CMP 713 Data Mining - Titanic Dataset

Erol ÖZKAN
Hacettepe University
Computer Engineering Department
Ankara, Turkey
erolozkan@outlook.com

31 Mayıs 2020

Abstract

In this work, we try to predict what kind of people are likely to survive in the Titanic dataset. We use several data mining methods to predict the survival information of the passengers. We do explanatory data analysis and show some illustrative data visualizations. We do feature engineering and create more meaningful variables. We do missing value imputation and outlier detection. We build RandomForrest, SVM and GBM models to predict the Survival information of the passengers. Finally, we compare these three models at the end of our work.

1 INTRODUCTION

The remainder of the document is organized as follows. In Section 2, we explain the Titanic dataset. In Section 3, we explain the explanatory data analysis steps. Missing value imputation and outlier detection steps are explained in Section 5 and Section 6. In Section 4 and Section 7 we show the feature engineering steps. Predictive models are explained in the Section 8. Results are shown in Section 9. Finally, we conclude our work in Section 10.

2 DATASET

Titanic dataset is used for this project. It contains the survival information of Titanic ship which sank after colliding with an iceberg on April 15, 1912. It killed 1502 out of 2224 including both all the passengers and its crew. One of the reasons this tragedy led to such loss of life was that there were not enough lifeboats for everyone. This tragedy resulted in better safety regulations for ships.

3 FIRST STEPS

3.1 Import Required Packages

Firstly, let's import the required packages.

```
library('ggplot2')
library('ggthemes')
library('scales')
```

```
library('mice')
library('knitr')
library('dplyr')
library('caret')
library('randomForest')
library('gridExtra')
library('corrplot')
library('Hmisc')
library('ROCR')
library('fpc')
library('DMwR2')
library('dbscan')
```

3.2 Read Data

Let's read the training and testing datasets.

```
train <- read.csv('dataset/train.csv', stringsAsFactors = F, na.strings = c("NA", ""))
test  <- read.csv('dataset/test.csv', stringsAsFactors = F, na.strings = c("NA", ""))
```

Let's combine the training and testing datasets.

```
full  <- bind_rows(train, test)
```

Note that Survived variable is missing from the testing dataset.

4 EXPLORATORY DATA ANALYSIS

4.1 Check Data

Let's check the dataset.

```
str(full)
```

```
## 'data.frame':    1309 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       : 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      : int  1 1 0 1 0 0 0 3 0 1 ...
##  $ Parch      : int  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" ...
```

There are 1309 observations in total;

- The training dataset has 891,
- The testing dataset has 418 observations.

4.2 Describe Variables

Description of the variables are shown below.

Variable Name	Description
Survived	Survived (1) or died (0)
Pclass	Passenger's class
Name	Passenger's name
Sex	Passenger's sex
Age	Passenger's age
SibSp	Number of siblings/spouses aboard
Parch	Number of parents/children aboard
Ticket	Ticket number
Fare	Fare
Cabin	Cabin
Embarked	Port of embarkation

Figure 1: Description of the Dataset

4.3 Investigate the Data

Let's investigate the variables that contain missing values.

```
sapply(full, function(x) {sum(is.na(x))})
```

```
## PassengerId    Survived      Pclass        Name         Sex         Age
##           0         418           0           0           0         263
##       SibSp       Parch      Ticket        Fare       Cabin    Embarked
##           0           0           0           1        1014           2
```

There are number missing variables in the dataset;

- 418 observations are missing the Survived variable in the testing dataset.
- Cabin is sparsely populated and has lots of mising values.
- Age is missing in some observations.
- Embarked is missing in two observations.
- Fare is missing in a observation.

4.4 Explore the Data

Let's convert Sex, Pclass, Survived and Embarked variables into factors.

```
full$Sex <- as.factor(full$Sex)
full$Survived <- as.factor(full$Survived)
full$Pclass <- as.factor(full$Pclass)
full$Embarked <- as.factor(full$Embarked)
```

4.4.1 Survived

Let's visualize the Survived variable. It describes the survival information of the people.

```
ggplot(full[!is.na(full$Survived),], aes(x = Survived, fill = Survived)) +
  geom_bar(stat='count') +
  labs(x = 'Survival Information of People') +
    geom_label(stat='count',aes(label=..count..), size=7)
```

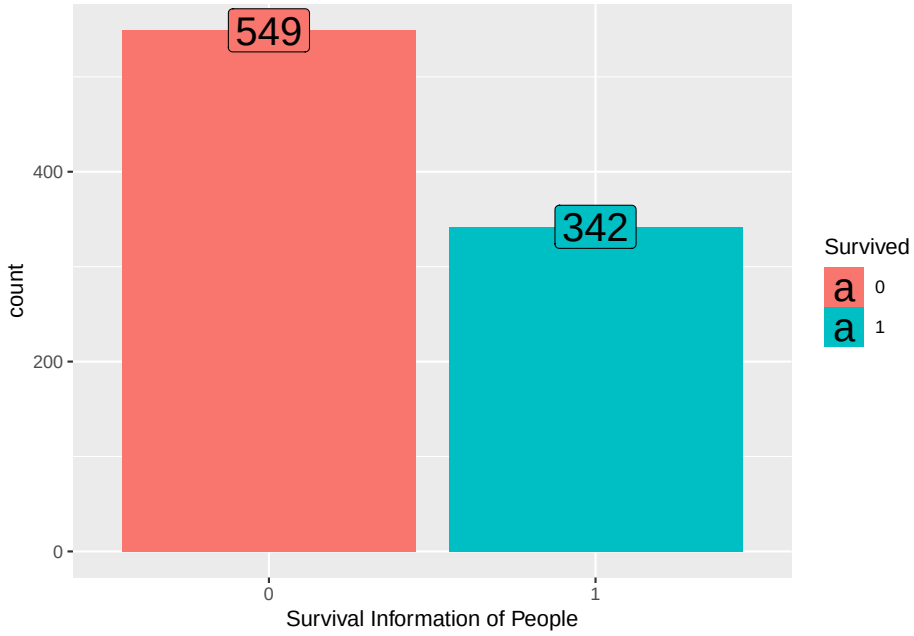


Figure 2: Survival Information of People

It is clear that 549 people died and 342 people survived in the training dataset.

