Titanic: Machine Learning From Disaster

Luke Kim (mkim14@stanford.edu)

#1. Introduction and preliminary observation of data

Given information about the passenger, we need to predict whether the passenger will survive the titanic accident or not. This is a classification problem. Let us read in the data first. The details of the data is given in this website: https://www.kaggle.com/c/titanic/data. Also, this is the link to the actual competition: https://www.kaggle.com/c/titanic/overview.

```
train <- read.csv('./train.csv')
test <- read.csv('./test.csv')
head(train)
```

```
##   PassengerId Survived Pclass
## 1           1        0      3
## 2           2        1      1
## 3           3        1      3
## 4           4        1      1
## 5           5        0      3
## 6           6        0      3
##                                                  Name    Sex Age SibSp Parch
## 1                             Braund, Mr. Owen Harris   male  22     1     0
## 2 Cumings, Mrs. John Bradley (Florence Briggs Thayer) female  38     1     0
## 3                              Heikkinen, Miss. Laina female  26     0     0
## 4        Futrelle, Mrs. Jacques Heath (Lily May Peel) female  35     1     0
## 5                            Allen, Mr. William Henry   male  35     0     0
## 6                                    Moran, Mr. James   male  NA     0     0
##             Ticket    Fare Cabin Embarked
## 1        A/5 21171  7.2500              S
## 2         PC 17599 71.2833   C85        C
## 3 STON/O2. 3101282  7.9250              S
## 4           113803 53.1000  C123        S
## 5           373450  8.0500              S
## 6           330877  8.4583              Q
```

```
head(test)
```

```
##   PassengerId Pclass                                         Name    Sex  Age
## 1         892      3                             Kelly, Mr. James   male 34.5
## 2         893      3             Wilkes, Mrs. James (Ellen Needs) female 47.0
## 3         894      2                    Myles, Mr. Thomas Francis   male 62.0
## 4         895      3                             Wirz, Mr. Albert   male 27.0
## 5         896      3 Hirvonen, Mrs. Alexander (Helga E Lindqvist) female 22.0
## 6         897      3                   Svensson, Mr. Johan Cervin   male 14.0
##   SibSp Parch  Ticket    Fare Cabin Embarked
## 1     0     0  330911  7.8292              Q
## 2     1     0  363272  7.0000              S
## 3     0     0  240276  9.6875              Q
## 4     0     0  315154  8.6625              S
## 5     1     1 3101298 12.2875              S
## 6     0     0    7538  9.2250              S
```

Let's extract the IdList for submission purposes.

```r
IdList <- test$PassengerId
```

Let's examine the data a little bit. We can first see the summary for each columns.

```r
summary(train)
```

```
##   PassengerId       Survived          Pclass     
##  Min.   :  1.0   Min.   :0.0000   Min.   :1.000  
##  1st Qu.:223.5   1st Qu.:0.0000   1st Qu.:2.000  
##  Median :446.0   Median :0.0000   Median :3.000  
##  Mean   :446.0   Mean   :0.3838   Mean   :2.309  
##  3rd Qu.:668.5   3rd Qu.:1.0000   3rd Qu.:3.000  
##  Max.   :891.0   Max.   :1.0000   Max.   :3.000  
##                                                  
##                                     Name         Sex           Age       
##  Abbing, Mr. Anthony                  :  1   female:314   Min.   : 0.42  
##  Abbott, Mr. Rossmore Edward          :  1   male  :577   1st Qu.:20.12  
##  Abbott, Mrs. Stanton (Rosa Hunt)     :  1                Median :28.00  
##  Abelson, Mr. Samuel                  :  1                Mean   :29.70  
##  Abelson, Mrs. Samuel (Hannah Wizosky):  1                3rd Qu.:38.00  
##  Adahl, Mr. Mauritz Nils Martin       :  1                Max.   :80.00  
##  (Other)                              :885                NA's   :177    
##      SibSp           Parch             Ticket         Fare       
##  Min.   :0.000   Min.   :0.0000   1601    :  7   Min.   :  0.00  
##  1st Qu.:0.000   1st Qu.:0.0000   347082  :  7   1st Qu.:  7.91  
##  Median :0.000   Median :0.0000   CA. 2343:  7   Median : 14.45  
##  Mean   :0.523   Mean   :0.3816   3101295 :  6   Mean   : 32.20  
##  3rd Qu.:1.000   3rd Qu.:0.0000   347088  :  6   3rd Qu.: 31.00  
##  Max.   :8.000   Max.   :6.0000   CA 2144 :  6   Max.   :512.33  
##                                   (Other) :852                   
##          Cabin     Embarked
##             :687    :  2   
##  B96 B98    :  4   C:168   
##  C23 C25 C27:  4   Q: 77   
##  G6         :  4   S:644   
##  C22 C26    :  3           
##  D          :  3           
##  (Other)    :186           
```

```r
summary(test)
```

```
##   PassengerId         Pclass     
##  Min.   : 892.0   Min.   :1.000  
##  1st Qu.: 996.2   1st Qu.:1.000  
##  Median :1100.5   Median :3.000  
##  Mean   :1100.5   Mean   :2.266  
##  3rd Qu.:1204.8   3rd Qu.:3.000  
##  Max.   :1309.0   Max.   :3.000  
##                                  
##                                         Name         Sex           Age       
##  Abbott, Master. Eugene Joseph            :  1   female:152   Min.   : 0.17  
##  Abelseth, Miss. Karen Marie              :  1   male  :266   1st Qu.:21.00  
##  Abelseth, Mr. Olaus Jorgensen            :  1                Median :27.00  
##  Abrahamsson, Mr. Abraham August Johannes :  1                Mean   :30.27  
##  Abrahim, Mrs. Joseph (Sophie Halaut Easu):  1                3rd Qu.:39.00  
##  Aks, Master. Philip Frank                :  1                Max.   :76.00
##  (Other)                                  :412                NA's   :86
##      SibSp            Parch             Ticket          Fare
##  Min.   :0.0000   Min.   :0.0000   PC 17608:  5   Min.   :  0.000
##  1st Qu.:0.0000   1st Qu.:0.0000   113503  :  4   1st Qu.:  7.896
##  Median :0.0000   Median :0.0000   CA. 2343:  4   Median : 14.454
##  Mean   :0.4474   Mean   :0.3923   16966   :  3   Mean   : 35.627
##  3rd Qu.:1.0000   3rd Qu.:0.0000   220845  :  3   3rd Qu.: 31.500
##  Max.   :8.0000   Max.   :9.0000   347077  :  3   Max.   :512.329
##                                    (Other) :396   NA's   :1
##             Cabin     Embarked
##                 :327   C:102
##  B57 B59 B63 B66:  3   Q: 46
##  A34            :  2   S:270
##  B45            :  2
##  C101           :  2
##  C116           :  2
##  (Other)        : 80
```

```
str(train)
```

```
## '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 58
##  $ 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 86 396 345 133 ...
##  $ 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 ...
```

Let's examine how many data points we have

```
nrow(train)
```

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

```
nrow(test)
```

```
## [1] 418
```

We have 891 for training and 418 for testing.

