Titanic - Predicting Surviors

Ryan Porter

3/5/2020

The goal is to predict how many people survived and didn't survive the Titanic disaster. Below are different methods that I used in order to predict surivorship.

Loading the Data

Loading the data was simple but I opted to work on the project on my local computer rather than kaggle's workbook. I did this because I wanted to use GitHub for the project since I work on my laptop and a desktop.

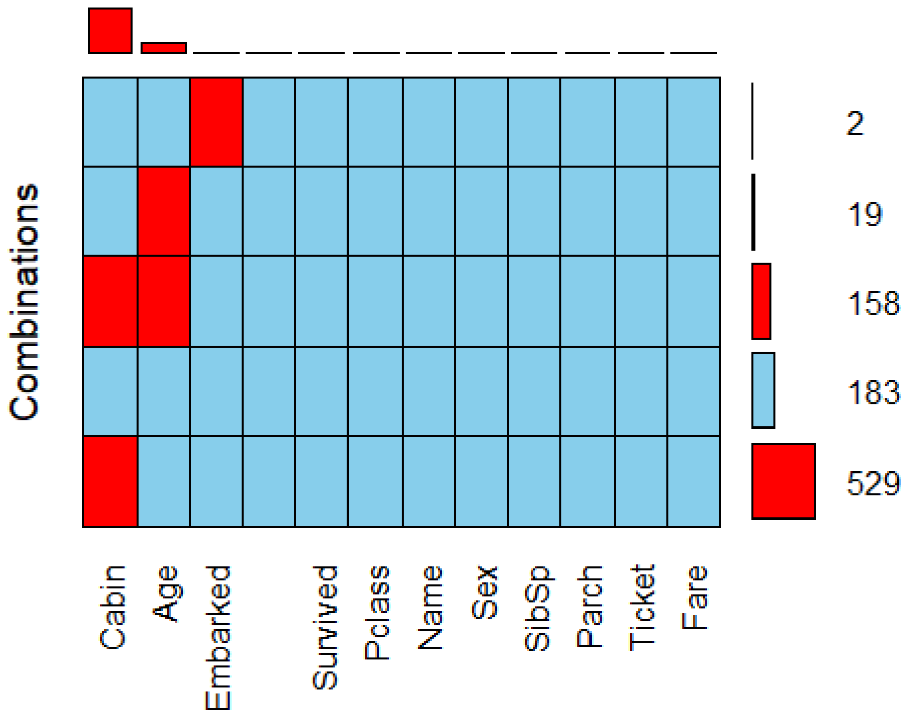
```
#loading packages
library(tidyverse)
library(caret)
library(rpart)
library(VIM)
library(naniar)
library(car)
library(MASS)
library(fastAdaboost)
library(mda)

#loading the data
list.files(path = "C:/Users/Ryan/Documents/titanic")

##  [1] "feature_engineering.R"  "gender_submission.csv"  "README.md"
##  [4] "submission.csv"         "submission_ada.csv"     "submission_fda.csv
"
##  [7] "submission_rf.csv"      "submission_svm.csv"     "test.csv"
## [10] "titanic.Rproj"          "titanic_code.R"         "titanic_markdown.d
ocx"
## [13] "titanic_markdown.Rmd"   "titanic_markdown_files" "train.csv"

train_data <- read_csv("C:/Users/Ryan/Documents/titanic/train.csv")
test_data <- read_csv("C:/Users/Ryan/Documents/titanic/test.csv")
```

Desciptive Statistics

I made a table of each of the variables that way I know what values they were able to have. I also graphed a few of the combinations in order to determine if there were any correlations that was easily detectable. First class and fare had the biggest spread when looking at the second graph. Also, I know sex is going to play an important factor since at that time men were expect to let women escape before them. Children are probably also more likely to survive since older people would want the kids to escape.

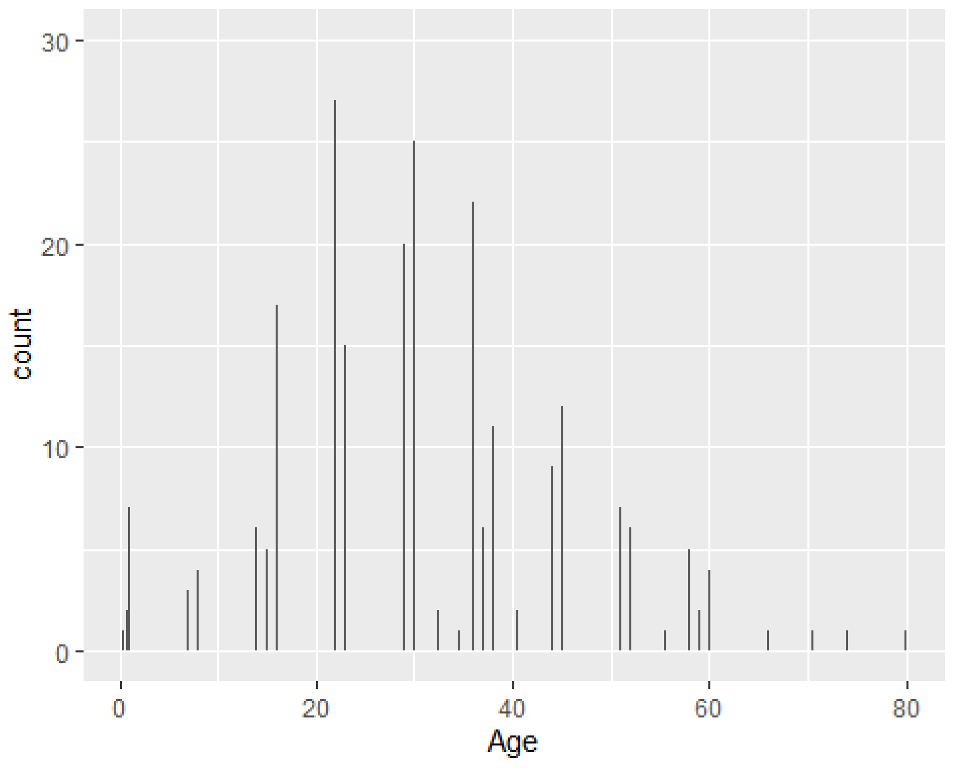
```
table(train_data$Pclass)

##
##   1   2   3
## 216 184 491

table(train_data$Survived)

##
##   0   1
## 549 342

table(train_data$Age)

##
## 0.42 0.67 0.75 0.83 0.92    1    2    3    4    5    6    7    8    9   10
11
##    1    1    2    2    1    7   10    6   10    4    3    3    4    8    2
4
##   12   13   14 14.5   15   16   17   18   19   20 20.5   21   22   23 23.5
24
##    1    2    6    1    5   17   13   26   25   15    1   24   27   15    1
30
## 24.5   25   26   27   28 28.5   29   30 30.5   31   32 32.5   33   34 34.5
35
##    1   23   18   18   25    2   20   25    2   17   18    2   15   15    1
18
##   36 36.5   37   38   39   40 40.5   41   42   43   44   45 45.5   46   47
48
##   22    1    6   11   14   13    2    6   13    5    9   12    2    3    9
9
##   49   50   51   52   53   54   55 55.5   56   57   58   59   60   61   62
63
##    6   10    7    6    1    8    2    1    4    2    5    2    4    3    4
2
##   64   65   66   70 70.5   71   74   80
##    2    3    1    2    1    2    1    1

table(train_data$Embarked)

##
##   C   Q   S
## 168  77 644

table(train_data$SibSp)

##
##   0   1   2   3   4   5   8
## 608 209  28  16  18   5   7
```

```
table(train_data$Parch)

##
##   0   1   2   3   4   5   6
## 678 118  80   5   4   5   1

aggr(train_data, prop = FALSE, combined = TRUE, numbers = TRUE, sortVars = TR
UE, sortCombs = TRUE)
```

