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.

Objectives and Motivation

Boston Marathon Runners are very concerned about their finish times, moreso than many other races. Boston has a limited course size, constrainted by the 30' width of road in Hopkington, MA (source). Due to the limitation, Boston has to be very selective in the choice of runners and has put restrictive qualifying times in order to run the marathon (source). In order to run the Boston Marathon, not only does an applicant had to have run a previous qualifying marathon, the candidate must also had to finish much faster than the typical marathoner. While other races like New York and Chicago have increased their participation size significantly in recent years, Boston is currently and the foreseeable future restricted to approximately 27,000 runners.

Mean Finish Time	Men	Women
Worldwide(*)	253	282
Boston(**)	216	235

(*) The Guardian

(**) See Data Sources, below

Boston is also known as a hilly course so a candidate who may have run a fast race on a flat course would be challenged by New England's rolling hills, especally between miles 16-21, which is among the most difficult of the course. The peak of course is affectionately known as Heartbreak Hill.

For all these reasons plus the long history of the race, the Boston Marathon is a very prestigious event in the running community. Forecasting an accurate finish time in the race is a key motivator for runners to improve their time during the hundreds of miles of training a marathon requires.

Data Sources

We obtained our data from two sources. First, we downloaded a previously assembled dataset from a group of researchers University of North Carolina-Chapel Hill who analyzed historic runner performance with the goal of predicting the anticpated finish times for the approximately 5000 runners who were unable to complete the 2013 Boston Marathon due the bombings at the finish line. This data sat contained that split and finish

times for 69923 runners in the 2010, 2011, and 2013 Boston Marathons. (The 2012 Boston marathon was intentionally excluded from their dataset because it was unusually hot that year.) We also obtained finish and split times for 78042 runners who completed the Chicago Marathon in 2014 and 2015. We obtained these results directly from the Chicago Marathon's results website using a custom python scraper.

Methodology

We are attempting to build the most accurate prediction model possible on the smallest amount of interesting data. We are not attempting to infer or analyze the data extensively. Our method consists of the following steps:

1. Create a baseline regression using OLS
2. Analyze the data and perform appropriate transformations
3. Cluster the data into subgroups using unsupervised learning, such as k-means
4. Verify the improved accuracy of our predictions.

Throughout we will be using 10-fold cross validation testing with sum of squares errors as the metric to improve.

Step 1: Baseline Regression

We ran a simple OLS regression using all the Boston data. The predictor variables were age, gender (as factor), and 5k split time.

Using k-Fold cross validation, and taking the sum of squares error for each fold resulted in a baseline mean score of 240 or approximately 15 minutes. While $\pm$ 15 minutes over the course of a four hour race may not seem like a large error, it makes a significant difference in terms of ability to qualify for Boston.

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:

```
# Step 1 - baseline regression
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
  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)
    }
  }
  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$totaltime = rowSums(times)
dfm<- dfm[c("totaltime","Age","Gender1F2M","K0.5")] # keep only columns we need
dfm$Gender1F2M = as.factor(dfm$Gender1F2M) # make gender into a factor
dfm = dfm[!is.na(dfm$totaltime), ]  # eliminate rows with no finish times
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_er
  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")

# 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

# 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_er
  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")

#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")
# Step 3: Clustering the Data into Subgroups
#Let's try subsetting the data set in subgroups using an unsupervised learning algorithm.
# Code inspired from https://datayo.wordpress.com/2015/05/06/using-k-means-to-cluster-wine-dataset/

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"],]

      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")
}
```

Appendix B - References

Place R Code in Appendix