# Predicting Boston qualifiers at the Boston Marathon


A data science & machine learning capstone project

**Don Wang**

Spring 2018

**Table of Contents**



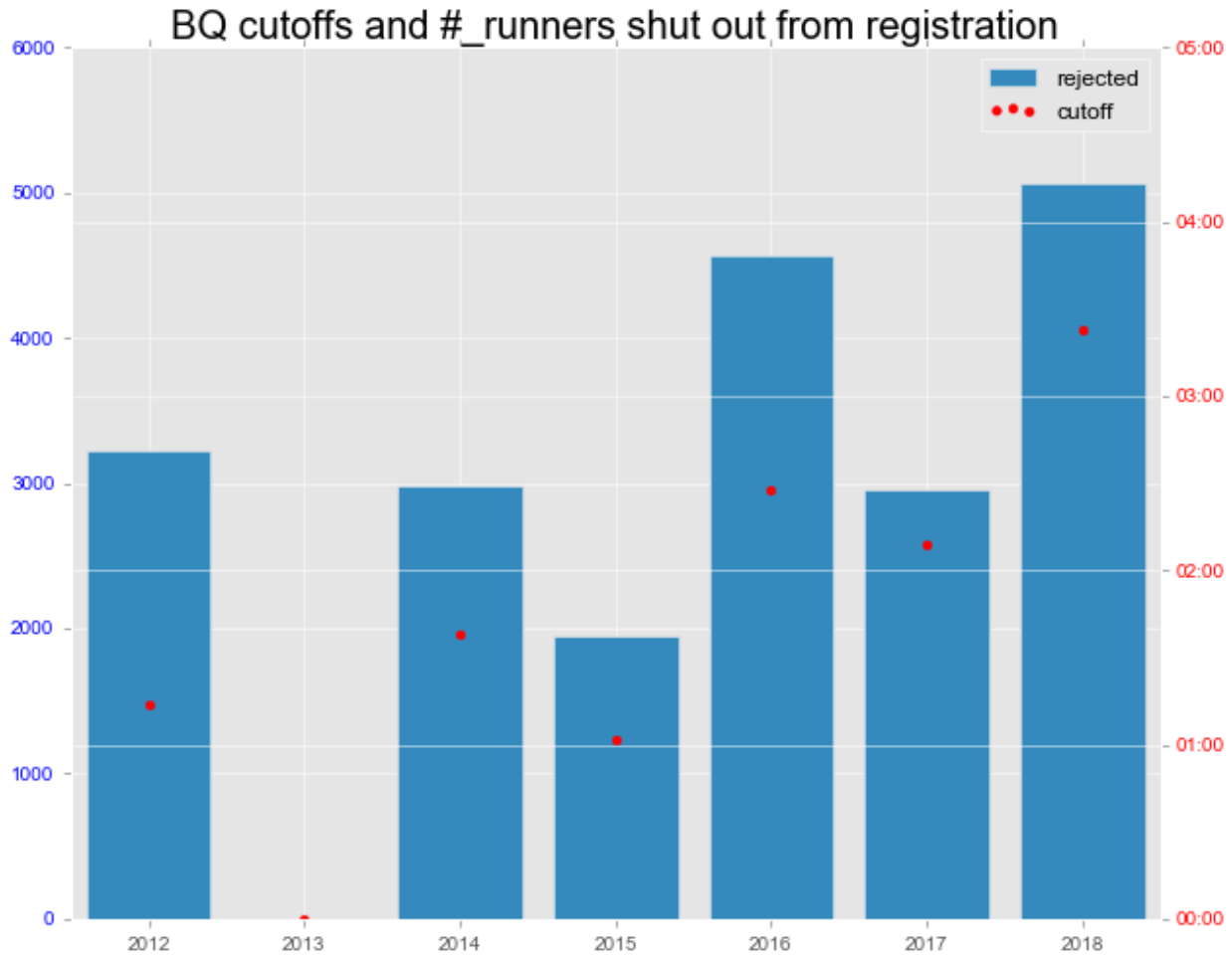
## Introduction

The Boston Marathon remains the only one in the US that has qualifying standards for its prospective entrants. (This doesn't apply to charity runners, who account for about 20% of the field size – 30,000 – in recent years.) Held annually in April, it's known as the "grand-daddy" of road races because of its longevity, dating all the way back to 1897. Even before the bombings that marred the finish line at the 2013 race, it had become increasingly difficult to get into Boston, due to its limited field size and the burgeoning demand for entries by runners worldwide.