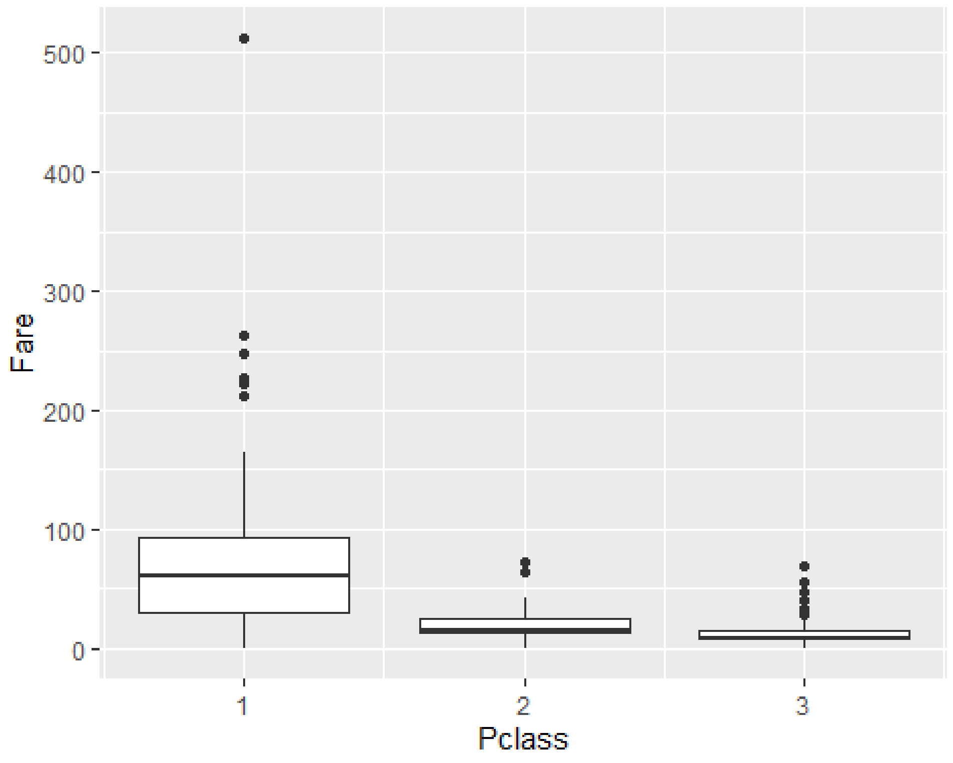


```
##
##  Variables sorted by number of missings:
##     Variable Count
##        Cabin   687
##          Age   177
##     Embarked     2
##  PassengerId     0
##     Survived     0
##       Pclass     0
##         Name     0
##          Sex     0
##        SibSp     0
##        Parch     0
##       Ticket     0
##         Fare     0
```

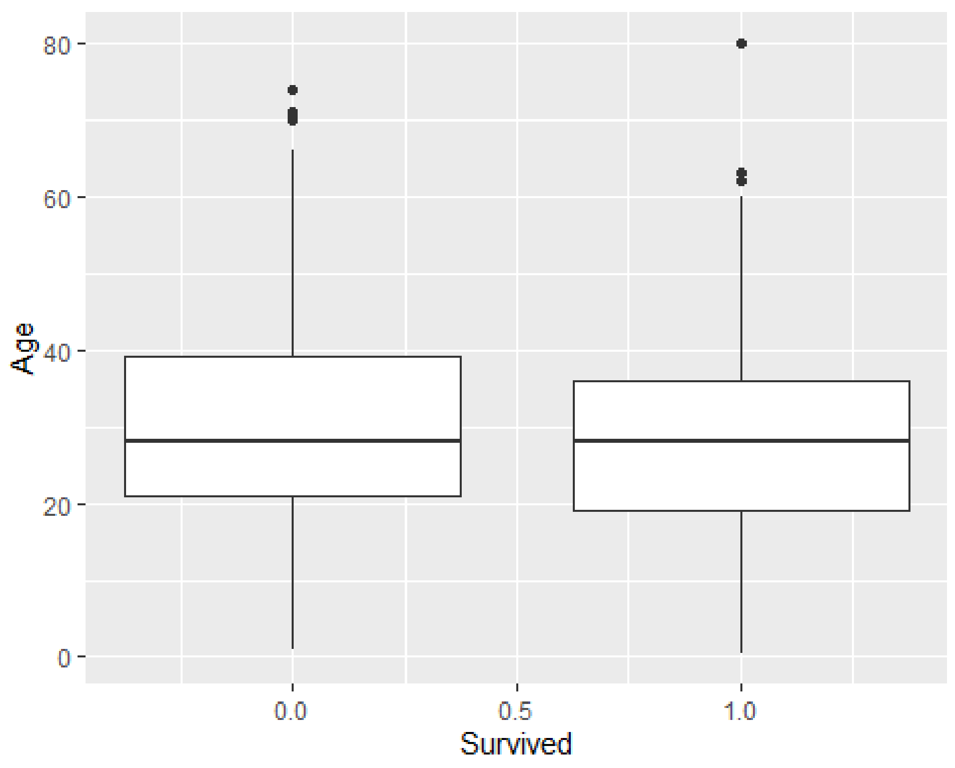
```
train_data %>%
  ggplot(aes(Age)) +
  geom_bar(stat = "count") #a lot of people under 40

## Warning: Removed 177 rows containing non-finite values (stat_count).
```



