STAT 351 Project

Charles Hwang

Professor Matthews

STAT 351-001

2 May 2020

(1) **Introduction**: The goal of this project is to build a model to predict which passengers survived the sinking of the RMS *Titanic*. I am doing this as part of the *Titanic* competition on Kaggle. Because this competition is a "Getting Started" competition, there are no prizes for certain positions on the public leaderboard, making it a good competition for students (like myself) to learn about Kaggle and machine learning.

(2) **Data**: The outcome variable is binary, with a 1 indicating a survival and a 0 indicating a death. There are many different variables in the data set to describe 1,309 of the 2,224 passengers onboard the maiden voyage of the RMS *Titanic*. The full list included ticket class, name, sex, age, numbers of siblings/spouses and parents/children of passenger onboard, ticket and cabin numbers, fare, and port of embarkation. There are designated training and test data sets containing 891 and 418 observations, respectively. There were 866 missing values in the training set and I was able to visualize which variables these missing values belonged to with two histograms and plots using the aggr() function. These can be found in the (6.1) **Data Cleaning and Imputation** subsection of this report.

(3) **Methods**: I read the data into RStudio and analyzed each variable. I built a classification and regression tree (CART) using the rpart() function from the "rpart" package and obtained its predicted values and cross-validation error. I pruned the tree with the prune() function but the pruned tree turned out to be too basic and potentially overlooking important splits. Finally, the predictions were copied to a newly-created data set "tpdf" and written to a CSV file to be uploaded to the Kaggle competition webpage. All of the code for this project can be found in the (6) **Appendix** of this report.

(4) **Results**: The tree I fit had splits on several variables. The results can be found in the (6.2) Tree Models subsection of this report. After uploading the CSV file to the Kaggle competition webpage, my calculated Kaggle score based part of the test data was 0.78468 ($\frac{328}{418}$). The Kaggle score is a fraction between 0 and 1 calculated by dividing the number of correctly-predicted observations by the total number of observations in the test set (418). Therefore, a higher score indicates a model that is more well-fit to the data provided. On the same lines, there are only 419 possible scores ($\frac{0}{418}, \frac{1}{418}, ..., \frac{418}{418}$).

(5) **Conclusions/Future Work**: I conclude that my Kaggle score was 0.78468. It is impractical to determine my true overall ranking because there are only 419 possible scores and my score is tied with at least 24 other scores above mine, and the Kaggle website automatically listed me at the bottom of this score. Because the leaderboard used a two-month rolling window, the total number of competitors is also not known. However, the Kaggle competition had 20,012 competitors in the past two months at the time of the creation of this report, many of which make numerous submissions over the course of the competition to improve upon their score. Additionally, because the full data set including the values of the response variable for each observation is publicly available online per the rules of the competition, many users may have simply searched online for this data set and uploaded it so that their username would be near the top of the leaderboard, even if it is only for two months and there are no further prizes or benefits for doing so. As such, at least the first fifty positions on the public leaderboard have a perfect score of 1.00000 and several of them have only one competition entry (presumably users who uploaded the data set online instead of obtaining a perfect score by chance). This leads to rank deflation, causing users' scores to be ranked lower than they would have if all

users had uploaded scores based on models that were entirely the result of their own work. In light of this information, this score is good.

There are many different things I would do in a similar project in the future if I had more time, including:

- Fitting a larger tree
- Fitting a tree or trees with functions from different packages or a different set.seed() value
  - I experimented with fitting a series of trees using the tree() function from the "tree" package; however, the resulting Kaggle score of the model was the same as the one that resulted from the model in this report.
- Using different tree-based methods like bagging
- Fitting a random forest
  - I decided not to grow a random forest due to time constraints in dealing with missing data.
- Experimenting with other types of prediction models like gradient boosted models (GBMs) to see how the data would be handled
- After some time, searching online for the dataset with the true values of the response variable to see which observations I predicted incorrectly and possibly why (if they were outliers, fell on the wrong side of a split in the CART model, etc.)

