Kaggle Titanic Competition

Matthew Sahagun

March 06, 2021

Introduction

The Kaggle Titanic competition asks you to predict whether a passenger will survive or not based on a variety of explanatory variables, including sex, fare, etc. For the competition, I will practice using a variety of machine learning algorithms and then submit my result to Kaggle. Specifically, I plan on using

0. Null Model
1. kNN
2. Boosted C5.0
3. Random Forest
4. Logistic Regression using regularization

Data

```
library(pacman)
p_load(titanic, tidyverse, janitor, naniar, DataExplorer, tidymodels, tidyr)
```

```
data(titanic_train)
data(titanic_test)

glimpse(titanic_train)
```

```
## Rows: 891
## Columns: 12
## $ PassengerId <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, ...
## $ Survived    <int> 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0...
## $ Pclass      <int> 3, 1, 3, 1, 3, 3, 1, 3, 3, 2, 3, 1, 3, 3, 3, 2, 3, 2, 3...
## $ Name        <chr> "Braund, Mr. Owen Harris", "Cumings, Mrs. John Bradley ...
## $ Sex         <chr> "male", "female", "female", "female", "male", "male", "...
## $ Age         <dbl> 22, 38, 26, 35, 35, NA, 54, 2, 27, 14, 4, 58, 20, 39, 1...
## $ SibSp       <int> 1, 1, 0, 1, 0, 0, 0, 3, 0, 1, 1, 0, 0, 1, 0, 0, 4, 0, 1...
## $ Parch       <int> 0, 0, 0, 0, 0, 0, 0, 1, 2, 0, 1, 0, 0, 5, 0, 0, 1, 0, 0...
## $ Ticket      <chr> "A/5 21171", "PC 17599", "STON/O2. 3101282", "113803", ...
## $ Fare        <dbl> 7.2500, 71.2833, 7.9250, 53.1000, 8.0500, 8.4583, 51.86...
## $ Cabin       <chr> "", "C85", "", "C123", "", "", "E46", "", "", "", "G6",...
## $ Embarked    <chr> "S", "C", "S", "S", "S", "Q", "S", "S", "S", "C", "S", ...
```

```r
glimpse(titanic_test)
```

```
## Rows: 418
## Columns: 11
## $ PassengerId <int> 892, 893, 894, 895, 896, 897, 898, 899, 900, 901, 902, ...
## $ Pclass      <int> 3, 3, 2, 3, 3, 3, 3, 2, 3, 3, 3, 1, 1, 2, 1, 2, 2, 3, 3...
## $ Name        <chr> "Kelly, Mr. James", "Wilkes, Mrs. James (Ellen Needs)",...
## $ Sex         <chr> "male", "female", "male", "male", "female", "male", "fe...
## $ Age         <dbl> 34.5, 47.0, 62.0, 27.0, 22.0, 14.0, 30.0, 26.0, 18.0, 2...
## $ SibSp       <int> 0, 1, 0, 0, 1, 0, 0, 1, 0, 2, 0, 0, 1, 1, 1, 1, 0, 0, 1...
## $ Parch       <int> 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ Ticket      <chr> "330911", "363272", "240276", "315154", "3101298", "753...
## $ Fare        <dbl> 7.8292, 7.0000, 9.6875, 8.6625, 12.2875, 9.2250, 7.6292...
## $ Cabin       <chr> "", "", "", "", "", "", "", "", "", "", "", "", "B45", ...
## $ Embarked    <chr> "Q", "S", "Q", "S", "S", "S", "Q", "S", "C", "S", "S", ...
```

