Titanic Competition

Trevor Aeschliman

July 10, 2017

Getting the Data

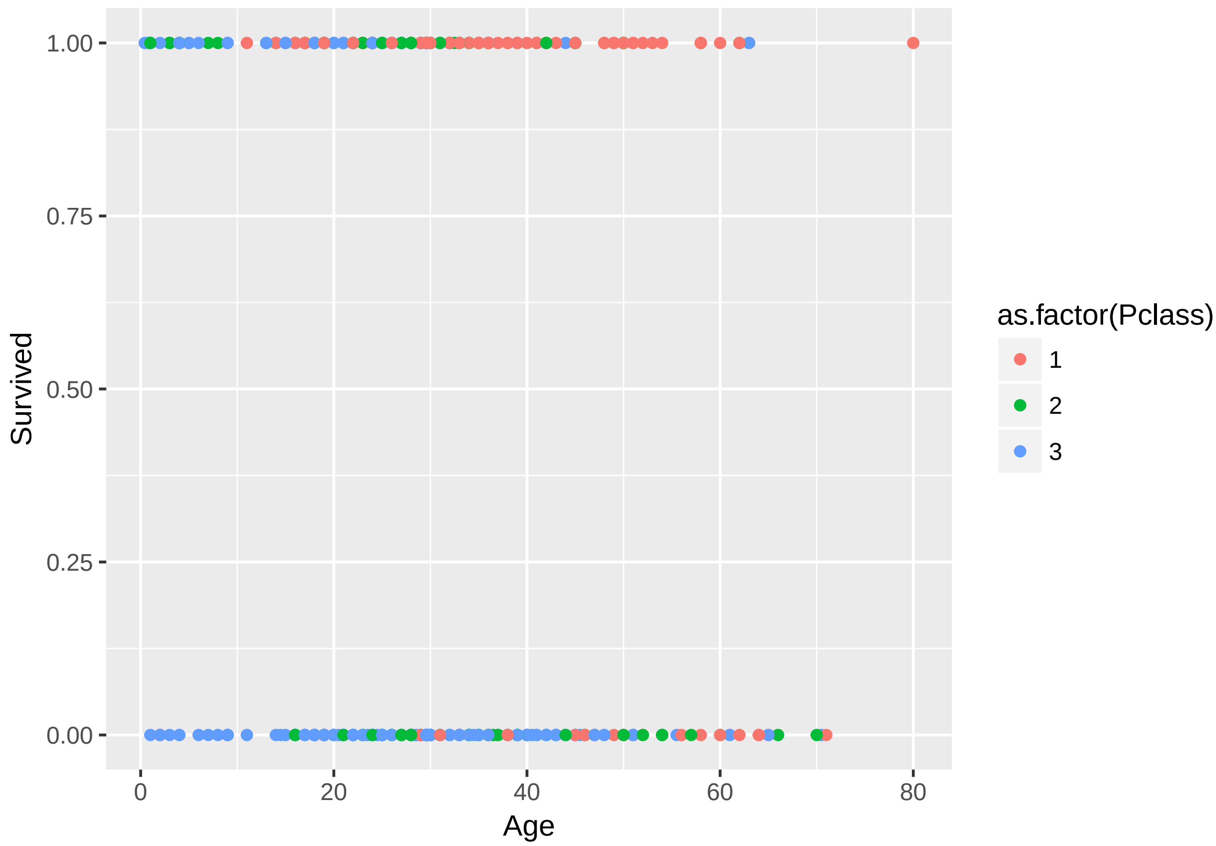
```
## Loading required package: lattice

## Loading required package: ggplot2

## Loading required package: gplots

##
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':
##
##     lowess
```