4.4.2 Sex

Let's visualize the Sex variable.

```
p1 <- ggplot(full, aes(x = Sex, fill = Sex)) +
  geom_bar(stat='count', position='dodge') + theme_grey() +
  labs(x = 'All data') +
        geom_label(stat='count', aes(label=..count..)) +
        scale_fill_manual("legend", values = c("female" = "pink", "male" = "green"))

p2 <- ggplot(full[!is.na(full$Survived),], aes(x = Sex, fill = Survived)) +
  geom_bar(stat='count', position='dodge') + theme_grey() +
  labs(x = 'Training data only') +
        geom_label(stat='count', aes(label=..count..))

grid.arrange(p1,p2, nrow=1)
```

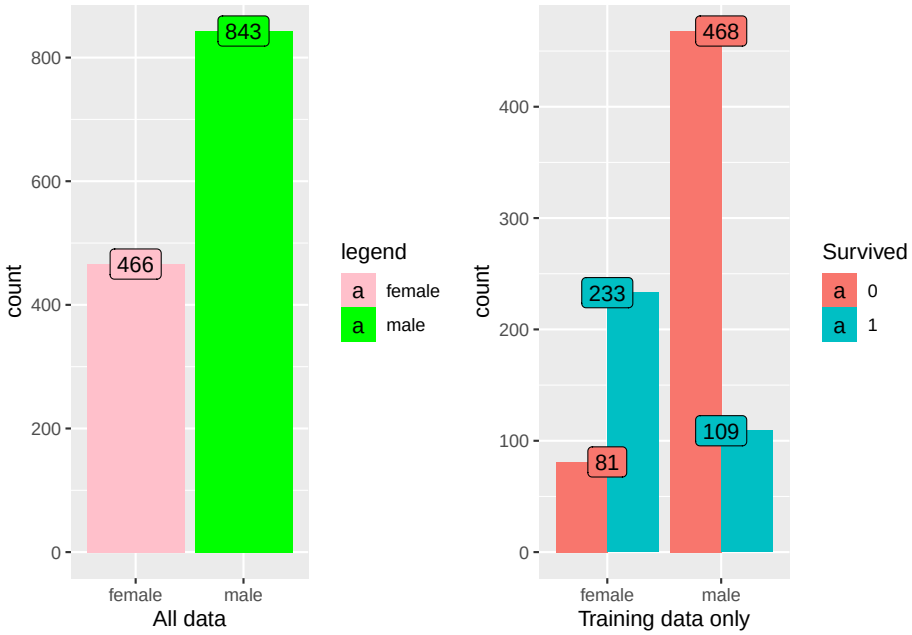


Figure 3: Survival rate of Male/Female Passangers

Let's investigate the plots;

- First plot states that there are 466 female, 843 male passengers.
- Second plot states that 81.1% of men, 25.8% of women died in the training dataset.

4.4.3 PClass

Let's visualize the PClass variable.

```
p3 <- ggplot(full, aes(x = Pclass, fill = Pclass)) +
  geom_bar(stat='count', position='dodge') +
  labs(x = 'Pclass, All data') + geom_label(stat='count', aes(label=..count..)) +
   theme(legend.position="none") + theme_grey()
p4 <- ggplot(full[!is.na(full$Survived),], aes(x = Pclass, fill = Survived)) +
  geom_bar(stat='count', position='dodge') + labs(x = 'Training data only') +
        theme(legend.position="none") + theme_grey()
p5 <- ggplot(full[!is.na(full$Survived),], aes(x = Pclass, fill = Survived)) +
  geom_bar(stat='count', position='stack') +
  labs(x = 'Training data only', y= "Count") + facet_grid(.~Sex) +
        theme(legend.position="none") + theme_grey()
p6 <- ggplot(full[!is.na(full$Survived),], aes(x = Pclass, fill = Survived)) +
  geom_bar(stat='count', position='fill') +
  labs(x = 'Training data only', y= "Percent") + facet_grid(.~Sex) +
        theme(legend.position="none") + theme_grey()
grid.arrange(p3, p4, p5, p6, ncol=2)
```

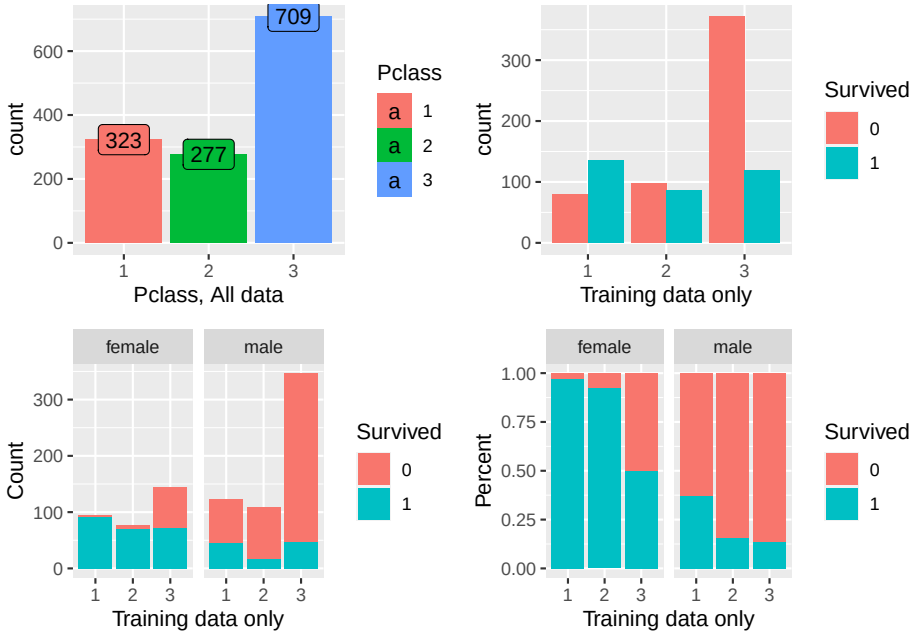


Figure 4: Passenger Class Information

Let's investigate the plots;

- First plot states that most people was in the 3rd class.
- Second plot states that majority of 1st and 2nd class passengers survived, while majority of 3rd class died.
- Third and fourth plots state that almost all women in 1st and 2nd class survived.