(6) Appendix

(6.1) Data Cleaning and Imputation

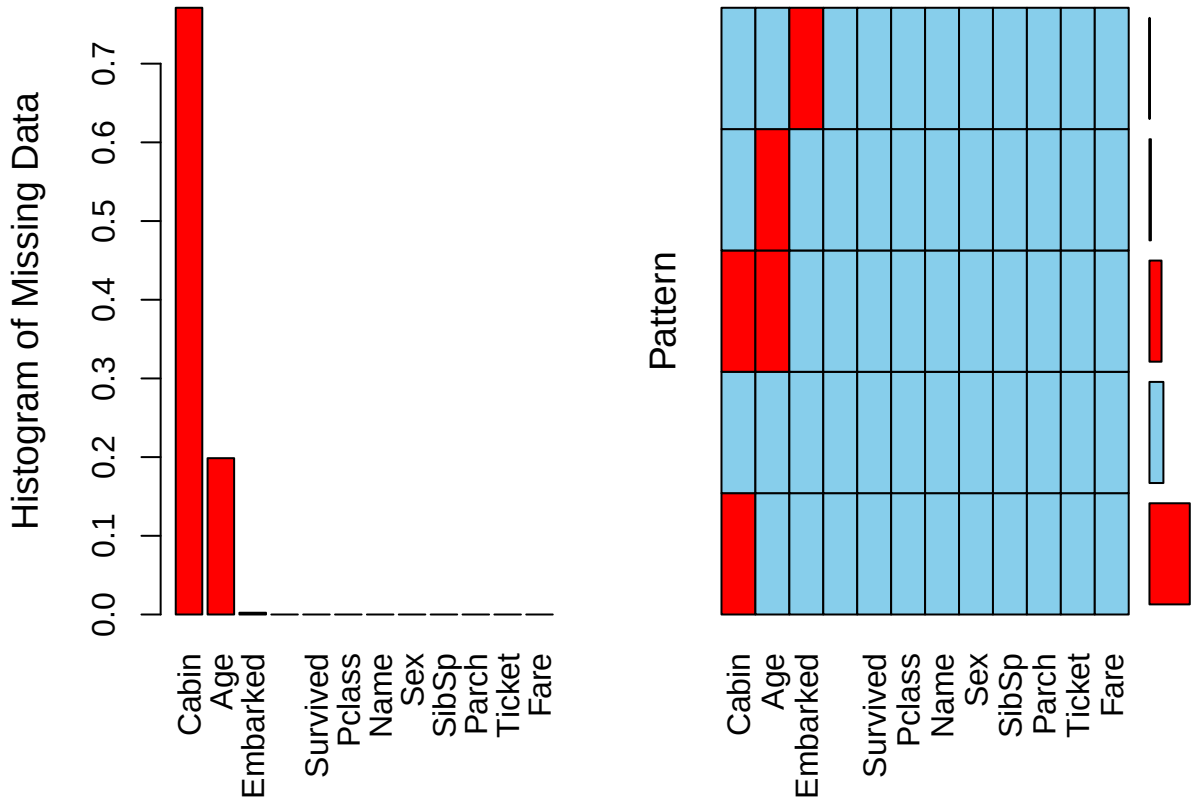
```
rm(list=ls())
sample_submission <- read.csv(file="/Users/newuser/Desktop/Notes/Undergraduate/STAT 351 - Nonparametric
test <- read.csv(file="/Users/newuser/Desktop/Notes/Undergraduate/STAT 351 - Nonparametric Statistical
train <- read.csv(file="/Users/newuser/Desktop/Notes/Undergraduate/STAT 351 - Nonparametric Statistical
library(mice)
library(rpart)
library(randomForest)
library(tidyr)
library(tree)
library(VIM)
train$Survived <- as.factor(train$Survived)
train[train==""] <- NA # Changing blank ("") values to "NA"
train$Name <- as.character(train$Name)
train$Ticket <- as.character(train$Ticket)
train$Cabin <- as.character(train$Cabin)
test$Name <- as.character(test$Name)
test$Ticket <- as.character(test$Ticket)
test$Cabin <- as.character(test$Cabin)
sum(is.na(train[,c("Survived")])) # There are no missing values in the predictor/response variable.
```

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

```
sum(is.na(train)) # However, there are 866 missing values in the rest of the data. Let's visualize that
```

```
## [1] 866
```

```
aggr(train,sortVars=TRUE,labels=names(train),ylab=c("Histogram of Missing Data","Pattern"))
```

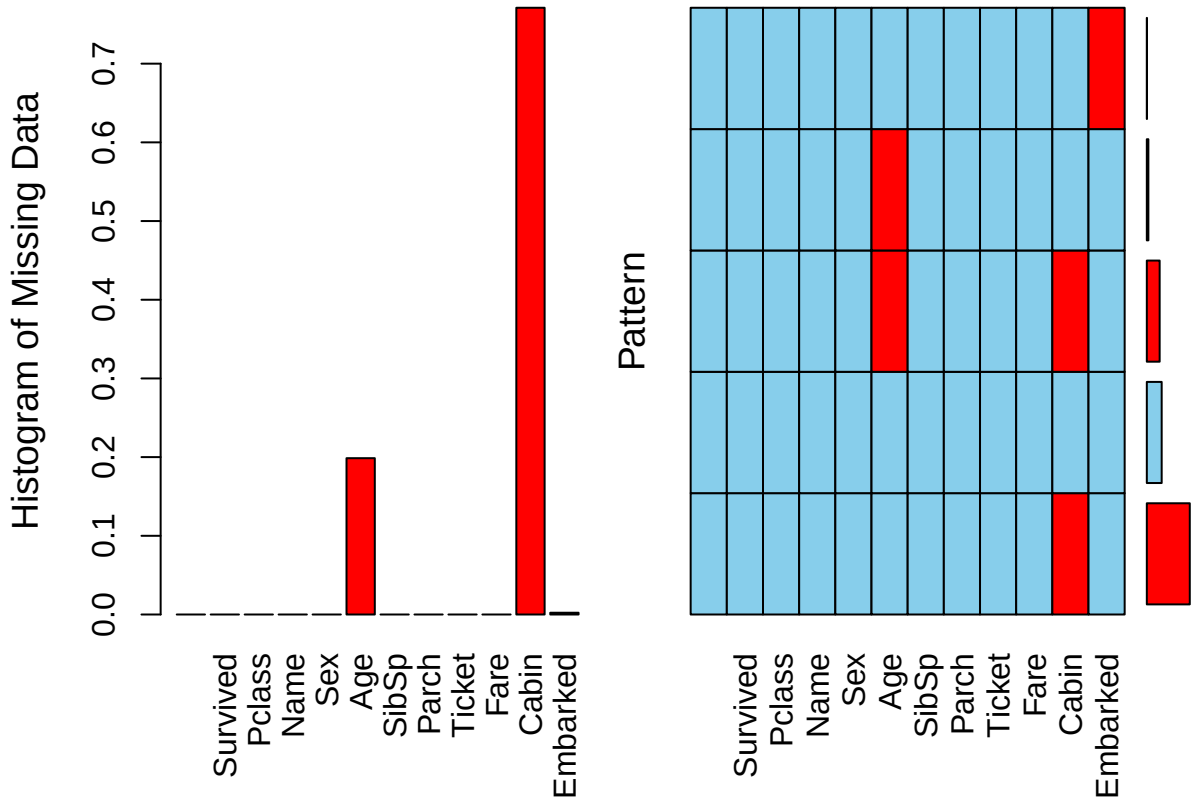



```
##
##  Variables sorted by number of missings:
##     Variable       Count
##        Cabin 0.771043771
##          Age 0.198653199
##     Embarked 0.002244669
##  PassengerId 0.000000000
##     Survived 0.000000000
##       Pclass 0.000000000
##         Name 0.000000000
##          Sex 0.000000000
##        SibSp 0.000000000
##        Parch 0.000000000
##       Ticket 0.000000000
##         Fare 0.000000000
```

```
cat("The table tells us that '",aggr(train,plot=FALSE,bars=FALSE)$missing$Variable[order(aggr(train,plo
```

```
## The table tells us that 'Cabin' and 'Age' are the variables with the large majority of missing values
```

```
aggr(train,labels=names(train),ylab=c("Histogram of Missing Data","Pattern"),gap=2.5) # The same histog
```




```
cart <- mice(train[,-c(4,9)],method="cart",seed=2025) # Removed "Name" and "Ticket" variables so that t
```

