pctg   60.1  17.8 11.9  3.2  1.5  2.5 1.3 0.8 0.8
```

From the above table we can see the following observations :

- 61.5% travelling alone
- 17.2% travelling with another family
- 11.3% travelling with another 2 family members
- The remaining travelling with more than 2 family members

Which groups had better chance to survive in this disaster? I use prop and table function again to show this.

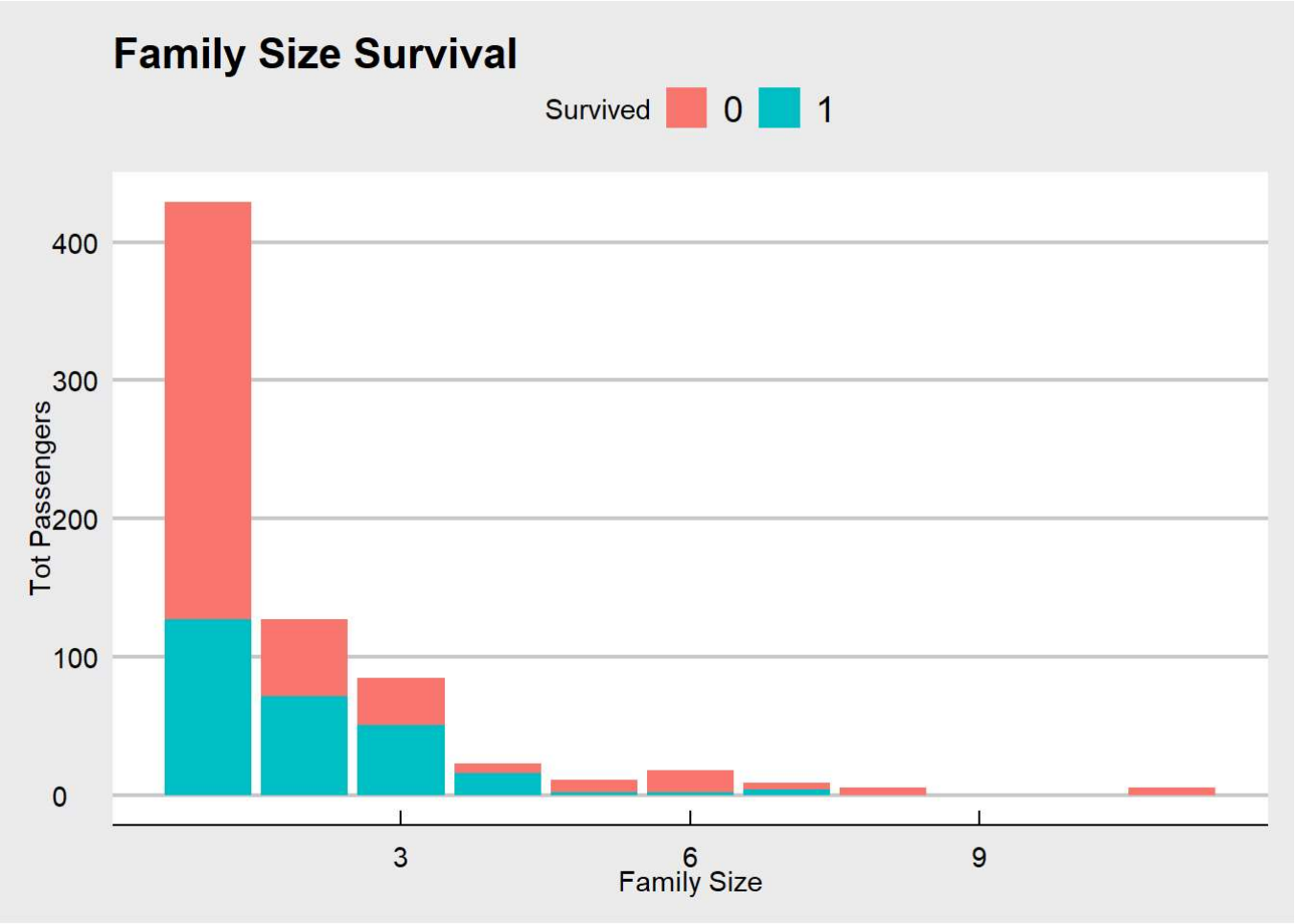
```
num <- table(train$FamilySize,train$Survived)
pct <- round(prop.table(table(train$FamilySize,train$Survived),1),3)*100
tbl <- cbind(num,pct)
colnames(tbl) <- c('Died','Survived','Died %','Survived %')
tbl
```

```
##    Died Survived Died % Survived %
## 1   302      127   70.4       29.6
## 2    55       72   43.3       56.7
## 3    34       51   40.0       60.0
## 4     7       16   30.4       69.6
## 5     9        2   81.8       18.2
## 6    16        2   88.9       11.1
## 7     5        4   55.6       44.4
## 8     6        0  100.0        0.0
## 11    6        0  100.0        0.0
```

The table above show the actual numbers on the left and percentages on the right. From there we can draw the following summary :

- Those passengers travelling in category 1,5,6 and 7 had between 16.7% to 42.9% chance to survive
- Those passengers with 2-3 family members had ~55% chance to survive
- Those passengers with 4 family members had 68% chance to survive
- Passengers travelling with 8 or more family members had 100% chance to die

```
ggplot(data = train, aes(x=FamilySize)) + geom_bar(aes(fill=Survived)) + labs(x='Family Size'
, y='Tot Passengers', title='Family Size Survival') + theme_economist_white()
```



When I plot the above statistic, we can see there were not that many families with 4 or more members. Those families who all died as per our train table probably only a small proportion of the whole dataset. Let's find out which families they were.

```
train %>% filter(FamilySize>7) %>% select(Name,Sex,Age, FamilySize, Pclass) %>% arrange(Name)
```

Name <fctr>	Sex <fctr>	Age <dbl>	FamilySize <dbl>	Pclass <fctr>
Goodwin, Master. Harold Victor	male	9.00000	8	3
Goodwin, Master. Sidney Leonard	male	1.00000	8	3
Goodwin, Master. William Frederick	male	11.00000	8	3
Goodwin, Miss. Lillian Amy	female	16.00000	8	3
Goodwin, Mr. Charles Edward	male	14.00000	8	3
Goodwin, Mrs. Frederick (Augusta Tyler)	female	43.00000	8	3
Sage, Master. Thomas Henry	male	29.69912	11	3
Sage, Miss. Constance Gladys	female	29.69912	11	3
Sage, Miss. Dorothy Edith "Dolly"	female	29.69912	11	3
Sage, Miss. Stella Anna	female	29.69912	11	3

We found that bigger families struggled to get everyone on lifeboats and they probably sticked around to each other if they did not make it. Especially if those families were also in passenger class 3.

3.4 Embarked