5 FEATURE ENGINEERING 1

5.1 Title

We can break the Name variable into more meaningful variables, including; title, name and surname.

```
full$Title <- gsub('(.*, )|(\\..*)', '', full$Name) # Create Title variable
table(full$Sex, full$Title) # Show Title counts
```

```
##
##          Capt Col Don Dona  Dr Jonkheer Lady Major Master Miss Mlle Mme  Mr Mrs
##   female    0   0   0    1   1        0    1     0      0  260    2   1   0 197
##   male      1   4   1    0   7        1    0     2     61    0    0   0 757   0
##
##           Ms Rev Sir the Countess
##   female   2   0   0            1
##   male     0   8   1            0
```

Let's replace low title counts.

```
rare_title <- c('Dona', 'Lady', 'the Countess','Capt', 'Col', 'Don',  'Dr', 'Major', 'Rev', 'Sir', 'Jonk
full$Title[full$Title == 'Mlle']        <- 'Miss'
full$Title[full$Title == 'Ms']          <- 'Miss'
full$Title[full$Title == 'Mme']         <- 'Mrs'
full$Title[full$Title %in% rare_title]  <- 'Rare Title'
table(full$Sex, full$Title)
```

```
##
##          Master Miss  Mr Mrs Rare Title
##   female      0  264   0 198          4
##   male       61    0 757   0         25
```

Let's visualize the Title variable.

```
ggplot(full[!is.na(full$Survived),], aes(x = Title, fill = Survived)) +
  geom_bar(stat='count', position='stack') +
  labs(x = 'Title') +theme_grey()
```

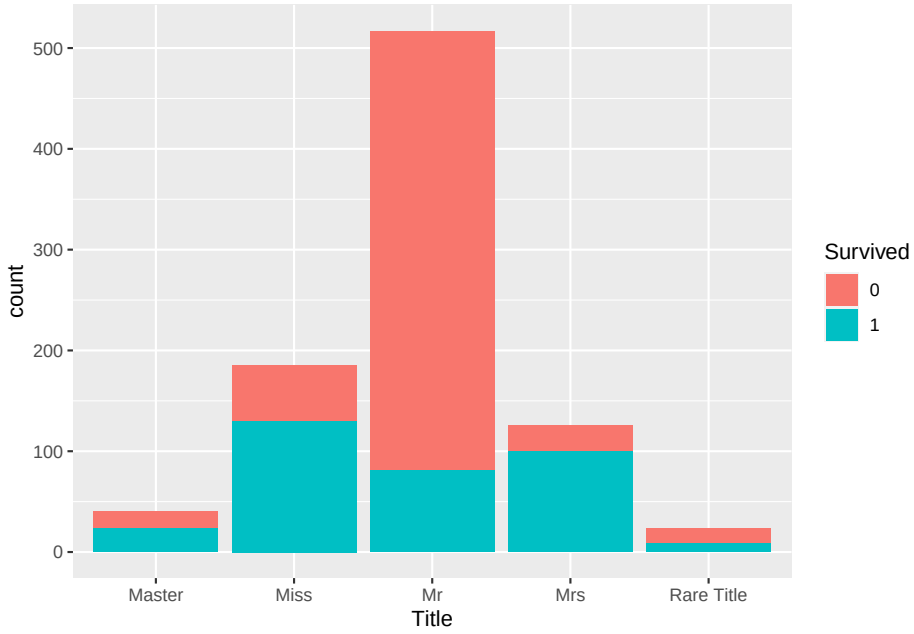


Figure 5: Visualization of Title Variable

It is clear that Title is highly correlated with Survival information.

5.2 Family Size

Let's create a Family size variable.

- It is based on number of siblings/spouse(s) and children/parents.
- We need to add 1 to include the passenger himself/herself.

```
full$Fsize <- full$SibSp + full$Parch + 1
```

To better understand the family size variable, let's use ggplot to visualize the family size and the survival.

```
ggplot(full[1:891,], aes(x = Fsize, fill = factor(Survived))) +
  geom_bar(stat='Count', position='dodge') +
  scale_x_continuous(breaks=c(1:11)) +
  labs(x = 'Family Size')
```

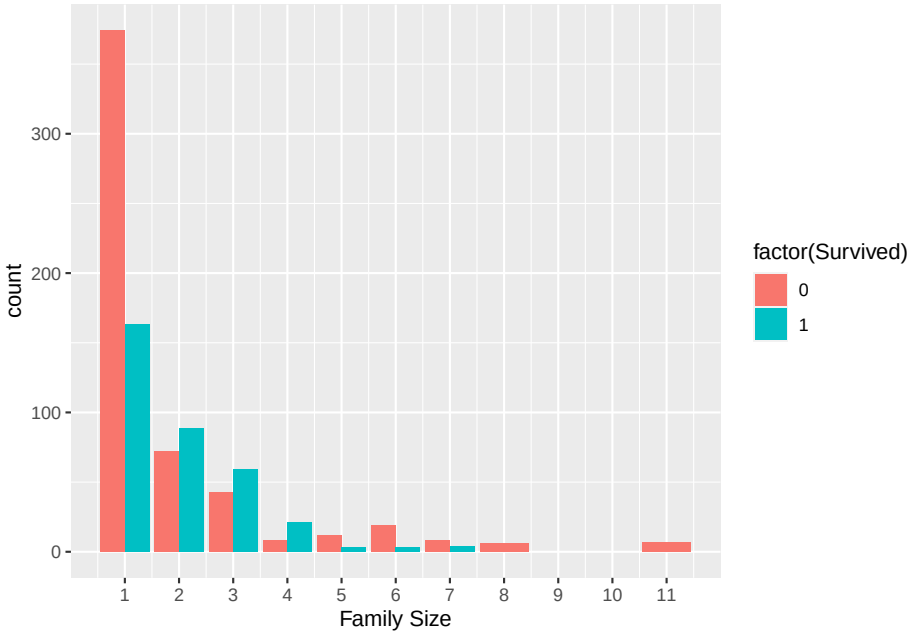
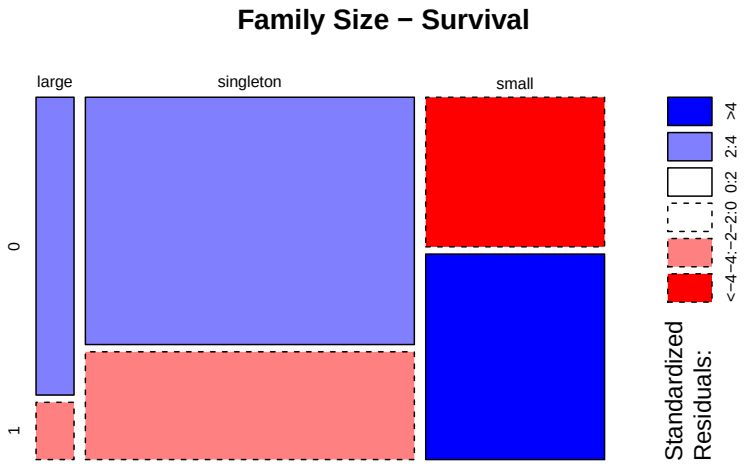