First, I will filter the columns. I will remove PassengerId, Name, and Ticket as these are ID variable. I will also remove Cabin and Embarked as I don't believe that these will be useful. I will also change sex to a numeric (1 for female, 0 for male) to allow for KNN.

```r
titanic_test_2 = titanic_test %>%
   mutate(
    Pclass = as_factor(Pclass)
  ) %>%
  mutate(Sex_num = ifelse(Sex == "male", 0, 1)) %>%
  select(-c(PassengerId, Name, Ticket, Cabin, Embarked, Sex))

colnames(titanic_test_2)
```

```
## [1] "Pclass"  "Age"     "SibSp"   "Parch"   "Fare"    "Sex_num"
```

```r
titanic_train_2 = titanic_train %>%
  mutate(
    Survived = as_factor(Survived),
    Pclass = as_factor(Pclass)
  ) %>%
  mutate(Sex_num = ifelse(Sex == "male", 0, 1)) %>%
  select(-c(PassengerId, Name, Ticket, Cabin, Embarked, Sex))

colnames(titanic_train_2)
```

```
## [1] "Survived" "Pclass"   "Age"      "SibSp"    "Parch"    "Fare"     "Sex_num"
```

I now want to take a look at na values. It would be helpful to know the percent of na's from each column.

```r
round(colMeans(is.na(titanic_test_2)), 4)
```

```
##  Pclass     Age   SibSp   Parch    Fare Sex_num 
##  0.0000  0.2057  0.0000  0.0000  0.0024  0.0000
```

```
round(colMeans(is.na(titanic_train_2)), 4)
```

```
## Survived   Pclass      Age    SibSp    Parch     Fare  Sex_num
##   0.0000   0.0000   0.1987   0.0000   0.0000   0.0000   0.0000
```

It looks like we have a single for fare, and many na's for age. I will imputate those values with the median for each column.

```
med_age = median(titanic_train_2$Age, na.rm = TRUE)
med_fare = median(titanic_train_2$Fare, na.rm = TRUE)

titanic_train_imp = titanic_train_2 %>%
  group_by(Age) %>%
  mutate(Age = replace_na(Age, med_age)) %>%
  mutate(Fare = replace_na(Fare, med_fare))

round(colMeans(is.na(titanic_train_imp)), 4)
```

```
## Survived   Pclass      Age    SibSp    Parch     Fare  Sex_num
##        0        0        0        0        0        0        0
```

```
titanic_test_imp = titanic_test_2 %>%
  group_by(Age) %>%
  mutate(Age = replace_na(Age, med_age))  %>%
  mutate(Fare = replace_na(Fare, med_fare))

round(colMeans(is.na(titanic_test_imp)), 4)
```

```
##  Pclass     Age   SibSp   Parch    Fare Sex_num
##       0       0       0       0       0       0
```

Now, I will take the training data, and split it between test and training data (20-80 split).

```
set.seed(42)

dat_parts <- titanic_train_imp %>%
  initial_split(prop = 0.8)

train <- dat_parts %>%
  training()

test <- dat_parts %>%
  testing()
```

Null Model

To build the null model, I will need to see what percentage of people survived from my training data set.

```r
train %>%
  select(Survived) %>%
  group_by(Survived) %>%
  summarise(n=n(), ratio = n/nrow(train))
```

```
## Adding missing grouping variables: `Age`

## `summarise()` ungrouping output (override with `.groups` argument)

## # A tibble: 2 x 3
##   Survived     n ratio
##   <fct>    <int> <dbl>
## 1 0          440 0.617
## 2 1          273 0.383
```

It looks like most people (61.6%) from the training set died on the Titanic.

I will now test the null model on the test dataset, and that will be my baseline for every other model's result.

```r
test %>%
  select(Survived) %>%
  group_by(Survived) %>%
  summarise(n=n(), ratio = n/nrow(test))
```

```
## Adding missing grouping variables: `Age`

## `summarise()` ungrouping output (override with `.groups` argument)

## # A tibble: 2 x 3
##   Survived     n ratio
##   <fct>    <int> <dbl>
## 1 0          109 0.612
## 2 1           69 0.388
```

It looks like the null model correctly identifies 62.8% of the people in the test data set.

