Boston Marathon Finish Time Predictions

Harvard Stats E139 Fall 2015

Nathaniel Burbank
Pooja Singh
David Wihl

December 21, 2015

Abstract

We wanted to build a reasonably accurate model for predicting Boston Marathon finish times based on a runner's gender, age, and the number of minutes it took for the runner to complete the first five kilometers of the race (5k split time). We first built a baseline model using untransformed linear regression. Then, we analyzed the data and performed appropriate transformations. After transforming the data, we clustered the race results into several different subgroups using an unsupervised learning algorithms. Using these different clusters of race results, we generated a new regression model for each subset. Finally, we compared the accuracy of predictions made via our clustered regression model with our baseline model.

Objectives and Motivation

Runners who participate in the Boston Marathon are very concerned about their finish times. The reason is simple. The Boston Athletic Association (BAA), which organizes the race each April, limits the field of participants to runners who are capable of completing a marathon in a very fast time. Starting in the 1970s, when the BAA was concerned that the race was becoming unwieldy as it had grown to include more than 1000 runners for the first time (source), the BAA established a set of qualifying times that runners had to meet(source) in a previously timed marathon in order to be granted acceptance to the Boston race. With marathoning having grown tremendously in popularity over the following four decades, today race organizers are physically constrained by the 30' width of the road in Hopkington, MA (source) where the race begins. While other races like New York and Chicago have increased their participation sizes significantly in recent years, Boston is currently, and will be for the foreseeable future, restricted to a size of approximately 27,000 runners. Thus, while the Boston Marathon course is considered one the harder marathon courses, the runners who participate are faster than average. The mean marathon finish time for Boston runners is significantly faster than global averages:

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

(*) The Guardian

(**) See Data Sources, below

Boston is also known as a hilly course. Race participants who have qualified for the race in marathons featuring flat courses are sometimes challenged by New England's rolling hills, especially those between miles 16-21, which are among the most difficult of the course. The peak of these hills is affectionately known as Heartbreak Hill.

For all of 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 at the University of North Carolina-Chapel Hill who analyzed historic runner performance with the goal of predicting the anticipated finish times for the approximately 5000 runners who were unable to complete the 2013 Boston Marathon due to the bombings at the finish line. This data set contained the split and finish times for 64106 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-built 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. Data cleansing
2. Create a baseline regression using ordinary least squares (OLS) regression
3. Analyze the data and perform appropriate transformations
4. Cluster the data into subgroups using unsupervised learning, such as k-means
5. Verify the improved accuracy of our predictions.

