Boston Marathon Finish Time Predictions

Harvard Stats E139 Fall 2015

Nathaniel Burbank
Pooja Singh
David Wihl

December 21, 2015

Abstract

We want to build a reasonably accurate model for predicting Boston Marathon finish times based on gender, age on 5k split times. We will build a baseline using untransformed linear regression. Then, we will analyze the data and perform appropriate transformations. After transforming the data, we will cluster the data into different subgroups using unsupervised learning algorithms. Finally, we will run different regression algorithms on the transformed and subsetted data to see our improvements over the baseline.

Baseline

We obtained the following data: *fill in here*

First let's read in the data and calculate finish times:

Step 1: Baseline Regression

Our baseline mean error is *whatever*, or approximately 15 minutes.

Step 2: Examination of the data and transformations.

Some preliminary EDA of the data:

The data appears somewhat right skewed. Let's try a log transform of the predicted variable:

That seems better.

Let's try a multiple regression on the log transformed data:

Our transformed mean error is

Let's plot the residuals vs predicted

This appears to be spreading out and trending up.

Let's try a polynomial transformation on the 5k time to see if that improves things:

Step 3: Clustering the Data into Subgroups

Let's try subsetting the data set in subgroups using an unsupervised learning algorithm.

Note: as you can see, the unsupervised learning found groupings that would have difficult if not impossible to us to find on our own

Conclusion

By transforming and subsetting the data we were able to bring down the size of the residuals significantly and improve the accuracy of the model…