Logistic Regression Model

I will now create a logistic regression model.

```r
model <- glm(Survived ~.,family=binomial(link='logit'),data=train)

summary(model)
```

```
##
## Call:
## glm(formula = Survived ~ ., family = binomial(link = "logit"),
##     data = train)
##
## Deviance Residuals:
```

```
##     Min       1Q   Median       3Q      Max
## -2.4025  -0.6299  -0.4106   0.6046   2.4467
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.103648   0.440896   2.503  0.01231 *
## Pclass2     -1.083903   0.328263  -3.302  0.00096 ***
## Pclass3     -2.240127   0.327633  -6.837 8.07e-12 ***
## Age         -0.038848   0.008634  -4.499 6.82e-06 ***
## SibSp       -0.396484   0.125575  -3.157  0.00159 **
## Parch       -0.055780   0.127568  -0.437  0.66193
## Fare         0.003246   0.002627   1.236  0.21657
## Sex_num      2.712258   0.224200  12.098  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 948.95  on 712  degrees of freedom
## Residual deviance: 632.27  on 705  degrees of freedom
## AIC: 648.27
##
## Number of Fisher Scoring iterations: 5
```

I will now rerun the model keeping only the statistically significant variables.

```r
model2 <- glm(Survived ~ Pclass + Age + SibSp + Sex_num, family=binomial(link='logit'),data=train)

summary(model2)
```

```
##
## Call:
## glm(formula = Survived ~ Pclass + Age + SibSp + Sex_num, family = binomial(link = "logit"),
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3504  -0.6126  -0.4078   0.6044   2.4655
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.361018   0.391034   3.481  0.00050 ***
## Pclass2     -1.278323   0.291524  -4.385 1.16e-05 ***
## Pclass3     -2.470783   0.274069  -9.015  < 2e-16 ***
## Age         -0.039541   0.008605  -4.595 4.33e-06 ***
## SibSp       -0.386656   0.118367  -3.267  0.00109 **
## Sex_num      2.711013   0.218604  12.402  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 948.95  on 712  degrees of freedom
```

```
## Residual deviance: 634.01  on 707  degrees of freedom
## AIC: 646.01
##
## Number of Fisher Scoring iterations: 5
```

I will now test to see how the logistic model runs using the test data.

```r
probs_test = predict(model2, newdata = test, type = "response")
length(probs_test)
```

```
## [1] 178
```

```r
preds_test = rep(0, 178)
preds_test[probs_test > 0.5] = 1
head(probs_test)
```

```
##          1          2          3          4          5          6
## 0.63949916 0.04571626 0.64998459 0.56313118 0.53119825 0.62107409
```

```r
head(preds_test)
```

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

Now I will make the confusion matrix

```r
tb = table(prediction = preds_test,
           actual = test$Survived)
addmargins(tb)
```

```
##           actual
## prediction   0   1 Sum
##        0    91  17 108
##        1    18  52  70
##        Sum 109  69 178
```

```r
141/178
```

```
## [1] 0.7921348
```

The logistic model performed better than the null model. It correctly identified 79.2% of the passengers.

KNN Model

I will create a KNN model. For KNN, all predictors need to be numeric.

```
titanic_train_imp_knn = titanic_train_imp %>%
  mutate(Pclass = as.numeric(Pclass))

set.seed(42)

dat_parts_knn <- titanic_train_imp_knn %>%
  initial_split(prop = 0.8)

train_knn <- dat_parts_knn %>%
  training()

test_knn <- dat_parts_knn %>%
  testing()
```

```
tit_rec <-
  recipe(Survived ~ ., data = train_knn) %>%
  step_normalize(all_predictors()) %>%
  prep()

tit_rec
```

```
## Data Recipe
##
## Inputs:
##
##       role #variables
##    outcome          1
##  predictor          6
##
## Training data contained 713 data points and no missing data.
##
## Operations:
##
## Centering and scaling for Pclass, Age, SibSp, Parch, Fare, Sex_num [trained]
```

```
summary(tit_rec)
```

```
## # A tibble: 7 x 4
##   variable type    role      source
##   <chr>    <chr>   <chr>     <chr>
## 1 Pclass   numeric predictor original
## 2 Age      numeric predictor original
## 3 SibSp    numeric predictor original
## 4 Parch    numeric predictor original
## 5 Fare     numeric predictor original
## 6 Sex_num  numeric predictor original
## 7 Survived nominal outcome   original
```