Figure 6: Family Size and Survival

It is clear that there is a survival penalty to single passengers and family sizes above 4.

We can further make Family variable a categorical variable.

```
full$FsizeD[full$Fsize == 1] <- 'singleton'
full$FsizeD[full$Fsize < 5 & full$Fsize > 1] <- 'small'
full$FsizeD[full$Fsize > 4] <- 'large'
mosaicplot(table(full$FsizeD, full$Survived), main='Family Size - Survival', shade=TRUE)
```



This plot shows that the survival rate of large families is worse than single and small families.

5.3 Cabin

Let's take a look into the Cabin variable.

```
full$Cabin[1:28]
```

```
##  [1] NA           "C85"        NA           "C123"       NA
##  [6] NA           "E46"        NA           NA           NA
## [11] "G6"         "C103"       NA           NA           NA
## [16] NA           NA           NA           NA           NA
## [21] NA           "D56"        NA           "A6"         NA
## [26] NA           NA           "C23 C25 C27"
```

As you can see, there are lots of missing values. But, we can also infer Deck variable from Cabin variable.

```
strsplit(full$Cabin[2], NULL)[[1]]
```

```
## [1] "C" "8" "5"
```

```
full$Deck<-factor(sapply(full$Cabin, function(x) strsplit(x, NULL)[[1]][1]))
full$Deck[1:28]
```

```
##  [1] <NA> C    <NA> C    <NA> <NA> E    <NA> <NA> <NA> G    C    <NA> <NA> <NA>
## [16] <NA> <NA> <NA> <NA> <NA> <NA> D    <NA> A    <NA> <NA> <NA> C
## Levels: A B C D E F G T
```

6 MISSING VALUE IMPUTATION

There are some missing values in the dataset. Since, we don't have many observations, we are better not to delete any of the observations or columns with missing values. Hence, we are trying to replace missing values with sensible values.

6.1 Embarked

Let's take a look at Embarked variable.

```
subset(full,is.na(full['Embarked']))
```

```
##     PassengerId Survived Pclass                                         Name
## 62           62        1      1                       Icard, Miss. Amelie
## 830         830        1      1 Stone, Mrs. George Nelson (Martha Evelyn)
##        Sex Age SibSp Parch Ticket Fare Cabin Embarked Title Fsize    FsizeD
## 62  female  38     0     0 113572   80   B28     <NA>  Miss     1 singleton
## 830 female  62     0     0 113572   80   B28     <NA>   Mrs     1 singleton
##     Deck
## 62     B
## 830    B
```

Passengers 62 and 830 are missing Embarkment variable.

```
full[c(62, 830), 'Embarked']
```

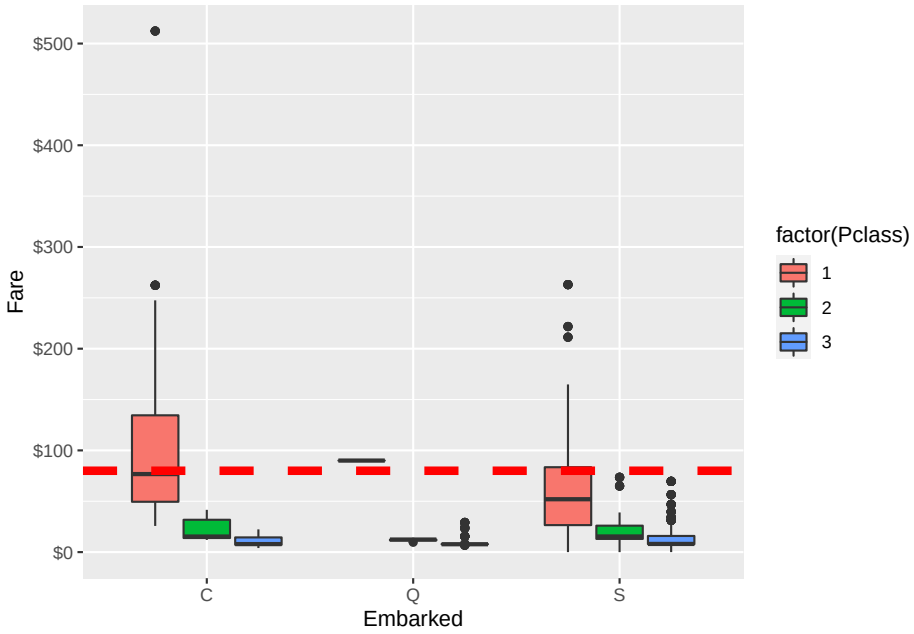
```
## [1] <NA> <NA>
## Levels: C Q S
```

We can infer their values based on passenger class and fare.

```
embark_fare <- full %>%
  filter(PassengerId != 62 & PassengerId != 830)
```