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

```
summary(cart)
```

```
## Class: mids
## Number of multiple imputations:  5
## Imputation methods:
## PassengerId    Survived      Pclass         Sex         Age       SibSp
##          ""          ""          ""          ""      "cart"          ""
##       Parch        Fare       Cabin    Embarked
##          ""          ""          ""          ""
## PredictorMatrix:
##             PassengerId Survived Pclass Sex Age SibSp Parch Fare Cabin Embarked
## PassengerId           0        1      1   0   1     1     1    1     0        0
## Survived              1        0      1   0   1     1     1    1     0        0
## Pclass                1        1      0   0   1     1     1    1     0        0
## Sex                   1        1      1   0   1     1     1    1     0        0
## Age                   1        1      1   0   0     1     1    1     0        0
## SibSp                 1        1      1   0   1     0     1    1     0        0
## Number of logged events:  3
##   it im dep     meth      out
## 1  0  0     constant      Sex
## 2  0  0     constant    Cabin
## 3  0  0     constant Embarked
```

```
sapply(complete(cart),function(x) sum(is.na(x))) # Checking that missing values were imputed
```

```
## PassengerId    Survived      Pclass         Sex         Age       SibSp
##           0           0           0           0           0           0
##       Parch        Fare       Cabin    Embarked
##           0           0         687           2
```

```
levels(train$Cabin)
```

```
## NULL
```

```
train$Cabin <- factor(train$Cabin) # Removing empty levels of factor variables
levels(train$Cabin) # Check
```

```
##   [1] "A10"             "A14"             "A16"             "A19"
##   [5] "A20"             "A23"             "A24"             "A26"
##   [9] "A31"             "A32"             "A34"             "A36"
##  [13] "A5"              "A6"              "A7"              "B101"
##  [17] "B102"            "B18"             "B19"             "B20"
##  [21] "B22"             "B28"             "B3"              "B30"
##  [25] "B35"             "B37"             "B38"             "B39"
##  [29] "B4"              "B41"             "B42"             "B49"
##  [33] "B5"              "B50"             "B51 B53 B55"     "B57 B59 B63 B66"
##  [37] "B58 B60"         "B69"             "B71"             "B73"
##  [41] "B77"             "B78"             "B79"             "B80"
##  [45] "B82 B84"         "B86"             "B94"             "B96 B98"
##  [49] "C101"            "C103"            "C104"            "C106"
##  [53] "C110"            "C111"            "C118"            "C123"
##  [57] "C124"            "C125"            "C126"            "C128"
##  [61] "C148"            "C2"              "C22 C26"         "C23 C25 C27"
##  [65] "C30"             "C32"             "C45"             "C46"
##  [69] "C47"             "C49"             "C50"             "C52"
##  [73] "C54"             "C62 C64"         "C65"             "C68"
```

```
##  [77] "C7"              "C70"             "C78"             "C82"
##  [81] "C83"             "C85"             "C86"             "C87"
##  [85] "C90"             "C91"             "C92"             "C93"
##  [89] "C95"             "C99"             "D"               "D10 D12"
##  [93] "D11"             "D15"             "D17"             "D19"
##  [97] "D20"             "D21"             "D26"             "D28"
## [101] "D30"             "D33"             "D35"             "D36"
## [105] "D37"             "D45"             "D46"             "D47"
## [109] "D48"             "D49"             "D50"             "D56"
## [113] "D6"              "D7"              "D9"              "E10"
## [117] "E101"            "E12"             "E121"            "E17"
## [121] "E24"             "E25"             "E31"             "E33"
## [125] "E34"             "E36"             "E38"             "E40"
## [129] "E44"             "E46"             "E49"             "E50"
## [133] "E58"             "E63"             "E67"             "E68"
## [137] "E77"             "E8"              "F E69"           "F G63"
## [141] "F G73"           "F2"              "F33"             "F38"
## [145] "F4"              "G6"              "T"
```

```
levels(train$Embarked)
```

```
## NULL
```

```
train$Embarked <- factor(train$Embarked)
levels(train$Embarked) # Check
```

```
## [1] "C" "Q" "S"
```

```
rf <- mice(train[,-c(4,9)],method="rf",seed=2025) # Data imputation with random forest method
```

```
##
##  iter imp variable
##   1   1  Age  Cabin  Embarked
##   1   2  Age  Cabin  Embarked
##   1   3  Age  Cabin  Embarked
##   1   4  Age  Cabin  Embarked
##   1   5  Age  Cabin  Embarked
##   2   1  Age  Cabin  Embarked
##   2   2  Age  Cabin  Embarked
##   2   3  Age  Cabin  Embarked
##   2   4  Age  Cabin  Embarked
##   2   5  Age  Cabin  Embarked
##   3   1  Age  Cabin  Embarked
##   3   2  Age  Cabin  Embarked
##   3   3  Age  Cabin  Embarked
##   3   4  Age  Cabin  Embarked
##   3   5  Age  Cabin  Embarked
##   4   1  Age  Cabin  Embarked
##   4   2  Age  Cabin  Embarked
##   4   3  Age  Cabin  Embarked
##   4   4  Age  Cabin  Embarked
##   4   5  Age  Cabin  Embarked
##   5   1  Age  Cabin  Embarked
##   5   2  Age  Cabin  Embarked
##   5   3  Age  Cabin  Embarked
##   5   4  Age  Cabin  Embarked
```

```
##   5   5  Age  Cabin  Embarked
```

```
summary(rf)
```

```
## Class: mids
## Number of multiple imputations:  5
## Imputation methods:
## PassengerId    Survived      Pclass         Sex         Age       SibSp
##          ""          ""          ""          ""        "rf"          ""
##       Parch        Fare       Cabin    Embarked
##          ""          ""        "rf"        "rf"
## PredictorMatrix:
##             PassengerId Survived Pclass Sex Age SibSp Parch Fare Cabin Embarked
## PassengerId           0        1      1   0   1     1     1    1     1        1
## Survived              1        0      1   0   1     1     1    1     1        1
## Pclass                1        1      0   0   1     1     1    1     1        1
## Sex                   1        1      1   0   1     1     1    1     1        1
## Age                   1        1      1   0   0     1     1    1     1        1
## SibSp                 1        1      1   0   1     0     1    1     1        1
## Number of logged events:  34
##   it im      dep     meth                out
## 1  0  0          constant                Sex
## 2  1  1      Age       rf CabinB78, CabinC128
## 3  1  2 Embarked       rf            CabinB28
## 4  1  4      Age       rf           CabinC106
## 5  1  4 Embarked       rf            CabinB28
## 6  1  5      Age       rf  CabinA19, CabinB78
```

```
sapply(complete(rf),function(x) sum(is.na(x))) # Checking that missing values were imputed
```

```
## PassengerId    Survived      Pclass         Sex         Age       SibSp
##           0           0           0           0           0           0
##       Parch        Fare       Cabin    Embarked
##           0           0           0           0
```