tune function allows you to not have to specify the number of neighbors. tune sets up a tuning grid

```r
tune_spec <-
  nearest_neighbor(neighbors = tune()) %>%
  set_engine("kknn") %>%
  set_mode("classification")
```

```r
tune_grid <- seq(5, 23, by = 2)
tune_grid
```

```
##  [1]  5  7  9 11 13 15 17 19 21 23
```

```r
tit_wflow <-
  workflow() %>%
  add_recipe(tit_rec) %>%
  add_model(tune_spec)

tit_wflow
```

```
## == Workflow ====================================================================
## Preprocessor: Recipe
## Model: nearest_neighbor()
##
## -- Preprocessor ----------------------------------------------------------------
## 1 Recipe Step
##
## * step_normalize()
##
## -- Model -----------------------------------------------------------------------
## K-Nearest Neighbor Model Specification (classification)
##
## Main Arguments:
##   neighbors = tune()
##
## Computational engine: kknn
```

Using Cross Validation, 5 folds

```r
folds <- vfold_cv(train_knn, v = 5)
folds
```

```
## #  5-fold cross-validation
## # A tibble: 5 x 2
##   splits            id
##   <list>            <chr>
## 1 <split [570/143]> Fold1
## 2 <split [570/143]> Fold2
## 3 <split [570/143]> Fold3
## 4 <split [571/142]> Fold4
## 5 <split [571/142]> Fold5
```

```r
tit_fit_rs <-
  tit_wflow %>%
  tune_grid(
    resamples = folds,
    grid = tune_grid
    )
```

```r
collect_metrics(tit_fit_rs)
```

```
## # A tibble: 8 x 7
##   neighbors .metric  .estimator  mean     n std_err .config
##       <int> <chr>    <chr>      <dbl> <int>   <dbl> <fct>
## 1         4 accuracy binary     0.763     5  0.0165 Preprocessor1_Model1
## 2         4 roc_auc  binary     0.827     5  0.0265 Preprocessor1_Model1
## 3         8 accuracy binary     0.799     5  0.0190 Preprocessor1_Model2
## 4         8 roc_auc  binary     0.839     5  0.0234 Preprocessor1_Model2
## 5        12 accuracy binary     0.799     5  0.0189 Preprocessor1_Model3
## 6        12 roc_auc  binary     0.843     5  0.0231 Preprocessor1_Model3
## 7        14 accuracy binary     0.801     5  0.0198 Preprocessor1_Model4
## 8        14 roc_auc  binary     0.844     5  0.0225 Preprocessor1_Model4
```

Showing which model had the best accuracy

```r
tit_fit_rs %>%
  show_best("accuracy")
```

```
## # A tibble: 4 x 7
##   neighbors .metric  .estimator  mean     n std_err .config
##       <int> <chr>    <chr>      <dbl> <int>   <dbl> <fct>
## 1        14 accuracy binary     0.801     5  0.0198 Preprocessor1_Model4
## 2        12 accuracy binary     0.799     5  0.0189 Preprocessor1_Model3
## 3         8 accuracy binary     0.799     5  0.0190 Preprocessor1_Model2
## 4         4 accuracy binary     0.763     5  0.0165 Preprocessor1_Model1
```

```r
best_knn <- tit_fit_rs %>%
  select_best("accuracy")

best_knn
```

```
## # A tibble: 1 x 2
##   neighbors .config
##       <int> <fct>
## 1        14 Preprocessor1_Model4
```

```r
final_wflow <-
  tit_wflow %>%
  finalize_workflow(best_knn)
```

```r
final_knn <-
  final_wflow %>%
  last_fit(dat_parts_knn)
```

```
final_knn %>%
    collect_metrics()
```

```
## # A tibble: 2 x 4
##   .metric  .estimator .estimate .config
##   <chr>    <chr>          <dbl> <fct>
## 1 accuracy binary         0.826 Preprocessor1_Model1
## 2 roc_auc  binary         0.871 Preprocessor1_Model1
```

The KNN model correctly identified the correct response 82.6% of the time, performing much better than either the logistic or null models.

Decision Tree

I will now run a C5.0 model.