We could then look at the existence of NAs for each columns for the train data

```
colnames(train)[colSums(is.na(train)) > 0]
```

```
## [1] "Age"
```

We do the same for the test data

```
colnames(test)[colSums(is.na(test)) > 0]
```

```
## [1] "Age"  "Fare"
```

There appears to be missing data for Age and Fare in the test set and Age for the train set. Let's try to take care of the missing values for Age first. Also, although there are no NA values, a lot of the data for Cabin are missing. Let's try to deal with these in the next section.

Also, just briefly looking over the data, there appears to be some correlation between Fare and Survived. Let's check if this is true. From the summary of the data above, the mean value of Fare is $32.

```r
business <- train$Survived[train$Fare > 32]
cnt = 0
for (i in 1:length(business)){
  if (business[i] == 1){
    cnt = cnt+1
  }
}
cat("Percent of survival for business = ",cnt*100/length(business),"\n")
```

```
## Percent of survival for business =  59.71564
```

```r
economy <- train$Survived[train$Fare <= 32]
cnt = 0
for (i in 1:length(economy)){
  if (economy[i] == 1){
    cnt = cnt+1
  }
}
cat("Percent of survival for economy = ",cnt*100/length(economy),"\n")
```

```
## Percent of survival for economy =  31.76471
```

Yes this kind of explains that those who paid more to board on the Titanic has close to two times more survival percentage than those who paid less. I believe that Fare actually is one of the most determining features of survival.

Also, let's think about Sex (gender). It would be common to think that women actually had a greater survival percentage than men because women and children were probably rescued first. Let's examine this.

```r
women <- train$Survived[train$Sex == "female"]
men <- train$Survived[train$Sex == "male"]
cnt1 = 0
cnt2 = 0
for (i in 1:length(women)){
  if (women[i] == 1){
    cnt1 = cnt1+1
  }
}
for (i in 1:length(men)){
  if (men[i] == 1){
    cnt2 = cnt2+1
  }
}
cat("Survival percentage of women = ",cnt1*100/length(women),"\n")
```

```
## Survival percentage of women =  74.20382
```

```r
cat("Survival percentage of men = ",cnt2*100/length(men),"\n")
```

```
## Survival percentage of men =  18.89081
```

So we can see that women had a much higher survival rate then men. So gender also plays an important role in predicting survival.

#2. Missing value imputation

There are many ways to impute the missing values for Age, for instance we could replace the NA values for Age to the median or the mean of Age. The method we are going to use for now is the impute the missing Age values using other features to predict Age by the (multiple) linear regression model. Some of the features that may be relevant to determining age could be SibSp (number of siblings), Pclass (ticket class), Parch (number of parents/children aboard), Fare (the amount the passenger had to pay), and Sex (gender of the passenger).

```
age.fit <- lm(Age~Pclass+Sex+SibSp+Fare+Parch,data=train)
summary(age.fit)
```

```
##
## Call:
## lm(formula = Age ~ Pclass + Sex + SibSp + Fare + Parch, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -35.449  -8.177  -1.215   6.785  45.703
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 46.22832    1.92047  24.071  < 2e-16 ***
## Pclass      -7.01138    0.70303  -9.973  < 2e-16 ***
## Sexmale      3.24259    1.03767   3.125  0.00185 **
## SibSp       -3.80316    0.56098  -6.779 2.54e-11 ***
## Fare        -0.01792    0.01133  -1.582  0.11417
## Parch       -0.77186    0.63276  -1.220  0.22293
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.74 on 708 degrees of freedom
##   (177 observations deleted due to missingness)
## Multiple R-squared:  0.2368, Adjusted R-squared:  0.2314
## F-statistic: 43.92 on 5 and 708 DF,  p-value: < 2.2e-16
```

The R-squared value is pretty low, but that is expected since the features I have may be relevant to predicting age, but are not really strong indicators of the age of the passenger. Let's make the prediction of Age based on our linear model, then replace NA values for the Age columns with our predictions.

```
age.prediction <- predict(age.fit,train)
age_list <- train$Age
for (i in 1:length(age_list)){
  if (is.na(age_list[i])){
    age_list[i] = age.prediction[i]
  }
}
train$Age <- age_list
sum(is.na(train$Age))
```

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

It appears that we have successfully imputed missing values for Age for the training data. Let us do the same for the test data.

```
age.fit.test <- lm(Age~Pclass+Sex+SibSp+Fare+Parch,data=test)
summary(age.fit.test)
```

```
##
## Call:
## lm(formula = Age ~ Pclass + Sex + SibSp + Fare + Parch, data = test)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -34.944  -7.643  -0.592   7.505  37.729
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 44.92480    2.70950  16.580  < 2e-16 ***
## Pclass      -7.37539    0.99191  -7.436 9.33e-13 ***
## Sexmale      1.58139    1.40883   1.122   0.2625
## SibSp       -1.35357    0.81864  -1.653   0.0992 .
## Fare         0.02632    0.01426   1.845   0.0659 .
## Parch       -0.86337    0.91524  -0.943   0.3462
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.13 on 325 degrees of freedom
##   (87 observations deleted due to missingness)
## Multiple R-squared:  0.2712, Adjusted R-squared:   0.26
## F-statistic: 24.19 on 5 and 325 DF,  p-value: < 2.2e-16
```

```
age.predict.test <- predict(age.fit.test,test)
age_list_test <- test$Age
for (i in 1:length(age_list_test)){
  if (is.na(age_list_test[i])){
    age_list_test[i] = age.predict.test[i]
  }
}
test$Age <- age_list_test
sum(is.na(test$Age))
```