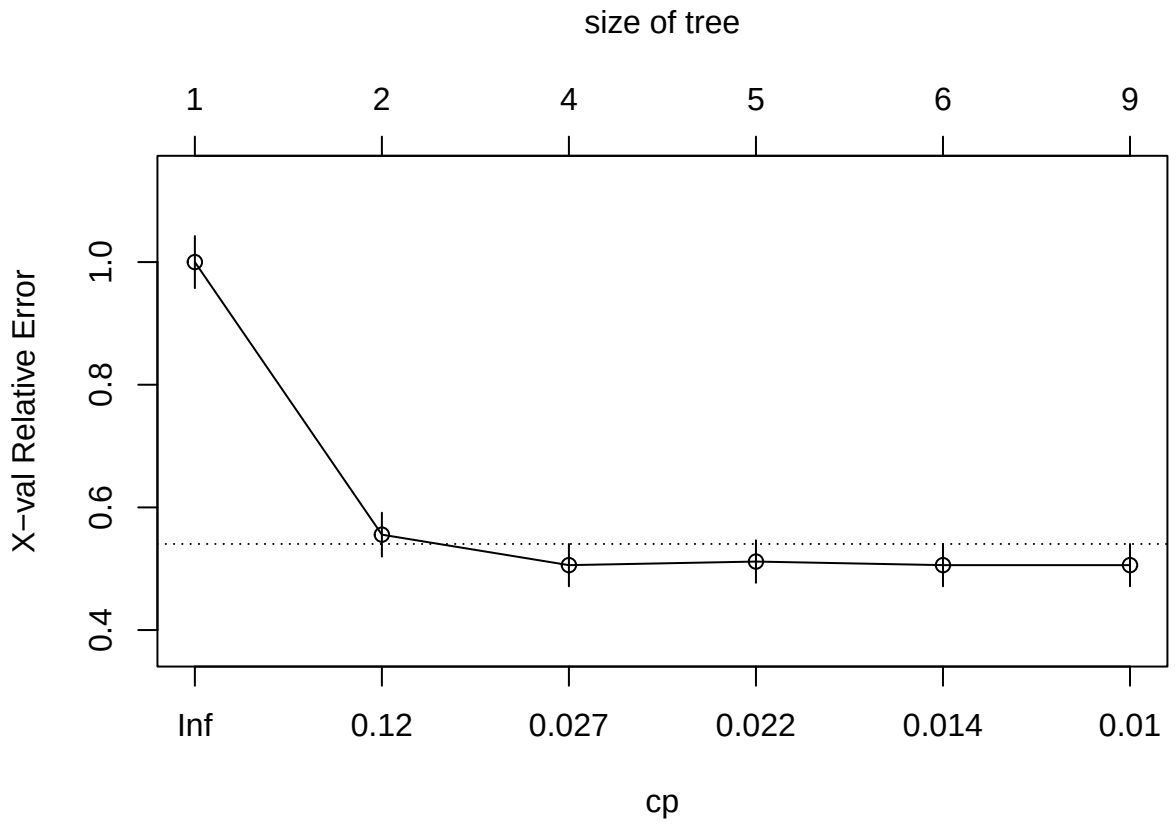
(6.2) Tree Models

```
set.seed(2025)
tree <- rpart(Survived~.,data=train[,-c(4,9,11)])
printcp(tree)
```

```
##
## Classification tree:
## rpart(formula = Survived ~ ., data = train[, -c(4, 9, 11)])
##
## Variables actually used in tree construction:
## [1] Age      Embarked Fare     Pclass   Sex      SibSp
##
## Root node error: 342/891 = 0.38384
##
## n= 891
##
##         CP nsplit rel error  xerror     xstd
## 1 0.444444      0   1.00000 1.00000 0.042446
## 2 0.030702      1   0.55556 0.55556 0.035750
```

```
## 3 0.023392      3   0.49415 0.50585 0.034524
## 4 0.020468      4   0.47076 0.51170 0.034675
## 5 0.010234      5   0.45029 0.50585 0.034524
## 6 0.010000      8   0.41813 0.50585 0.034524
```

```
plotcp(tree)
```




```
par(mfrow=c(1,2))
plot(tree,uniform=TRUE,main="Classification Tree")
text(tree,use.n=TRUE,all=TRUE,cex=.7)
summary(tree)
```

```
## Call:
## rpart(formula = Survived ~ ., data = train[, -c(4, 9, 11)])
##   n= 891
##
##           CP nsplit rel error    xerror       xstd
## 1 0.44444444      0 1.0000000 1.0000000 0.04244576
## 2 0.03070175      1 0.5555556 0.5555556 0.03574957
## 3 0.02339181      3 0.4941520 0.5058480 0.03452394
## 4 0.02046784      4 0.4707602 0.5116959 0.03467453
## 5 0.01023392      5 0.4502924 0.5058480 0.03452394
## 6 0.01000000      8 0.4181287 0.5058480 0.03452394
##
## Variable importance
##         Sex        Fare      Pclass       SibSp       Parch PassengerId
##          45          17          13           6           6           4
##         Age    Embarked
##           4           4
```