```
TotalPassenger <- table(train$Embarked)
Pctg <- round(prop.table(table(train$Embarked)),3)*100
Pctg <- Pctg[-1]
TotalPassenger <- TotalPassenger[-1]
rbind(TotalPassenger,Pctg)
```

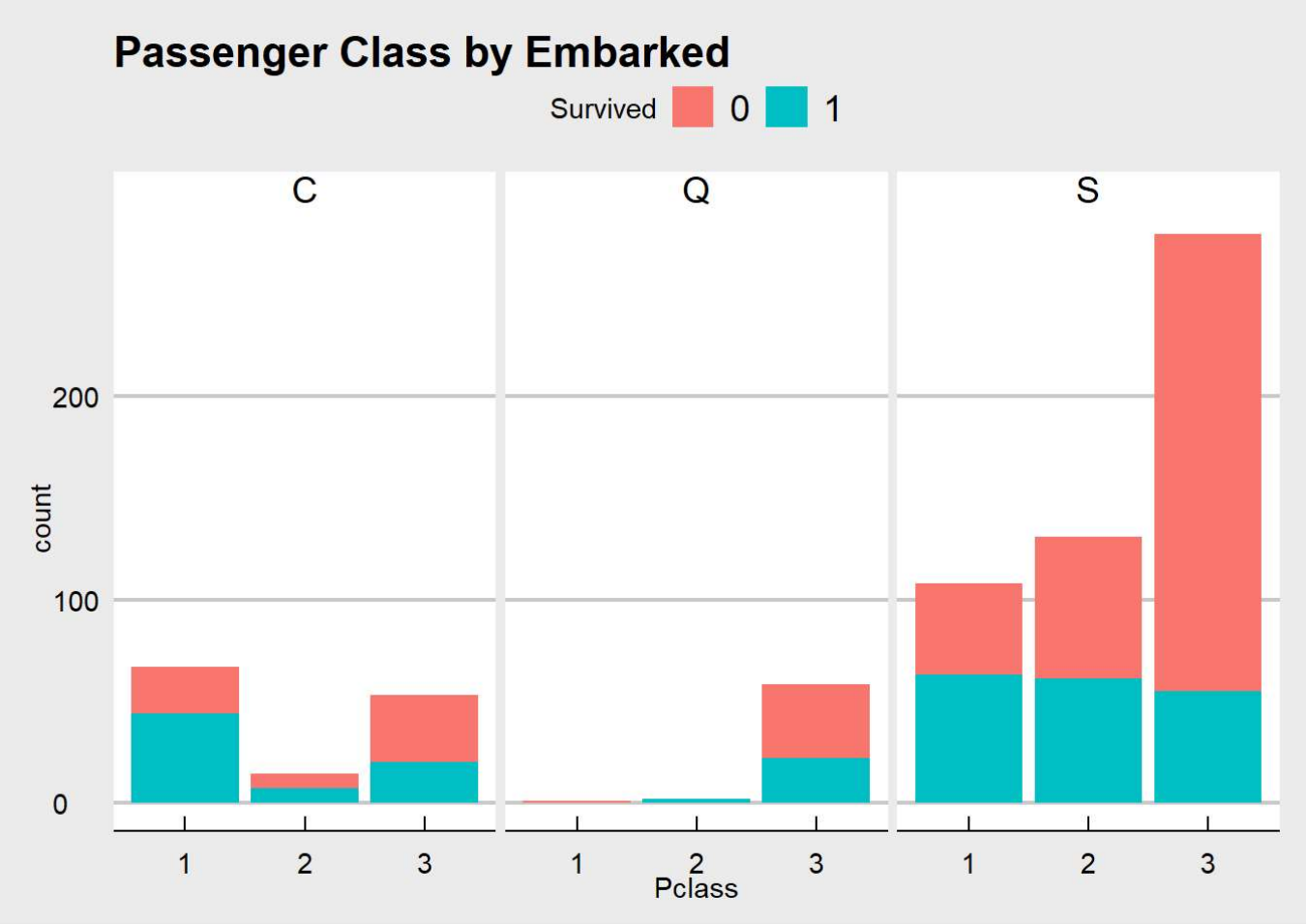
```
##                    C    Q     S
## TotalPassenger 134.0 61.0 519.0
## Pctg            18.8  8.5  72.7
```

There were three embarkation points for Titanic :

- C - Cherbourg in France
- Q - Queenstown in Ireland
- S - Southampton in England

You can see from table, most people embarked from Southampton followed by Cherbourg and Queenstown. Here is the question: Would it be logical to determine if someone survived based on his/her embarkation point?

```
train %>% ggplot(aes(x=Pclass, fill=Survived)) + geom_bar() + facet_wrap(~Embarked) + theme_e
conomist_white() + labs(title="Passenger Class by Embarked")
```



The plot above showing passengers Southampton had higher chance to die than people embarked from other ports. That is because most people from Southampton sat in class-3 and they also outnumbered passengers from other ports. It seems the passenger class is the key feature instead of where they embarked. This feature will not be used for my model to avoid overfitting by adding unecessary column for prediction.

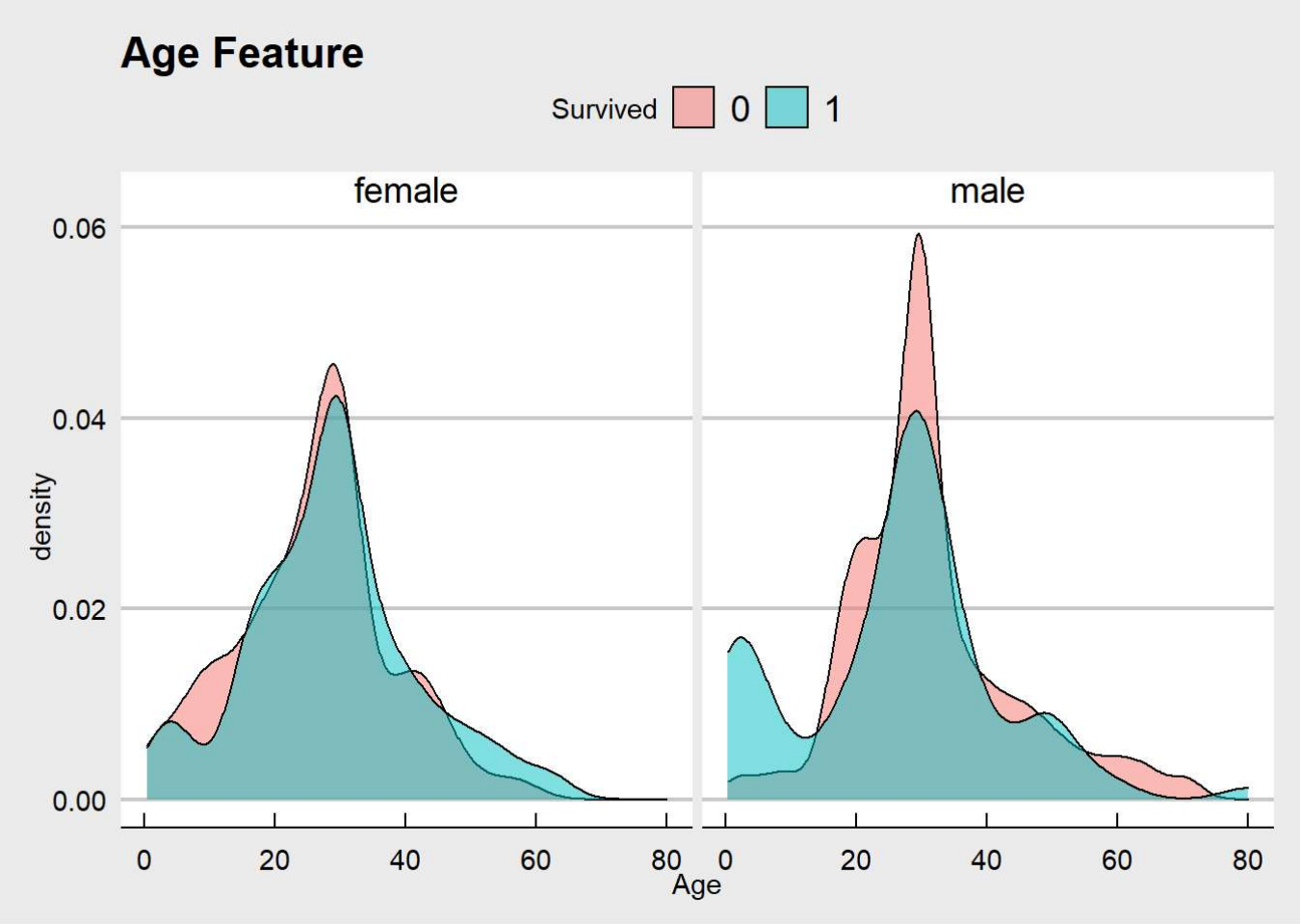
3.5 Age

The next feature is Age and need to figure out how to group the age to help machine learning use it as another categorical feature.

There are two points that I want to analyse further :

1. How many kids survive?
2. How about the older people like over 60?