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

Let us now fill in the missing value for Fare.

```
sum(is.na(test$Fare))
```

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

So there appears to be only one value of Fare that is missing. Let's use linear regression to impute the values for Fare. Factors that may affect the Fare price is Cabin (cabin number),Embarked (the port of embarkation), Pclass ("Ticket class").

```
fare.fit <- lm(Fare~Cabin+Embarked+Pclass,data=test)
fare.predict <- predict(fare.fit,test)
fare_list <- test$Fare
for (i in 1:length(fare_list)){
  if (is.na(fare_list[i])){
    fare_list[i] = fare.predict[i]
  }
}
test$Fare <- fare_list
sum(is.na(test$Fare))
```

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

We noted above that there are also many missing values for Cabin, but they are treated as missing and not exactly filled with NAs. Because there are too many missing values for Cabin and this is not something that we can predict (unless we decide to randomly assign passengers to some seat number - this may add to a lot of noise in the data), we will just let it be.

Logistic Regression

Now let's try to fit a model for prediction. The most famous model for binary classification is logistic regression. In R, we fit logistic regression using the glm() function. First of all, let's get rid of some features that would obviously not help in our prediction. These would include the Id number of the passengers, the name of the passengers and the ticket number, because these are just random information that most likely does not have any relation to survival.

```
df_train <- train
df_train$PassengerId <- NULL
df_train$Name <- NULL
df_train$Ticket <- NULL
glm.fit <- glm(Survived~.,data=df_train,
               family="binomial")
summary(glm.fit)
```

```
##
## Call:
## glm(formula = Survived ~ ., family = "binomial", data = df_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9065  -0.5272  -0.3226   0.2487   2.5069
##
## Coefficients: (1 not defined because of singularities)
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)          4.681e+00  7.471e-01   6.265 3.72e-10 ***
## Pclass              -9.721e-01  2.085e-01  -4.663 3.11e-06 ***
## Sexmale             -2.682e+00  2.236e-01 -11.997  < 2e-16 ***
## Age                 -4.982e-02  1.029e-02  -4.840 1.30e-06 ***
## SibSp               -5.262e-01  1.250e-01  -4.212 2.54e-05 ***
## Parch               -5.240e-02  1.298e-01  -0.404   0.6864
## Fare                 7.633e-03  5.318e-03   1.435   0.1512
## CabinA10            -1.838e+01  6.523e+03  -0.003   0.9978
## CabinA14            -1.792e+01  6.523e+03  -0.003   0.9978
## CabinA16             1.720e+01  6.523e+03   0.003   0.9979
## CabinA19            -1.770e+01  6.523e+03  -0.003   0.9978
## CabinA20             1.980e+01  6.523e+03   0.003   0.9976
## CabinA23             2.130e+01  6.523e+03   0.003   0.9974
## CabinA24            -1.843e+01  6.523e+03  -0.003   0.9977
## CabinA26             1.978e+01  6.523e+03   0.003   0.9976
## CabinA31             1.902e+01  6.523e+03   0.003   0.9977
## CabinA32            -1.790e+01  6.523e+03  -0.003   0.9978
## CabinA34             1.722e+01  6.523e+03   0.003   0.9979
## CabinA36            -1.765e+01  6.523e+03  -0.003   0.9978
## CabinA5             -1.660e+01  6.523e+03  -0.003   0.9980
## CabinA6              1.866e+01  6.523e+03   0.003   0.9977
## CabinA7             -1.731e+01  6.523e+03  -0.003   0.9979
## CabinB101             1.510e+01  6.523e+03   0.002   0.9982
## CabinB102            -1.748e+01  6.523e+03  -0.003   0.9979
## CabinB18              1.582e+01  4.418e+03   0.004   0.9971
## CabinB19             -1.681e+01  6.523e+03  -0.003   0.9979
## CabinB20              1.806e+01  3.668e+03   0.005   0.9961
## CabinB22              7.258e-02  2.616e+00   0.028   0.9779
## CabinB28              1.684e+01  4.466e+03   0.004   0.9970
## CabinB3               1.544e+01  6.523e+03   0.002   0.9981
## CabinB30             -1.705e+01  6.523e+03  -0.003   0.9979
## CabinB35              1.525e+01  4.612e+03   0.003   0.9974
## CabinB37             -1.721e+01  6.523e+03  -0.003   0.9979
## CabinB38             -1.755e+01  6.523e+03  -0.003   0.9979
## CabinB39              1.540e+01  6.523e+03   0.002   0.9981
## CabinB4               1.656e+01  6.523e+03   0.003   0.9980
## CabinB41              2.023e+01  6.523e+03   0.003   0.9975
## CabinB42              1.557e+01  6.523e+03   0.002   0.9981
## CabinB49              1.730e+01  3.782e+03   0.005   0.9964
## CabinB5               1.440e+01  4.567e+03   0.003   0.9975
## CabinB50              1.862e+01  6.523e+03   0.003   0.9977
## CabinB51 B53 B55     -1.395e+00  2.466e+00  -0.566   0.5717
## CabinB57 B59 B63 B66  1.471e+01  4.610e+03   0.003   0.9975
## CabinB58 B60         -2.638e+00  1.873e+00  -1.409   0.1589
## CabinB69              1.731e+01  6.523e+03   0.003   0.9979
## CabinB71             -1.792e+01  6.523e+03  -0.003   0.9978
## CabinB73              1.564e+01  6.523e+03   0.002   0.9981
## CabinB77              1.577e+01  4.610e+03   0.003   0.9973
## CabinB78              1.562e+01  6.523e+03   0.002   0.9981
## CabinB79              1.499e+01  6.523e+03   0.002   0.9982
## CabinB80              1.635e+01  6.523e+03   0.003   0.9980
## CabinB82 B84         -1.818e+01  6.523e+03  -0.003   0.9978
## CabinB86             -1.928e+01  6.523e+03  -0.003   0.9976
## CabinB94             -1.760e+01  6.523e+03  -0.003   0.9978
## CabinB96 B98          1.761e+01  2.700e+03   0.007   0.9948
## CabinC101             1.816e+01  6.523e+03   0.003   0.9978
## CabinC103             1.754e+01  6.523e+03   0.003   0.9979
## CabinC104             1.990e+01  6.523e+03   0.003   0.9976
## CabinC106             1.939e+01  6.523e+03   0.003   0.9976
## CabinC110            -1.765e+01  6.523e+03  -0.003   0.9978
## CabinC111            -1.859e+01  6.523e+03  -0.003   0.9977
## CabinC118            -1.820e+01  6.523e+03  -0.003   0.9978
## CabinC123            -4.532e-01  1.799e+00  -0.252   0.8010
## CabinC124            -1.763e+01  4.609e+03  -0.004   0.9969
## CabinC125             1.622e+01  4.517e+03   0.004   0.9971
## CabinC126             1.895e+01  3.675e+03   0.005   0.9959
## CabinC128            -1.778e+01  6.523e+03  -0.003   0.9978
## CabinC148             1.833e+01  6.523e+03   0.003   0.9978
## CabinC2              -1.079e+00  1.874e+00  -0.576   0.5646
## CabinC22 C26         -3.767e+00  1.532e+00  -2.459   0.0139 *
## CabinC23 C25 C27     -1.375e+00  1.742e+00  -0.789   0.4300
## CabinC30             -1.709e+01  6.523e+03  -0.003   0.9979
## CabinC32              1.534e+01  6.523e+03   0.002   0.9981
## CabinC45              1.474e+01  6.523e+03   0.002   0.9982
## CabinC46             -1.787e+01  6.523e+03  -0.003   0.9978
## CabinC47              1.912e+01  6.523e+03   0.003   0.9977
## CabinC49             -2.028e+01  6.523e+03  -0.003   0.9975
## CabinC50              1.679e+01  6.523e+03   0.003   0.9979
## CabinC52              1.907e+01  4.570e+03   0.004   0.9967
## CabinC54              1.514e+01  6.523e+03   0.002   0.9981
## CabinC62 C64          1.427e+01  6.523e+03   0.002   0.9983
## CabinC65             -2.078e+00  1.832e+00  -1.134   0.2568
## CabinC68             -7.204e-01  1.949e+00  -0.370   0.7117
## CabinC7               1.525e+01  6.523e+03   0.002   0.9981
## CabinC70              1.737e+01  6.523e+03   0.003   0.9979
## CabinC78             -6.574e-01  2.155e+00  -0.305   0.7603
## CabinC82             -2.003e+01  6.523e+03  -0.003   0.9975
## CabinC83             -4.858e-01  1.924e+00  -0.253   0.8006
## CabinC85              1.646e+01  6.523e+03   0.003   0.9980
## CabinC86             -1.766e+01  6.523e+03  -0.003   0.9978
## CabinC87             -1.671e+01  6.523e+03  -0.003   0.9980
## CabinC90              1.540e+01  6.523e+03   0.002   0.9981
## CabinC91             -1.882e+01  6.523e+03  -0.003   0.9977
## CabinC92              1.842e+01  3.652e+03   0.005   0.9960
## CabinC93              1.826e+01  3.827e+03   0.005   0.9962
## CabinC95             -1.937e+01  6.523e+03  -0.003   0.9976
## CabinC99              1.557e+01  6.523e+03   0.002   0.9981
## CabinD                4.006e-01  1.537e+00   0.261   0.7943
## CabinD10 D12          1.798e+01  6.523e+03   0.003   0.9978
## CabinD11              1.733e+01  6.523e+03   0.003   0.9979
## CabinD15              1.559e+01  6.523e+03   0.002   0.9981
## CabinD17              1.708e+01  4.612e+03   0.004   0.9970
## CabinD19              1.976e+01  6.523e+03   0.003   0.9976
## CabinD20              1.715e+01  4.611e+03   0.004   0.9970
## CabinD21              1.671e+01  6.523e+03   0.003   0.9980
## CabinD26             -1.845e+01  4.352e+03  -0.004   0.9966
## CabinD28              1.541e+01  6.523e+03   0.002   0.9981
## CabinD30             -1.853e+01  6.523e+03  -0.003   0.9977
## CabinD33              1.863e+01  3.888e+03   0.005   0.9962
## CabinD35              1.870e+01  4.027e+03   0.005   0.9963
## CabinD36              1.560e+01  4.595e+03   0.003   0.9973
## CabinD37              1.752e+01  6.523e+03   0.003   0.9979
## CabinD45              1.940e+01  6.523e+03   0.003   0.9976
## CabinD46             -1.751e+01  6.523e+03  -0.003   0.9979
## CabinD47              1.571e+01  6.523e+03   0.002   0.9981
## CabinD48             -1.774e+01  6.523e+03  -0.003   0.9978
## CabinD49              1.802e+01  6.523e+03   0.003   0.9978
## CabinD50             -1.680e+01  6.523e+03  -0.003   0.9979
## CabinD56              2.011e+01  6.523e+03   0.003   0.9975
## CabinD6              -1.838e+01  6.523e+03  -0.003   0.9978
## CabinD7               1.793e+01  6.523e+03   0.003   0.9978
## CabinD9               1.531e+01  6.523e+03   0.002   0.9981
## CabinE10              2.102e+01  6.523e+03   0.003   0.9974
## CabinE101             1.716e+01  3.754e+03   0.005   0.9964
## CabinE12              1.973e+01  6.523e+03   0.003   0.9976
## CabinE121             1.903e+01  4.195e+03   0.005   0.9964
## CabinE17              1.988e+01  6.523e+03   0.003   0.9976
## CabinE24              1.927e+01  4.599e+03   0.004   0.9967
## CabinE25              1.913e+01  4.612e+03   0.004   0.9967
## CabinE31             -1.724e+01  6.523e+03  -0.003   0.9979
## CabinE33              1.602e+01  4.549e+03   0.004   0.9972
## CabinE34              1.613e+01  6.523e+03   0.002   0.9980
## CabinE36              1.564e+01  6.523e+03   0.002   0.9981
## CabinE38             -1.656e+01  6.523e+03  -0.003   0.9980
## CabinE40              1.560e+01  6.523e+03   0.002   0.9981
## CabinE44             -5.113e-02  1.930e+00  -0.026   0.9789
## CabinE46             -1.730e+01  6.523e+03  -0.003   0.9979
## CabinE49              1.647e+01  6.523e+03   0.003   0.9980
## CabinE50              1.861e+01  6.523e+03   0.003   0.9977
## CabinE58             -1.745e+01  6.523e+03  -0.003   0.9979
## CabinE63             -1.755e+01  6.523e+03  -0.003   0.9979
## CabinE67             -1.302e-01  1.969e+00  -0.066   0.9473
## CabinE68              1.525e+01  6.523e+03   0.002   0.9981
## CabinE77             -1.854e+01  6.523e+03  -0.003   0.9977
## CabinE8               1.810e+01  3.965e+03   0.005   0.9964
## CabinF E69            1.794e+01  6.523e+03   0.003   0.9978
## CabinF G63           -1.561e+01  6.523e+03  -0.002   0.9981
## CabinF G73           -1.662e+01  4.602e+03  -0.004   0.9971
## CabinF2               1.598e+00  1.304e+00   1.225   0.2206
## CabinF33              1.720e+01  3.750e+03   0.005   0.9963
## CabinF38             -1.661e+01  6.523e+03  -0.003   0.9980
## CabinF4               1.843e+01  3.919e+03   0.005   0.9962
## CabinG6              -8.094e-01  1.081e+00  -0.749   0.4539
## CabinT               -1.762e+01  6.523e+03  -0.003   0.9978
## EmbarkedC             2.771e-01  2.982e-01   0.929   0.3527
## EmbarkedQ             3.102e-01  3.472e-01   0.894   0.3716
## EmbarkedS                    NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1186.66  on 890  degrees of freedom
## Residual deviance:  619.99  on 735  degrees of freedom
## AIC: 931.99
##
## Number of Fisher Scoring iterations: 17
```