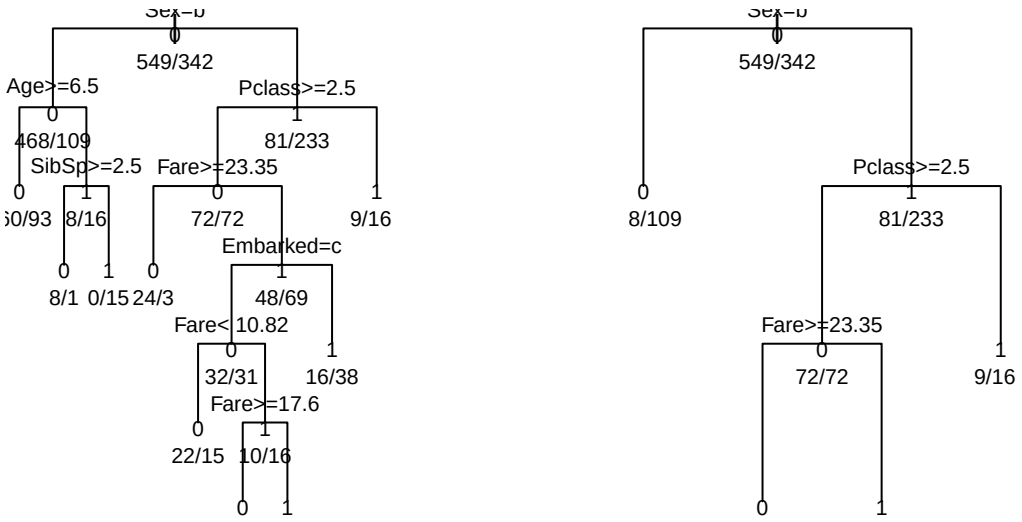
```
##
## Node number 1: 891 observations,    complexity param=0.4444444
##   predicted class=0  expected loss=0.3838384  P(node) =1
##     class counts:   549   342
##    probabilities: 0.616 0.384
##   left son=2 (577 obs) right son=3 (314 obs)
##   Primary splits:
##       Sex      splits as  RL,           improve=124.426300, (0 missing)
##       Pclass   < 2.5      to the right, improve= 43.781830, (0 missing)
##       Fare     < 10.48125 to the left,  improve= 37.941940, (0 missing)
##       Embarked splits as  RLL,          improve= 12.131190, (2 missing)
##       Parch    < 0.5      to the left,  improve=  9.157774, (0 missing)
##   Surrogate splits:
##       Fare        < 77.6229  to the left,  agree=0.679, adj=0.089, (0 split)
##       Parch       < 0.5      to the left,  agree=0.678, adj=0.086, (0 split)
##       PassengerId < 4.5      to the right, agree=0.650, adj=0.006, (0 split)
##
## Node number 2: 577 observations,    complexity param=0.02339181
##   predicted class=0  expected loss=0.1889081  P(node) =0.647587
##     class counts:   468   109
##    probabilities: 0.811 0.189
##   left son=4 (553 obs) right son=5 (24 obs)
##   Primary splits:
##       Age      < 6.5      to the right, improve=10.788930, (124 missing)
##       Fare     < 26.26875 to the left,  improve=10.216720, (0 missing)
##       Pclass   < 1.5      to the right, improve=10.019140, (0 missing)
##       Parch    < 0.5      to the left,  improve= 3.350327, (0 missing)
##       Embarked splits as  RLL,          improve= 3.079304, (0 missing)
##
## Node number 3: 314 observations,    complexity param=0.03070175
##   predicted class=1  expected loss=0.2579618  P(node) =0.352413
##     class counts:    81   233
##    probabilities: 0.258 0.742
##   left son=6 (144 obs) right son=7 (170 obs)
##   Primary splits:
##       Pclass   < 2.5      to the right, improve=31.163130, (0 missing)
##       Fare     < 48.2     to the left,  improve=10.114210, (0 missing)
##       SibSp    < 2.5      to the right, improve= 9.372551, (0 missing)
##       Parch    < 3.5      to the right, improve= 5.140857, (0 missing)
##       Embarked splits as  RLL,          improve= 3.542239, (2 missing)
##   Surrogate splits:
##       Fare        < 25.69795 to the left,  agree=0.799, adj=0.563, (0 split)
##       Embarked    splits as  RLR,          agree=0.631, adj=0.194, (0 split)
##       PassengerId < 256.5    to the left,  agree=0.608, adj=0.146, (0 split)
##       SibSp       < 1.5      to the right, agree=0.592, adj=0.111, (0 split)
##       Parch       < 1.5      to the right, agree=0.567, adj=0.056, (0 split)
##
## Node number 4: 553 observations
##   predicted class=0  expected loss=0.1681736  P(node) =0.620651
##     class counts:   460    93
##    probabilities: 0.832 0.168
##
## Node number 5: 24 observations,    complexity param=0.02046784
##   predicted class=1  expected loss=0.3333333  P(node) =0.02693603
```

```
##     class counts:     8    16
##    probabilities: 0.333 0.667
##   left son=10 (9 obs) right son=11 (15 obs)
##   Primary splits:
##       SibSp       < 2.5      to the right, improve=8.8888890, (0 missing)
##       Pclass      < 2.5      to the right, improve=3.8095240, (0 missing)
##       PassengerId < 189      to the left,  improve=2.8683470, (0 missing)
##       Fare        < 20.825   to the right, improve=2.6666670, (0 missing)
##       Age         < 1.5      to the right, improve=0.6095238, (0 missing)
##   Surrogate splits:
##       PassengerId < 178      to the left,  agree=0.792, adj=0.444, (0 split)
##       Pclass      < 2.5      to the right, agree=0.792, adj=0.444, (0 split)
##       Fare        < 26.95    to the right, agree=0.750, adj=0.333, (0 split)
##       Embarked    splits as  RLR,          agree=0.708, adj=0.222, (0 split)
##
## Node number 6: 144 observations,    complexity param=0.03070175
##   predicted class=0  expected loss=0.5  P(node) =0.1616162
##     class counts:    72    72
##    probabilities: 0.500 0.500
##   left son=12 (27 obs) right son=13 (117 obs)
##   Primary splits:
##       Fare        < 23.35    to the right, improve=10.051280, (0 missing)
##       Embarked    splits as  RRL,          improve= 7.071429, (0 missing)
##       SibSp       < 2.5      to the right, improve= 4.571429, (0 missing)
##       PassengerId < 396      to the right, improve= 4.020870, (0 missing)
##       Age         < 38.5     to the right, improve= 3.875163, (42 missing)
##   Surrogate splits:
##       SibSp       < 2.5      to the right, agree=0.882, adj=0.370, (0 split)
##       Parch       < 1.5      to the right, agree=0.882, adj=0.370, (0 split)
##       