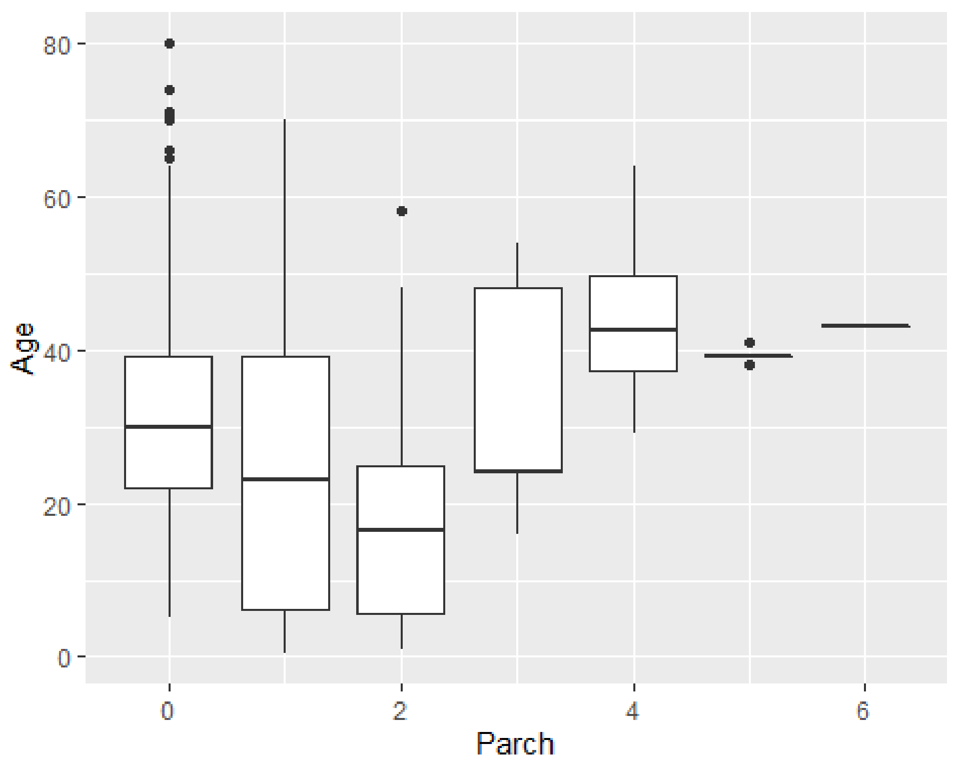
```
train_data %>%
  ggplot(aes(Pclass, Fare)) +
  geom_boxplot(aes(group = Pclass)) #first class has biggest spread
```



```
train_data %>%
  ggplot(aes(Survived, Age)) +
  geom_boxplot(aes(group = Survived))

## Warning: Removed 177 rows containing non-finite values (stat_boxplot).
```



```
train_data %>%
  ggplot(aes(Parch, Age)) +
  geom_boxplot(aes(group = Parch))

## Warning: Removed 177 rows containing non-finite values (stat_boxplot).
```



Feature Engineering

There were only two different features that I decided to create. I was looking at doing something with tickets and cabin but after looking at public workbooks decided against it. In cabin there were so many of the values missing that it would not be a very accurate model with all the imposed numbers. Ticket did not seem to improve many of the scores that I look at so decided to spend time elsewhere.

Family size, I saw the idea on a lot of the public workbooks so I decided to see if it would help my model. I categorized it into three different categories so it would be easy to run analysis and not have too many categories. The other variable that I created was the title that the person had. This varaible ended up not helping me since it added no value when prediciting but decide to leave it. There were two missing values in Embarked, so I gave them the most common embarked location since they fit the demographic.

```
table(train_data$Embarked)

## 
##   C   Q   S 
## 168  77 644

table(is.na(train_data$Embarked)) #two missing values

## 
## FALSE  TRUE 
##   889     2
```

```
table(is.na(test_data$Embarked))

##
## FALSE
##   418

#only two embarked where missing from all the data
train_data <- train_data %>% mutate(Embarked = ifelse(is.na(Embarked), "S", E
mbarked))

#family size
train_data$FamilySize <-train_data$SibSp + train_data$Parch + 1 #plus one so
everyone has a value
train_data$FamilySized[train_data$FamilySize == 1] <- 'Single'

train_data$FamilySized[train_data$FamilySize < 5 & train_data$FamilySize >= 2
] <- 'Small'
train_data$FamilySized[train_data$FamilySize >= 5] <- 'Big'
train_data$FamilySized=as.factor(train_data$FamilySized)

table(train_data$FamilySized)

##
##    Big Single  Small
##     62    537    292

test_data$FamilySize <-test_data$SibSp + test_data$Parch + 1
test_data$FamilySized[test_data$FamilySize == 1] <- 'Single'

test_data$FamilySized[test_data$FamilySize < 5 & test_data$FamilySize >= 2] <
- 'Small'
test_data$FamilySized[test_data$FamilySize >= 5] <- 'Big'
test_data$FamilySized=as.factor(test_data$FamilySized)

table(test_data$FamilySized)

##
##    Big Single  Small
##     20    253    145

##Engineer features based on title
train_data <- mutate(train_data, title_orig = factor(str_extract(Name, "[A-Z]
[a-z]*\\.")))
test_data <- mutate(test_data, title_orig = factor(str_extract(Name, "[A-Z][a
-z]*\\.")))

table(train_data$title_orig)

##
##     Capt.      Col. Countess.      Don.       Dr. Jonkheer.     Lady.    M
ajor.
```

```
##         1         2         1         1         7         1         1
2
##   Master.     Miss.     Mlle.      Mme.       Mr.      Mrs.       Ms.
Rev.
##        40       182         2         1       517       125         1
6
##      Sir.
##         1
```

```
table(test_data$title_orig)
```

```
##
##      Col.    Dona.      Dr. Master.    Miss.      Mr.     Mrs.      Ms.     Rev.
##         2        1        1      21       78      240       72        1        2
```

Dealing with Missing Data

The category that had the most missing data was Age, but I still wanted to use it as a factor. I used the same process that was covered in class since it seemed to be the best method for imposing records. The second part is preprocessing the data sets, so I don't have to do it later with each model.