It appears that Cabin information may actually just be adding noise to our prediction. Let's get rid of it and fit another logistic regression model

```
df_train$Cabin <- NULL
glm.fit <- glm(Survived~.,data=df_train,
               family="binomial")
summary(glm.fit)
```

```
##
## Call:
## glm(formula = Survived ~ ., family = "binomial", data = df_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7013  -0.6056  -0.4086   0.6247   2.4742
##
## Coefficients:
```

```
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)  17.908894 607.444100   0.029 0.976480
## Pclass       -1.194229   0.150561  -7.932 2.16e-15 ***
## Sexmale      -2.693986   0.201671 -13.358  < 2e-16 ***
## Age          -0.043953   0.008229  -5.341 9.24e-08 ***
## SibSp        -0.403245   0.107173  -3.763 0.000168 ***
## Parch        -0.081576   0.119801  -0.681 0.495919
## Fare          0.001596   0.002370   0.674 0.500522
## EmbarkedC   -12.285422 607.443874  -0.020 0.983864
## EmbarkedQ   -12.381050 607.443929  -0.020 0.983738
## EmbarkedS   -12.680140 607.443858  -0.021 0.983346
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1186.66  on 890  degrees of freedom
## Residual deviance:  780.99  on 881  degrees of freedom
## AIC: 800.99
##
## Number of Fisher Scoring iterations: 13
```

Let's now make prediction with the fitted logistic regression model. We say that the person will survive if the probability of predicted survival is greater than 0.5. Let's also write this prediction to our submission file.

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
df_test <- test
df_test$PassengerId <- NULL
df_test$Name <- NULL
df_test$Ticket <- NULL
df_test$Cabin <- NULL
glm.probs <- predict(glm.fit,df_test,type="response")
glm.binary <- ifelse(glm.probs > 0.5,1,0)
submission <- data_frame('PassengerId' = IdList, 'Survived' = glm.binary)
```

```
## Warning: `data_frame()` is deprecated, use `tibble()`.
## This warning is displayed once per session.
```

```
write_csv(submission,'titanic_logistic_regression.csv')
```

Upon submission, we get a result of 0.76076 which means we managed to predict approximately 76% of the outcomes correctly. This places us in the top 80%. Our aim in the later sections would be to improve this result!

Suppor Vector Machines

Next up after logistic regression is support vector machines. We first use SVMs with linear kernel and cost 10. The cost argument simply means that when cost is small, then the margins will be wide and many support vectors will be on the margin or will violate the margin, whereas if the cost is large the margins will be narrow and there will be few support vectors on the margin or violating the margin.

```r
library(e1071)
svm.lin <- svm(as.factor(Survived)~Pclass+Sex+Age+SibSp+Parch+Fare,
               data=train,kernel="linear",cost=10)
svm.lin.pred <- predict(svm.lin,test)
submission <- data_frame('PassengerId' = IdList, 'Survived' = svm.lin.pred)
write_csv(submission,'titanic_svmlinear1.csv')
```

Upon submission, we get a result of 76.5% a slight improvement from our previous submission with logistic regression. Let's see if we can do better with different kernels (something more flexible than linear).

```r
svm.rad <- svm(as.factor(Survived)~Pclass+Sex+Age+SibSp+Parch+Fare,
               data=train,kernel="radial",cost=10,gamma=1)
svm.rad.pred <- predict(svm.rad,test)
submission <- data_frame('PassengerId' = IdList, 'Survived' = svm.rad.pred)
write_csv(submission,'titanic_svmradial1.csv')
```

We did slightly worse at 76%. We can play around with different values of costs and gamma and different kernels, but the improvement seems minimal, so let's switch to a different model where we might be able to see dramatic improvements in prediction accuracy.

Random Forests

Random forests is when we first get a bootstrapped sample of the tree and consider only a subset of the variables at each step to form a decision tree. We repeat this many times to get a wide variety of trees. We test the data on all the trees we made and take the one with the most number of votes (since we are dealing with a binary classification problem). We also call this bagging, since we use bootstrapping to generate data and use the aggregate to make a decision. Let's see how well our random forest works for predicting the titanic data.

```r
library(randomForest)
```

```
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

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

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
set.seed(1)
rf <- randomForest(as.factor(Survived)~Pclass+Sex+Age+SibSp+Parch+Fare,
                   data = train,ntree=100,importance=TRUE)
rf.predict <- predict(rf,test)
```

```
submission <- data_frame('PassengerId' = IdList,'Survived' = rf.predict)
write_csv(submission,'titanic_rf1.csv')
```

Upon submission, we see a slight increase in accuraccy to 77.5%. Let's try more number of trees.