PassengerId < 884.5    to the right, agree=0.826, adj=0.074, (0 split)
##
## Node number 7: 170 observations
##   predicted class=1  expected loss=0.05294118  P(node) =0.1907969
##     class counts:     9   161
##    probabilities: 0.053 0.947
##
## Node number 10: 9 observations
##   predicted class=0  expected loss=0.1111111  P(node) =0.01010101
##     class counts:     8     1
##    probabilities: 0.889 0.111
##
## Node number 11: 15 observations
##   predicted class=1  expected loss=0  P(node) =0.01683502
##     class counts:     0    15
##    probabilities: 0.000 1.000
##
## Node number 12: 27 observations
##   predicted class=0  expected loss=0.1111111  P(node) =0.03030303
##     class counts:    24     3
##    probabilities: 0.889 0.111
##
## Node number 13: 117 observations,    complexity param=0.01023392
##   predicted class=1  expected loss=0.4102564  P(node) =0.1313131
##     class counts:    48    69
```

```
##    probabilities: 0.410 0.590
##   left son=26 (63 obs) right son=27 (54 obs)
##   Primary splits:
##       Embarked    splits as  RRL,          improve=2.6048030, (0 missing)
##       Age         < 16.5     to the right, improve=2.4685870, (34 missing)
##       Fare        < 7.8875   to the right, improve=2.0325270, (0 missing)
##       PassengerId < 396      to the right, improve=1.1554920, (0 missing)
##       SibSp       < 0.5      to the right, improve=0.3076923, (0 missing)
##   Surrogate splits:
##       Fare        < 7.7625   to the right, agree=0.667, adj=0.278, (0 split)
##       PassengerId < 571      to the left,  agree=0.598, adj=0.130, (0 split)
##
## Node number 26: 63 observations,    complexity param=0.01023392
##   predicted class=0  expected loss=0.4920635  P(node) =0.07070707
##     class counts:    32    31
##    probabilities: 0.508 0.492
##   left son=52 (37 obs) right son=53 (26 obs)
##   Primary splits:
##       Fare        < 10.825   to the left,  improve=1.34653300, (0 missing)
##       Age         < 27.5     to the right, improve=0.97840760, (5 missing)
##       Parch       < 0.5      to the left,  improve=0.71428570, (0 missing)
##       PassengerId < 634      to the left,  improve=0.55500060, (0 missing)
##       SibSp       < 0.5      to the right, improve=0.08821734, (0 missing)
##   Surrogate splits:
##       SibSp       < 0.5      to the left,  agree=0.746, adj=0.385, (0 split)
##       Parch       < 0.5      to the left,  agree=0.746, adj=0.385, (0 split)
##       PassengerId < 93.5     to the right, agree=0.651, adj=0.154, (0 split)
##       Age         < 11       to the right, agree=0.635, adj=0.115, (0 split)
##
## Node number 27: 54 observations
##   predicted class=1  expected loss=0.2962963  P(node) =0.06060606
##     class counts:    16    38
##    probabilities: 0.296 0.704
##
## Node number 52: 37 observations
##   predicted class=0  expected loss=0.4054054  P(node) =0.04152637
##     class counts:    22    15
##    probabilities: 0.595 0.405
##
## Node number 53: 26 observations,    complexity param=0.01023392
##   predicted class=1  expected loss=0.3846154  P(node) =0.0291807
##     class counts:    10    16
##    probabilities: 0.385 0.615
##   left son=106 (10 obs) right son=107 (16 obs)
##   Primary splits:
##       Fare        < 17.6     to the right, improve=3.23269200, (0 missing)
##       PassengerId < 138      to the left,  improve=0.80442430, (0 missing)
##       SibSp       < 0.5      to the right, improve=0.72599300, (0 missing)
##       Age         < 13       to the right, improve=0.33893560, (2 missing)
##       Parch       < 0.5      to the left,  improve=0.04578755, (0 missing)
##   Surrogate splits:
##       PassengerId < 65       to the left,  agree=0.692, adj=0.2, (0 split)
##       SibSp       < 1.5      to the right, agree=0.692, adj=0.2, (0 split)
##       Parch       < 1.5      to the right, agree=0.654, adj=0.1, (0 split)
```

```
##
## Node number 106: 10 observations
##   predicted class=0  expected loss=0.3  P(node) =0.01122334
##     class counts:     7     3
##    probabilities: 0.700 0.300
##
## Node number 107: 16 observations
##   predicted class=1  expected loss=0.1875  P(node) =0.01795735
##     class counts:     3    13
##    probabilities: 0.188 0.812
```

```
ptree <- prune(tree,cp=tree$cptable[which.min(tree$cptable[,"xerror"]),"CP"])
plot(ptree,uniform=TRUE,main="Pruned Classification Tree") # The pruned tree actually appears too simpl
text(ptree,use.n=TRUE,all=TRUE,cex=.7)
```




```
cat("Cross validation error:",min(tree$cptable[,"xerror"]))
```

```
## Cross validation error: 0.505848
```

```
par(mfrow=c(1,1))
T <- tree(Survived~.,data=train[,-c(4,9,11)]) # Fitting different tree
```