```
p <- train %>%
  ggplot(aes(x = Age, fill=Survived))

p + geom_density(alpha=0.5) + facet_wrap(~Sex) + labs(title="Age Feature") + theme_economist_
white()
```



The above density plots give me the answers to my previuos questions

- More male under 15 survived
- More male over 60 died
- Strange pattern in female under 15 requires further investigation

Let's drill down the strange pattern for female under 15 using below code.

```
tmp <- train %>% filter(Age<15 & Sex=='female')
table(tmp$FamilySize,tmp$Survived)
```

```
##
##     0 1
##   1 1 2
##   2 2 4
##   3 2 8
##   4 1 4
##   5 1 0
##   6 2 0
##   7 2 1
```

```
tmp %>% filter(FamilySize>4) %>% select(Name,Pclass,FamilySize,Sex,Age) %>% arrange(desc(Fami
lySize,Name))
```

Name <fctr>	Pclass <fctr>	FamilySize <dbl>	Sex <fctr>	... <dbl>
Asplund, Miss. Lillian Gertrud	3	7	female	5
Andersson, Miss. Ingeborg Constanzia	3	7	female	9
Andersson, Miss. Sigrid Elisabeth	3	7	female	11
Skoog, Miss. Mabel	3	6	female	9
Skoog, Miss. Margit Elizabeth	3	6	female	2
Palsson, Miss. Torborg Danira	3	5	female	8

6 rows

It seems, being both a part of passenger class 3 and a part of big family as a girl is not lucky at the Titanic at all.

I will create another categorical feature for Age and call it AgeGrp with three categories :

1. Kids
2. Adults
3. Seniors

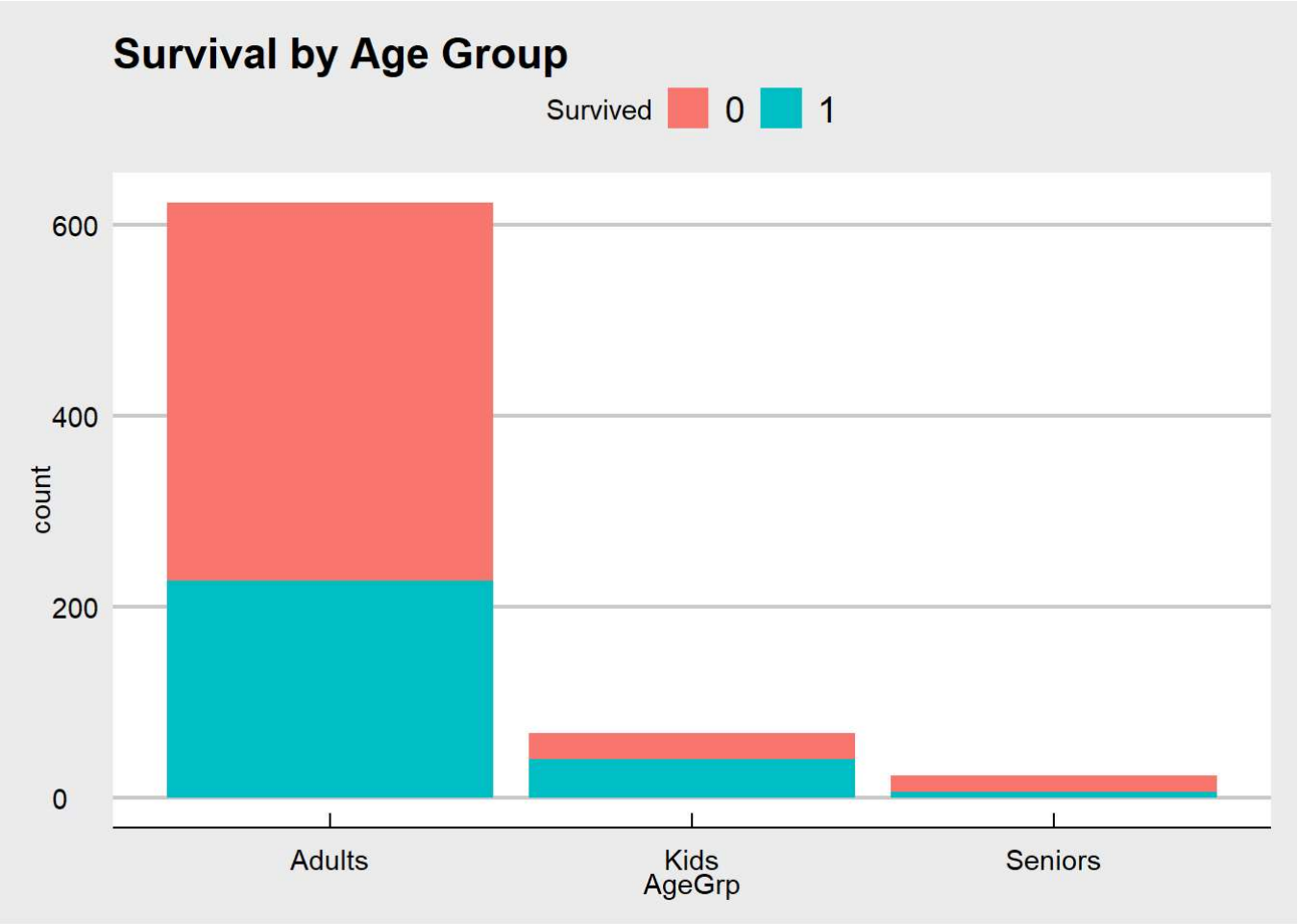
```
train$AgeGrp[train$Age<=15] <- 'Kids'
train$AgeGrp[train$Age>15 & train$Age<=59] <- 'Adults'
train$AgeGrp[train$Age>59] <- 'Seniors'
train$AgeGrp <- factor(train$AgeGrp)
```