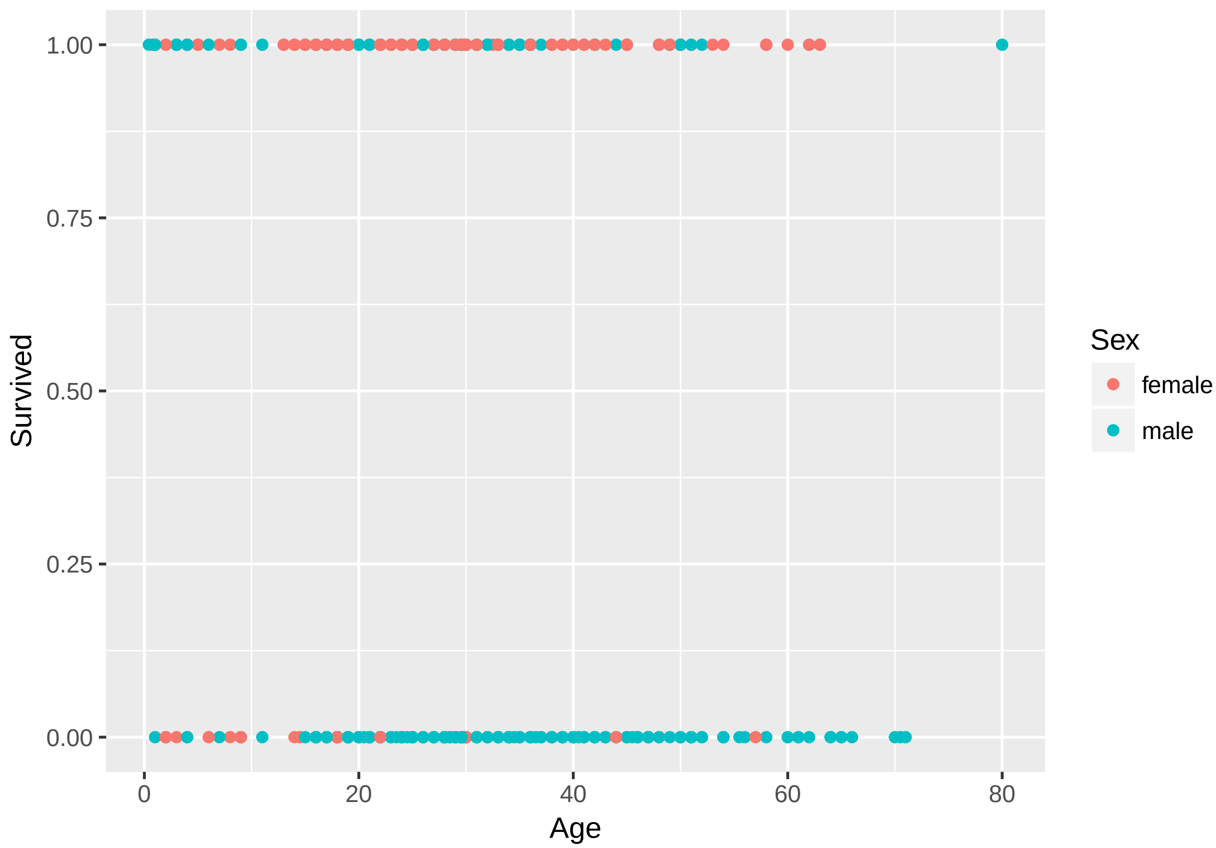
```
train<-read.csv("train.csv",sep=",",header=TRUE,na.strings=c(""))
inTrain<-createDataPartition(y=train$Survived,p=0.7,list=FALSE)
training<-train[inTrain,]
trainingX<-train[inTrain,]
testing<-train[-inTrain,]

#Replacing missing Age values with mean age
training$Age[is.na(training$Age)]<-mean(training$Age,na.rm=T)
testing<-testing[complete.cases(testing$Age),]

head(training)
```

```
##    PassengerId Survived Pclass
## 1            1        0      3
## 2            2        1      1
## 9            9        1      3
## 10          10        1      2
## 11          11        1      3
## 12          12        1      1
##                                                    Name    Sex Age SibSp
## 1                               Braund, Mr. Owen Harris   male  22     1
## 2  Cumings, Mrs. John Bradley (Florence Briggs Thayer) female  38     1
## 9    Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg) female  27     0
## 10                  Nasser, Mrs. Nicholas (Adele Achem) female  14     1
## 11                      Sandstrom, Miss. Marguerite Rut female   4     1
## 12                             Bonnell, Miss. Elizabeth female  58     0
##    Parch    Ticket    Fare Cabin Embarked
## 1      0 A/5 21171  7.2500  <NA>        S
## 2      0  PC 17599 71.2833   C85        C
## 9      2    347742 11.1333  <NA>        S
## 10     0    237736 30.0708  <NA>        C
## 11     1   PP 9549 16.7000    G6        S
## 12     0    113783 26.5500  C103        S
```

```r
#Test Set For Submission
test<-read.csv("test.csv",sep=",",header=TRUE,na.strings=c(""))
test<-test[complete.cases(test),]
```