For example, in the lead-up to the 2012 edition, the Boston Athletic Association (BAA) – the non-profit organization that organizes the race – had to impose cutoffs on top of the qualifying times for the first time in its history. The registration process begins months in advance of the race; for 2012, runners submitted their marathon times in September 2011, and when the dust settled, they found out that they had to have beaten their Boston qualifying (BQ) time by at least 1:14 (one minute and fourteen seconds) to get in.

Here is a plot of the number of Boston finishers in recent years –

And here is a plot of the number of qualified registrants who were shut out in recent years, along with the margin by which entrants had to have beaten their BQ time in order to get in –




A word on nomenclature: a runner BQs (qualifies for Boston) by finishing a marathon faster than his/her BQ (Boston qualifying) requirement. What "BQ" stands for depends on the context in which it is used, but to runners its meaning and part-of-speech is not often a subject of confusion.

The goal of this project is **to predict who will BQ at Boston, given all the data leading up to and including the halfway point**: runner info (age, gender, bib number), starting temperature, split times, and cumulative and split paces.

It is a machine learning **classification** problem: will a runner BQ or not?

## Bottom line up front (BLUF)

Using 2013-2016 Boston data as the training set and 2017 Boston data as the test set, I built two different classifiers (gradient boosting machines and K-nearest neighbors) that were both able to predict BQs/non-BQs with slightly higher than 90% accuracy. The gradient boosting classifier performs slightly better, but the K-neighbors approach might be more useful to end-users.

## Business application

Runners generally train weeks and months before racing a marathon: its length requires that runners "respect the distance," lest they "hit the wall" toward the latter stages and slow to a crawl (by doing the "death shuffle"). Boston qualifiers are speedier and more experienced compared to the average runner, but they still have to train well and form a sensible plan in order to BQ at Boston.

The goal is **to help runners get a sense of their BQ chances, given their planned paces leading up to and including the halfway point**. It would serve them well if they can see how previous runners have fared in the second half when those past runners started out in a way that's similar to a planned strategy for the first half. In terms of marathoning "best practices," it's generally advised to "be conservative" and "not go out too fast". A machine learning model can help in this regard, by showing what's considered too conservative or aggressive in order to BQ, for someone of a certain age group.

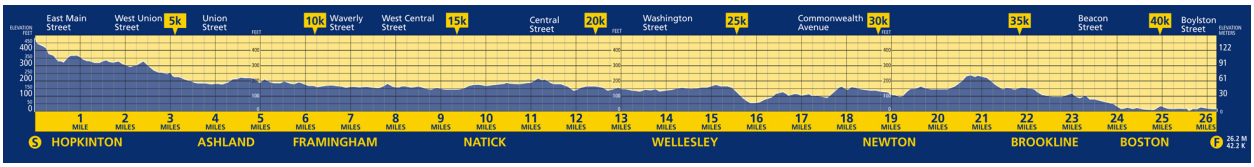
Why should anyone care about how to run Boston? There are a number of reasons:

1. It's considered a tough course, and a sound pacing strategy can go a long way toward having a successful race.
2. It has a high entry fee, one of the highest in the US. (For the 2018 race, it cost $185 for US residents and $250 for international residents.)
3. Travel and lodging costs for race weekend also have to be factored into the experience. Most Boston entrants are not international, but most stateside entrants reside outside of the New England area. Why spend all that money to get there, only to have a bad run?

It's also important to point out that not everyone tries to BQ at Boston even if they ran much faster times than their BQ requirements. A runner might have just gotten injured, or is in the process of returning from a life event, or might have plans to run with a much slower friend. Using a classifier such as K-nearest neighbors would allow someone to not only get a sense of her BQ chances, but also see specific outcomes from the past that are similar to her plan of attack.

## Why Boston?

There are several advantages of using data from the Boston Marathon. It's a popular race, so lots of people might be interested in what its data look like. It's also considered a challenging course, given its elevation profile –




If it looks easy for the first 16 miles, that's because it is – since it's mostly downhill with a net elevation drop of around 450 feet. But then the Newton Hills appear, from Miles 17-21 – a "killer chain" of four uphill portions, culminating with the infamous Heartbreak Hill, located just before the Boston College campus. The early miles, then, serve as a trap to those who find it easy and proceed to get carried away and run faster than intended. Because of the course's deceptive level of difficulty, a prospective runner might be well served by knowing how the data translate to a prospective pacing plan (i.e., how a plan compares to similar pacing profiles and outcomes from the past).