Let's use ggplot to visualize Embarked, Class and median Fare variables.

```
ggplot(embark_fare, aes(x = Embarked, y = Fare, fill = factor(Pclass))) +
  geom_boxplot() +
  geom_hline(aes(yintercept=80),
    colour='red', linetype='dashed', lwd=2) +
  scale_y_continuous(labels=dollar_format())
```



The median fare for a first class passenger departing from Charbourg ('C') matches with the $80 Fare variable. Hence, we can replace the NA values with 'C'.

```
full$Embarked[c(62, 830)] <- 'C'
full[c(62, 830), 'Embarked']
```

```
## [1] C C
## Levels: C Q S
```

6.2 Fare

Similar to Embarked variable, Fare variable on row 1044 has an NA value.

```
full[1044, "Fare"]
```

```
## [1] NA
```

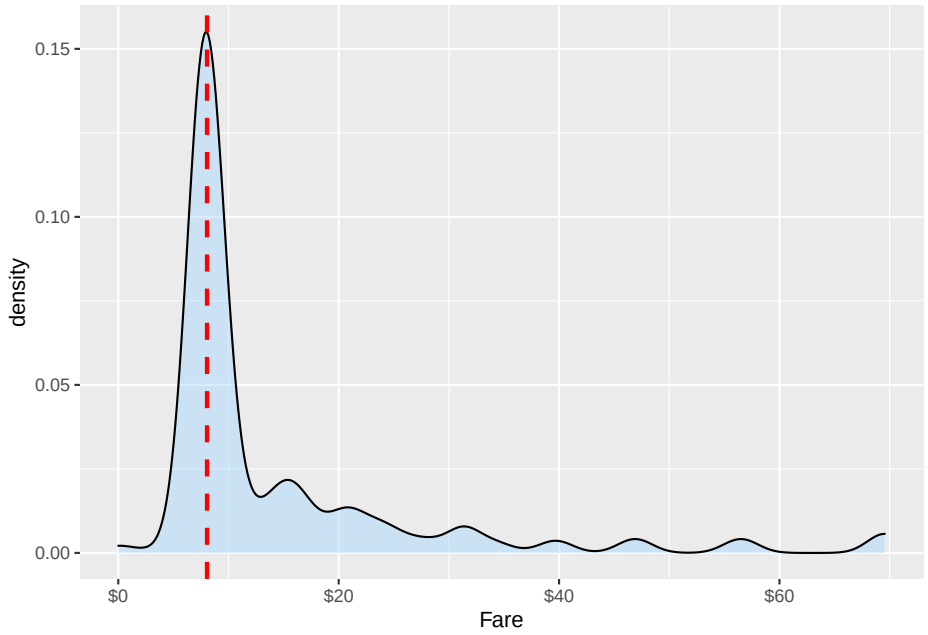
Let's show all of the observation.

```
full[1044,]
```

```
##      PassengerId Survived Pclass               Name  Sex  Age SibSp Parch
## 1044        1044     <NA>      3 Storey, Mr. Thomas male 60.5     0     0
##      Ticket Fare Cabin Embarked Title Fsize    FsizeD Deck
## 1044   3701   NA  <NA>        S    Mr     1 singleton <NA>
```

This passenger departed from Southampton ('S'). Let's visualize the Fares variable with same class and embarkment values.

```
ggplot(full[full$Pclass == '3' & full$Embarked == 'S', ],
  aes(x = Fare)) +
  geom_density(fill = '#99d6ff', alpha=0.4) +
  geom_vline(aes(xintercept=median(Fare, na.rm=T)),
    colour='red', linetype='dashed', lwd=1) +
  scale_x_continuous(labels=dollar_format())
```



From this visualization, it seems quite reasonable to replace the NA values with median for their class and embarkment.

```
full$Fare[1044] <- median(full[full$Pclass == '3' & full$Embarked == 'S', ]$Fare, na.rm = TRUE)
```

```
full[1044, "Fare"]
```

```
## [1] 8.05
```

6.3 Age

There are many missing Age variable in the dataset. Let's look into the number of missing Age variables.

```
sum(is.na(full$Age))
```

```
## [1] 263
```

There are 263 missing Age variables. We can use mice package to guess the missing Age variables.

```
set.seed(129) # Set a random seed
mice_mod <- mice(full[, !names(full) %in% c('PassengerId','Name','Ticket','Cabin','Family','Surname','Su
```

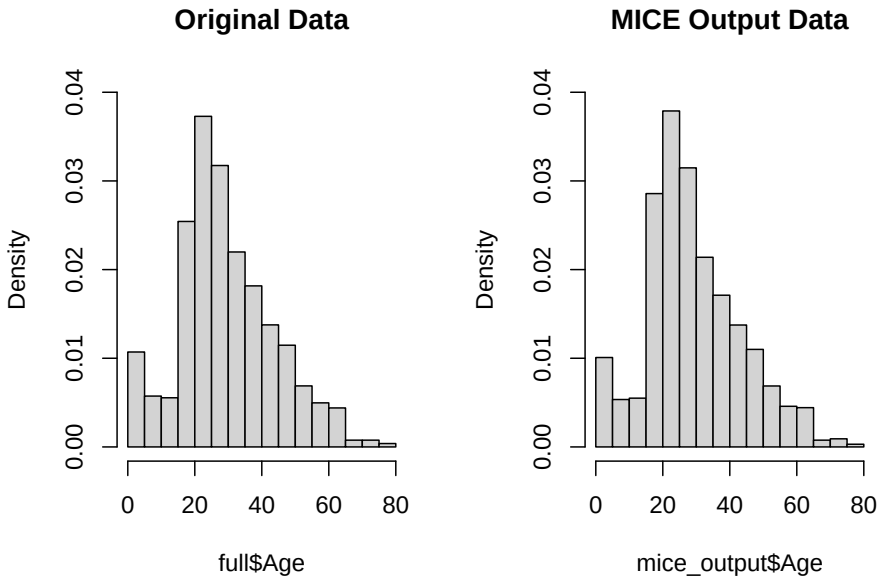
```
##
##  iter imp variable
##   1   1  Age  Deck
##   1   2  Age  Deck
##   1   3  Age  Deck
##   1   4  Age  Deck
##   1   5  Age  Deck
##   2   1  Age  Deck
##   2   2  Age  Deck
##   2   3  Age  Deck
##   2   4  Age  Deck
##   2   5  Age  Deck
##   3   1  Age  Deck
##   3   2  Age  Deck
##   3   3  Age  Deck
##   3   4  Age  Deck
##   3   5  Age  Deck
##   4   1  Age  Deck
##   4   2  Age  Deck
##   4   3  Age  Deck
##   4   4  Age  Deck
##   4   5  Age  Deck
##   5   1  Age  Deck
##   5   2  Age  Deck
##   5   3  Age  Deck
##   5   4  Age  Deck
##   5   5  Age  Deck

## Warning: Number of logged events: 52
```

```
mice_output <- complete(mice_mod) # Save the output
```