```
a <- table(train$AgeGrp)
b <- prop.table(table(train$AgeGrp)) * 100
rbind(Count=a,Pctg=round(b,1))
```

```
##       Adults Kids Seniors
## Count  623.0 68.0    23.0
## Pctg    87.3  9.5     3.2
```

The above is the new category for age group and we can see there were not many people seniors (above 59), only 2.9% of the sample size. But there were 65 kids (under 15) on board, roughly about 9.1% of total passengers in training dataset. Based on this new grouping, I am quite satisfied it can help predicting the outcome of test dataset later on.

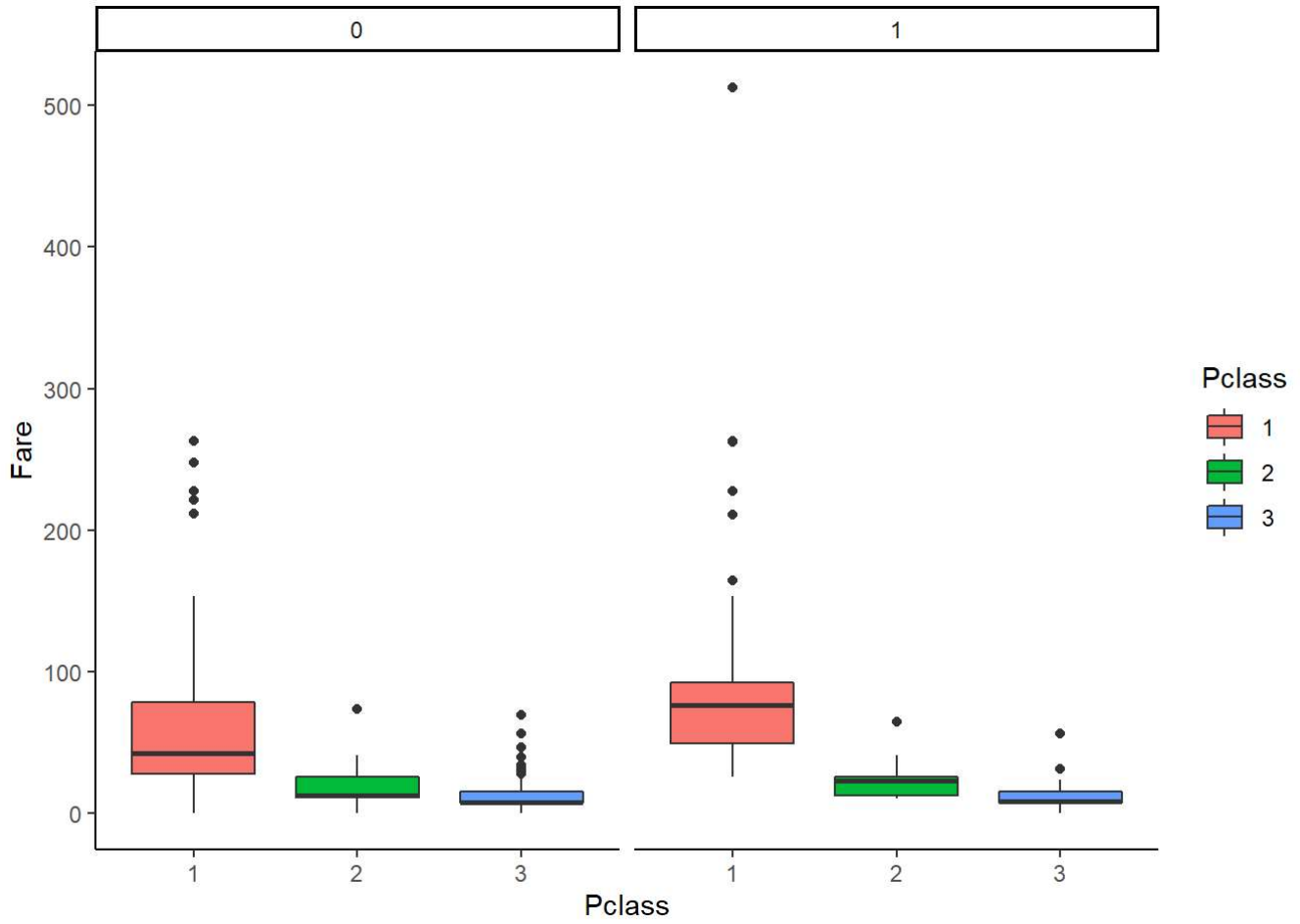
```
train %>% ggplot(aes(x=AgeGrp, fill=Survived)) + geom_bar() + theme_economist_white() + labs
(title="Survival by Age Group")
```



3.6 Fare

Fare is closely related to passenger class. Let's review this using box plot below.

```
train %>% ggplot(aes(x = Pclass, y = Fare, fill=Pclass)) + geom_boxplot() + facet_grid(~Survi
ved) + theme_classic()
```



The summary of passenger class 3 below showing 3rd quantile around 15.5 whilst the max value is 69.55. There is a huge gap caused by these outliers.