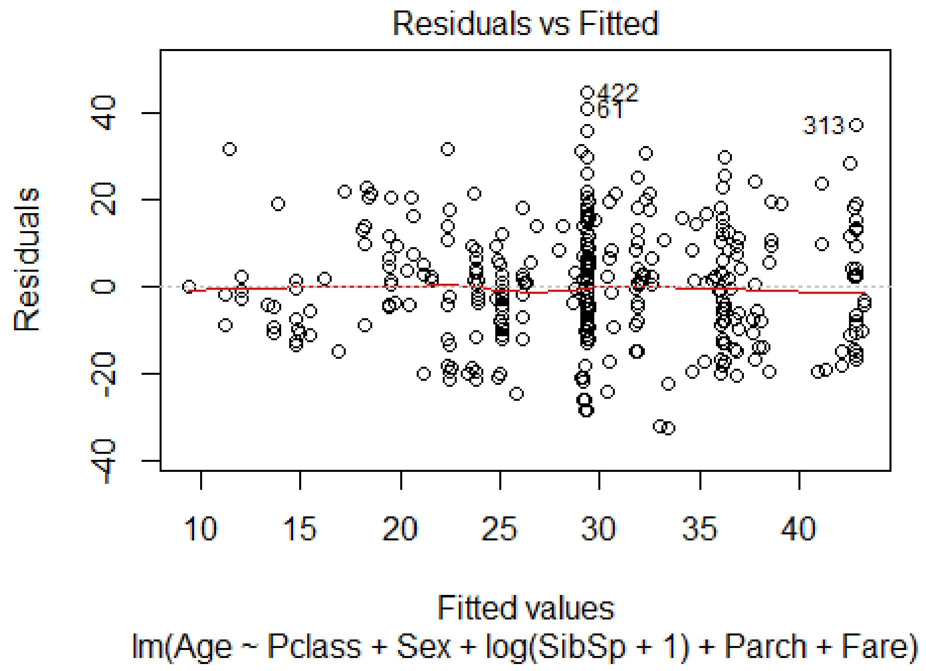
```
## Create test & train datasets ##
test_index <- createDataPartition(train_data$Survived, times=1,p=0.5, list=FA
LSE)
test_set <- train_data[test_index,]
train_set <- train_data[-test_index,]

#predict ages into misses values
#log sibsip + 1, because it has a big negative, sibsip right scewed

miss_var_summary(train_set)
```

```
## # A tibble: 15 x 3
##    variable    n_miss pct_miss
##    <chr>        <int>    <dbl>
##  1 Cabin          343     77.1
##  2 Age             74     16.6
##  3 PassengerId      0      0
##  4 Survived         0      0
##  5 Pclass           0      0
##  6 Name             0      0
##  7 Sex              0      0
##  8 SibSp            0      0
##  9 Parch            0      0
## 10 Ticket           0      0
## 11 Fare             0      0
## 12 Embarked         0      0
## 13 FamilySize       0      0
## 14 FamilySized      0      0
## 15 title_orig       0      0
```

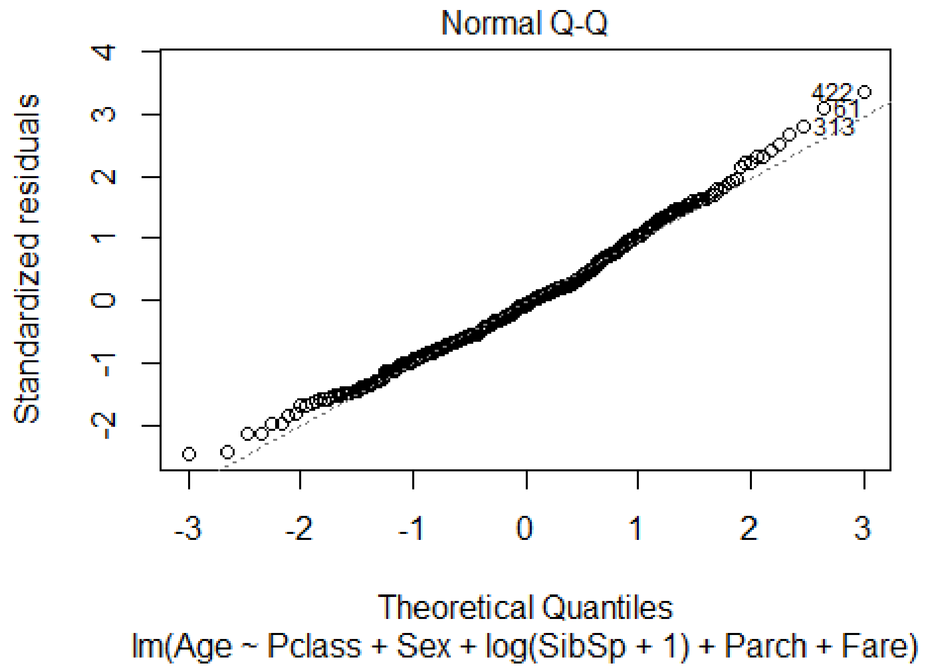
```r
linearMod <- lm(Age ~ Pclass + Sex + log(SibSp+1) + Parch + Fare, data=train_
set)
#look at the model
summary(linearMod)
```

```
##
## Call:
## lm(formula = Age ~ Pclass + Sex + log(SibSp + 1) + Parch + Fare,
##     data = train_set)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -32.607  -9.276  -0.799   8.535  44.594
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    45.79439    2.68864  17.033  < 2e-16 ***
## Pclass         -6.86452    1.00383  -6.838 3.38e-11 ***
## Sexmale         4.31549    1.52285   2.834  0.00486 **
## log(SibSp + 1) -8.00873    1.63687  -4.893 1.49e-06 ***
## Parch          -1.26685    0.86098  -1.471  0.14204
## Fare           -0.01414    0.01561  -0.906  0.36565
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.4 on 365 degrees of freedom
##   (74 observations deleted due to missingness)
## Multiple R-squared:  0.2324, Adjusted R-squared:  0.2218
## F-statistic:  22.1 on 5 and 365 DF,  p-value: < 2.2e-16
```