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

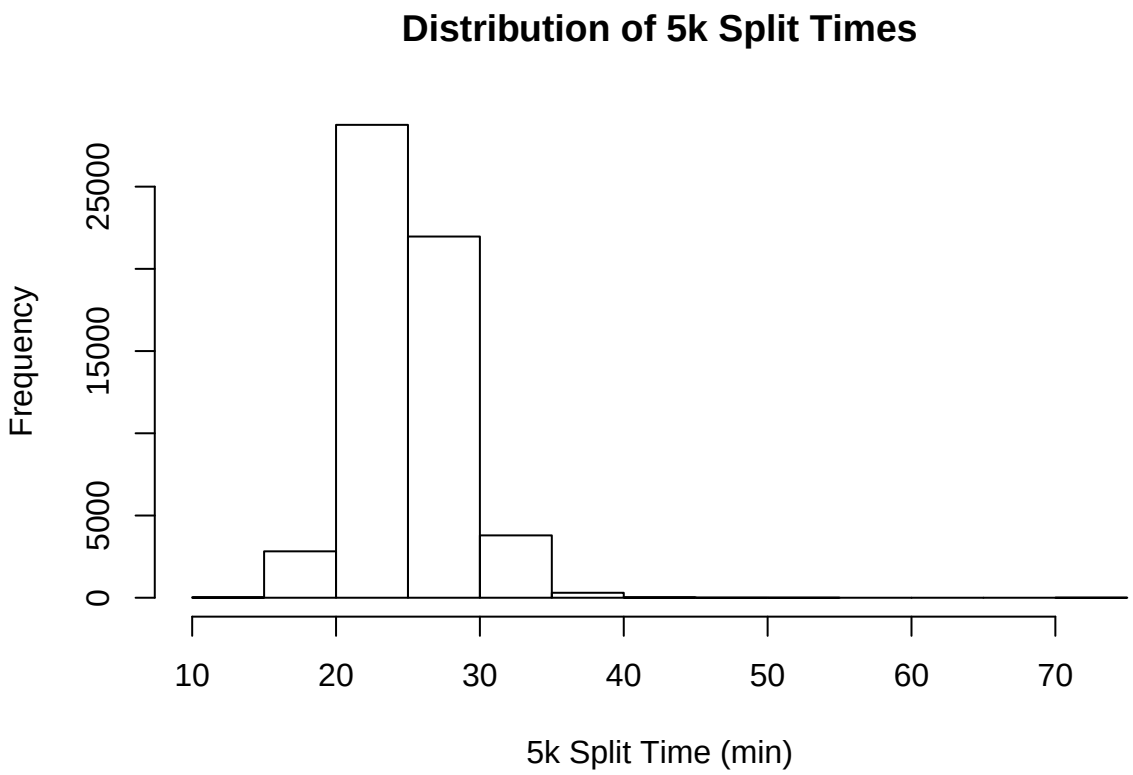
Data Cleansing

Even though we downloaded our primary data from another study where the researchers had already performed some scrubbing, we needed to perform additional data cleansing to format the data for our needs. As our response variable (finish times) was calculated as a sum of a runner's individual split times, we needed to eliminate, rather than impute, any rows that were missing any split times as their inclusion might have tainted the predictive model. Since we were using k-Fold cross validation, we randomized the order of the data to ensure that each fold had a reasonable sample size. Finally, gender had a value of $\{1, 2\}$ which we had to convert into a factor in R.

Set aside a validation set.

In order to maintain the ability confirm the accuracy of our final model on unseen data, we set aside 10% of our Boston dataset as a validation set.

Removing Outliers



As the histogram reveals, there were several outliers in the 5k split times. These could be due to timing errors, or an injury or accident at the start of the race. Due to the sensitivity of OLS regression, we elected to remove the outliers that were more than three standard deviations (σ) greater than the mean 5k split time. In total there were 370 rows like this out of 57696 (0.6%).

Age was in the range [18,81]. (The minimum age to run the Boston Marathon is 18). No outliers were removed. Males represented 58.8% of total runners, which seemed reasonable.

Baseline Regression - 5K Split

We ran a simple OLS regression model using the training portion of our Boston data. The predictor variables were age, gender (as a factor), and 5k split time. The response variable was the finish time in minutes.

The baseline model was quite good, but we felt we could do better.

```
##
## Call:
## lm(formula = totaltime ~ . - HalfMar, data = dfm)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -79.280  -9.213  -2.610   6.410 187.521
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -24.279752   0.568517 -42.707   <2e-16 ***
## Age           0.059676   0.006526   9.145   <2e-16 ***
```

```
## Gender1F2M2   3.161281   0.145559  21.718   <2e-16 ***
## K0.5          9.821244   0.022675 433.123   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.15 on 57322 degrees of freedom
## Multiple R-squared:  0.8041, Adjusted R-squared:  0.8041
## F-statistic: 7.844e+04 on 3 and 57322 DF,  p-value: < 2.2e-16
```

Using k-Fold cross validation, and taking the sum of squares error for each fold resulted in a baseline mean score of 229 or approximately 15 minutes. While $\pm$ 15 minutes over the course of a four hour race may not seem like a large error, $\pm$ 15 minutes could be the difference between qualifying and not qualifying for a competitive race such as the Boston Marathon.

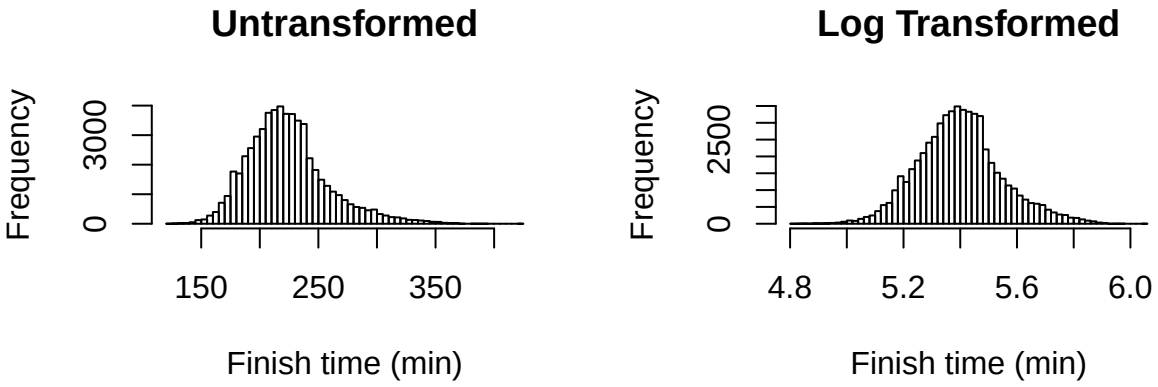
Baseline Regression - Half Marathon Split

As a comparison, we also ran a regression using half marathon instead of 5k split time. The sum of squares error for each fold resulted in a baseline mean score of 113 or approximately 10 minutes, a significant improvement. R^2 went from 0.8041308 to 0.902876. Adjusted R^2 improved from 0.8041205 to 0.9028709.

This was an expected result as closer we are to finish line, the better our predictive analytics will be, hence half marathon time was a better predictor than the 5k split time.