```
s <- train %>% filter(Pclass==3) %>% .$Fare
summary(s)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    7.75    8.05   13.62   15.50   69.55
```

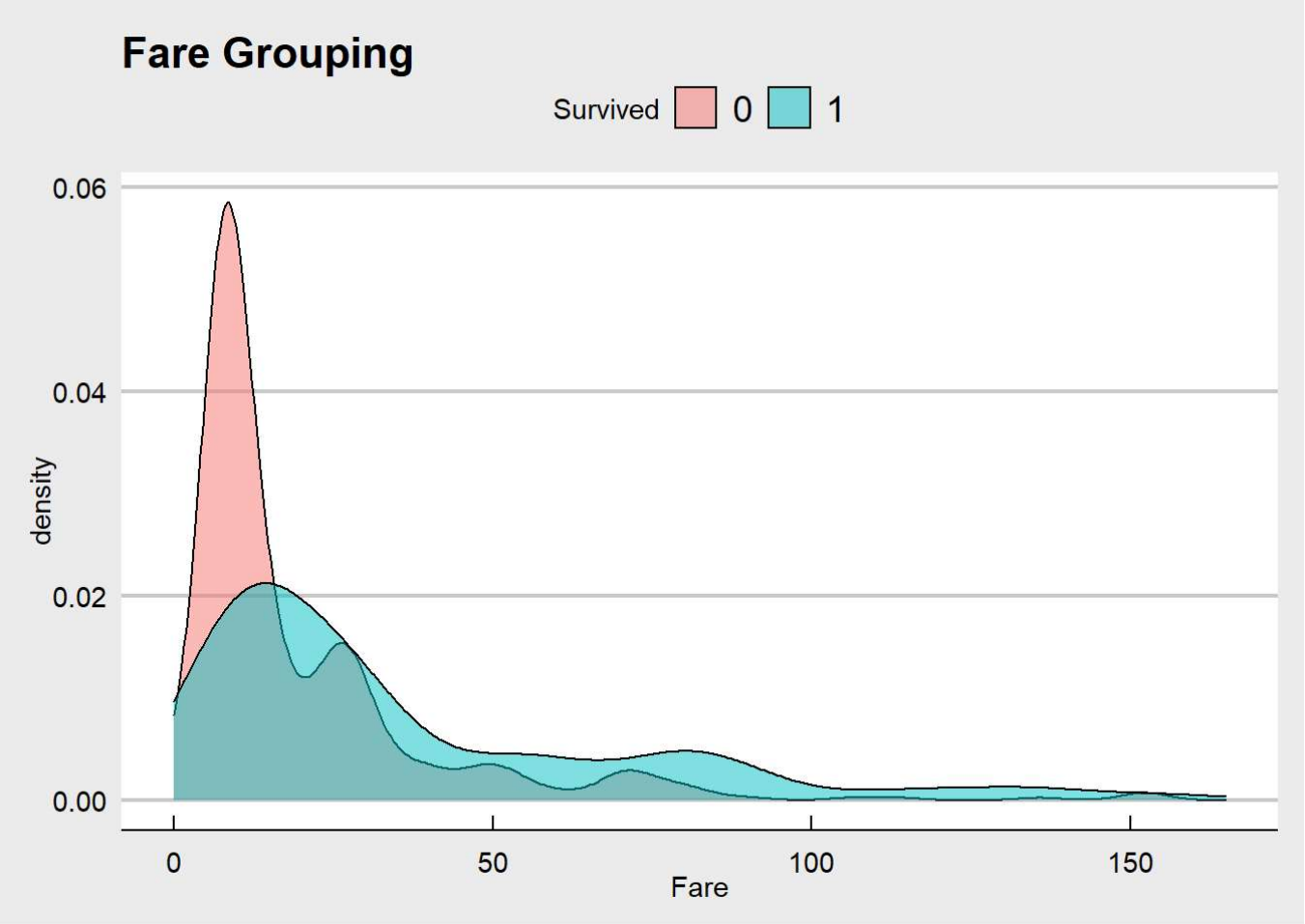
```
train %>% filter(Pclass==3 & Fare>15.50) %>% select(Name, FamilySize, Age, Embarked) %>% arra
nge(desc(FamilySize,Name))
```

Name <fctr>	**FamilySize** <dbl>	**Age** <dbl>
Sage, Master. Thomas Henry	11	29.69912
Sage, Miss. Constance Gladys	11	29.69912
Sage, Mr. Frederick	11	29.69912
Sage, Mr. George John Jr	11	29.69912
Sage, Miss. Stella Anna	11	29.69912
Sage, Miss. Dorothy Edith "Dolly"	11	29.69912
Goodwin, Master. William Frederick	8	11.00000
Goodwin, Miss. Lillian Amy	8	16.00000

Name <fctr>	FamilySize <dbl>	Age <dbl>
Goodwin, Master. Sidney Leonard	8	1.00000
Goodwin, Master. Harold Victor	8	9.00000

1-10 of 96 rows Previous **1** 2 3 4 5 6 ... 10 Next

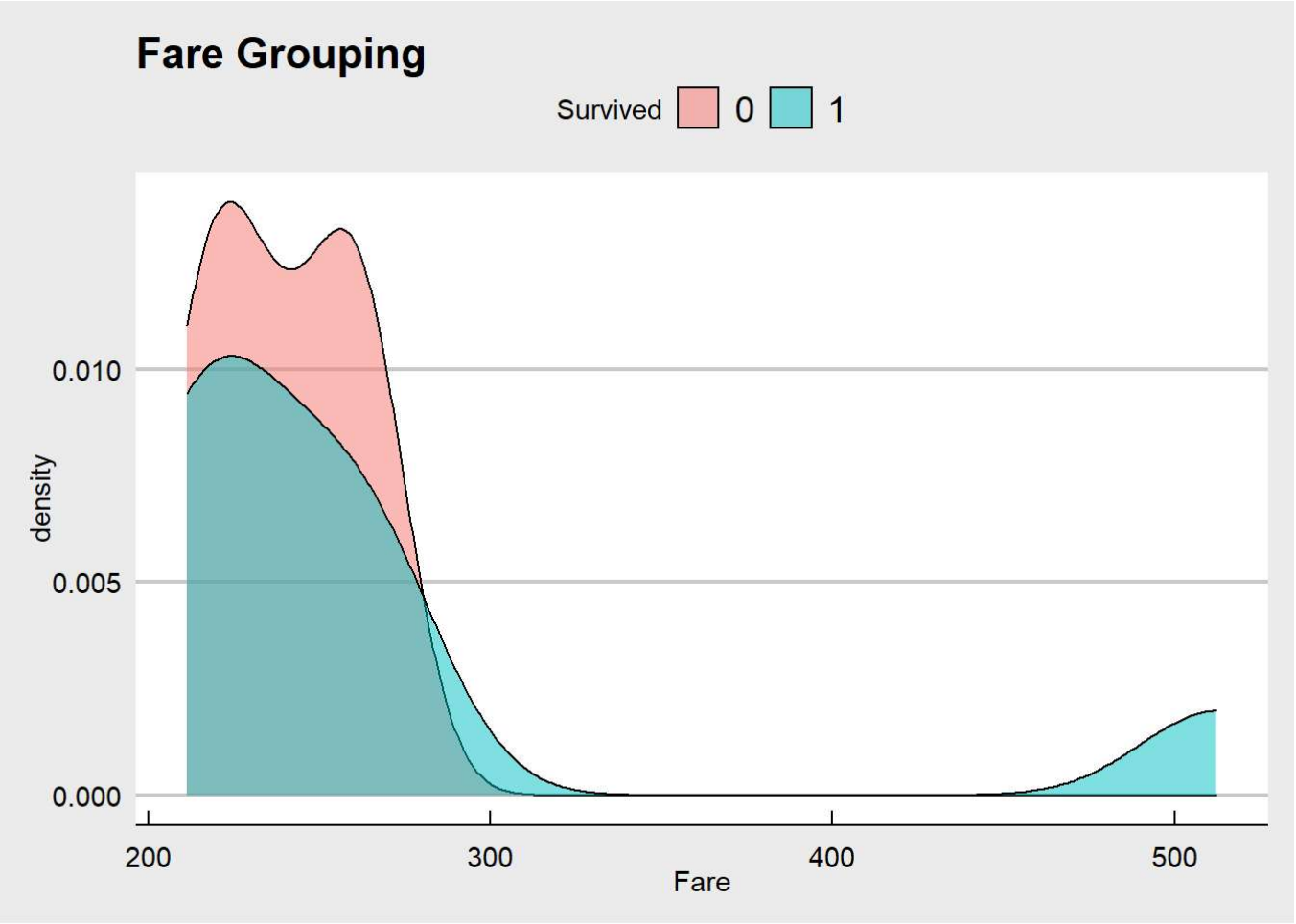
```
p <- train %>% filter(Fare<200) %>%
  ggplot(aes(x = Fare, fill=Survived))

p + geom_density(alpha=0.5) + labs(title="Fare Grouping") + theme_economist_white()
```



Based on the fare grouping plot above, we can see the threshold is around 9 dollars before your chance to survive getting better. I can probably create a two factor field based on this.

There are some outliers beyond 90, let's see the plot to see how we can group this.