Exporatory Analysis





You can see that most of the females in the training set survived, and that age was spread pretty evenly for survivors between age 0 and age 55. Also, most of the survivors were from Pclass "1".

Model Fitting with Boosted Logistic Regression

We first try a boosted logistic regression using sex, age, siblings, parents, and class as predictors.

```r
#Fit model BOOSTED LOGISTIC REGRESSION
modelFit<- train(factor(Survived)~Sex+Age+SibSp+Parch+Pclass,data=training,method="LogitBoost")
```

```
## Loading required package: caTools
```

```r
#Make predictions on test set
predictions<-predict(modelFit,newdata=testing,type="raw")

#Confusion Matrix
confusionMatrix(predictions,testing$Survived)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 117  19
##          1  22  67
##
##                Accuracy : 0.8178
##                  95% CI : (0.761, 0.8659)
```

```
##     No Information Rate : 0.6178
##     P-Value [Acc > NIR] : 6.377e-11
##
##                   Kappa : 0.6167
##  Mcnemar's Test P-Value : 0.7548
##
##             Sensitivity : 0.8417
##             Specificity : 0.7791
##          Pos Pred Value : 0.8603
##          Neg Pred Value : 0.7528
##              Prevalence : 0.6178
##          Detection Rate : 0.5200
##    Detection Prevalence : 0.6044
##       Balanced Accuracy : 0.8104
##
##        'Positive' Class : 0
##
```

Because we saw that the majority of survivors were female and in Pclass 1, we will try boosted logistic regression with only sex and Pclass predictors.

```r
#Fit model BOOSTED LOGISTIC REGRESSION
modelFitLB<- train(factor(Survived)~Sex+Pclass,method="LogitBoost",data=training)

#Make predictions on test set
predictionsLB<-predict(modelFitLB,newdata=testing,type="raw")

#Confusion Matrix
confusionMatrix(predictionsLB,testing$Survived)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 98 12
##          1 41 74
##
##                Accuracy : 0.7644
##                  95% CI : (0.7035, 0.8183)
##     No Information Rate : 0.6178
##     P-Value [Acc > NIR] : 2.058e-06
##
##                   Kappa : 0.5313
##  Mcnemar's Test P-Value : 0.00012
##
##             Sensitivity : 0.7050
##             Specificity : 0.8605
##          Pos Pred Value : 0.8909
##          Neg Pred Value : 0.6435
##              Prevalence : 0.6178
##          Detection Rate : 0.4356
##    Detection Prevalence : 0.4889
##       Balanced Accuracy : 0.7828
##
##        'Positive' Class : 0
```

```
##
```

This lowered the accuracy significantly, so we try the same method with only sex as a predictor.

```
#Fit model BOOSTED LOGISTIC REGRESSION
modelFitLB<- train(factor(Survived)~Sex,method="LogitBoost",data=training)

#Make predictions on test set
predictionsLB<-predict(modelFitLB,newdata=testing,type="raw")

#Confusion Matrix
confusionMatrix(predictionsLB,testing$Survived)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 123  23
##          1  16  63
##
##                Accuracy : 0.8267
##                  95% CI : (0.7708, 0.8737)
##     No Information Rate : 0.6178
##     P-Value [Acc > NIR] : 7.696e-12
##
##                   Kappa : 0.6272
##  Mcnemar's Test P-Value : 0.3367
##
##             Sensitivity : 0.8849
##             Specificity : 0.7326
##          Pos Pred Value : 0.8425
##          Neg Pred Value : 0.7975
##              Prevalence : 0.6178
##          Detection Rate : 0.5467
##    Detection Prevalence : 0.6489
##       Balanced Accuracy : 0.8087
##
##        'Positive' Class : 0
##
```

Surprisingly, this increased the accuracy a bit, and we can keep this is mind for later.