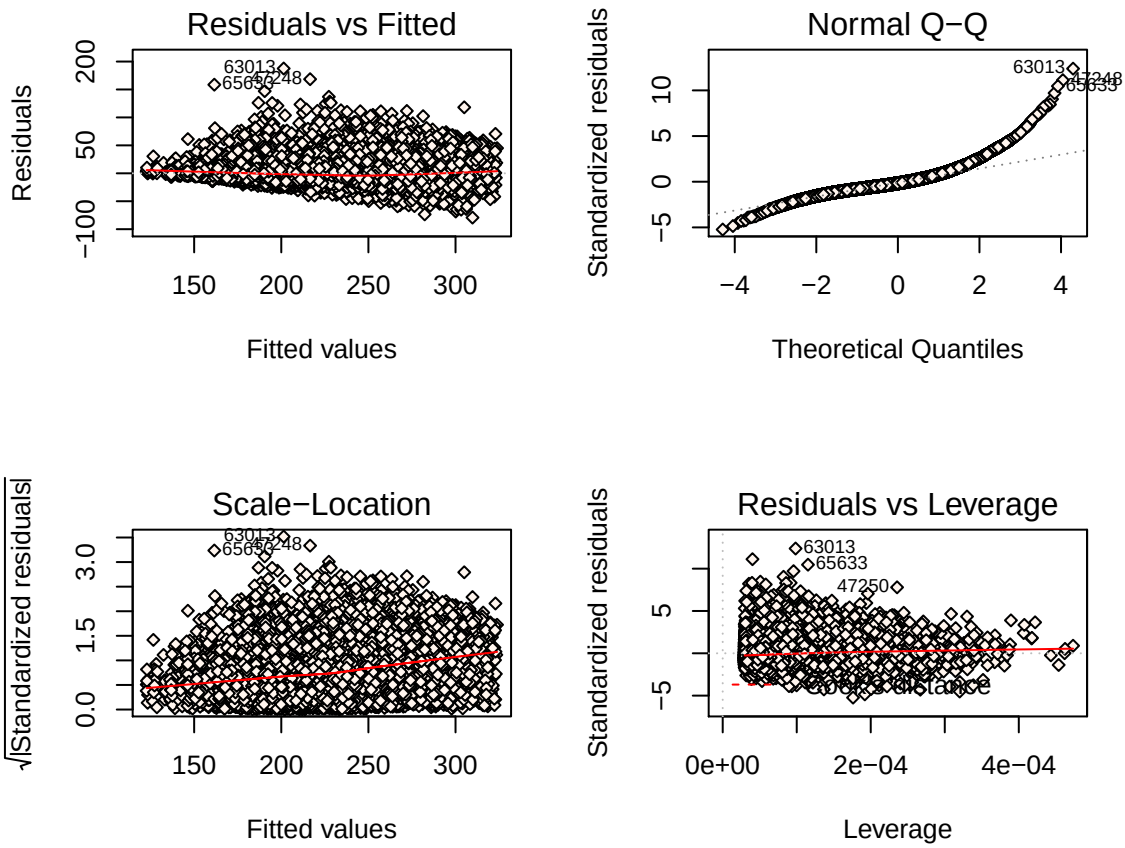
While this may be useful to someone planning to meet the runner at the finish, we felt it would be more challenging to minimize prediction errors using the 5k split time.

Examination of the Data and Transformations



The untransformed data appeared to be slightly right skewed. A log transformation of the response variable would result in a more normal distribution.

We then examined the base model regression diagnostics such as residuals vs. predicted, Cook's Distance, and heteroskedasticity.



Given the right-tail distribution of the QQplot, the fan out of the residuals vs. predicted, and the slight right-skewness of the finish time histogram, we elected to perform a log transformation on the response variable.

Using k-Fold cross validation, and taking the sum of squares error for each fold resulted in a log-transformed mean score of 226 or approximately 15 minutes, a negligible improvement.