```
library(randomForest)
set.seed(1)
rf <- randomForest(as.factor(Survived)~Pclass+Sex+Age+SibSp+Parch+Fare,
                   data = train,ntree=1000,importance=TRUE)
rf.predict <- predict(rf,test)
submission <- data_frame('PassengerId' = IdList,'Survived' = rf.predict)
write_csv(submission,'titanic_rf2.csv')
```

A slight decrease in performance at 77% accuracy. Let's move on to a different model, namely the gradient boosting model.

Boosting

Random forest models build an ensemble of deep independent trees, whereas the GBM builds an ensemble of shallow and weak successive trees with each tree learning and improving on the previous, and when these trees are combined, they produce a powerful committee that outperforms many other models. For GBM, it is important to tune the parameters. - n.trees: the number of iterations (trees) in the model - interaction.depth: the numbers of splits it has to perform on a tree - shrinkage: learning rate
- n.minobsinnode: minimum number of observations in the tree terminal nodes.

Let's just try some combination of parameters I like to use:

```
library(gbm)
```

```
## Loaded gbm 2.1.5
```

```
set.seed(1)
boost.model <- gbm(Survived~Pclass+Sex+Age+SibSp+Parch+Fare,
                   data=train,n.trees = 10000,interaction.depth=4,
                   shrinkage=0.001,distribution="bernoulli")
boost.predict <- predict(boost.model,newdata=test,n.trees=10000,type="response")
boost.binary <- ifelse(boost.predict > 0.5,1,0)
submission <- data_frame('PassengerId' = IdList,'Survived' = boost.binary)
write_csv(submission,'titanic_gbm2.csv')
```

Upon submission, this gbm model gives a prediction accuracy of 78.46%. Again a small improvement. Although it was a small improvement, we jumped from top 80% to top 31% of all participants. Let's see how we can further improve our score.

Let's try the caret package for tuning the number of trees (3 fold repated cross validation). Let's try with smaller number of trees from the range 50~300.

```
library(caret)
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```r
fitControl <- trainControl(method='repeatedcv',
                           number = 3,
                           repeats = 3)

newGrid <- expand.grid(n.trees = c(50, 100, 150, 200, 250, 300),
                       interaction.depth = 6,
                       shrinkage = 0.01,
                       n.minobsinnode = 10
                       )

gbmFit <- train(Survived ~Pclass+Sex+Age+SibSp+Parch+Fare, data=train,
                 method = 'gbm',
                 trControl = fitControl,
                 tuneGrid =  newGrid,
                 bag.fraction = 0.5,
                 verbose = FALSE)
```

```
## Warning in train.default(x, y, weights = w, ...): You are trying to do
## regression and your outcome only has two possible values Are you trying to do
## classification? If so, use a 2 level factor as your outcome column.
```

```r
gbmFit$bestTune
```

```
##   n.trees interaction.depth shrinkage n.minobsinnode
## 6     300                 6      0.01             10
```

We now tune the interaction depth and the learning rate.

```r
newGrid <- expand.grid(n.trees = 300,
                       interaction.depth = c(4:10),
                       shrinkage = c(0.01,0.001),
                       n.minobsinnode = 10
                       )

gbmFit <- train(Survived ~Pclass+Sex+Age+SibSp+Parch+Fare, data=train,
                 method = 'gbm',
                 trControl = fitControl,
                 tuneGrid =  newGrid,
                 bag.fraction = 0.5,
                 verbose = FALSE)
```

```
## Warning in train.default(x, y, weights = w, ...): You are trying to do
## regression and your outcome only has two possible values Are you trying to do
## classification? If so, use a 2 level factor as your outcome column.
```

```r
gbmFit$bestTune
```

```
##    n.trees interaction.depth shrinkage n.minobsinnode
## 14     300                10      0.01             10
```

Let's try best tune values (among the values we tried)

```r
set.seed(1)
boost.tune <- gbm(Survived~Pclass+Sex+Age+SibSp+Parch+Fare,
                   data=train,n.trees = 300,interaction.depth=10,
                   shrinkage=0.01,n.minobsinnode = 10,distribution="bernoulli")
boost.predict <- predict(boost.tune,newdata=test,n.trees=300,type="response")
boost.binary <- ifelse(boost.predict > 0.5,1,0)
submission <- data_frame('PassengerId' = IdList,'Survived' = boost.binary)
write_csv(submission,'titanic_gbmtune1.csv')
```

We get an accuracy of 79.4%. Great! an improvement. We are now within the top 17% of the participants. Perhaps we could try some more values for tuning

```r
library(caret)
fitControl <- trainControl(method='repeatedcv',
                           number = 3,
                           repeats = 3)

newGrid <- expand.grid(n.trees = c(350,400,450,500,550,600),
                       interaction.depth = c(4:12),
                       shrinkage = c(0.01,0.001),
                       n.minobsinnode = 10
                       )

gbmFit <- train(Survived~Pclass+Sex+Age+SibSp+Parch+Fare, data=train,
                method = 'gbm',
                trControl = fitControl,
                tuneGrid =  newGrid,
                bag.fraction = 0.5,
                verbose = FALSE)

gbmFit$bestTune
```

We obtained the best tune to be 550 trees, 6 for interaction depth, 0.01 shrinkage. Let's try these parameters.

```r
set.seed(1)
boost.tune <- gbm(Survived~Pclass+Sex+Age+SibSp+Parch+Fare,
                  data=train,n.trees = 550,interaction.depth=6,
                  shrinkage=0.01,n.minobsinnode = 10,distribution="bernoulli")
boost.predict <- predict(boost.tune,newdata=test,n.trees=550,type="response")
boost.binary <- ifelse(boost.predict > 0.5,1,0)
submission <- data_frame('PassengerId' = IdList,'Survived' = boost.binary)
write_csv(submission,'titanic_gbmtune2.csv')
```

Unfortunately, we have a lower accuracy at 77%. How about we try more features?

Feature Engineering

We can do some feature engineering and come up with some additionally interesting features such as family size, which is a variable denoting how many familiy members a person has on board on Titanic.

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

We can also generate features Title and marriage status.

Title means "Mr","Mrs","Dr","Miss" etc. Notice that the title is between the delimiters "," and "." in the Name column.

