Titanc

Alok Kumar

March 22, 2018

Contents

Data Collection :

There is two type of data set

```
> train.df = read.csv("Data/train.csv",
+                      stringsAsFactors = FALSE,
+                      na.strings = c(""),
+                      header = T)
> test.df = read.csv("Data/test.csv",
+                     stringsAsFactors = FALSE,
+                     na.strings = c(""),
+                     header = T)
> sub.df = read.csv('Data/gender_submission.csv', header = T, na.string=c(""))
> test.df = merge(test.df, sub.df, by = "PassengerId")
```

Exploring Data:

The trained dataset contains 891 observations and 12 features (Variable), and the tested dataset contains 418 observations.

```
> str(train.df)
```

```
'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       : chr  "Braund, Mr. Owen Harris" "Cumings, Mrs. John Bradley (Floren
 $ 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" ...
```

Pclass need to be converted in factor

```
> train.df$Pclass = as.factor(train.df$Pclass)
> test.df$Pclass = as.factor(test.df$Pclass)
```

Preparion Data:

Missing Data Analysis

We have to analyse is there any missing data.

Train Data:

```
> sapply(train.df, function(x) sum(is.na(x)))
```

```
PassengerId    Survived      Pclass        Name         Sex         Age
          0           0           0           0           0         177
      SibSp       Parch      Ticket        Fare       Cabin    Embarked
          0           0           0           0         687           2
```

Test Data:

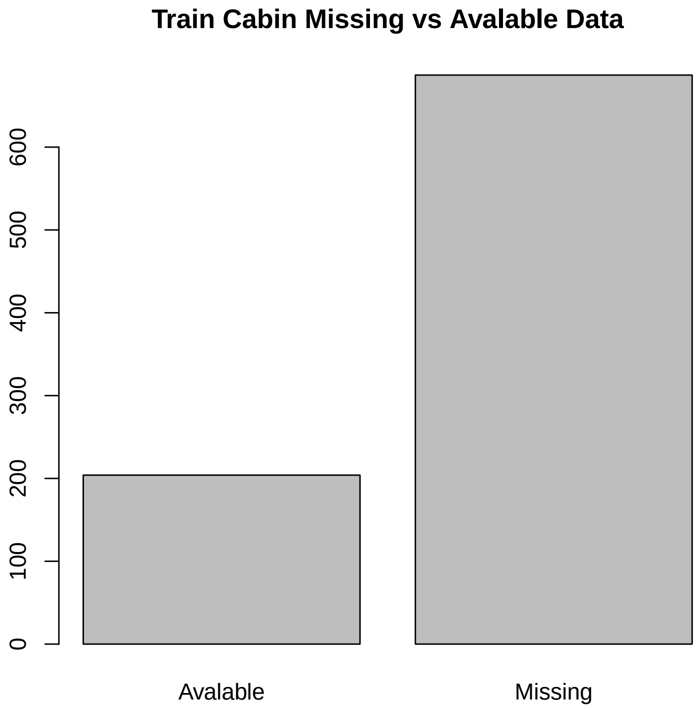
```
> sapply(test.df, function(x) sum(is.na(x)))
```

```
PassengerId      Pclass        Name         Sex         Age       SibSp
          0           0           0           0          86           0
      Parch      Ticket        Fare       Cabin    Embarked    Survived
          0           0           1         327           0           0
```

Age and cabin need to handle. Lets check what fraction of data is missing in both.

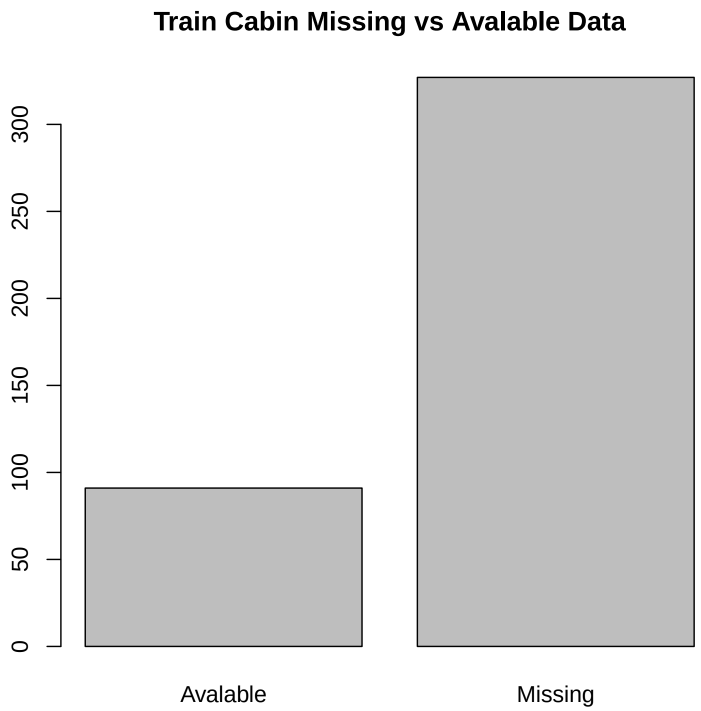
Cabin in train data set

```
> train.df['CabinMissing'] =  sapply(train.df$Cabin,
+                                    function(x)
+                                      ifelse(is.na(x),
+                                         "Missing","Avalable")
+                                    )
> barplot(table(factor(train.df$CabinMissing)),
+         main = "Train Cabin Missing vs Avalable Data")
>
```