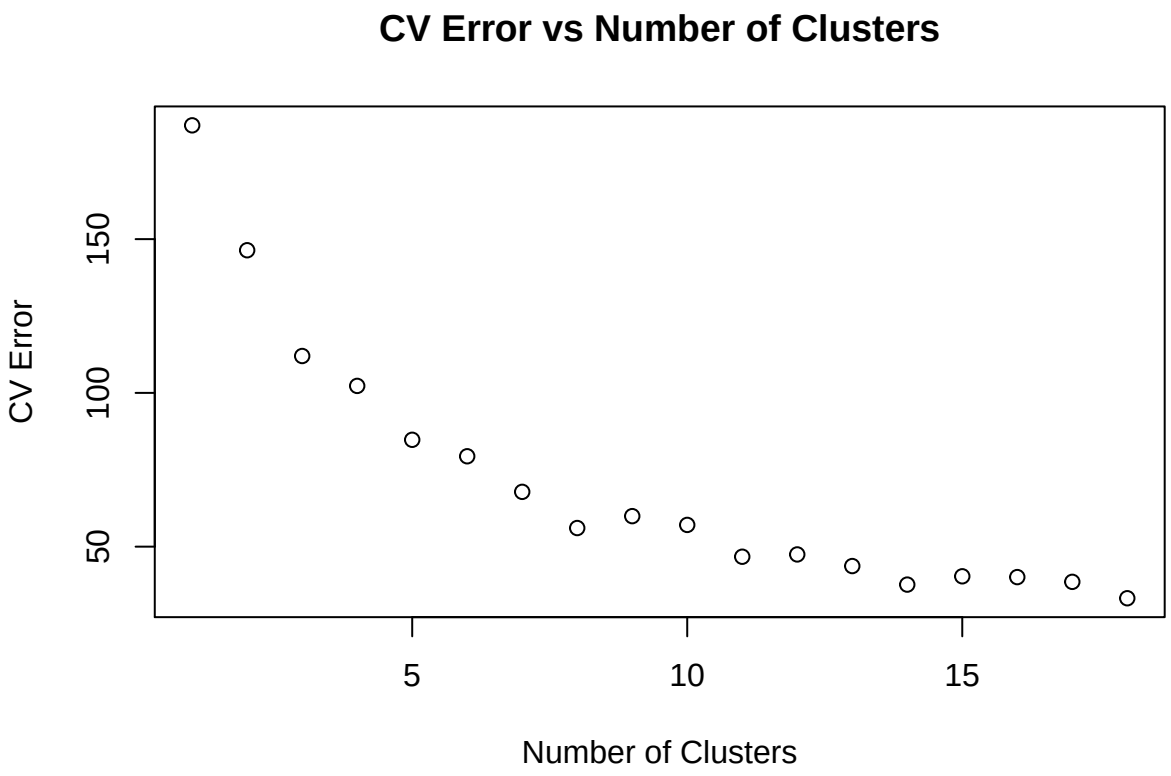
R^2 went from 0.8041308 to 0.8227841, a slight improvement. Adjusted R^2 improved by an equally small amount.

We also attempted a range of polynomial transformations on the 5K times using an exponent in the range [1.1,2.5]. However, this did not improve model's accuracy significantly. We have omitted these results in order to be concise within the report, but they are available on our project GitHub site (See Appendix B).

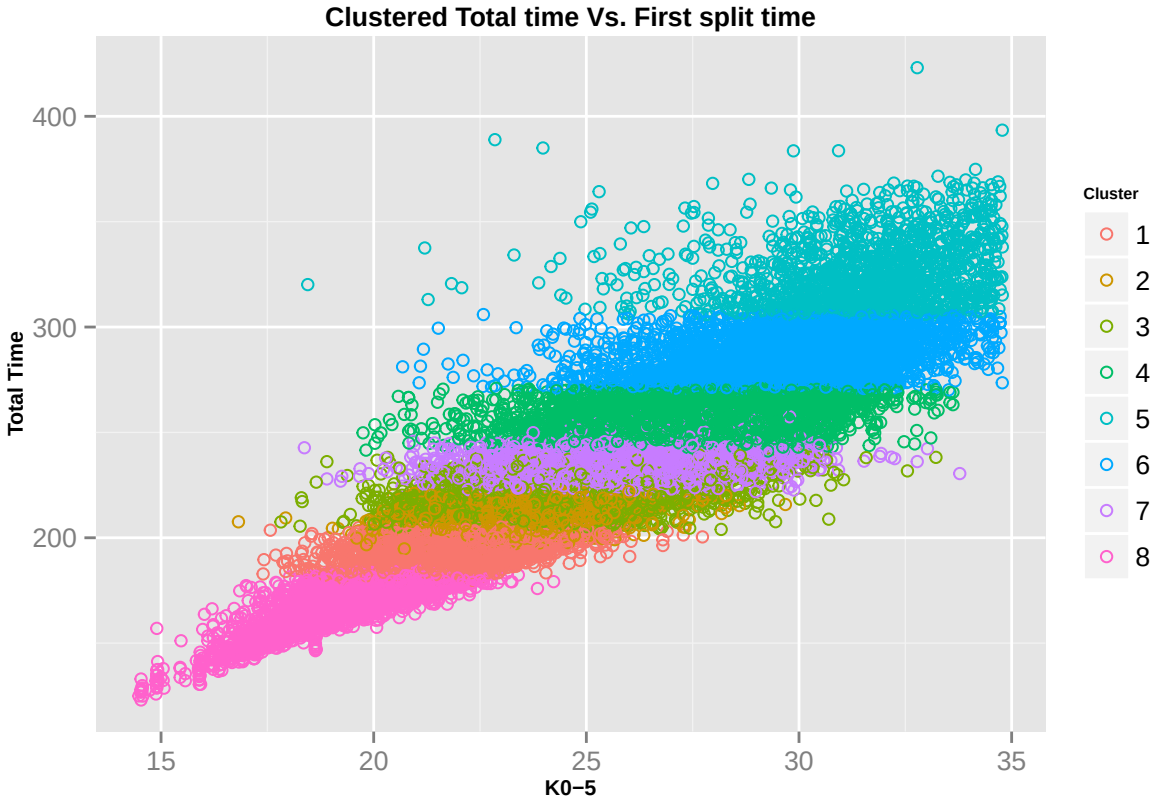
Clustering the Data into Subgroups

Since the transformation yielded only a small improvement, we elected to try clustering the data into subgroups using k-Means. We attempted $k = [3, 20]$.