The race offers several advantages from a data standpoint:

1. It's one of the largest marathons in the US, with over 26,000 finishers in recent years. (Most US marathons have less than 1,000 finishers.)
2. It has many timing mats that record runners' progress through the race, with 9 "checkpoints" between the start and finish: every 5K, plus the halfway point. (Most marathons have 3 or fewer checkpoints, and some don't have any.)
3. The course/route is well-established and hasn't changed for the past few decades, meaning that the timing mat locations are static and provide an "apples-to-apples" comparison of pacing from year to year. (Many marathons tinker with their courses from time to time, due to expiring city permits, road construction, and public backlash.)
4. The race uses a bib-numbering system that ranks (non-charity) entrants by the order of their qualifying times. (This helps prospective runners compare themselves to runners who had similar bib numbers in previous Boston races, because similar bib numbers generally imply similar running ability.)
5. The BQ percentage is much closer to 50% than most races, meaning that the classification problem is somewhat balanced. (Overall, around 41% of all 2013-17 finishers BQed. Many marathons have a BQ rate in the single digits, and a BQ rate of 20% at any non-Boston race is considered high.)

Because it's a large race, we need only a few years' worth of results to form a large training set. And because there are a lot of timing mats, we have more possible features to choose from and feed into a machine learning algorithm. These are the main reasons for the suitability of the race as a predictive problem to solve.

## Data gathering & exploratory data analysis

I took 2013-17 data from the BAA's site. The BAA's archive contains official results but with no split times. To get split times, I looked elsewhere on the site and found them at the following URLs: http://registration.baa.org/**[YEAR]**/cf/public/iframe_ResultsSearch.cfm. For example, the link for **2013** results is as follows: http://registration.baa.org/**2013**/cf/public/iframe_ResultsSearch.cfm

(The unofficial results with split times can be found at the following URLs: http://boston.r.mikatiming.de/**[YEAR]**/

Sometimes the unofficial results will contain records for people who aren't in the official results (and vice versa). And sometimes a runner's unofficial chip time will show up as the same as her gun time, due to her chip not registering at the starting line, and her chip time will be corrected in the official results (after race officials review video from the starting line to see when exactly she starts). Later, in the data wrangling section, we'll see that the split times might need to be adjusted for people whose official and unofficial times are different.)

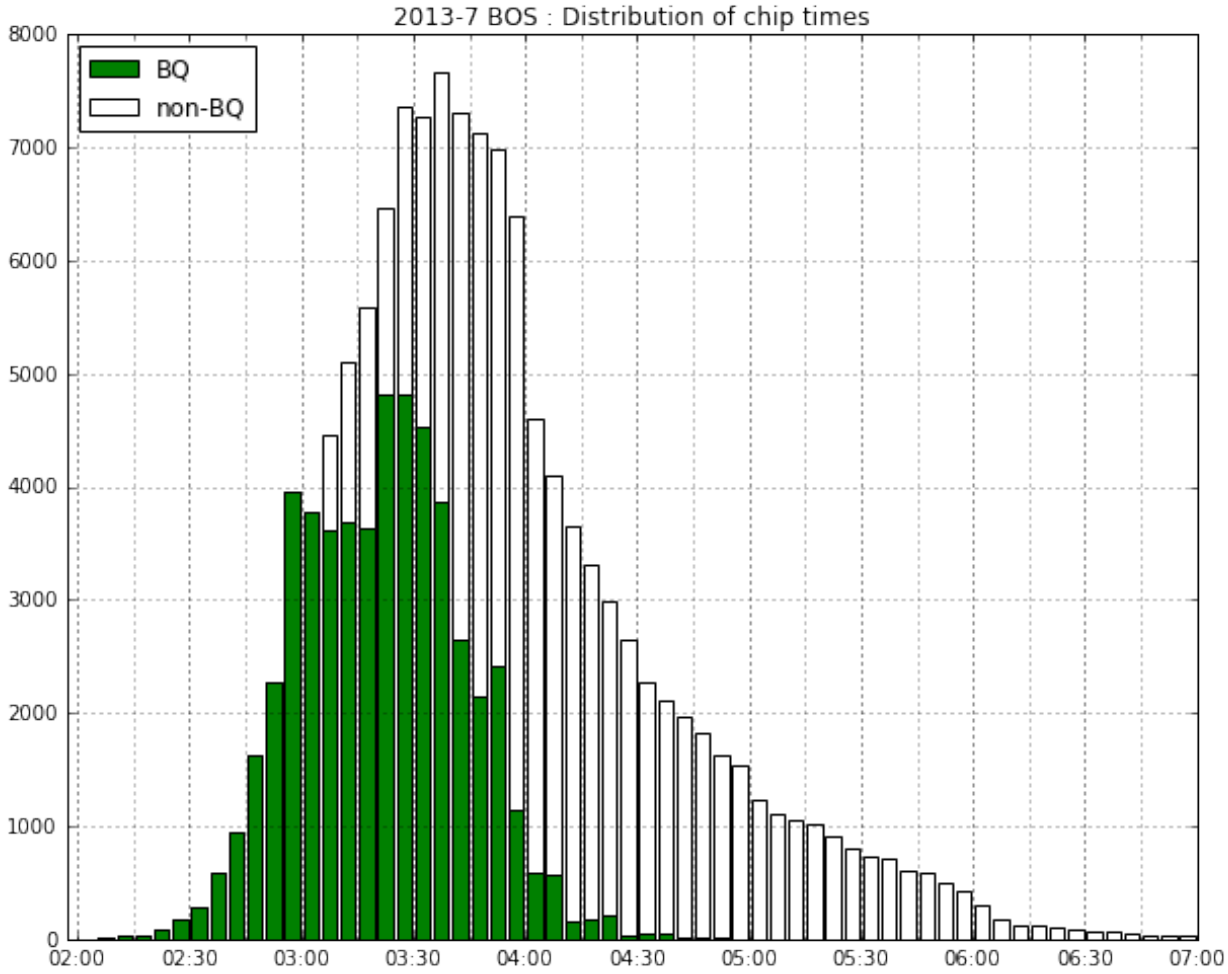
Across the 5 years (2013-17), there were around 129,000 official finishers. Here's a screenshot of what one record for an official finisher looks like –