```r
title.extract <- function(x){
    strsplit(toString(x), split = "[,.]")[[1]][2]
}
```

```r
train$Title <- sapply(train$Name,FUN=title.extract)
test$Title <- sapply(test$Name,FUN=title.extract)
train$Title <- sub(" ","",train$Title)
test$Title <- sub(" ","",test$Title)
# convert to factor data type
train$Title <- as.factor(train$Title)
test$Title <- as.factor(test$Title)
```

To get the marital status, if the title is Mrs, we say the person is married (1) and if the title is something else, we label it 0 (we do not know if the person is married or not). However, the Title column is probably enough for now (we'll see if we should add marriage status later).

Perhaps we should apply feature engineering for Cabin too. Our plan is to extract the first letter of the cabin and replace the cabin information with this first letter. For missing values (empty string) for cabin, we will replace it with the character 'NO' to indicate that we do not have information about this Cabin.

```r
cabin.group.extract <- function(x){
  substring(x,1,1)
}
train$CabinGroup <- sapply(train$Cabin,cabin.group.extract)
test$CabinGroup <- sapply(test$Cabin,cabin.group.extract)
for (i in 1:length(train$CabinGroup)){
  if (train$CabinGroup[i]==""){
    train$CabinGroup[i] =  "NO"
  }
}
for (i in 1:length(test$CabinGroup)){
  if (test$CabinGroup[i] == ""){
    test$CabinGroup[i] = "NO"
  }
}
# convert to factors
train$CabinGroup <- as.factor(train$CabinGroup)
test$CabinGroup <- as.factor(test$CabinGroup)
```

With these new features, let's run caret again for hyper parameter optimization.

```r
library(caret)
fitControl <- trainControl(method='repeatedcv',
                           number = 3,
                           repeats = 3)

newGrid <- expand.grid(n.trees = c(50,100,150,200,250,300,350,400,450,500,550,600),
                       interaction.depth = c(4:12),
                       shrinkage = c(0.01,0.001),
                       n.minobsinnode = 10
                       )

gbmFit <- train(Survived~Pclass+Sex+Age+SibSp+Parch+Fare+
                  FamilySize+Title+CabinGroup,
                data=train,
                 method = 'gbm',
                 trControl = fitControl,
                 tuneGrid =  newGrid,
                 bag.fraction = 0.5,
                 verbose = FALSE)

gbmFit$bestTune
```

It took a while to run, but the parameters we get are n.trees=350, interaction.depth=9 and shrinkage = 0.01.

```
set.seed(1)
boost.tune <- gbm(Survived~Pclass+Sex+Age+SibSp+Parch+Fare+FamilySize+Title+CabinGroup,
                  data=train,n.trees = 350,interaction.depth=9,
                  shrinkage=0.01,n.minobsinnode = 10,distribution="bernoulli")
boost.predict <- predict(boost.tune,newdata=test,n.trees=350,type="response")
boost.binary <- ifelse(boost.predict > 0.5,1,0)
head(boost.binary)
```

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

```
submission <- data_frame('PassengerId' = IdList,'Survived' = boost.binary)
write_csv(submission,'titanic_gbm_added.csv')
```

77% accuracy upon submission. 79.4% is our best accuracy so far.

Logistic Regression with Regularization

Let's try logistic regression with L1 and L2 regularization. First is a ridge regression, with L2 regularization.

```
df_train <- train
df_test <- test
df_test$Survived <- NA
combined <- rbind(df_train,df_test)
df_train <- combined[1:nrow(df_train),]
df_test <- combined[nrow(df_train)+1:nrow(test),]
library(glmnet)
```

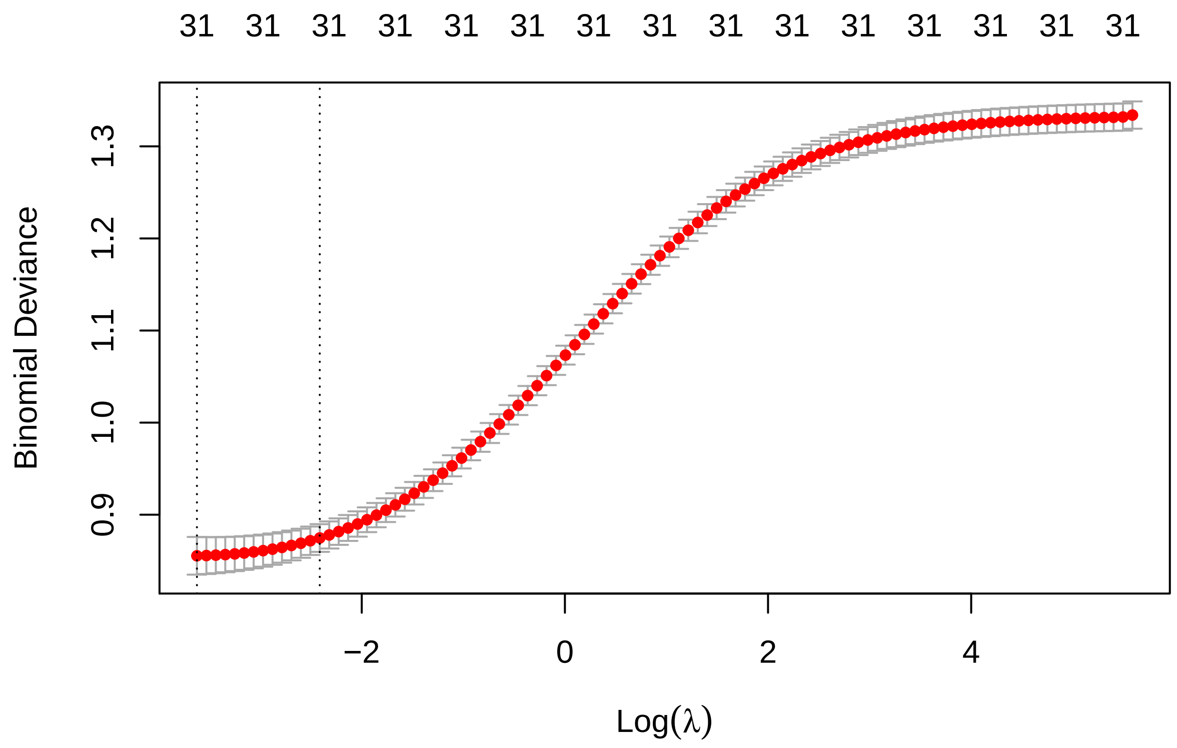
```
## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