```
p <- train %>% filter(Fare>200) %>%
  ggplot(aes(x = Fare, fill=Survived))

p + geom_density(alpha=0.5) + labs(title="Fare Grouping") + theme_economist_white()
```



Based on the above two plots, I finally can create grouping to factorise fare as per below code.

```
train$Fare2[train$Fare<10] <- 'Low Fare'
train$Fare2[train$Fare>=10 & train$Fare<=200] <- 'Normal Fare'
train$Fare2[train$Fare>200 & train$Fare<=300] <- 'Outlier1'
train$Fare2[train$Fare>300] <- 'Outlier2'
```

4. Machine Learning

After analysing all the features in Titanic dataset, 5 final features are:

- Pclass
- AgeGrp
- FamilySize
- Fare2
- Sex

In the next section, I will be running a few algorithms using ***Caret*** to decide which one to choose.

4.1 Finding the best model

The good thing about Caret, you can try multiple algorithms already included in this package. In here, I will try to apply some of the most popular algorithms below to decide which one I am going to use

- Decision Tree
- Random Forest
- XgBoost
- KNN
- Support Vector Machine

- GLM

I use caret to train the above algorithms using default values without any tuning.

```
set.seed(7)

fit.rf <- train(Survived ~ Sex + Pclass + AgeGrp + FamilySize + Fare2, data=train, method="r
f")
fit.rpart <- train(Survived ~ Sex + Pclass + AgeGrp + FamilySize + Fare2, data=train, method=
"rpart")
fit.glm <- train(Survived ~ Sex + Pclass + AgeGrp + FamilySize + Fare2, data=train, method="g
lm")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

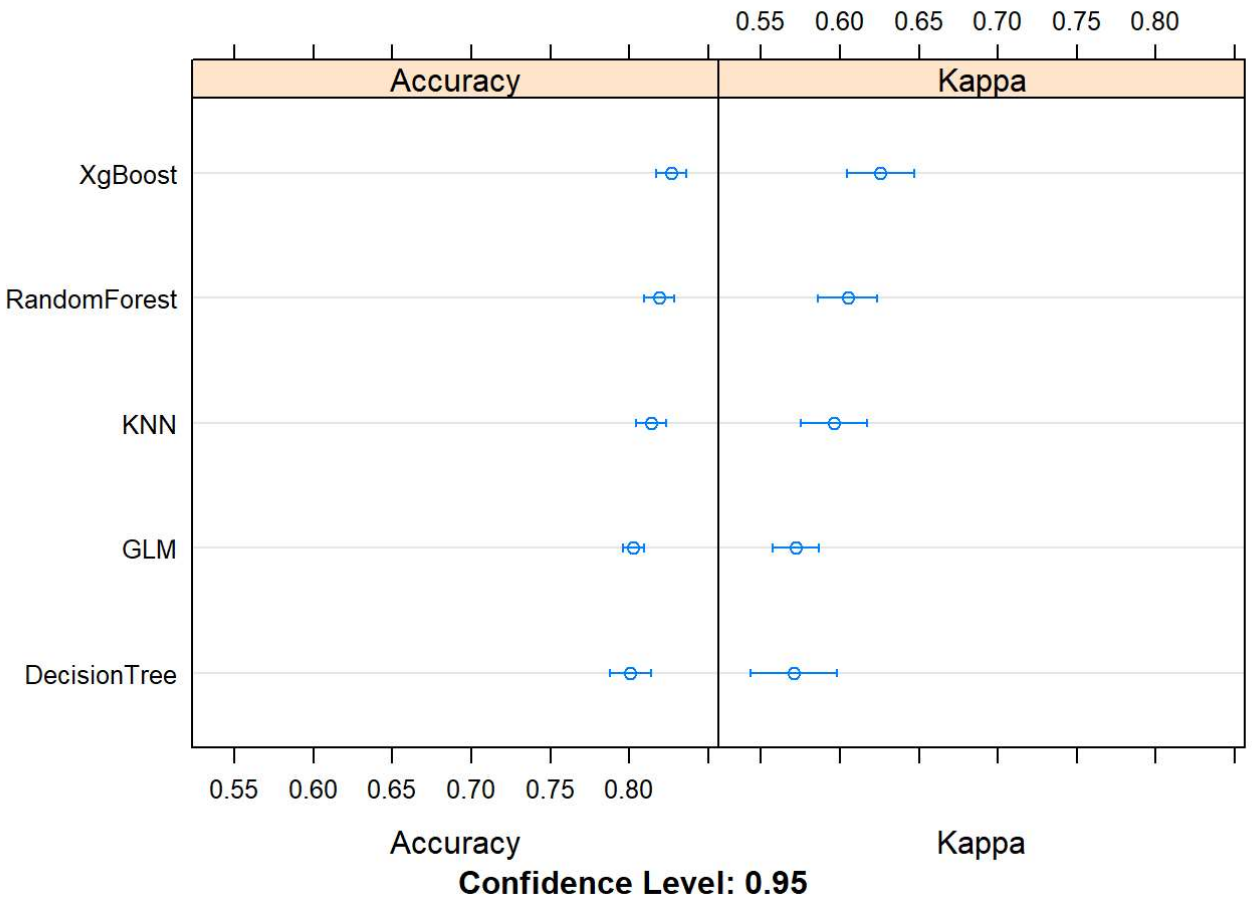
```
fit.knn <- train(Survived ~ Sex + Pclass + AgeGrp + FamilySize + Fare2, data=train, method="k
nn")
fit.xgbTree <- train(Survived ~ Sex + Pclass + AgeGrp + FamilySize + Fare2, data=train, metho
d="xgbTree")
```

```
# summarize accuracy of models

set.seed(7)

results <- resamples(list(DecisionTree=fit.rpart,RandomForest=fit.rf,XgBoost=fit.xgbTree,  KN
N=fit.knn, GLM=fit.glm))
summary(results)
```

```
##
## Call:
## summary.resamples(object = results)
##
## Models: DecisionTree, RandomForest, XgBoost, KNN, GLM
## Number of resamples: 25
##
## Accuracy
##                   Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## DecisionTree 0.7335907 0.7781955 0.8023256 0.8008643 0.8164062 0.8560311    0
## RandomForest 0.7732342 0.8000000 0.8208955 0.8189879 0.8384615 0.8498024    0
## XgBoost      0.7811321 0.8087649 0.8301887 0.8264805 0.8365759 0.8793774    0
## KNN          0.7566540 0.7969925 0.8208955 0.8138677 0.8302583 0.8467433    0
## GLM          0.7723881 0.7950820 0.8015564 0.8025204 0.8142292 0.8358779    0
##
## Kappa
##                   Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## DecisionTree 0.4312765 0.5193262 0.5718191 0.5708943 0.6144578 0.6861314    0
## RandomForest 0.5182750 0.5673808 0.6009926 0.6049052 0.6481481 0.6844144    0
## XgBoost      0.5205864 0.5841331 0.6371951 0.6256995 0.6599195 0.7450478    0
## KNN          0.4797230 0.5629111 0.6052250 0.5961454 0.6319134 0.6682344    0
## GLM          0.4974176 0.5549630 0.5742555 0.5720247 0.5927978 0.6442690    0
```