Model Fitting with Regularized Logistic Regression

Next, we try regularized logistic regression.

```
##REGULARIZED LOGISTIC REGRESSION
trainingGLM<-training
trainingGLM$Survived<-as.factor(trainingGLM$Survived)
modelFitGLM<- train(Survived~Sex+Age+SibSp+Parch+Pclass,method="glmnet",data=trainingGLM)
```

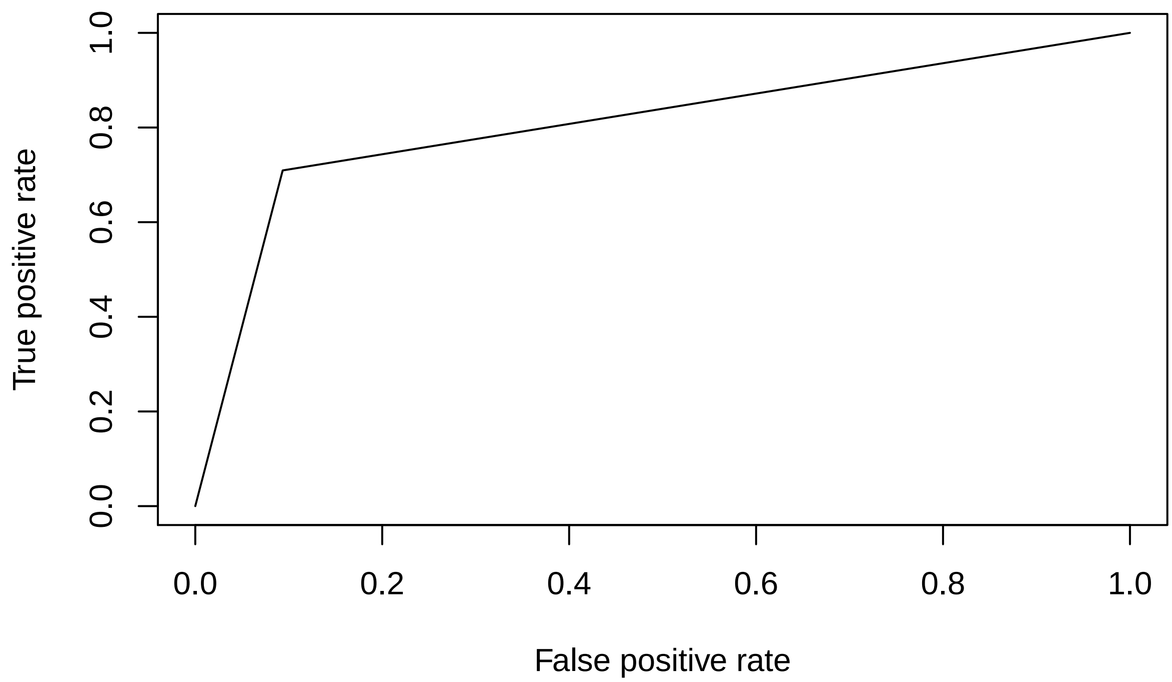
```
## Loading required package: glmnet

## Loading required package: Matrix

## Loading required package: foreach
```

```
## Loaded glmnet 2.0-10
```

```r
#Make predictions on test set
predictionsGLM<-predict(modelFitGLM,newdata=testing,type="raw")

#Confusion Matrix
testingGLM<-testing
testingGLM$Survived<-as.factor(testingGLM$Survived)
confusionMatrix(predictionsGLM,testingGLM$Survived)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 126  25
##          1  13  61
##
##                Accuracy : 0.8311
##                  95% CI : (0.7756, 0.8776)
##     No Information Rate : 0.6178
##     P-Value [Acc > NIR] : 2.556e-12
##
##                   Kappa : 0.6326
##  Mcnemar's Test P-Value : 0.07435
##
##             Sensitivity : 0.9065
##             Specificity : 0.7093
##          Pos Pred Value : 0.8344
##          Neg Pred Value : 0.8243
##              Prevalence : 0.6178
##          Detection Rate : 0.5600
##    Detection Prevalence : 0.6711
##       Balanced Accuracy : 0.8079
##
##        'Positive' Class : 0
##
```

```r
###ROC Curve
predGLM<-ifelse(predictionsGLM=="1",1,0)
testPrediction<-prediction(predGLM,as.numeric(testingGLM$Survived))
perf <- performance(testPrediction, "tpr", "fpr")
plot(perf)
```