Let's compare the results with the original distribution.

```
par(mfrow=c(1,2))
hist(full$Age, freq=F, main='Original Data', ylim=c(0,0.04))
hist(mice_output$Age, freq=F, main='MICE Output Data', ylim=c(0,0.04))
```



Distributions looks very similar, Let's replace the Age variables with the mice model.

```
full$Age <- mice_output$Age
```

Let's check the Age variable again.

```
sum(is.na(full$Age))
```

```
## [1] 0
```

7 OUTLIER DETECTION

In this section, we are trying to find the outliers. We experiment with three different methods; dbscan, Torgo (2007) and Breunig et al. (2000).

7.1 DBSCAN

Let's define the "dbscan.outliers" function.

```
dbscan.outliers <- function(data, ...) {
  require(fpc, quietly=TRUE)
  cl <- dbscan(data, ...)
  posOuts <- which(cl$cluster == 0)
  list(positions = posOuts,
       outliers = data[posOuts,],
       dbscanResults = cl)
}
```

Let's investigate the Fare variable.

```
outs <- dbscan.outliers(full['Fare'],  eps = 3)
full[outs$positions,'Fare']
```

```
##  [1] 146.5208 247.5208 146.5208 512.3292 247.5208 164.8667 120.0000 120.0000
##  [9] 221.7792 512.3292 512.3292 120.0000 120.0000 164.8667 221.7792 221.7792
## [17] 221.7792 247.5208 164.8667 146.5208 512.3292 164.8667
```

Fare values looks reasonable. So, the output of dbscan on the Fare variable didn't reveal any outliers. Let's investigate the Age variable.

```
outs <- dbscan.outliers(full['Age'],  eps = 3)
full[outs$positions,'Age']
```

```
## [1] 80
```

There is someone who is aged 80. But, it is not an outlier either.

7.2 Torgo (2007)

Let's investigate the Fare variable.

```
outs <- outliers.ranking(scale(full['Fare']))
full[outs$rank.outliers[1:10],'Fare']
```

```
##  [1]  8.0292  8.0500  8.0500  8.0500 13.0000  8.0500 13.0000  8.0500 13.0000
## [10]  8.0500
```

These values are very low, but still possible. So, we will not replace these values. Let's investigate the Age variable.

```
outs <- outliers.ranking(scale(full['Age']))
full[outs$rank.outliers[1:10],'Age']
```

```
##  [1] 24 24 18 18 24 18 24 21 22 24
```

These values are not outliers.

7.3 Breunig et al. (2000)

Let's investigate the Fare variable.

```
out.scores <- lofactor(scale(full['Fare']), 15)
top_outliers <- order(out.scores, decreasing = T)[1:10]
full[top_outliers,'Fare']
```

```
##  [1]  8.4583 11.1333  8.0292  7.8792 11.2417  7.8792  7.8000  8.1583  9.0000
## [10]  7.7875
```

These values are very low, but still possible. So, we will not replace these values. Let's investigate the Age variable.

```
out.scores <- lofactor(scale(full['Age']), 15)
top_outliers <- order(out.scores, decreasing = T)[1:10]
full[top_outliers,'Age']
```

```
##  [1] 15.0 49.0 28.5 46.0 32.5 32.5 40.5 36.5 51.0 40.5
```

These values are not outliers.

8 FEATURE ENGINEERING 2

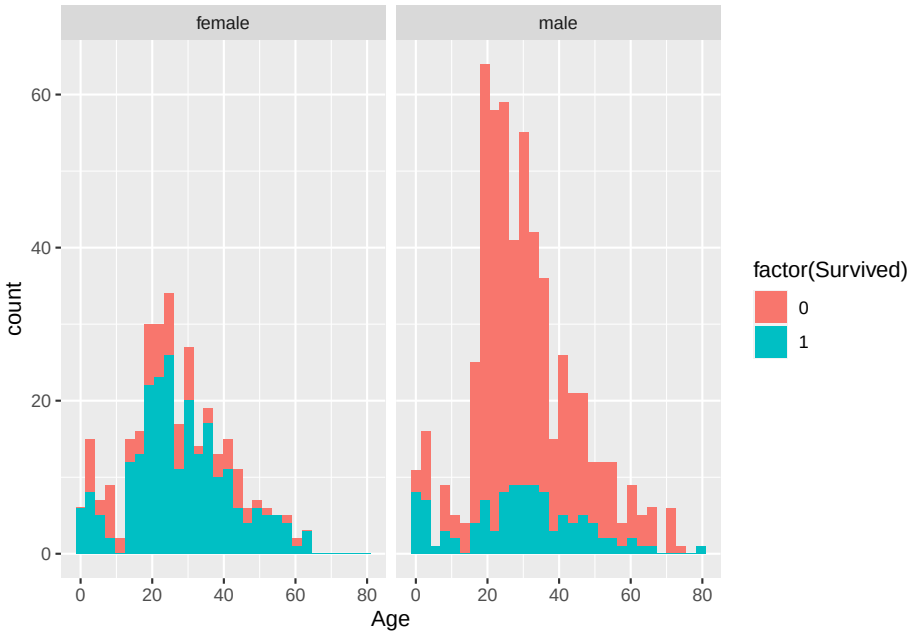
We can create a couple of new age-dependent variables: Child and Mother.

- A child is someone under 18 years.
- A mother is someone who is female, older than 18, doesn't have the 'Miss' title and has more than 0 children.