## Loaded glmnet 3.0-1
```

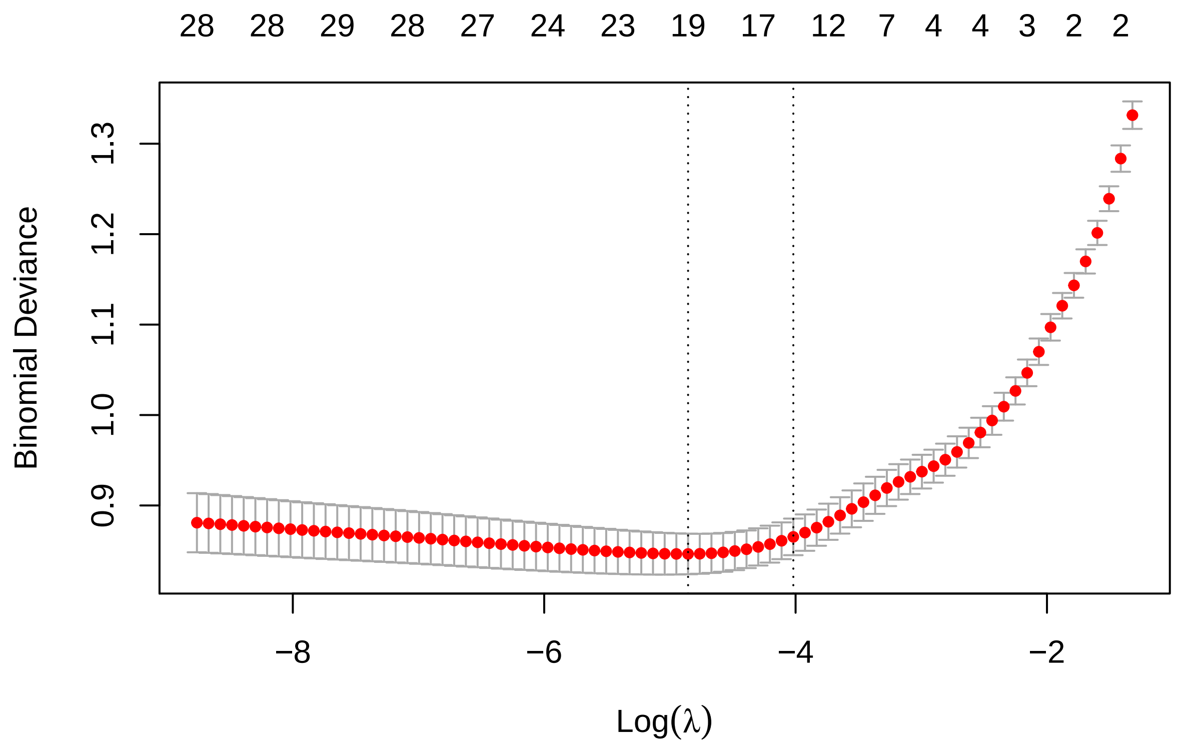
```
set.seed(1)
x <- model.matrix(Survived~Pclass+Sex+Age+SibSp+Parch+Fare+FamilySize+Title+CabinGroup,data = df_train)
y <- df_train$Survived
newx <- model.matrix(~Pclass+Sex+Age+SibSp+Parch+Fare+FamilySize+Title+CabinGroup,data=df_test)
ridge.fit <- cv.glmnet(x,y,alpha=0,family='binomial',type.measure='deviance')
plot(ridge.fit)
```



```
ridge.pred <- predict(ridge.fit,newx=newx,s='lambda.min',type='response')
ridge.binary <- ifelse(ridge.pred > 0.5,1,0)
submission <- data_frame('PassengerId' = test$PassengerId,'Survived' = ridge.binary)
write.csv(submission,'titanic_logistic_ridge.csv',row.names=FALSE)
```

We then try L1 regularization, or the Lasso.

```
set.seed(1)
x <- model.matrix(Survived~Pclass+Sex+Age+SibSp+Parch+Fare+FamilySize+Title+CabinGroup,data = df_train)
y <- df_train$Survived
newx <- model.matrix(~Pclass+Sex+Age+SibSp+Parch+Fare+FamilySize+Title+CabinGroup,data=df_test)
lasso.fit <- cv.glmnet(x,y,alpha=1,family='binomial',type.measure='deviance')
plot(lasso.fit)
```



```
lasso.pred <- predict(lasso.fit,newx=newx,s='lambda.min',type='response')
lasso.binary <- ifelse(lasso.pred > 0.5,1,0)
submission <- data_frame('PassengerId' = test$PassengerId,'Survived' = lasso.binary)
write.csv(submission,'titanic_logistic_lasso.csv',row.names=FALSE)
```

Both record a score of 77.5%. Somewhat similar to logistic regression without any regularization.

Ensemble

We use the majority vote method to ensemble the three best performing results.

```
gbm_best <- read.csv('./titanic_gbmtune1.csv')$Survived
gbm_second_best <- read.csv('./titanic_gbm2.csv')$Survived
lasso_best <- read.csv('./titanic_logistic_lasso.csv')$Survived
Ensemble <- c()
for (i in 1:length(gbm_best)){
  cnt1 <- 0
  cnt0 <- 0
  if (gbm_best[i] == 1) cnt1 = cnt1+1
  else cnt0 = cnt0+1
  if (gbm_second_best[i] == 1) cnt1 = cnt1+1
  else cnt0 = cnt0+1
  if (lasso_best[i] == 1) cnt1 = cnt1+1
  else cnt0 = cnt0+1
  if (cnt0 > cnt1){
    Ensemble <- c(Ensemble,0)
  }else{
    Ensemble <- c(Ensemble,1)
  }
}
submission <- data_frame('PassengerId' = test$PassengerId,'Survived' = Ensemble)
write.csv(submission,'titanice_ensemble.csv')
```

Unfortunately, a worse performance of 74% accuracy.

Possible next steps

- More feature engineering: we have not looked at features such as Ticket and Embarked in great detail. Also, we could create more relevant features from Title, such as the significance of Mrs (is this woman a mother?) or Master (what does it mean for a male to be labeled as a master?) Also, especially for GBM, it appeared that after feature engineering the results showed a decrease in prediction accuracy, and this might mean that some of the features I came up with could be adding noise to the data.
- Trying more models after better feature engineering: We could try xgboost, lightgbm, logistic regression with regularization etc and if we achieve reasonably high results on the public leaderboard, we could use the results from these models to form an ensemble. We would have to be using majority vote ensemble as we are dealing with a binary classification problem.
- Possibly get rid of anomalies if there exists any