| BIB | NAME | AGE | M/F | CITY | ST | CTRY | CTZ |
|---|---|---|---|---|---|---|---|
| 19 | Keflezighi, Meb | 38 | M | San Diego | CA | USA | |

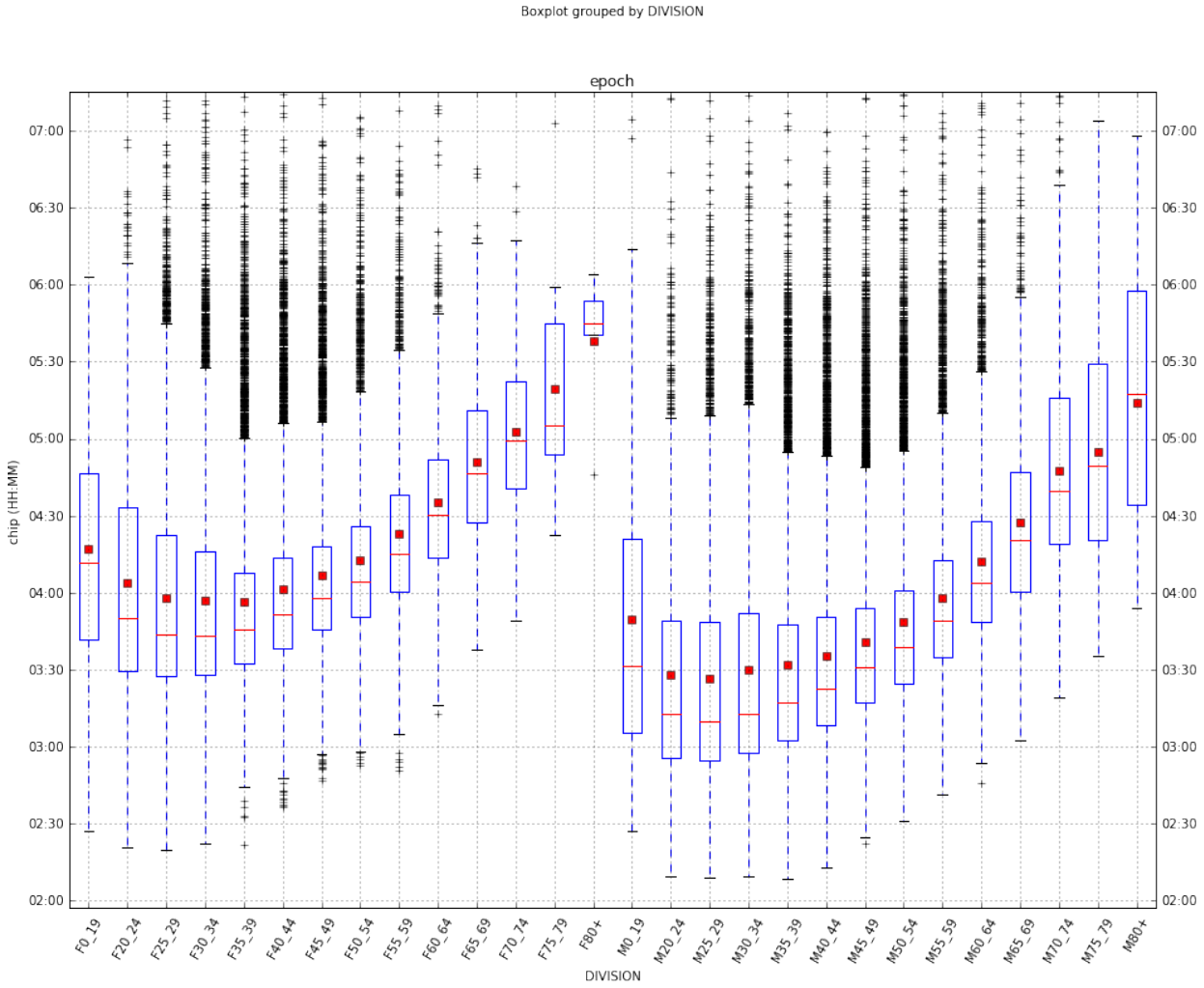
| 5k | 10k | 15k | 20k | Half | 25k | 30k | 35k | 40k |
|---|---|---|---|---|---|---|---|---|
| 0:15:09 | 0:30:29 | 0:45:47 | 1:01:05 | 1:04:21 | 1:16:00 | 1:31:10 | 1:46:37 | 2:01:49 |
| Finish: | | | Pace | Proj. Time | Offl. Time | Overall | Gender | Division |
| | | | 0:04:55 | 2:08:37 | 2:08:37 | 1 | 1 | 1 |