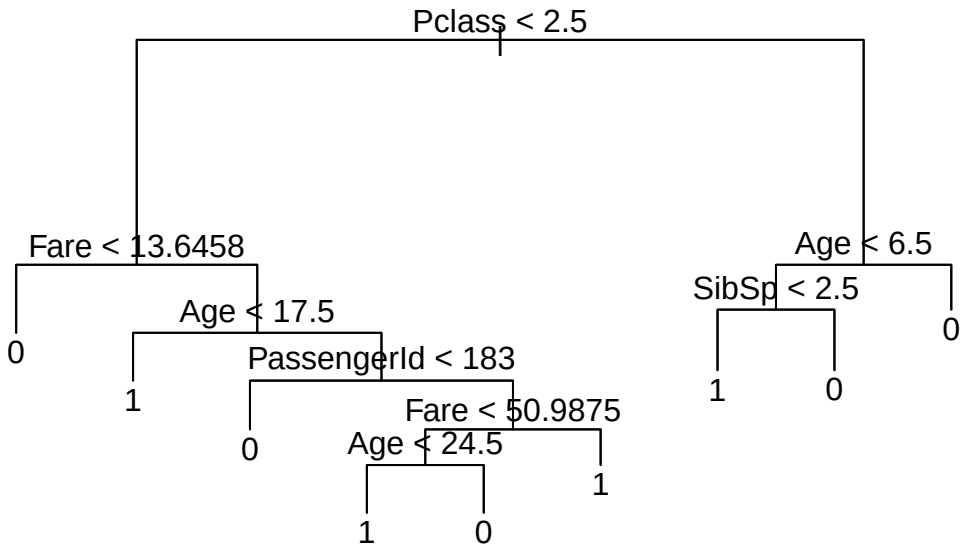
```
## Warning in tree(Survived ~ ., data = train[, -c(4, 9, 11)]): NAs introduced by
## coercion
```

```
T
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 712 960.900 0 ( 0.5955 0.4045 )
##    2) Pclass < 2.5 357 488.200 1 ( 0.4314 0.5686 )
##      4) Fare < 13.6458 84 106.900 0 ( 0.6667 0.3333 ) *
##      5) Fare > 13.6458 273 356.400 1 ( 0.3590 0.6410 )
##       10) Age < 17.5 32  14.960 1 ( 0.0625 0.9375 ) *
##       11) Age > 17.5 241 324.100 1 ( 0.3983 0.6017 )
##         22) PassengerId < 183 44  55.040 0 ( 0.6818 0.3182 ) *
```

```
##         23) PassengerId > 183 197 251.200 1 ( 0.3350 0.6650 )
##           46) Fare < 50.9875 99 136.400 1 ( 0.4545 0.5455 )
##             92) Age < 24.5 13   0.000 1 ( 0.0000 1.0000 ) *
##             93) Age > 24.5 86 119.000 0 ( 0.5233 0.4767 ) *
##           47) Fare > 50.9875 98 101.800 1 ( 0.2143 0.7857 ) *
##    3) Pclass > 2.5 355 390.800 0 ( 0.7606 0.2394 )
##      6) Age < 6.5 30  41.050 1 ( 0.4333 0.5667 )
##       12) SibSp < 2.5 16   7.481 1 ( 0.0625 0.9375 ) *
##       13) SibSp > 2.5 14  11.480 0 ( 0.8571 0.1429 ) *
##      7) Age > 6.5 325 333.400 0 ( 0.7908 0.2092 ) *
```

```
plot(T) # This tree ended up yielding the same score as the other two.
text(T)
```