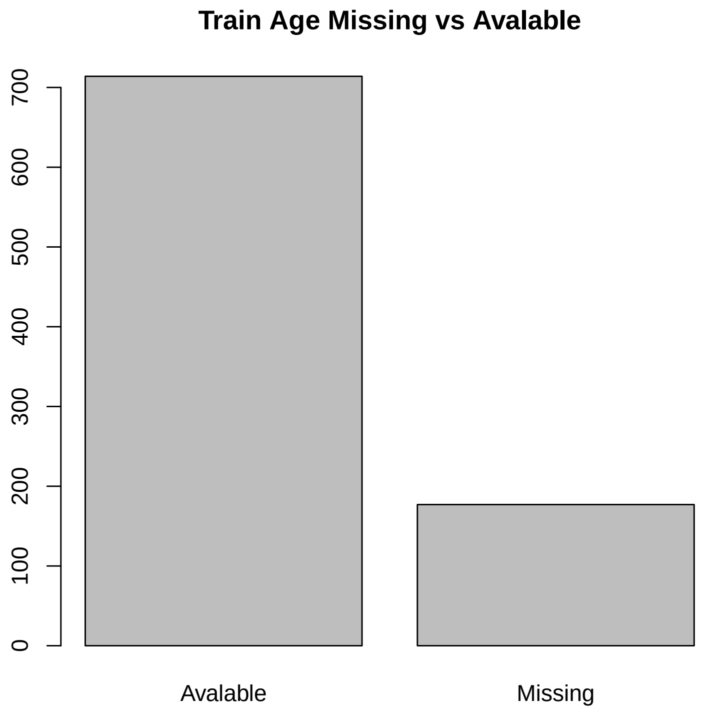
Cabin in test data set

```
> test.df['CabinMissing'] =  sapply(test.df$Cabin,
+                                   function(x)
+                                     ifelse(is.na(x),
+                                         "Missing","Avalable")
+                                     )
> barplot(table(factor(test.df$CabinMissing)),
+         main = "Train Cabin Missing vs Avalable Data")
>
```



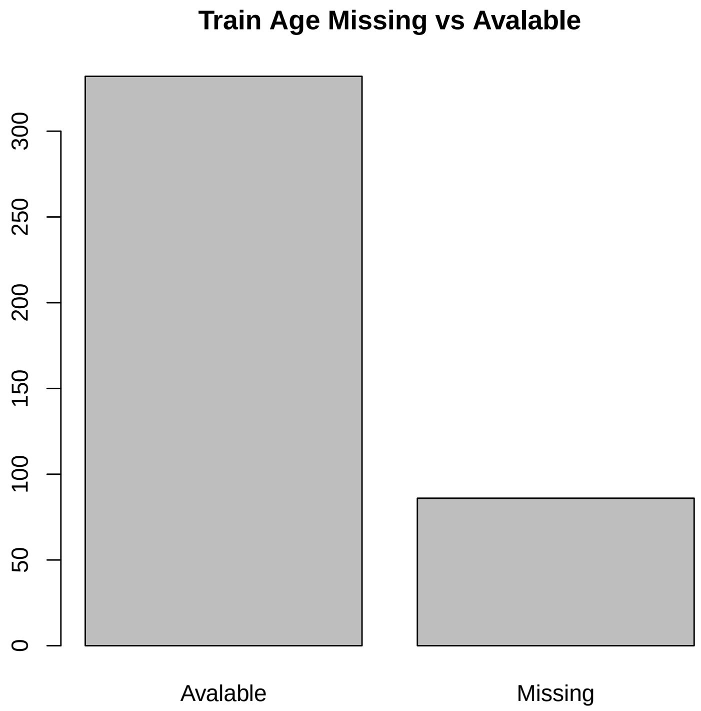
Age in train dataset

```
> train.df['AgeMissing'] =  sapply(train.df$Age,
+                                  function(x)
+                                   ifelse(is.na(x),
+                                     "Missing","Avalable")
+                                  )
> barplot(table(factor(train.df$AgeMissing)),
+         main = "Train Age Missing vs Avalable")
```