8.1 Child

Let's look at the relationship between age and survival.

```
ggplot(full[1:891,], aes(Age, fill = factor(Survived))) +
  geom_histogram() +
  facet_grid(.~Sex)
```



Let's create a Child variable.

```
full$Child[full$Age < 18] <- 'Child'
full$Child[full$Age >= 18] <- 'Adult'
table(full$Child, full$Survived)
```

```
##
##            0   1
##   Adult 479 271
##   Child  70  71
```

```
full$Child  <- factor(full$Child)
```

8.2 Mother

Similarly, let's create a Mother variable.

```
full$Mother <- 'Not Mother'
full$Mother[full$Sex == 'female' & full$Parch > 0 & full$Age > 18 & full$Title != 'Miss'] <- 'Mother'
table(full$Mother, full$Survived)
```

```
##
##                 0   1
##   Mother       16  38
##   Not Mother 533 304
```

```
full$Mother <- factor(full$Mother)
```

9 PREDICTION

We successfully fill the missing values and created new variables in the Titanic datasets. In this section, we will build some models to predict the Survival variable. Let's first check the datasets again using the completeness function.

```
sapply(full, function(x) {sum(is.na(x))})
```

```
## PassengerId    Survived      Pclass        Name         Sex         Age
##           0         418           0           0           0           0
##       SibSp       Parch      Ticket        Fare       Cabin    Embarked
##           0           0           0           0        1014           0
##       Title       Fsize      FsizeD        Deck       Child      Mother
##           0           0           0        1014           0           0
```

Let's factorize some variables.

```
full$Sex <- as.factor(full$Sex)
full$Embarked <- as.factor(full$Embarked)
full$Title<- as.factor(full$Title)
full$FsizeD<- as.factor(full$FsizeD)
```

As you can see, we have many new variables. Also, we replaced some missing values with their sensible values. Again, we need to convert some variables into factor.

9.1 Split Dataset

Let's split the data back into a training and test datasets.

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

9.2 RandomForest Model

Let's build the RandomForest model.

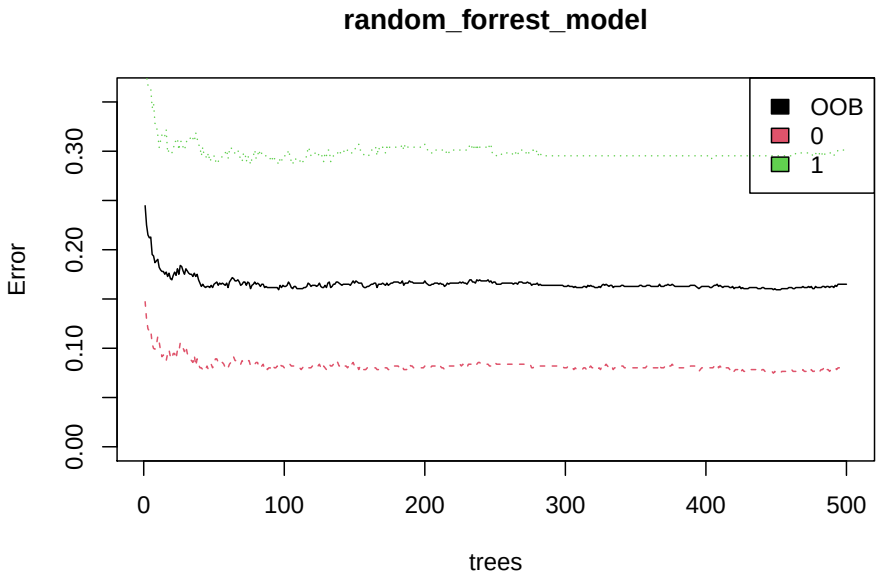
```
set.seed(2017)
random_forrest_model <- randomForest(factor(Survived) ~ Pclass + Sex + Age + SibSp + Parch +
                                     Fare + Embarked + Title +
                                     FsizeD + Child + Mother, data = train)
```

Let's predict the Survival column and write output into a file.

```
random_forrest_predictions <- predict(random_forrest_model, test)
random_forrest_solutions <- data.frame(PassengerID = test$PassengerId, Survived = random_forrest_predic
write.csv(random_forrest_solutions, file = 'output/random_forrest_solutions.csv', row.names = F)
```

The final result for RandomForest model is **0.78947**. Let's visualize the error rate.

```
plot(random_forrest_model, ylim=c(0,0.36))
legend('topright', colnames(random_forrest_model$err.rate), col=1:3, fill=1:3)
```

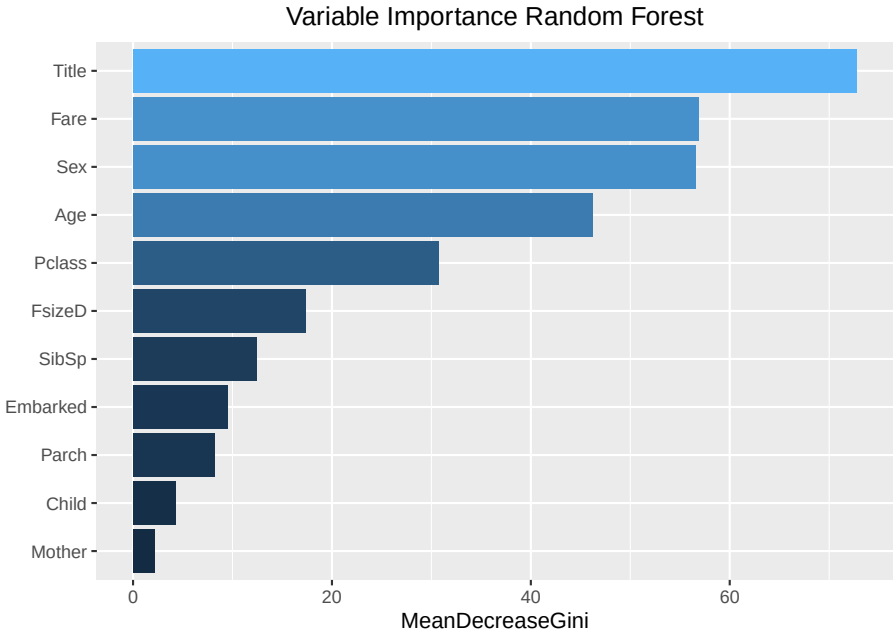


The black line shows the overall error rate. The red and green lines show the error rate of 'died' and 'survived'.