By clustering, the resulting error rates in the training set decreased significantly. Using $k = 3$, the mean error rate was 186 or $\pm$ 13 min. At $k = 20$, the mean error rate on the training dataset was 33 or $\pm$ 5 min, a significant improvement.



Using the elbow method, we determined that the optimal number of clusters is $k = 8$, which results in a cross validation error of 79 or $\pm$ 8 min on tn the training dataset, which is approximately half the error as compared to unclustered linear regression.

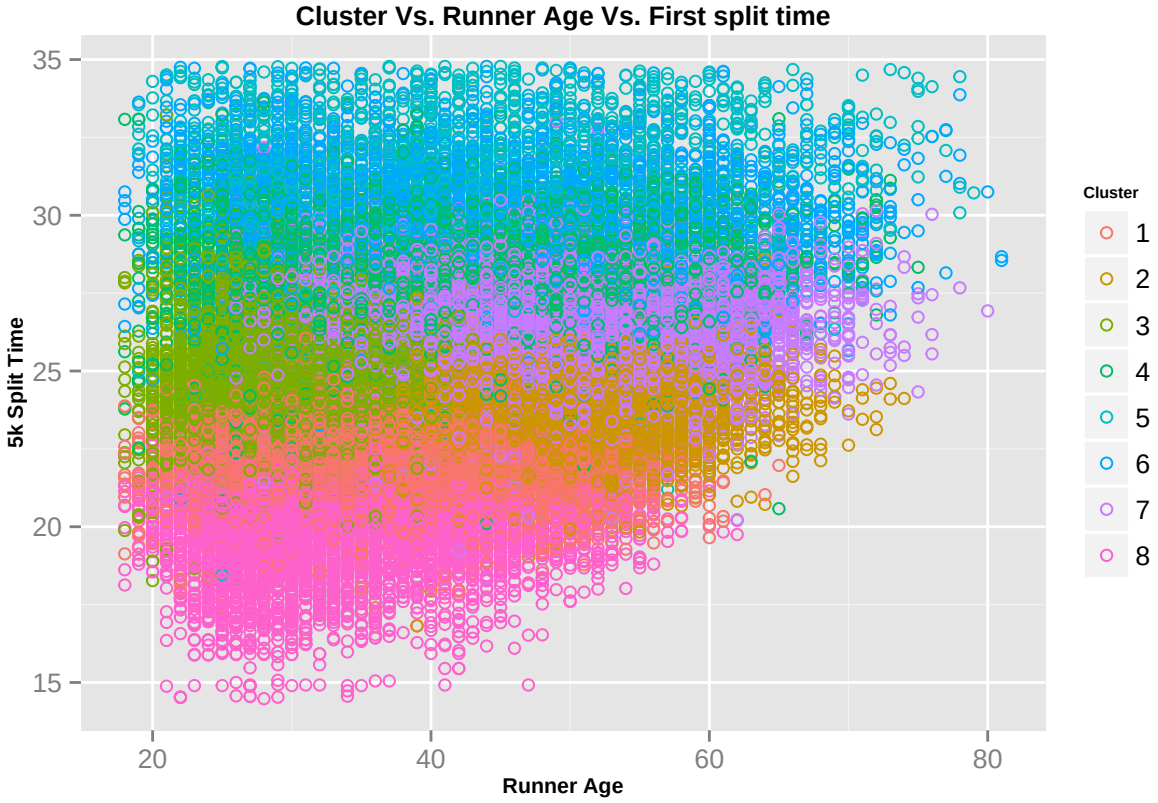


As you can see, the unsupervised learning found groupings, especially in the dense middle of the field that

would have difficult if not impossible to us to find on our own.

In a given race, there are multiple categories of runners. Elite runners are significantly faster of course. Even though the Boston Marathon requires fast qualification times, an allowance of approximately 10% of the field is made for runners who are running for a charity such as Dana-Farber. The charity runners do not need to qualify and are much slower than an average qualified runner.

The number of runners two σ faster than the mean is 375. There are 2650 runners two σ slower than the mean. (This also confirms our earlier right skewness to the data). The difference in slope for fast vs. slow runners is indicative that a single regression model could be improved upon.



Even when we look at the clusters without include our response variable (finish time). as we do in the chart above, at appears there are significant distinctions based on age, gender, and 5k split time that we should be able to classify runners and gain additional predictive power.