Age in test dataset

```
> test.df['AgeMissing'] =  sapply(test.df$Age,
+                                  function(x)
+                                   ifelse(is.na(x),
+                                     "Missing","Avalable")
+                                  )
> barplot(table(factor(test.df$AgeMissing)),
+         main = "Train Age Missing vs Avalable")
```



Since cabin have many missing data we can remove this feather from out data set.

```
> train.df  = subset(train.df, select = c(2,3,5,6,7,8,10,12))
> test.df = subset(test.df, select = c(2,4,5,6,7,9,11,12))
> names(train.df)

[1] "Survived" "Pclass"   "Sex"      "Age"      "SibSp"    "Parch"    "Fare"
[8] "Embarked"

> names(test.df)

[1] "Pclass"   "Sex"      "Age"      "SibSp"    "Parch"    "Fare"     "Embarked"
[8] "Survived"
```

To handle missing value of Age we can apply one of these method.

- Throw out any data with missing values

- Assign the average value
- Use a regression or another simple model to predict the values of missing variables

For checking weather we can use regression or not lets check correlation in predicter.

```
> corr = na.omit(train.df)
> cormat <- round(cor(corr[, c('Age','Fare')]),2)
> cormat

      Age Fare
Age  1.00 0.09
Fare 0.09 1.00

> cormat <- round(cor(corr[, c('Age','Parch')]),2)
> cormat

        Age Parch
Age    1.00 -0.19
Parch -0.19  1.00
```

No correlation found in Age and other predictor. hence regression is not used for missing age. we will fill age by avarage value.

```
> agetrain = train.df$Age
> avgTrainAge = mean(agetrain, na.rm = T)
> train.df$Age[is.na(train.df$Age)] = avgTrainAge
> ageTest = test.df$Age
> avgTestAge = mean(ageTest, na.rm = T)
> test.df$Age[is.na(test.df$Age)] = avgTestAge
>
```

We can remove row with missing

```
> train.df = train.df[!is.na(train.df$Embarked),]
> test.df = test.df[!is.na(test.df$Fare),]
```

Now again checking missing value count

Train data

```
> sapply(train.df, function(x) sum(is.na(x)))

Survived   Pclass      Sex      Age    SibSp    Parch     Fare Embarked 
       0        0        0        0        0        0        0        0 
```

Test data

```
> sapply(test.df, function(x) sum(is.na(x)))

  Pclass      Sex      Age    SibSp    Parch     Fare Embarked Survived 
       0        0        0        0        0        0        0        0 
```

Model on Train data

Logistic Regression

```
> glm1 = glm(train.df$Survived ˜., family = binomial(link = 'logit'), data = tra
> summary(glm1)

Call:
glm(formula = train.df$Survived ˜ ., family = binomial(link = "logit"), 
    data = train.df)

Deviance Residuals: 
    Min       1Q   Median       3Q      Max  
-2.6235  -0.6098  -0.4237   0.6112   2.4512  

Coefficients:
             Estimate Std. Error z value Pr(>|z|)    
(Intercept)  4.102784   0.476303   8.614  < 2e-16 ***
Pclass2     -0.924047   0.297882  -3.102  0.00192 ** 
Pclass3     -2.149626   0.297749  -7.220 5.21e-13 ***
Sexmale     -2.709611   0.201336 -13.458  < 2e-16 ***
Age         -0.039320   0.007888  -4.984 6.21e-07 ***
SibSp       -0.322143   0.109545  -2.941  0.00327 ** 
Parch       -0.095061   0.119028  -0.799  0.42450    
Fare         0.002261   0.002462   0.918  0.35842    
```

```
EmbarkedQ   -0.029839   0.381534  -0.078  0.93766
EmbarkedS   -0.445754   0.239730  -1.859  0.06297 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1182.82  on 888  degrees of freedom
Residual deviance:  783.74  on 879  degrees of freedom
AIC: 803.74

Number of Fisher Scoring iterations: 5
```

Evauvate model

```
> testPredict = predict(glm1, newdata = subset(test.df, select = c(1:7)),
+                             type = 'response')
> testPredict = ifelse(testPredict >0.5, 1, 0)
> cm = table(test.df[, 'Survived'], testPredict > 0.5)
> cm
```

```
    FALSE TRUE
  0   252   13
  1    11  141
```

Acuracy

```
> misClassifiError = mean(testPredict != test.df$Survived)
> #print(paste('Accuracy', 1 - testPredict))
> 1- misClassifiError
```

```
[1] 0.942446
```