The optimal true positive rate is 1.00, meaning that all positive survival outcomes are predicted as positive. The optimal false positive rate is 0.00, which means that out of all the negative survival outcomes, none were predicted as positive. Thus, in our ROC curve, we are looking for values close to the upper left corner. There is an optimal point on the plot, located where the true positive rate (the "Sensitivity") is about 0.72.

Next, we try using cross-validation on a regularized logistic regression model with our original set of features/predictors. This partitions the training set into 10 subsamples, and randomly chooses one of them as the test set and the other 9 as training sets, taking the average of the results of each of the 10 model runs.

```
trainGCV<-train
trainGCV$Survived<-as.factor(trainGCV$Survived)
trainGCV$Age[is.na(trainGCV$Age)]<-mean(trainGCV$Age,na.rm=T)
control<- trainControl(method="cv", number=10, savePredictions = TRUE)


modelFitGCV<- train(Survived~Sex+Age+SibSp+Parch+Pclass,data=trainGCV, trControl=control,method="glmnet

#Make predictions on test set
predictionsGCV<-predict(modelFitGCV,newdata=testing,type="raw")

#Confusion Matrix
confusionMatrix(predictionsGCV,testing$Survived)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
```

```
##          0 124  25
##          1  15  61
##
##                Accuracy : 0.8222
##                  95% CI : (0.7659, 0.8699)
##     No Information Rate : 0.6178
##     P-Value [Acc > NIR] : 2.249e-11
##
##                   Kappa : 0.615
##  Mcnemar's Test P-Value : 0.1547
##
##             Sensitivity : 0.8921
##             Specificity : 0.7093
##          Pos Pred Value : 0.8322
##          Neg Pred Value : 0.8026
##              Prevalence : 0.6178
##          Detection Rate : 0.5511
##    Detection Prevalence : 0.6622
##       Balanced Accuracy : 0.8007
##
##        'Positive' Class : 0
##
```

Model Fitting With Extreme Gradient Boosting

Finally, we try XGBOOST, a tree ensemble.

```r
#remove columns with values of class "character", transform dataframe into matrix
trainingXG<-trainingX[c(1:3,6:8,10)]
trainingXG<-as.matrix(trainingXG)
label<-trainingXG[,2]
dat<-trainingXG[,-2]

#Fit xgboost model to training set, using "Survived" variable as outcome
modelFitXG <- xgboost(data = dat, label=label, nrounds=5,objective = "binary:logistic")
```