9.2.1 Visualize the Importance

Let's visualize the importance of variables.

```
random_forrest_importance <- varImp(random_forrest_model, scale = FALSE)
importance_scores <- data.frame(Variables = row.names(random_forrest_importance), MeanDecreaseGini = ra
ggplot(importance_scores, aes(x=reorder(Variables, MeanDecreaseGini), y=MeanDecreaseGini, fill=MeanDecr
  geom_bar(stat='identity') + coord_flip() + theme(legend.position="none") + labs(x="") +
  ggtitle('Variable Importance Random Forest') + theme(plot.title = element_text(hjust = 0.5))
```



It is clear in the plot that, the Title variable has the highest relative importance.

9.3 SVM Model

```
set.seed(2017)
svm_model <- train(Survived~ Pclass + Sex + Age + SibSp + Parch + Fare + Embarked + Title +
                   FsizeD + Child + Mother, data=train, method='svmRadial', preProcess= c('center', '
svm_model
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 891 samples
##  11 predictor
##   2 classes: '0', '1'
##
## Pre-processing: centered (17), scaled (17)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 712, 713, 712, 713, 714
## Resampling results across tuning parameters:
```

```
##
##   C     Accuracy   Kappa
##   0.25  0.8260540  0.6227622
##   0.50  0.8260666  0.6215805
##   1.00  0.8271903  0.6224679
##
## Tuning parameter 'sigma' was held constant at a value of 0.07157105
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.07157105 and C = 1.
```

```
svm_model$results
```

```
##        sigma    C  Accuracy     Kappa AccuracySD    KappaSD
## 1 0.07157105 0.25 0.8260540 0.6227622 0.01919075 0.04108654
## 2 0.07157105 0.50 0.8260666 0.6215805 0.02023424 0.04181027
## 3 0.07157105 1.00 0.8271903 0.6224679 0.01837269 0.03867780
```

```
svm_predictions <- predict(svm_model, test)
svm_solutions <- data.frame(PassengerID = test$PassengerId, Survived = svm_predictions)
write.csv(svm_solutions, file = 'output/svm_solutions.csv', row.names = F)
```

The final result for SVM model is **0.80382**.

9.4 GBM Model

```
set.seed(2017)
gbm_model <- train(Survived~ Pclass + Sex + Age + SibSp + Parch + Fare + Embarked + Title +
                     FsizeD + Child + Mother, data=train, method='gbm', preProcess= c('center', 'scal
print(gbm_model)
```

```
## Stochastic Gradient Boosting
##
## 891 samples
##  11 predictor
##   2 classes: '0', '1'
##
## Pre-processing: centered (17), scaled (17)
## Resampling: Cross-Validated (7 fold)
## Summary of sample sizes: 763, 763, 763, 764, 765, 764, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.8047906  0.5823664
##   1                  100      0.8103711  0.5950456
##   1                  150      0.8080859  0.5903117
##   2                   50      0.8148616  0.5997138
##   2                  100      0.8204595  0.6089840
##   2                  150      0.8170848  0.6039685
##   3                   50      0.8238787  0.6176522
##   3                  100      0.8238341  0.6195332
```

```
##   3                   150      0.8271735  0.6259423
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 150, interaction.depth =
##  3, shrinkage = 0.1 and n.minobsinnode = 10.
```

```
gbm_predictions <- predict(gbm_model, test)
gbm_solutions <- data.frame(PassengerID = test$PassengerId, Survived = gbm_predictions)
write.csv(gbm_solutions, file = 'output/gbm_solutions.csv', row.names = F)
```

The final result for GBM model is **0.76555**.

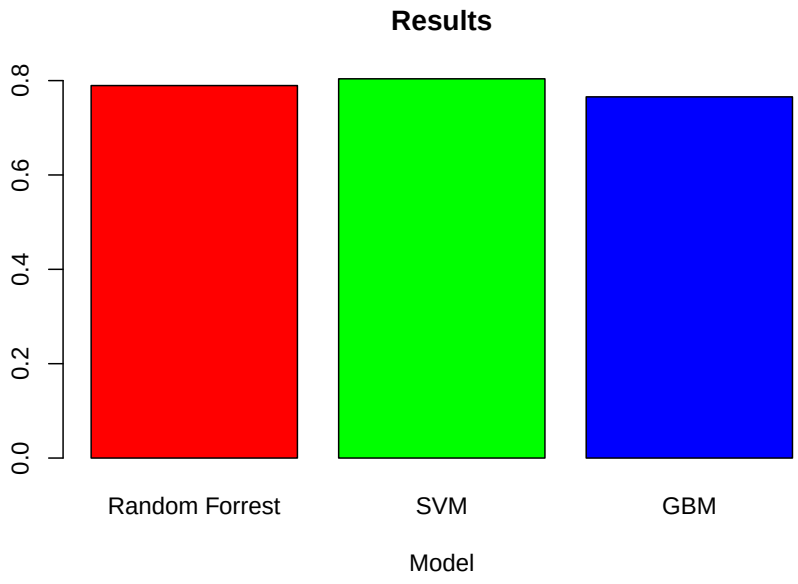
10 RESULTS

Results are evaluated in Kaggle. Scores are shown below.

```
random_forrest_model_result <- 0.78947
svm_model_result <- 0.80382
gbm_model_result <- 0.76555
```

Let's compare the results in a plot.

```
barplot(c(random_forrest_model_result, svm_model_result, gbm_model_result), main="Results", names.arg=c
```



It is clear that SVM model achives the best results. RandomForrest gets the second best result. Finally GBM model is the third model.

11 CONCLUSION

In this work, we tried to predict the survival information of the passengers in the Titanic shipwreck. We did explanatory data analysis and showed some illustrative data visualizations. We did feature engineering and create more meaningful variables. We did missing value imputation and outlier detection. Finally, we built RandomForrest, SVM and GBM models to predict the survival information of the passengers.