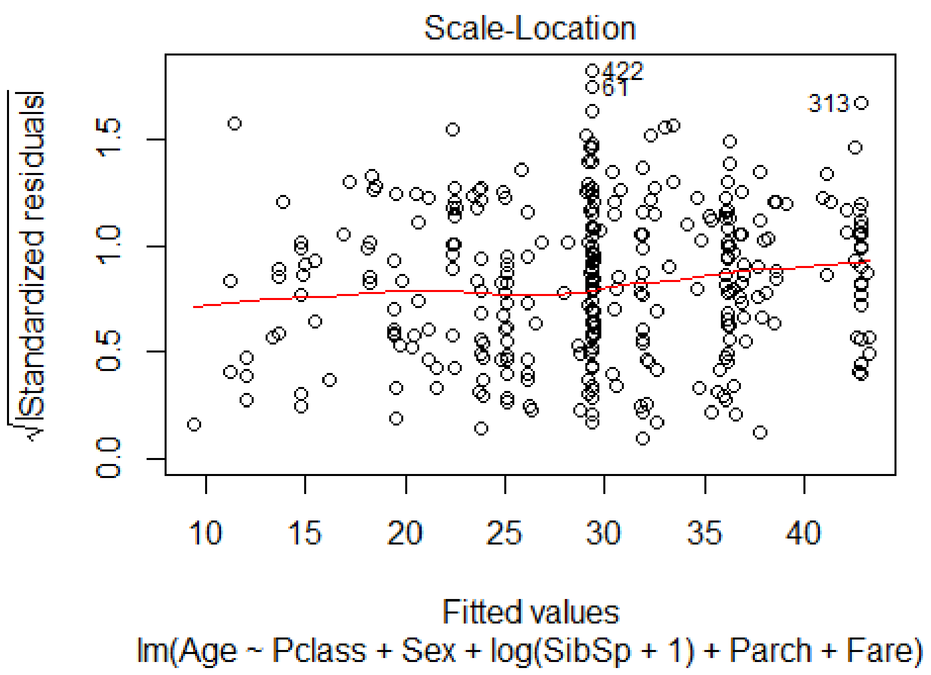
```r
#does it meet the assumptions of a linear model? How do the residuals look
plot(linearMod) # all assumptions are meet
```

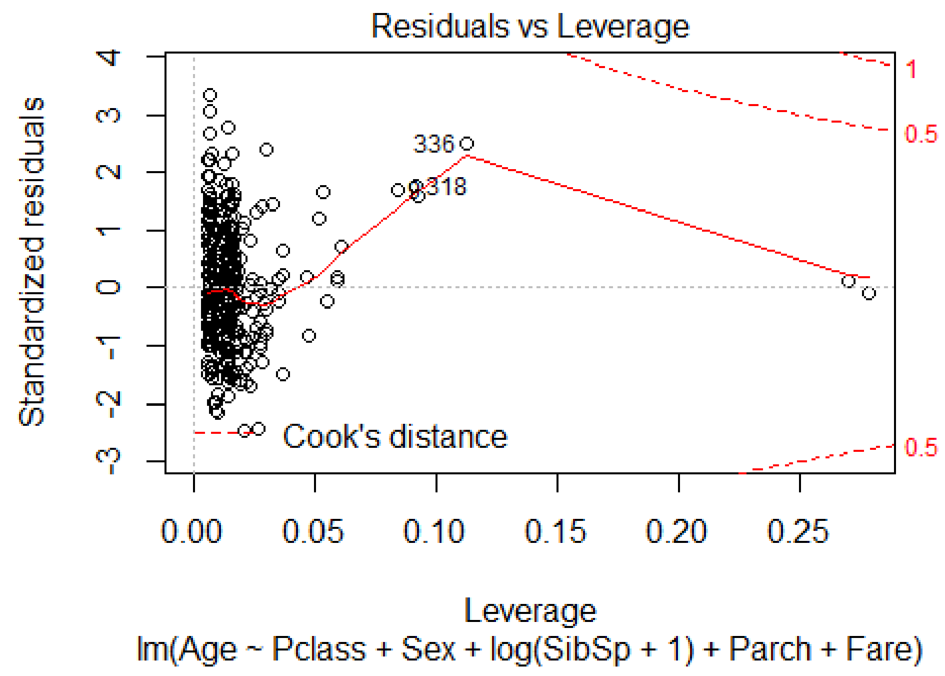


Residuals vs Fitted
Residuals
Fitted values
lm(Age ~ Pclass + Sex + log(SibSp + 1) + Parch + Fare)
422
61
313



Normal Q-Q
Standardized residuals
Theoretical Quantiles
lm(Age ~ Pclass + Sex + log(SibSp + 1) + Parch + Fare)
422
61
313





```
#check other assumptions of linear model
dwt(linearMod) #check for independent errors - want values close to 2.
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1     -0.05550828      2.106647   0.288
##  Alternative hypothesis: rho != 0
```

```
vif(linearMod) #check the variance inflation factor - values greater than 10
are problematic
```

```
##         Pclass            Sex log(SibSp + 1)          Parch           Fare
##       1.507563       1.094314       1.166366       1.239050       1.524788
```

```
#predict the age based on the model
Age_pre_train <- round(predict(linearMod, train_set))
Age_pre_test_set <- round(predict(linearMod, test_set))
Age_pre_test_data <- round(predict(linearMod, test_data))
#set our ages for all three sets
train_set <- train_set %>% mutate(Age_new = ifelse(is.na(Age), Age_pre_train,
Age))
test_set <- test_set %>% mutate(Age_new = ifelse(is.na(Age), Age_pre_test_set
, Age))
test_data <- test_data %>% mutate(Age_new = ifelse(is.na(Age), Age_pre_test_d
ata, Age))


#preprocess for the models you want to run after feature engineering
#it is important for models like knn
preProc <- preProcess(test_set[, c("Age_new", "SibSp", "Parch", "Fare")], met
hod=c("center", "scale"))
test_set <- predict(preProc, test_set)
preProc2 <- preProcess(train_set[, c("Age_new", "SibSp", "Parch", "Fare")], m
ethod=c("center", "scale"))
train_set <- predict(preProc2, train_set)
preProc3 <- preProcess(test_data[, c("Age_new", "SibSp", "Parch", "Fare")], m
ethod=c("center", "scale"))
test_data <- predict(preProc3, test_data)
```

Feature Selection

I used a few different methods for feature selection. This first method I used was looking at other public workbooks and determining what variables a lot of the higher scoring methods used. I also used trial and error to see if certain features where valuable in changing the accuracy score. The best model from the linear regression method was also helpful determining which variables were useful. The variables that I determined as important are Pclass, Age, Sex, Family Size, and Embarked.

```
#summary(bestModel1) table used for feature selection

#aggr(train_data, prop = FALSE, combined = TRUE, numbers = TRUE, sortVars = T
RUE, sortCombs = TRUE)
```