Build the simplest decision tree

```
library(C50)
tit5_model <- C5.0(Survived ~ ., data = train, trials = 1)
```

Display simple facts about the tree

```
tit5_model
```

```
##
## Call:
## C5.0.formula(formula = Survived ~ ., data = train, trials = 1)
##
## Classification Tree
## Number of samples: 713
## Number of predictors: 6
##
## Tree size: 6
##
## Non-standard options: attempt to group attributes
```

Display detailed information about the tree

```
summary(tit5_model)
```

```
##
## Call:
## C5.0.formula(formula = Survived ~ ., data = train, trials = 1)
##
##
## C5.0 [Release 2.07 GPL Edition]      Sat Mar 06 21:10:44 2021
## -------------------------------
##
## Class specified by attribute `outcome'
##
## Read 713 cases (7 attributes) from undefined.data
##
## Decision tree:
##
## Sex_num <= 0:
## :...Age > 13: 0 (433/72)
## :   Age <= 13:
## :   :...SibSp <= 2: 1 (17)
## :       SibSp > 2: 0 (12)
## Sex_num > 0:
## :...Pclass in {1,2}: 1 (135/6)
##     Pclass = 3:
##     :...Fare <= 23.25: 1 (92/40)
##         Fare > 23.25: 0 (24/3)
```

```
##
##
## Evaluation on training data (713 cases):
##
##      Decision Tree
##    ----------------
##    Size      Errors
##
##       6  121(17.0%)   <<
##
##
##     (a)   (b)    <-classified as
##    ----  ----
##     394    46    (a): class 0
##      75   198    (b): class 1
##
##
##  Attribute usage:
##
##  100.00% Sex_num
##   64.80% Age
##   35.20% Pclass
##   16.27% Fare
##    4.07% SibSp
##
##
## Time: 0.0 secs
```

```
#this also provides error for training data. IN this case, it is 18%
```

I will now evaluate the performance of the C5.0 model.

```
tit5_pred <- predict(tit5_model, test, type="class")
sum(tit5_pred == test$Survived ) / length(tit5_pred)
```

```
## [1] 0.8370787
```

Cross tabulation of predicted versus actual classes

```
library(gmodels)
```

```
## Warning: package 'gmodels' was built under R version 3.6.3
```

```
CrossTable(test$Survived, tit5_pred,
           prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
           dnn = c('actual', 'predicted'))
```

```
##
##
##    Cell Contents
## |-------------------------|
```

```
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  178
##
##
##              | predicted
##       actual |         0 |         1 | Row Total |
## -------------|-----------|-----------|-----------|
##            0 |        97 |        12 |       109 |
##              |     0.545 |     0.067 |           |
## -------------|-----------|-----------|-----------|
##            1 |        17 |        52 |        69 |
##              |     0.096 |     0.292 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       114 |        64 |       178 |
## -------------|-----------|-----------|-----------|
##
##
```

The C5.0 Decision Tree model correctly predicted 82.0% of the passengers on the training set. This is better than the null model.