We can see that there are split times, but no split paces or cumulative paces. This isn't a major issue since we can calculate those ourselves.

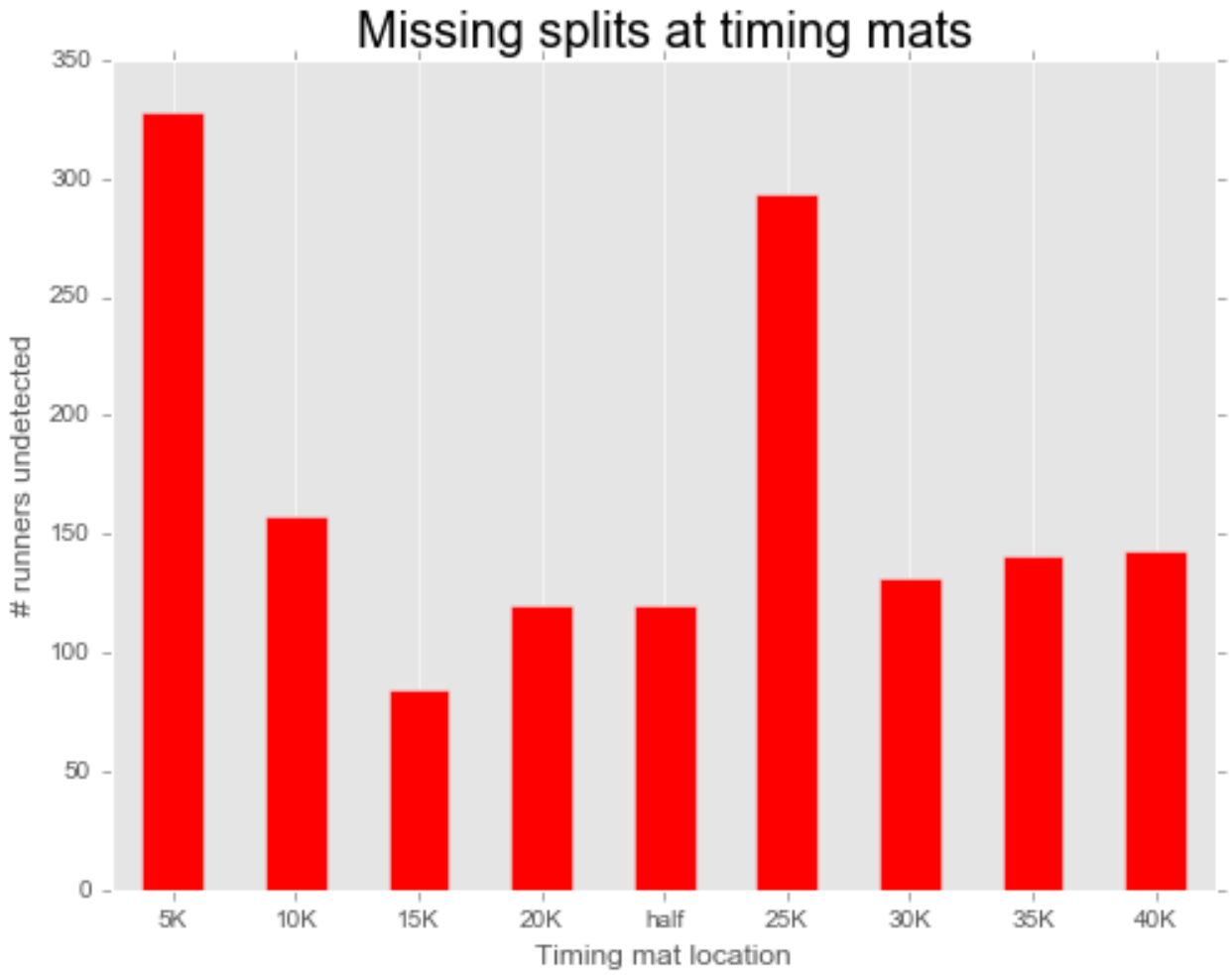
Here's a histogram of the finish times –