Appendix A - Code

The following is the code used to prepare this report:

```r
create_folds = function (df, k){
  # Perform k-Fold Cross Validation and return average score
  # Inspired by Scikit-Learn's cross_val_score
  if (missing(k)) {
    k = 5
  }

  train1_start = train1_end = train2_start = train2_end = test_start = test_end = c()
  testsize = nrow(df) / k
#  cat("train1 start \t train1_end \t test_start \t test_end \t train2_start \t train2_end \n")
  for (i in 1:k){
    if (i < k){
      train1_start[i] = 1
      train1_end[i] = floor((k-i) * testsize)
    } else {
      train1_start[i] = 0
      train1_end[i] = 0
    }
    test_start[i] = max(ceiling((k-i) * testsize),1)
    test_end[i]   = floor((k-i+1) * testsize)
    train2_start[i] = 0
    train2_end[i] = 0
    if (i > 1){
      train2_start[i] = ceiling((k-i+1) * testsize)
      train2_end[i] = nrow(df)
    }
#    cat(train1_start, "\t\t",train1_end,
#        "\t\t", test_start, "\t\t", test_end,
#        "\t\t",train2_start, "\t\t", train2_end,"\n")

  }
  return (data.frame(train1_start, train1_end, train2_start, train2_end, test_start, test_end))
}


k = 10 # we'll do 10-fold cross validation

# First let's read in the data and calculate finish times:
dfm <- read.csv("Previous Boston Marathon study/BAA data.txt",header=T,sep=" ")
times = as.matrix(dfm[,7:15], ncol=9)
dfm$totaltime = rowSums(times)
dfm<- dfm[c("totaltime","Age","Gender1F2M","K0.5")] # keep only columns we need
dfm = dfm[!is.na(dfm$totaltime), ]  # eliminate rows with no finish times
```

```r
#dfm = dfm[sample(nrow(dfm)),]  # in case the data is sorted, randomize the order

# Create a set of k-fold sets for cross-validation
folds = create_folds(dfm,k)
score = c()

# Baseline regression
base.mod = lm(totaltime~.,data=dfm)
summary(base.mod)

# get baseline score
for (i in 1:k) {
  train = dfm[c(folds[i,"train1_start"]:folds[i,"train1_end"],folds[i,"train2_start"]:folds[i,"train2_e
  test = dfm[folds[i,"test_start"]:folds[i,"test_end"],]

  model = lm(totaltime~.,data=train)
  score[i] = sum((test$totaltime - predict(model,new=test))^2) / as.numeric(nrow(test))
}
cat ("Baseline mean score is: ", mean(score),"\n")

#par(mfrow=c(2,2))
#plot(model, pch=23 ,bg="chocolate1",cex=.8)


#y.hat = predict(base.mod)
#xydata = data.frame(x=y.hat, y=resid(base.mod))
#xydata = xydata[sample(1:nrow(xydata), 5000, replace=FALSE),]
#plot(xydata$x,xydata$y,ylim=c(-100,100), xlab="Predicted", ylab="Residuals")

## Step 2: Examination of the data and transformations.
par(mfrow=c(1,2))
hist(dfm$totaltime,breaks=50, main="Boston Marathon Finish Time (min) Distribution for '10, '11, '13")

#The data appears somewhat right skewed. Let's try a log transform of the predicted variable:
hist(log(dfm$totaltime),breaks=50, main="Boston Marathon Finish Time (log-min) Distribution for '10, '1

# That seems better.

# Let's try a multiple regression on the log transformed data:
tx.mod = lm(log(totaltime)~., data=dfm)
summary(tx.mod)

score = c()
for (i in 1:k) {
  train = dfm[c(folds[i,"train1_start"]:folds[i,"train1_end"],folds[i,"train2_start"]:folds[i,"train2_e
  test = dfm[folds[i,"test_start"]:folds[i,"test_end"],]

  model = lm(log(totaltime)~.,data=train)
  score[i] = sum((test$totaltime - exp(predict(model,new=test)))^2) / as.numeric(nrow(test))
}
cat ("Log transformed mean score is: ", mean(score),"\n")
```

```
#plot(dfms$totaltime,model.resid,ylim=c(-100,100))
#y.hat = predict(tx.mod)
#xydata = data.frame(x=y.hat, y=resid(tx.mod))
#xydata = xydata[sample(1:nrow(xydata), 5000, replace=FALSE),]
#plot(xydata$x,xydata$y,ylim=c(-100,100), xlab="Predicted", ylab="Residuals")
#par(mfrow=c(2,2))
#plot(tx.mod, pch=23 ,bg="chocolate1",cex=.8)

#bestresidsum = 9e9
#bestpoly = 0
#for (i in seq(from=1.0, to= 2.5, by=0.1)){
#  dfms$k5xform = dfms$K0.5 ^ i
#  model = lm(totaltime~Age+Gender1F2M+k5xform, data=dfms)
#  model.residsum = sum(resid(model)^2)
# if (model.residsum < bestresidsum) {
#    bestresid = resid(model)
#    bestresidsum = model.residsum
#    bestpoly = i
#  }
#}

#cat("total error went from ",sum(model.resid^2), " (untransformed) to ", bestresidsum," (transformed)\
#plot(dfms$totaltime,bestresid,ylim=c(-100,100))
#xydata = data.frame(x=dfms$totaltime, y=bestresid)
#xydata = xydata[sample(1:nrow(xydata), 5000, replace=FALSE),]
#plot(xydata$x,xydata$y,ylim=c(-100,100), xlab="total time(min)", ylab="Residuals")

# Code inspired from https://datayo.wordpress.com/2015/05/06/using-k-means-to-cluster-wine-dataset/

# Warning: using NbClust never finished even when I tried 2-4 instead of 2-15 clusters.
# install.packages("NbClust")
#library(NbClust)
#nc <- NbClust(data=dfm2,
#               min.nc=2, max.nc=4,
#               method="kmeans")
#barplot(table(nc$Best.n[1,]),
#        xlab="Numer of Clusters",
#        ylab="Number of Criteria",
#        main="Number of Clusters Chosen by 26 Criteria")


for (num_clusters in 4:20) {
  fit.km <- kmeans(dfm, num_clusters)
  cat ("Number of clusters:",num_clusters,"\n")
  overall = c()
  for (i in 1:num_clusters){
    df = dfm[ fit.km$cluster == i, ]
    folds = create_folds(df,k)
    score = c()
    for (j in 1:k) {
      train = df[c(folds[j,"train1_start"]:folds[j,"train1_end"],folds[j,"train2_start"]:folds[j,"train
      test = df[folds[j,"test_start"]:folds[j,"test_end"],]
```

```r
      model = lm(totaltime~.,data=train)
      score[j] = sum((test$totaltime - predict(model,new=test))^2) / as.numeric(nrow(test))
    }
    cat("For cluster ",i," the mean error is ",mean(score), "\n")
    overall[num_clusters-3] = mean(score)
  }
  cat ("Mean error rate overall:", mean(overall),"\n\n")
}
# Warning: this takes awhile to run because it includes all data points
# install.packages("fpc")
#library(fpc)
#plotcluster(dfm2, fit.km$cluster)
```

Appendix B - References

Place R Code in Appendix