(6.3) Predictions

```
treepred <- predict(tree,test)
table(treepred)
```

```
## treepred
##                   0 0.0529411764705882  0.111111111111111  0.168173598553345
##                   6                  80                  8                258
##              0.1875  0.296296296296296                0.3  0.405405405405405
##                  13                  31                  4                 18
##  0.594594594594595                 0.7  0.703703703703704             0.8125
##                  18                   4                  31                13
##  0.831826401446655  0.888888888888889  0.947058823529412                 1
##                258                   8                  80                 6
```

```
treepred[treepred>=.5] <- 1 # Assigning values to outcome variable
treepred[treepred<.5] <- 0
table(treepred[,2])
```

```
##
##   0   1
## 288 130
```

```
treepred[,2]
```

```
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
##   0   0   0   0   1   0   1   0   1   0   0   0   1   0   1   1   0   0   0   1
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
##   0   0   1   0   1   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0
##  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
##   0   0   0   1   1   0   0   0   1   1   0   0   1   1   0   0   0   0   0   1
##  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80
##   0   0   0   1   0   1   1   0   0   1   1   0   0   0   1   0   0   1   0   1
##  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100
##   1   0   0   0   0   0   1   0   1   1   1   0   1   0   0   0   1   0   0   0
## 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
##   1   0   0   0   1   0   0   0   0   0   0   1   1   1   1   0   0   1   0   1
## 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140
##   1   0   1   0   0   1   0   1   0   0   0   0   0   0   0   0   0   0   0   0
## 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160
##   0   1   0   0   0   0   0   0   0   0   1   0   0   1   0   0   1   0   0   1
## 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180
##   1   0   1   0   0   0   0   0   1   0   0   0   0   0   0   1   1   0   1   1
## 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200
##   0   0   1   0   1   0   1   0   0   0   0   0   0   0   0   0   1   0   0   0
## 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220
##   1   1   0   1   0   0   1   0   1   0   0   0   0   1   0   0   1   0   1   0
## 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240
##   1   0   1   0   1   1   0   1   0   0   0   1   0   0   0   0   0   0   1   1
## 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260
##   1   1   0   0   0   0   1   0   1   1   1   0   0   0   0   0   0   0   1   0
## 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280
##   0   0   1   1   0   0   0   0   0   0   0   0   1   1   0   1   0   0   0   0
## 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300
##   0   1   1   1   0   0   0   0   0   0   0   1   0   0   0   0   1   0   0   0
## 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320
##   0   0   0   0   1   1   0   1   0   1   0   0   0   1   1   1   0   0   0   0
## 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340
##   0   0   0   0   1   0   1   0   0   0   1   0   0   1   0   0   0   0   0   0
## 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360
##   0   0   0   1   0   0   0   1   0   1   1   0   0   0   0   0   1   0   0   1
## 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380
##   0   1   1   0   1   0   0   0   1   0   0   1   0   0   1   1   0   0   0   0
## 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400
##   0   0   1   1   0   1   0   0   0   0   0   1   0   0   0   1   0   1   0   0
## 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418
##   1   0   1   0   0   0   0   0   1   1   1   1   0   0   1   0   0   0
```

```
tpdf <- as.data.frame(rep(0,nrow(treepred))) # Exporting predictions to CSV file for Kaggle submission
tpdf$PassengerId <- test$PassengerId
tpdf$Survived <- treepred[,2]
tpdf$`rep(0, nrow(treepred))` <- NULL
write.csv(as.data.frame(tpdf),"tpdf Submission.csv",row.names=FALSE)
```

(6.4) Score

```
# Kaggle score: 0.78468 (328/418, 71st percentile)
```

This score, as previously discussed in (5) **Conclusions/Future Work**, is relatively good, given the resources that other competitors have and the shorter time period I had to complete this project.