Applying the Model to Chicago Data

To evaluate the robustness of our model, we attempted to verify the predictability of the model to a different marathon. The Chicago marathon does not require qualification and is a flatter course.

The SSE using clustering and individual regression models is 48. Using the Boston Marathon baseline linear regression is 24. So this model does not generalize well to a marathon with a different profile.

Conclusion and Next Steps

The following table summarizes our results:

Method	Mean Error Train (min)	Mean Error Validate	Mean Error Chicago
Baseline 5k	15	15	24

Method	Mean Error Train (min)	Mean Error Validate	Mean Error Chicago
Baseline Half Marathon	10	10	15
Log(y) Transform 5k	15	15	27
8-Fold Cluster	8	26	48

By subsetting the data by cluster, we were able to bring down the cross validation error significantly and improve the accuracy of the model on our training data. However, this improvement in accuracy did not carry over to our validation dataset (the 10% of race results we reserved from the Boston dataset) or the sample of data from the Chicago marathons. In fact, the clustering model had the largest error of any of the models we tested on both datasets.

We believe that two distinct forms of overfitting explain this outcome. First, it's important to note that the cluster identifications in our training data were themselves based on groupings that aimed to minimize the Euclidean distances between age, gender, 5k split time, as well as our response variable, finish time. By building OLS regression models on the subset of race results that represented each cluster, and then analyzing the mean error rate across our cluster-specific models on the training data, a derivation of each runner's final finish time leaked into our model as a predictor variable. This lead to an overly optimistic reduction in mean squared error when running cross validations on our training data alone. This would not be a problem if we were then able to reliably classify runners into an appropriate cluster based on our primary predictor variables alone. Unfortunately we were unable to do this with enough accuracy to add real additional predictive power to our model through estimated cluster classification. Even on the training dataset, where, because these were the exact records that our cluster definitions were developed from, we expected to see the highest levels of accuracy, our classifier was only able to correctly categorize 38.6% of runners when we attempt to label them without including their final finish time in the calculation. Thus, the reduction in mean squared error we saw when running cross validations on our training data was impossible to recreate on unseen data where a runner's cluster-classification was not known in advance.

The second way we overfit our data has to do with the race itself. While it does not beat our other models, the clustering model still works significantly better for the Boston validation data set than the Chicago validation set. Our clustering process may have homed in on timing characteristics that are specific to the Boston Marathon and our model does not appear to generalize well to different profile marathons.

Potential next steps of this project include:

- Elaborate the data set to build a more robust model that would apply well to multiple marathons.
- Attempt other clustering models besides k-Means (example).
- Attempt other algorithms of regression, such as Random Forests.
- Using other split times, such as half marathon, predict the probability of a runner not finishing the race.
- Assuming we can improve generalization, make the prediction model available on a public website using R/Shiny.