Linear Regression

Linear Regression with stepwise. I submitted this one first since I would use it as a baseline for the future models that I used.

```
stepwise1 <- glm(Survived ~ factor(Pclass) + Age_new + Sex + FamilySized + Em
barked,
                 data=train_set,
                 family = "binomial")

bestModel1 <- stepAIC(stepwise1, direction="both") #stepwise in both directio
ns

## Start:  AIC=411.16
## Survived ~ factor(Pclass) + Age_new + Sex + FamilySized + Embarked
## 
##                  Df Deviance    AIC
## - Embarked        2   393.77 407.77
## <none>                393.16 411.16
## - FamilySized     2   410.34 424.34
## - Age_new         1   422.31 438.31
## - factor(Pclass)  2   456.54 470.54
## - Sex             1   471.30 487.30
## 
## Step:  AIC=407.77
## Survived ~ factor(Pclass) + Age_new + Sex + FamilySized
## 
##                  Df Deviance    AIC
## <none>                393.77 407.77
## + Embarked        2   393.16 411.16
## - FamilySized     2   412.56 422.56
## - Age_new         1   423.01 435.01
## - factor(Pclass)  2   460.53 470.53
## - Sex             1   476.71 488.71

summary(bestModel1)

## 
## Call:
## glm(formula = Survived ~ factor(Pclass) + Age_new + Sex + FamilySized, 
##     family = "binomial", data = train_set)
## 
## Deviance Residuals: 
##     Min       1Q   Median       3Q      Max  
## -2.4784  -0.6378  -0.3630   0.6154   2.4951  
## 
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)    
## (Intercept)         0.4172     0.6339   0.658 0.510500    
## factor(Pclass)2    -1.1462     0.3755  -3.052 0.002272 ** 
## factor(Pclass)3    -2.6664     0.3680  -7.246 4.28e-13 ***
## Age_new            -0.8286     0.1662  -4.984 6.22e-07 ***
## Sexmale            -2.3484     0.2805  -8.371  < 2e-16 ***
## FamilySizedSingle   2.4477     0.6539   3.743 0.000182 ***
## FamilySizedSmall    2.3515     0.6406   3.671 0.000242 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 598.16  on 444  degrees of freedom
## Residual deviance: 393.77  on 438  degrees of freedom
## AIC: 407.77
##
## Number of Fisher Scoring iterations: 5
```

```
survived_hat <- predict(bestModel1, test_set, type="response")
survived_pred <- factor(ifelse(survived_hat >0.5, 1, 0))
confusionMatrix(survived_pred, factor(test_set$Survived)) #82.06%
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 243  42
##          1  38 123
##
##                Accuracy : 0.8206
##                  95% CI : (0.7818, 0.8551)
##     No Information Rate : 0.63
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.6133
##
##  Mcnemar's Test P-Value : 0.7373
##
##             Sensitivity : 0.8648
##             Specificity : 0.7455
##          Pos Pred Value : 0.8526
##          Neg Pred Value : 0.7640
##              Prevalence : 0.6300
##          Detection Rate : 0.5448
##    Detection Prevalence : 0.6390
##       Balanced Accuracy : 0.8051
##
##        'Positive' Class : 0
##
```

Support Vector Machine

I saw a few people that used SVM on Kaggle so decided to try it out for myself. It was my first time using this model but was pretty straight forward and worked pretty well.

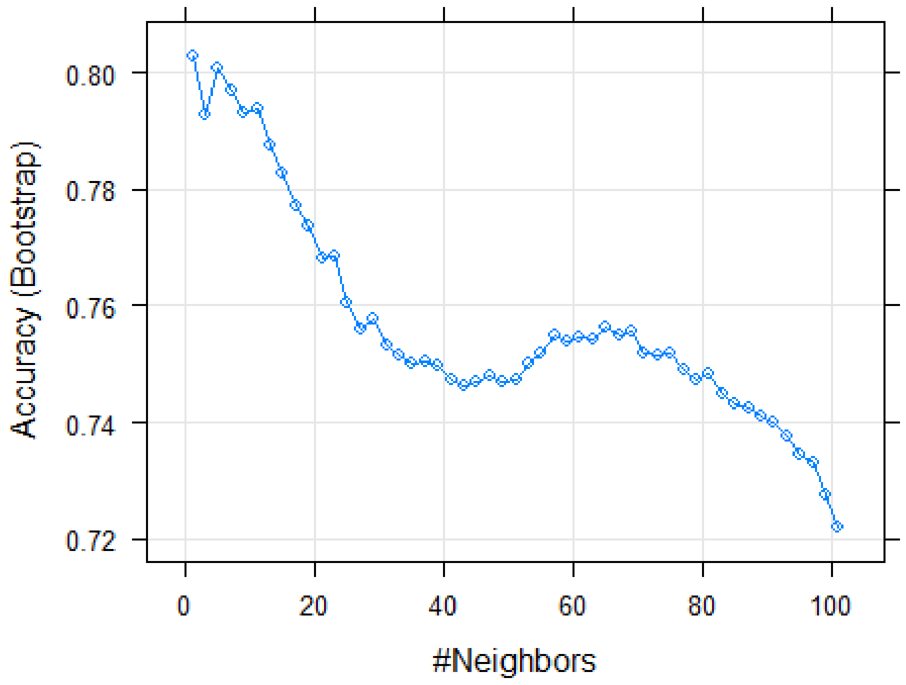
```
caret_svm <- train(factor(Survived) ~ factor(Pclass) + Age_new + Sex + Family
Sized + Embarked,
                   data=train_set, method='svmRadial',
                   trControl=trainControl(method="cv", number=5))
caret_svm

## Support Vector Machines with Radial Basis Function Kernel
##
## 445 samples
##   5 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 355, 357, 356, 356, 356
## Resampling results across tuning parameters:
##
##   C     Accuracy   Kappa
##   0.25  0.7842555  0.5246954
##   0.50  0.7954670  0.5528177
##   1.00  0.7953910  0.5553140
##
## Tuning parameter 'sigma' was held constant at a value of 0.1672242
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.1672242 and C = 0.5.

solution_svm <- predict(caret_svm, test_set, type = "raw")
#survived_svm <- factor(ifelse(solution_svm > 0.5, 1, 0))
confusionMatrix(solution_svm, factor(test_set$Survived)) #82.96%

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 261  56
##          1  20 109
##
##                Accuracy : 0.8296
##                  95% CI : (0.7914, 0.8633)
##     No Information Rate : 0.63
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6172
##
##  Mcnemar's Test P-Value : 5.95e-05
```

```
##
##             Sensitivity : 0.9288
##             Specificity : 0.6606
##          Pos Pred Value : 0.8233
##          Neg Pred Value : 0.8450
##              Prevalence : 0.6300
##          Detection Rate : 0.5852
##    Detection Prevalence : 0.7108
##       Balanced Accuracy : 0.7947
##
##        'Positive' Class : 0
##
```