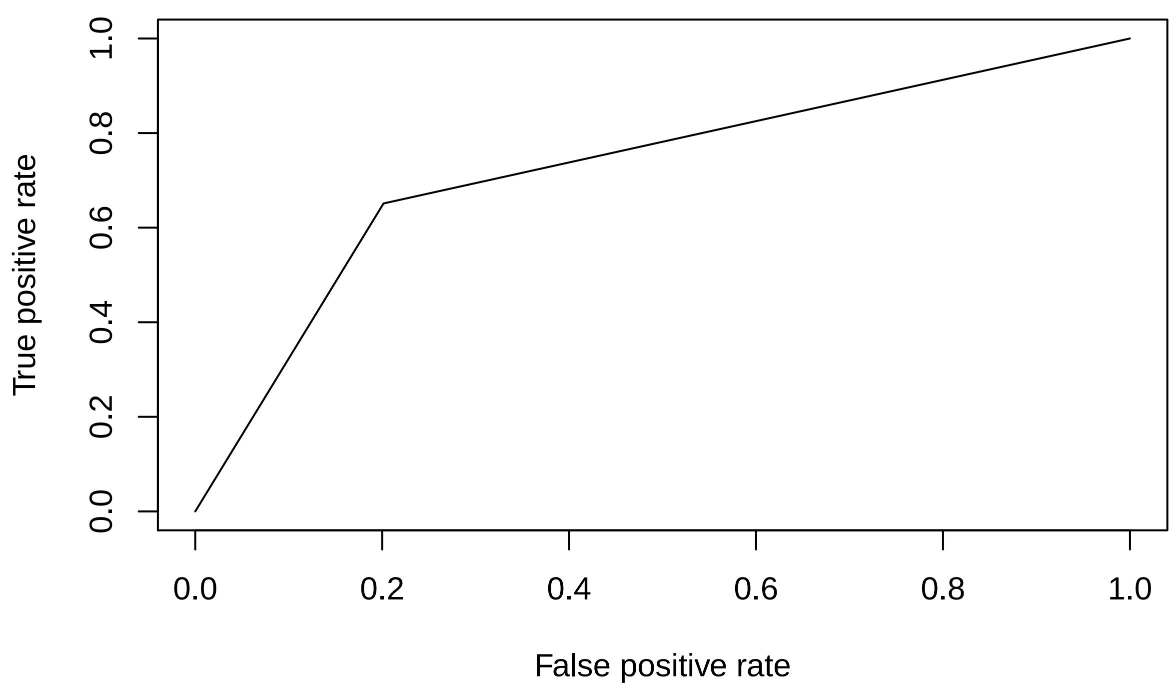
```
## [1]  train-error:0.189103
## [2]  train-error:0.160256
## [3]  train-error:0.144231
## [4]  train-error:0.144231
## [5]  train-error:0.133013
```

```r
#Remove "Survived" variable from testing set
testingX<-as.matrix(testing[c(1,3,6:8,10)])

predictions<-predict(modelFitXG,newdata=testingX)
modelPred <- as.numeric(predictions > 0.5)

confusionMatrix(modelPred,testing$Survived)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 111  30
```

```
##          1  28  56
##
##                Accuracy : 0.7422
##                  95% CI : (0.6799, 0.7981)
##     No Information Rate : 0.6178
##     P-Value [Acc > NIR] : 5.443e-05
##
##                   Kappa : 0.4517
##  Mcnemar's Test P-Value : 0.8955
##
##             Sensitivity : 0.7986
##             Specificity : 0.6512
##          Pos Pred Value : 0.7872
##          Neg Pred Value : 0.6667
##              Prevalence : 0.6178
##          Detection Rate : 0.4933
##    Detection Prevalence : 0.6267
##       Balanced Accuracy : 0.7249
##
##        'Positive' Class : 0
##
```

```r
#ROC curve (true positive rate vs. false positive rate)
testPrediction<-prediction(modelPred,as.numeric(testing$Survived))
perf <- performance(testPrediction, "tpr", "fpr")
plot(perf)
```



Our optimal Sensitivity for this model is around 0.65.

Conclusions

Boosted logistic regression, which models with logistic regression several times and determines an average of the several runs, performed very well on the data. We saw that removing all predictors except Pclass and Sex actually lowered the accuracy slightly, while removing all predictors except Sex increased the accuracy. Gender proved to be a strong predictor for survival.

Regularized logistic regression, which prevents overfitting by limiting and expanding the impact of certain predictors, gave us the highest accuracy of any method, at around 83%. Strangely, when cross validation was applied using this same model, accuracy dropped slightly*** FIGURE THIS OUT.

Finally, Xgboost is a popular, high speed and highly efficient algorithm for regression, but proved to have lower accuracy than our logistic regression models when applied to a binary classification problem.