Boosted C5.0 Decision Tree

I will now try to improve on this using boosting.

```
tit5_boost10 <- C5.0(Survived ~ ., data = train,
                      trials = 10) #build 10 trees
tit5_boost10
```

```
##
## Call:
## C5.0.formula(formula = Survived ~ ., data = train, trials = 10)
##
## Classification Tree
## Number of samples: 713
## Number of predictors: 6
##
## Number of boosting iterations: 10
## Average tree size: 5.9
##
## Non-standard options: attempt to group attributes
```

```
summary(tit5_boost10)
```

```
##
## Call:
## C5.0.formula(formula = Survived ~ ., data = train, trials = 10)
##
##
## C5.0 [Release 2.07 GPL Edition]      Sat Mar 06 21:10:44 2021
## -------------------------------
##
## Class specified by attribute `outcome'
##
## Read 713 cases (7 attributes) from undefined.data
##
## -----  Trial 0:  -----
##
## Decision tree:
##
## Sex_num <= 0:
## :...Age > 13: 0 (433/72)
## :   Age <= 13:
## :   :...SibSp <= 2: 1 (17)
## :       SibSp > 2: 0 (12)
## Sex_num > 0:
## :...Pclass in {1,2}: 1 (135/6)
##     Pclass = 3:
##     :...Fare <= 23.25: 1 (92/40)
##         Fare > 23.25: 0 (24/3)
##
## -----  Trial 1:  -----
##
## Decision tree:
```

```
##
## Pclass = 1: 1 (186.8/53.6)
## Pclass in {2,3}:
## :...Sex_num <= 0: 0 (334.2/85)
##     Sex_num > 0:
##     :...Pclass = 2: 1 (48.7/7.9)
##         Pclass = 3: 0 (143.3/47.6)
##
## -----  Trial 2:  -----
##
## Decision tree:
##
## Sex_num > 0: 1 (255.7/95.4)
## Sex_num <= 0:
## :...Age <= 13: 1 (28.4/8.1)
##     Age > 13: 0 (428.9/148.4)
##
## -----  Trial 3:  -----
##
## Decision tree:
##
## Fare > 52: 1 (124.1/38.9)
## Fare <= 52:
## :...SibSp > 2: 0 (30.3/4.4)
##     SibSp <= 2:
##     :...Age <= 8: 1 (29.8/2)
##         Age > 8: 0 (528.9/216.1)
##
## -----  Trial 4:  -----
##
## Decision tree:
##
## SibSp > 2: 0 (35.8/6.2)
## SibSp <= 2:
## :...Age <= 8: 1 (27.6/2.3)
##     Age > 8:
##     :...Sex_num > 0:
##         :...Pclass in {1,2}: 1 (96.4/15.5)
##         :   Pclass = 3: 0 (138.4/60.9)
##         Sex_num <= 0:
##         :...Fare <= 7.7417: 0 (45.8/9.1)
##             Fare > 7.7417:
##             :...Pclass = 2: 0 (61.9/21.2)
##                 Pclass in {1,3}:
##                 :...Age <= 53: 1 (285.1/133.3)
##                     Age > 53: 0 (22/4.1)
##
## -----  Trial 5:  -----
##
## Decision tree:
##
## Sex_num > 0: 1 (258.6/99)
## Sex_num <= 0:
## :...SibSp > 1: 0 (28.9/5.2)
```

```
##     SibSp <= 1:
##     :...Age <= 16: 1 (22.2/5.1)
##         Age > 16: 0 (403.3/160.3)
##
## -----  Trial 6:  -----
##
## Decision tree:
##
## SibSp > 2: 0 (36.7/6.6)
## SibSp <= 2:
## :...Age <= 9: 1 (29.5/6.2)
##     Age > 9:
##     :...Fare <= 7.7292: 0 (39.4/4.3)
##         Fare > 7.7292:
##         :...Sex_num > 0:
##             :...Pclass in {1,2}: 1 (81.5/15.9)
##             :   Pclass = 3: 0 (140/59)
##             Sex_num <= 0:
##             :...Pclass = 2: 0 (33.9)
##                 Pclass = 1: 1 (147.2/66)
##                 Pclass = 3:
##                 :...Fare <= 42.4: 0 (181.3/73.6)
##                     Fare > 42.4: 1 (12.7/1.9)
##
## -----  Trial 7:  -----
##
## Decision tree:
##
## Sex_num > 0:
## :...Pclass in {1,2}: 1 (65.5)
## :   Pclass = 3:
## :   :...Age > 38: 0 (15.5/2.4)
## :       Age <= 38:
## :       :...Age > 32.5: 1 (10.5)
## :           Age <= 32.5:
## :           :...Fare <= 24.15: 1 (172.5/69.9)
## :               Fare > 24.15: 0 (19.7/3.4)
## Sex_num <= 0:
## :...SibSp > 1: 0 (23.9/3.1)
##     SibSp <= 1:
##     :...Age > 53: 0 (36.7/4.9)
##         Age <= 53:
##         :...Age <= 17: 1 (29/10.7)
##             Age > 17:
##             :...Fare <= 26: 0 (132.7)
##                 Fare > 26:
##                 :...Fare <= 27: 1 (23/2.8)
##                     Fare > 27: 0 (145/54.3)
##
## -----  Trial 8:  -----
##
## Decision tree:
##
## Pclass = 3: 0 (382.6/119.7)
```

```
## Pclass in {1,2}:
## :...Sex_num > 0: 1 (54)
##     Sex_num <= 0:
##     :...Fare <= 16.7: 0 (21)
##         Fare > 16.7:
##         :...Age <= 53: 1 (187.7/89.9)
##             Age > 53: 0 (26.7/3.4)
##
## -----  Trial 9:  -----
##
## Decision tree:
##
## Sex_num <= 0: 0 (377.6/95.9)
## Sex_num > 0:
## :...Pclass in {1,2}: 1 (47.4)
##     Pclass = 3:
##     :...Fare > 23.25: 0 (21.8)
##         Fare <= 23.25:
##         :...Age <= 28.5: 1 (179.9/74.6)
##             Age > 28.5: 0 (34.3/10.5)
##
##
## Evaluation on training data (713 cases):
##
## Trial        Decision Tree
## -----      ----------------
##    Size      Errors
##
##    0      6  121(17.0%)
##    1      4  175(24.5%)
##    2      3  151(21.2%)
##    3      4  194(27.2%)
##    4      8  265(37.2%)
##    5      4  147(20.6%)
##    6      9  150(21.0%)
##    7     11  126(17.7%)
##    8      5  173(24.3%)
##    9      5  133(18.7%)
## boost            125(17.5%)   <<
##
##
##     (a)   (b)    <-classified as
##    ----  ----
##     432     8    (a): class 0
##     117   156    (b): class 1
##
##
##  Attribute usage:
##
##  100.00% Pclass
##  100.00% SibSp
##  100.00% Fare
##  100.00% Sex_num
##   99.72% Age
```

```
##
##
## Time: 0.0 secs
```

```
tit5_boost_pred10 <- predict(tit5_boost10, test)
CrossTable(test$Survived, tit5_boost_pred10,
           prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
           dnn = c('actual', 'predicted'))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  178
##
##
##              | predicted
##       actual |         0 |         1 | Row Total |
## -------------|-----------|-----------|-----------|
##            0 |       105 |         4 |       109 |
##              |     0.590 |     0.022 |           |
## -------------|-----------|-----------|-----------|
##            1 |        33 |        36 |        69 |
##              |     0.185 |     0.202 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       138 |        40 |       178 |
## -------------|-----------|-----------|-----------|
##
##
```