Knn - K Nearest Neighbors

I used Knn because it is a model that I am fairly familiar with since we used it a lot last semester. It performed one of the worsts but that could have been improved with better tuning of the parameters.

```
train_knn <- train(factor(Survived) ~ Age_new + Sex + factor(Pclass) + Family
Sized,
                   method = "knn",
                   data = train_set,
                   tuneGrid = data.frame(k=seq(1,101,2)))

plot(train_knn)
```



```
train_knn$bestTune #13 neighbors

##   k
## 1 1

y_hat_knn_prob <- predict(train_knn, test_set, type = "prob")
y_hat_knn <- predict(train_knn, test_set)
confusionMatrix(y_hat_knn, factor(test_set$Survived)) #78.7%

## Confusion Matrix and Statistics
## 
##           Reference
## Prediction   0   1
##          0 244  58
##          1  37 107
##                                           
##                Accuracy : 0.787           
##                  95% CI : (0.746, 0.8241)
##     No Information Rate : 0.63            
##     P-Value [Acc > NIR] : 6.271e-13       
##                                           
##                   Kappa : 0.5308          
##                                           
##  Mcnemar's Test P-Value : 0.04017         
##                                           
##             Sensitivity : 0.8683          
##             Specificity : 0.6485          
##          Pos Pred Value : 0.8079          
##          Neg Pred Value : 0.7431          
##              Prevalence : 0.6300          
##          Detection Rate : 0.5471          
##    Detection Prevalence : 0.6771          
##       Balanced Accuracy : 0.7584          
##                                           
##        'Positive' Class : 0               
## 
```

Random Forest

I actually got one of the best predictions from random forest which is to be expect since it's such a robust model. This model tied with FDA for second best performing model when I submitted to Kaggle.

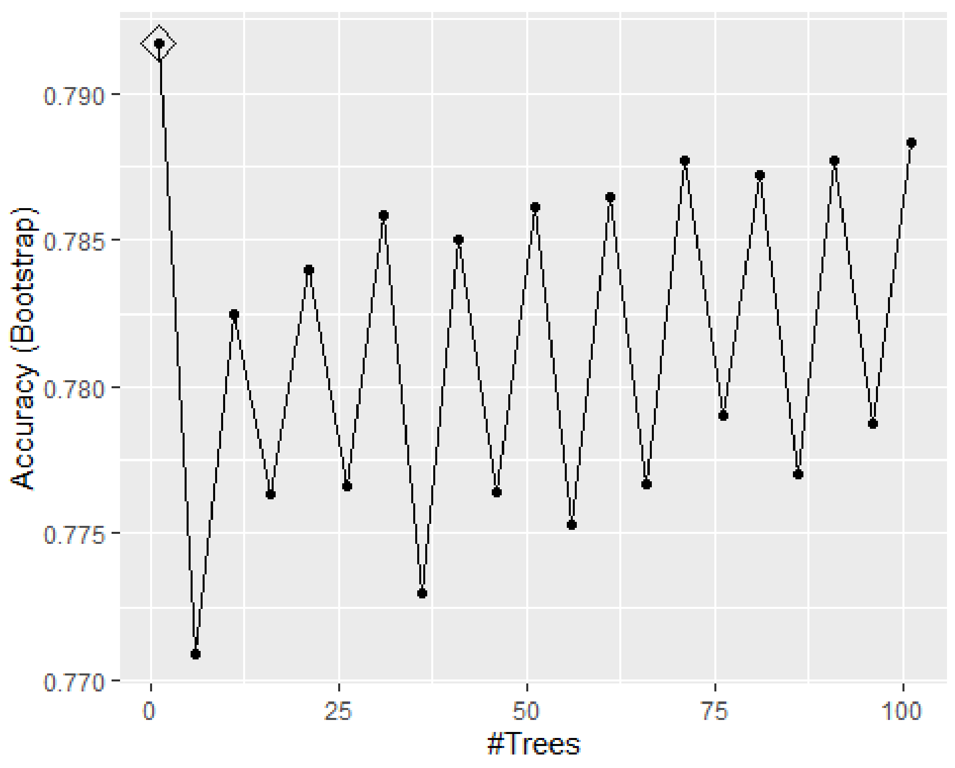
```
train_rf <- train(factor(Survived) ~  factor(Pclass) + Age_new + Sex + Family
Sized + Embarked,
                  method = "rf",
                  data=train_set,
                  tuneGrid = data.frame(mtry=seq(1,20,2)))

y_hat_rf_prob <-predict(train_rf, test_set, type="prob")
```

```
y_hat_rf <- predict(train_rf, test_set)
confusionMatrix(y_hat_rf, factor(test_set$Survived)) #81.84

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 255  55
##          1  26 110
##
##                Accuracy : 0.8184
##                  95% CI : (0.7794, 0.8531)
##     No Information Rate : 0.63
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5958
##
##  Mcnemar's Test P-Value : 0.001864
##
##             Sensitivity : 0.9075
##             Specificity : 0.6667
##          Pos Pred Value : 0.8226
##          Neg Pred Value : 0.8088
##              Prevalence : 0.6300
##          Detection Rate : 0.5717
##    Detection Prevalence : 0.6951
##       Balanced Accuracy : 0.7871
##
##        'Positive' Class : 0
##
```