Appendix A - Code

The following is the code used to prepare this report:

```
# Boston Marathon Finish Time Predictions
# Harvard Stats E139 Fall 2015
# Nathaniel Burbank, Pooja Singh, David Wihl
#
# December 21, 2015

library(ggplot2)

# Common Functions


percent <- function(x, digits = 1, format = "f", ...) {
  paste0(formatC(100 * x, format = format, digits = digits, ...), "%")
}

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 = 10
  }

  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))
}


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

# Read in the data for multiple years
dfm <- read.csv("Previous Boston Marathon study/BAA data.txt",header=T,sep=" ")
```

```
dfm$Age2014 = NULL # remove unneeded column which is mostly NA
ok = complete.cases(dfm)
dfm = dfm[ok,] # remove rows that have any NA values
times = as.matrix(dfm[,7:15], ncol=9)
dfm$totaltime = rowSums(times)
dfm<- dfm[c("totaltime","Age","Gender1F2M","K0.5","HalfMar")] # 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

# Find mean finish time by gender
agg = aggregate(dfm$totaltime, by=list(dfm$Gender1F2M), FUN=mean)[2]
men = as.integer(agg$x[2])
women = as.integer(agg$x[1])
#Set aside 10% of our data as a validation set
indexes = sample(1:nrow(dfm), size=0.1*nrow(dfm))
validate_dfm = dfm[indexes,]
dfm = dfm[-indexes,]

# Remove outliers (5k > 3 sigma from mean)
hist (dfm$K0.5, breaks=15, main="Distribution of 5k Split Times", xlab="5k Split Time (min)")
mean5k = mean(dfm$K0.5)
sd5k = sd(dfm$K0.5)
outliers5k = dfm$K0.5>(mean5k + 3*sd5k)
outliers5ksum = sum(outliers5k)
dfm.rows = nrow(dfm)
dfm = dfm[!outliers5k,]
malepct = table(dfm$Gender1F2M)[2]  / (table(dfm$Gender1F2M)[2] + table(dfm$Gender1F2M)[1])
# Create a set of k-fold sets for cross-validation
folds = create_folds(dfm,k)
score = c()

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

# 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_en
  test = dfm[folds[i,"test_start"]:folds[i,"test_end"],]

  cvmodel = lm(totaltime~.-HalfMar,data=train)
  score[i] = sum((test$totaltime - predict(cvmodel,new=test))^2) / as.numeric(nrow(test))
}
score.mean.5k = mean(score)
summary(base.mod)
# Half Marathon Baseline regression
base.mod.half = lm(totaltime~.-K0.5,data=dfm)

for (i in 1:k) {
  train = dfm[c(folds[i,"train1_start"]:folds[i,"train1_end"],folds[i,"train2_start"]:folds[i,"train2_en
  test = dfm[folds[i,"test_start"]:folds[i,"test_end"],]

  cvmodel = lm(totaltime~.-K0.5,data=train)
```

```
  score[i] = sum((test$totaltime - predict(cvmodel,new=test))^2) / as.numeric(nrow(test))
}
score.mean.half = mean(score)
# Display histograms of response variable untransformed and log transformed
par(mfrow=c(1,2))
hist(dfm$totaltime,breaks=50, main="Untransformed", xlab="Finish time (min)")
hist(log(dfm$totaltime),breaks=50, main="Log Transformed", xlab="Finish time (min)")
par(mfrow=c(2,2))
plot(base.mod, pch=23 ,bg="seashell",cex=.8)
# Try regression on log(response)
dfm$logtotaltime = log(dfm$totaltime)
folds = create_folds(dfm,k)
score = c()

tx.mod = lm(logtotaltime~.-totaltime-HalfMar,data=dfm)

for (i in 1:k) {
  train = dfm[c(folds[i,"train1_start"]:folds[i,"train1_end"],folds[i,"train2_start"]:folds[i,"train2_en
  test = dfm[folds[i,"test_start"]:folds[i,"test_end"],]

  cv.tx.mod = lm(logtotaltime~.-totaltime-HalfMar,data=train)
  score[i] = sum((test$totaltime - exp(predict(cv.tx.mod,new=test)))^2) / as.numeric(nrow(test))
}
score.mean.tx = mean(score)

# Clustering the Data into Subgroups
# Let's try subsetting the data set in subgroups using an unsupervised learning algorithm.

minclusters = 3
maxclusters = 20
overallclustererror = c()
dfm_clus <- dfm[c("Age","Gender1F2M","K0.5","totaltime")] #Ensure that we're clustering on our predicto
for (num_clusters in minclusters:maxclusters) {
  fit.km <- kmeans(dfm_clus, num_clusters)
  meanclustererror = c()
  # Find cv score for each cluster
  for (i in 1:num_clusters){
    df = dfm_clus[ fit.km$cluster == i, ]
    folds = create_folds(df,k)
    foldscore = c()
    for (j in 1:k) {
      train = df[c(folds[j,"train1_start"]:folds[j,"train1_end"],folds[j,"train2_start"]:folds[j,"train1
      test = df[folds[j,"test_start"]:folds[j,"test_end"],]

      model = lm(totaltime~.,data=train)
      foldscore[j] = sum((test$totaltime - predict(model,new=test))^2) / as.numeric(nrow(test))
    }
    meanclustererror[i] = mean(foldscore)
  }
  # store mean error for a given k clusters
  overallclustererror[num_clusters-2] = mean(meanclustererror)
}
plot(overallclustererror, xlab="Number of Clusters", ylab="CV Error", main="CV Error vs Number of Cluste
```

```
# Show the resulting clusters
fit.km = kmeans(dfm_clus, 8)
dfm$cluster = fit.km$cluster
# Create indicator variables for fast and slow runners to plot different slopes
mean.total = mean(dfm$totaltime)
sd.total = sd(dfm$totaltime)