If you look at the results from Accuracy table, you can see Random Forest has the highest mean ouf of the other models. Thats why I decided to use **Random Forest** as my model to predict the outcome.

```
dotplot(results)
```



4.2 Best Model - Random Forest

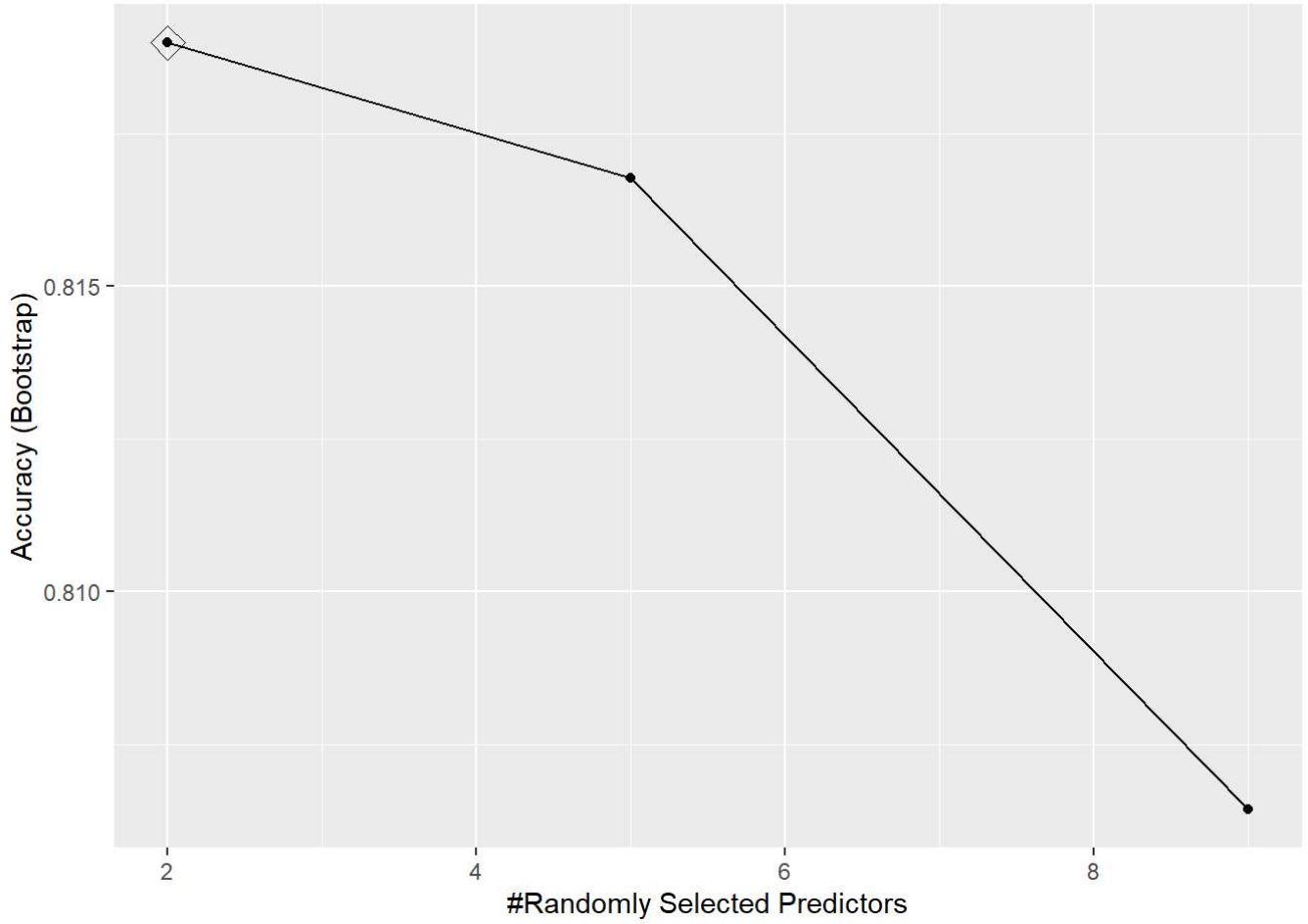
Using the default setting in Caret for Random Forest, we can see the details of the model. The default parameter is mtry using values of 2,4 and 6. Mtry is basically a random predictors chosen by the model when it building various different trees. We can see the best accuracy was achieved using mtry of 2.

```
print(fit.rf)
```

```
## Random Forest
##
## 714 samples
##   5 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 714, 714, 714, 714, 714, 714, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   2     0.8189879  0.6049052
##   5     0.8167650  0.6044899
##   9     0.8064317  0.5834307
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

The plot below shows the three default values in Random Forest and 2 predictos give the maximum accuracy around 81.9%.