AdaBoost

Had a lot of hope for this model and was the last model for my submission which actually gave me my best score of 80%. It took forever to run so with more time I would like to go back through and try tuning it even better.

```
train_ada <- train(factor(Survived) ~ factor(Pclass) + Age_new + Sex + Family
Sized + Embarked,
                   data = train_set,
                   method = "adaboost",
                   tuneGrid = data.frame(nIter = seq(1,101,5), method = "adab
oost"))

ggplot(train_ada, highlight = TRUE)
```



```
y_hat_ada_prob <- predict(train_ada, test_set, type = "prob")
y_hat_ada <- predict(train_ada, test_set, type = "raw")
confusionMatrix(y_hat_ada, factor(test_set$Survived)) #80.27

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 258  65
##          1  23 100
##
##                Accuracy : 0.8027
##                  95% CI : (0.7627, 0.8386)
##     No Information Rate : 0.63
##     P-Value [Acc > NIR] : 1.899e-15
##
##                   Kappa : 0.5533
##
##  Mcnemar's Test P-Value : 1.239e-05
##
##             Sensitivity : 0.9181
##             Specificity : 0.6061
##          Pos Pred Value : 0.7988
##          Neg Pred Value : 0.8130
##              Prevalence : 0.6300
##          Detection Rate : 0.5785
##    Detection Prevalence : 0.7242
##       Balanced Accuracy : 0.7621
##
##        'Positive' Class : 0
##
```

FDA

Decided to try Flexible Discriminant Analysis since it was the model that I gave the presentation on. It did not work that well with all the different combinations that I used for tuning it. There are two tuning parameters (degree and nprune) but got worse results when tuning so left it at default.

```
train_fda2 <- fda(factor(Survived) ~ Age_new + Sex + factor(Pclass) + FamilyS
ized + Embarked,
                  method = mars,
                  degree = 2,
                  data = train_set)

y_hat_fda2 <- predict(train_fda2, test_set)
confusionMatrix(y_hat_fda2, factor(test_set$Survived)) #84.08%
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 264  54
##          1  17 111
##
##                Accuracy : 0.8408
##                  95% CI : (0.8035, 0.8735)
##     No Information Rate : 0.63
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6419
##
##  Mcnemar's Test P-Value : 1.934e-05
##
##             Sensitivity : 0.9395
##             Specificity : 0.6727
##          Pos Pred Value : 0.8302
##          Neg Pred Value : 0.8672
##              Prevalence : 0.6300
##          Detection Rate : 0.5919
##    Detection Prevalence : 0.7130
##       Balanced Accuracy : 0.8061
##
##        'Positive' Class : 0
##
```

Prediction

I did five submissions which all gave me scores from 77% - 80%. The best scoring one was AdaBoost with the score of 80.382%. I did try using ensemble models, but they actually gave me a worse score so decided to focus my work in another direction. I do want to come back sometime over Spring Break and see if I can get an ensemble model to predict in the 80% range. I spent a lot of time getting the models to run and tried to tune them, but there is still a lot of room for improvement in that category.

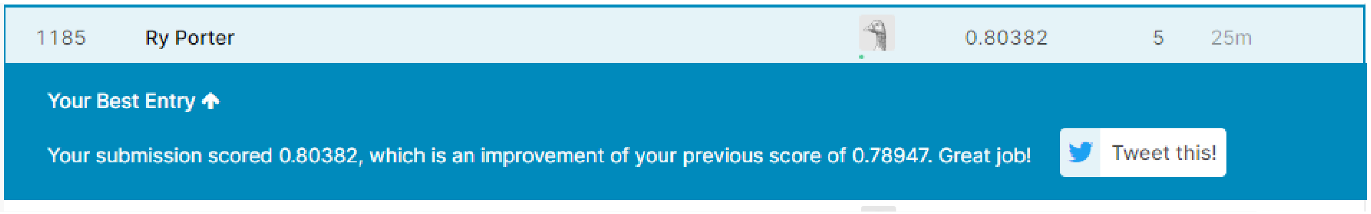
```
survive_test <- predict(bestModel1, test_data, type = "response") #0 to 1 pre
diction of surival
prediction <- factor(ifelse(survive_test > 0.5, 1, 0))
submission <- test_data %>% dplyr::select(PassengerId) %>% mutate(Survived =
prediction)
head(submission)
```

```
## # A tibble: 6 x 2
##   PassengerId Survived
##         <dbl> <fct>
## 1         892 0
## 2         893 0
## 3         894 0
## 4         895 0
## 5         896 1
## 6         897 0
```

```
write.csv(submission, "submission.csv", row.names=FALSE)

survive_test_svm <- predict(caret_svm, test_data, type = "raw") #0 to 1 predi
ction of surival
submission_svm <- test_data %>% dplyr::select(PassengerId) %>% mutate(Survive
d = survive_test_svm)
head(submission_svm)
```

```
## # A tibble: 6 x 2
##   PassengerId Survived
##         <dbl> <fct>
## 1         892 0
## 2         893 0
## 3         894 0
## 4         895 0
## 5         896 0
## 6         897 0
```

```
write.csv(submission_svm, "submission_svm.csv", row.names=FALSE)

survive_test_rf <- predict(train_rf, test_data, type = "raw") #0 to 1 predict
ion of surival
submission_rf <- test_data %>% dplyr::select(PassengerId) %>% mutate(Survived
= survive_test_rf)
head(submission_rf)

## # A tibble: 6 x 2
##   PassengerId Survived
##         <dbl> <fct>
## 1         892 0
## 2         893 0
## 3         894 0
## 4         895 0
## 5         896 0
## 6         897 0

write.csv(submission_rf, "submission_rf.csv", row.names=FALSE)

survive_test_fda <- predict(train_fda2, test_data) #0 to 1 prediction of suri
val
submission_fda <- test_data %>% dplyr::select(PassengerId) %>% mutate(Survive
d = survive_test_fda)
head(submission_rf)

## # A tibble: 6 x 2
##   PassengerId Survived
##         <dbl> <fct>
## 1         892 0
## 2         893 0
## 3         894 0
## 4         895 0
## 5         896 0
## 6         897 0

write.csv(submission_fda, "submission_fda.csv", row.names=FALSE)

survive_test_ada <- predict(train_ada, test_data, type = "raw") #0 to 1 predi
ction of surival
submission_ada <- test_data %>% dplyr::select(PassengerId) %>% mutate(Survive
d = survive_test_ada)
head(submission_rf)

## # A tibble: 6 x 2
##   PassengerId Survived
##         <dbl> <fct>
## 1         892 0
## 2         893 0
## 3         894 0
## 4         895 0
## 5         896 0
## 6         897 0

write.csv(submission_ada, "submission_ada.csv", row.names=FALSE)
```



1185 Ry Porter 0.80382 5 25m

Your Best Entry

Your submission scored 0.80382, which is an improvement of your previous score of 0.78947. Great job!