dfm$fastrunner = (dfm$totaltime < (mean.total - (2*sd.total)))
dfm$slowrunner = (dfm$totaltime > (mean.total + (2*sd.total)))
slope.model = lm(totaltime~.,data=dfm)
ggplot(dfm, aes(x = K0.5, y = totaltime, color=factor(cluster))) + geom_point(shape=1) +
  labs(list(title = "Clustered Total time Vs. First split time", x = "K0-5", y = "Total Time", colour="
  theme(legend.title = element_text(size=6, face="bold") , title = element_text(size=8, face="bold"))
# Show the resulting clusters
ggplot(dfm, aes(x = Age, y = K0.5, color=factor(cluster))) + geom_point(shape=1) +
  labs(list(title = "Cluster Vs. Runner Age Vs. First split time", x = "Runner Age", y = "5k Split Time
  theme(legend.title = element_text(size=6, face="bold") , title = element_text(size=8, face="bold"))
# Save the different regression models per cluster
clust.mod = c()
for (i in 1:length(fit.km$size)) {
  df = dfm[which(dfm$cluster==i), ]
  df = df[c("Age","Gender1F2M","K0.5","totaltime")]
  clust.mod[[i]] = lm(totaltime~.,data=df)
}


# Validating model against Chicago Marathon
dfChi14 <- read.csv("ChicagoScraper/Chicago2014Formated.csv",header=T)
dfChi15 <- read.csv("ChicagoScraper/Chicago2015Formated.csv",header=T)
dfChi <- rbind(dfChi14,dfChi15)


dfChi$Gender1F2M  <- as.factor(dfChi$Gender1F2M) # added converter to assign.cluster to handle gender a
Chitimes = as.matrix(dfChi[,7:15], ncol=9)
dfChi$StartHr = NULL
dfChi$StartMin = NULL
dfChi$totaltime = rowSums(Chitimes)
dfChi$logtotaltime = log(dfChi$totaltime)
#Filter out any rows with NAs -- they'll cause headaches later
dfChi = dfChi[!is.na(dfChi$totaltime), ]
dfChi = dfChi[!is.na(dfChi$HalfMar), ]
dfChi = dfChi[!is.na(dfChi$K0.5), ]
dfChi = dfChi[!is.na(dfChi$Age), ]

#Draw a random sample
dfChi = dfChi[sample(nrow(dfChi), 2500), ][c("Age","Gender1F2M","K0.5","totaltime","HalfMar")]
# Remove outliers
mean5k = mean(dfChi$K0.5)
sd5k = sd(dfChi$K0.5)
outliers5k = dfChi$K0.5>(mean5k + 3*sd5k)
dfChi = dfChi[!outliers5k,]

assign.cluster <- function(df, centers) {
  # compute squared Euclidean distance from each sample to each cluster center
```

```
  # There is probably a vectorized way of doing this.
  cols = c("Age","Gender1F2M","K0.5")
  clusters = c()
  # If gender in incoming DF is a factor, convert it to numeric
  if (is.factor(df$Gender1F2M)){
    df$Gender1F2M <- as.numeric(levels(df$Gender1F2M))[df$Gender1F2M]
  }
  for (i in 1:nrow(df)){
    diff = c()
    for (j in 1:nrow(centers) ){
      y = centers[j,cols]
      x = as.matrix(df[i,cols])
      diff[j] = sum((x-y)^2)
    }
    clusters[i] = which.min(diff)
  }
  return (clusters)
}

# Calculates predicted finish time using assigned cluster and cluster-specific regression models
predict.finish <- function(df) {
  newtot = c()
  for (n in 1:nrow(df)) {
    newtot[n] = predict(clust.mod[[df[n,"cluster"]]], newdata=df[n,])
  }
  return(newtot)
}

# Calculate sum of squared errors (SSE) on submitted DF for both baseline model and clustered model
assign.SSE <- function(df){

  y.hat.base = predict(base.mod,new=df)
  y.hat.half = predict(base.mod.half,new=df)
  df$logtotaltime = log(df$totaltime)
  y.hat.log = exp(predict(tx.mod,new=df))

  base.mod.SSE <- as.integer(sqrt(sum((y.hat.base - df$totaltime)^2) / nrow(df)))
  log.mod.SSE <- as.integer(sqrt(sum((y.hat.log - df$totaltime)^2) / nrow(df)))
  half.mod.SSE <- as.integer(sqrt(sum((y.hat.half - df$totaltime)^2) / nrow(df)))
  clus.mod.SSE <- as.integer(sqrt(sum((predict.finish(df) - df$totaltime)^2) / nrow(df)))

  return (list("base.mod" = base.mod.SSE, "log.mod"=log.mod.SSE, "half.mod" = half.mod.SSE ,"clus.mod" =
}

# Test on Chicago sample
# find which cluster would be most appropriate
# dfChi$Cluster = predict(sum,dfChi )

dfChi$cluster = assign.cluster(dfChi, fit.km[["centers"]])
SSE.Chi = assign.SSE(dfChi)

# Test on our validation set:
# validate_dfm$Cluster = predict(sum,validate_dfm)
```

```
validate_dfm$cluster = assign.cluster(validate_dfm, fit.km[["centers"]])
SSE.Bos.validate = assign.SSE(validate_dfm)

# As a final check, let's see how our classifier does on the training data.
# When we remove finish time from the equation, how often does our Euclidian distance
# based algorithm pick the right cluster ID?

dfm$PredictedCluster = assign.cluster(dfm, fit.km[["centers"]])
predicted.cluster.accuracy  = nrow(dfm[dfm$PredictedCluster == dfm$cluster,])/nrow(dfm)
#I get about 40% correct with distance classifer

##
```

Appendix B - References

This entire project code including source data can be found on GitHub (https://github.com/wihl/statse139-project)

K-Means Clustering in R Sample Code

Solutions Found on StackOverflow

Place R Code in Appendix

Takes mean by subgroup in R

Formatting R Values as Percent

R Markdown Page Break

Remove Rows with NA in DataFrame (use of complete.cases)

Assign a new datapoint to existing clusters