```
ggplot(fit.rf, highlight = TRUE)
```



4.3 Tuning Random Forest

If I change the parameter using below code to run from 2 to 5 features to find the best mtry and to improve accuracy from the the current rate of 81.9%. The control function is changed to run 5 times against 20% of data to validate the accuracy of my model. In the below table, you can see accuracy improved to 82.6% and mtry reduced from 5 to 4.

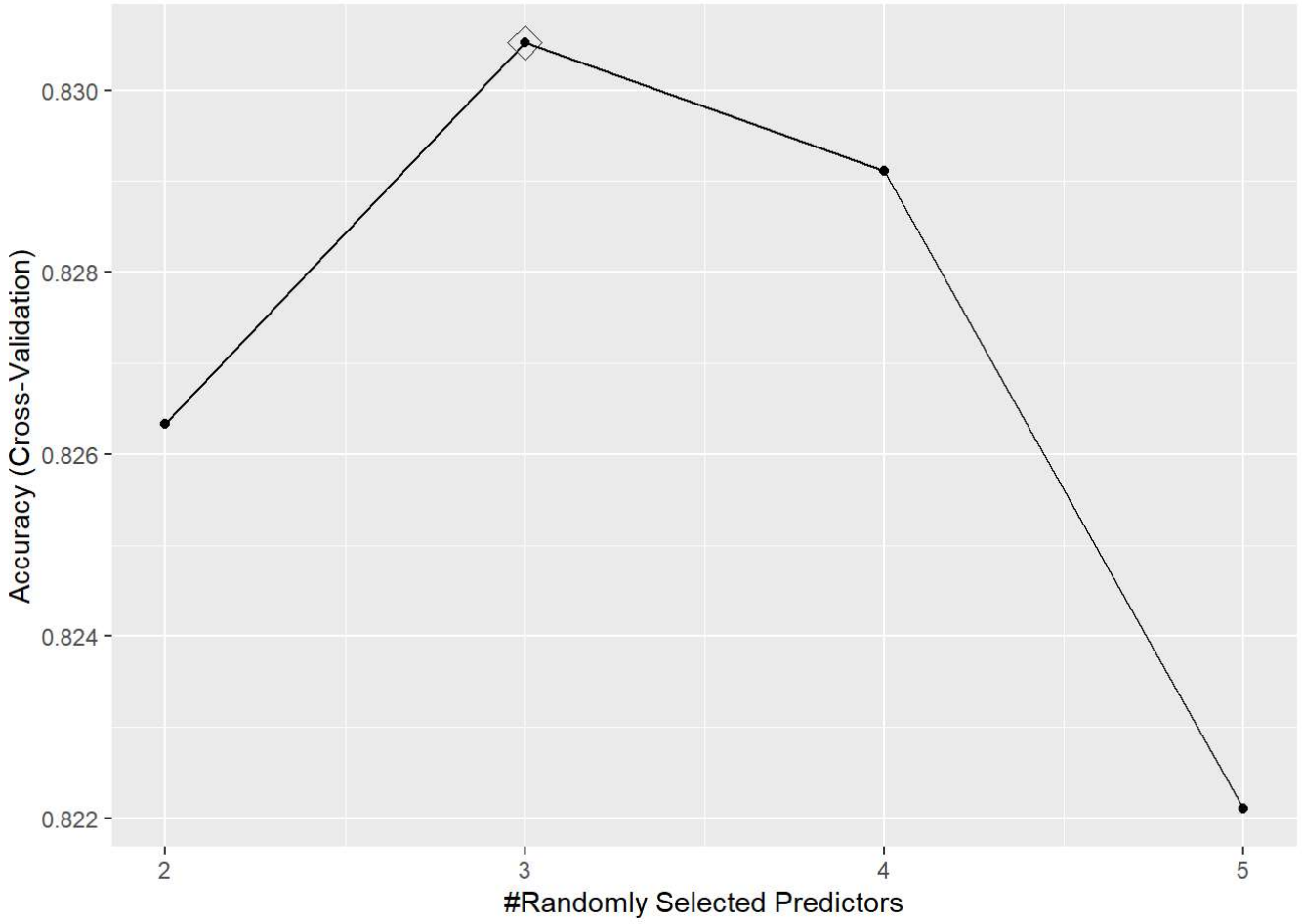
```
seq(2,5,1)
```

```
## [1] 2 3 4 5
```

```
# Machine learning one using 7 fold validation
library(caret)
train$Survived <- factor(train$Survived)
set.seed(7)

# create 7 fold validation
control <- trainControl(method="cv", number=5, p=0.8)
metric <- "Accuracy"

# Random Forest
fit.rf_final <- train(Survived ~ Sex + Pclass + AgeGrp + FamilySize + Fare2, data=train, metr
ic=metric, method="rf", tuneGrid = data.frame(mtry = seq(2, 5, 1)), trControl=control)

# print model
print(fit.rf_final)
```

```
## Random Forest
##
## 714 samples
##   5 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 571, 571, 571, 572, 571
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   2     0.8263272  0.6225234
##   3     0.8305230  0.6324212
##   4     0.8291145  0.6307745
##   5     0.8221018  0.6141416
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 3.
```

```
ggplot(fit.rf_final, highlight = TRUE)
```




```
fit.rf_final$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 16.53%
## Confusion matrix:
##     0   1 class.error
## 0 397  43  0.09772727
## 1  75 199  0.27372263
```

To compare the results against the train data to validate the accuracy. The prediction is added to a new field called **Survived2**. The accuracy is around 84.7% with balanced accuracy around 82.7% so this is a good balance in predicting 0 and 1. I am not expecting the model to predict the outcome 100% since there were a lot of factors not captured in the dataset and I think to get around 80% accuracy is pretty good result.

```
train$Survived2 <- predict(fit.rf_final,train)
train$Survived2 <- factor(train$Survived2)
confusionMatrix(train$Survived2,train$Survived)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 398  71
##          1  42 203
##
##                Accuracy : 0.8417
##                  95% CI : (0.8129, 0.8678)
##     No Information Rate : 0.6162
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6586
##
##  Mcnemar's Test P-Value : 0.008438
##
##             Sensitivity : 0.9045
##             Specificity : 0.7409
##          Pos Pred Value : 0.8486
##          Neg Pred Value : 0.8286
##              Prevalence : 0.6162
##          Detection Rate : 0.5574
##    Detection Prevalence : 0.6569
##       Balanced Accuracy : 0.8227
##
##        'Positive' Class : 0
##
```

4.4 Final - Predicting Test Data

Before testing the dataset, I need to fix some of the features such as family size and age group in test dataset.

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


test$AgeGrp[test$Age<=15] <- 'Kids'
test$AgeGrp[test$Age>15 & test$Age<=59] <- 'Adults'
test$AgeGrp[test$Age>59] <- 'Seniors'
test$AgeGrp <- factor(test$AgeGrp)

test$Fare2[test$Fare<10] <- 'Low Fare'
test$Fare2[test$Fare>=10 & test$Fare<=200] <- 'Normal Fare'
test$Fare2[test$Fare>200 & test$Fare<=300] <- 'Outlier1'
test$Fare2[test$Fare>300] <- 'Outlier2'
```

The final outcome against test dataset showing accuracy of 80.8% with balanced accuracy about 78.6%. Thanks for reading this report and I hope you also learnt something from this report.

```
test$Survived2 <- predict(fit.rf_final,test)
test$Survived2 <- factor(test$Survived2)
confusionMatrix(test$Survived2,test$Survived)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 94 20
##          1 15 48
##
##                Accuracy : 0.8023
##                  95% CI : (0.7359, 0.8582)
##     No Information Rate : 0.6158
##     P-Value [Acc > NIR] : 7.432e-08
##
##                   Kappa : 0.5762
##
##  Mcnemar's Test P-Value : 0.499
##
##             Sensitivity : 0.8624
##             Specificity : 0.7059
##          Pos Pred Value : 0.8246
##          Neg Pred Value : 0.7619
##              Prevalence : 0.6158
##          Detection Rate : 0.5311
##    Detection Prevalence : 0.6441
##       Balanced Accuracy : 0.7841
##
##        'Positive' Class : 0
##
```

5. Conclusion

The titanic dataset is a great example for a person who wants to see the most basics of data science and I believe that many people is already familiar with the issue from both the movie and from the lecture.

The report taught me a lot in R and data science. Another thing I see that the predicts are not always the only thing that we have to carry. There were some people who lived in the same situation with those who don't.

It was a great learning curve for me in the last 4 months and this course is one of the best and most competitive course that I have taken as online and I have learnt a lot from the course materials provided by Professor Rafael Irizarry. Thanks for providing good and challenging materials for us to help us jump into the world of data science.