The boosted tree correctly identified 82.6% of the passengers on the training set. While better than the null model, the boosted decision tree is only slightly better than the original tree.

Random Forest

I will now build a random forest model with the data

```
library(ranger)
```

```
## Warning: package 'ranger' was built under R version 3.6.3
```

```
## Random Forests ----

set.seed(42)
rf <- ranger(Survived ~ ., data = train, num.threads = 2)
rf
```

```
## Ranger result
##
## Call:
##  ranger(Survived ~ ., data = train, num.threads = 2)
##
## Type:                             Classification
## Number of trees:                  500
## Sample size:                      713
## Number of independent variables:  6
## Mtry:                             2
## Target node size:                 1
## Variable importance mode:         none
## Splitrule:                        gini
## OOB prediction error:             17.67 %
```

```
rf$confusion.matrix
```

```
##     predicted
## true   0   1
##    0 398  42
##    1  84 189
```

```
p2 <- predict(rf, test, type="response" )
sum(p2$predictions == test$Survived ) / length( p2$predictions )
```

```
## [1] 0.8539326
```

The decision tree correctly classified 82.0% of the test data.

Conclusion

The KNN and boosted decision tree models outperformed the others, correctly identifying 82.6% and 82.4% of the passengers.

Kaggle Final Model

For the Kaggle competition, I will choose to upload results for the boosted decision tree models.

```
pred_tree_final = predict(tit5_boost10, titanic_test_2)
```

```
final_df = data.frame("PassengerID" = titanic_test$PassengerId, "Survived" = pred_tree_final)
head(final_df)
```

```
##   PassengerID Survived
## 1         892        0
## 2         893        0
## 3         894        0
## 4         895        0
## 5         896        0
## 6         897        0
```

```
#write.csv(final_df,"sahagun_matthew_titanic_kaggle.csv", row.names = FALSE)
```

When I submitted the file to Kaggle, I was informed that I correctly identified 77.5% of the passengers.