Here's the boxplot of finish times across different age groups –




Runners are instructed to wear their bib numbers on the front. These bibs contain RFID tags that are detected by timing mats in order to register a runner's split times at those specific locations. It's possible that timing mats along the course don't detect a runner, but these "anomalies" aren't common at Boston. Here's a plot of those numbers –




I chose to drop cases in which a runner had a missing split at any of the first five timing mats. It's not a major loss, since this accounts for less than 1% of the dataset. There are a few other records that I'll drop from consideration, and we explain those cases in the BOS_3_machine.ipynb notebook file.

Notes:

- I chose not to use 2012's data because the weather was especially hot during the race that year, and only 12% of the finishers BQed – an abnormally low number for Boston. (In the days leading up to the race, the BAA was aware of the forecast and offered all entrants the choice to defer to 2013 for free. Still, over 21000 toughed it out and finished.)
- I chose not to use 2011's data because the qualifying standards at 2011 were different than the qualifying standards in subsequent years. To elaborate, runners at 2011 Boston had easier BQ requirements than at 2012 Boston and onward, and this might have affected the way that people paced themselves prior to 2012, even if they were going after the same goal. (For example, a 30-year-old man had a BQreq of 3:10:59 prior to 2012 Boston. If his goal at 2011 Boston were 3:00:00 and he

started to fade, he had a wider margin to BQ than if his BQreq were 3:05:00, the current BQreq for men under 35.)

## Data wrangling and cleaning

It was a fairly straightforward process to scrape the official results with split times from the BAA site. We used Selenium to extract results in batches based on first and last initials. (It was a bit more involved to scrape the unofficial results with split times.)

We expect time data to be in the standard "H:MM:SS" format, and for the most part things were correct. But here's an example of a wrong format (2.46.50) –

2014 BOSTON MARATHON RESULTS

SEARCH AGAIN  REFINE YOUR SEARCH

**All checkpoints are official times.**

bad format

| BIB | NAME | AGE | M/F | CITY | ST | CTRY | CTZ |
|---|---|---|---|---|---|---|---|
| 1665 | Knott, Timothy J. | 35 | M | Douglas, Isle Of Man | | GBR | |

| 5k | 10k | 15k | 20k | Half | 25k | 30k | 35k | 40k |
|---|---|---|---|---|---|---|---|---|
| 0:19:18 | 0:38:06 | 0:57:03 | 1:16:19 | 1:20:29 | 1:35: | 1:56:25 | 2:17:19 | 2:38:12 |
| Finish: | | | Pace | Proj. Time | Offi. ime | Overall | Gender | Division |
| | | | 0:06:24 | - | 2.46.50 | 698 | 661 | 545 |

| BIB | NAME | AGE | M/F | CITY | ST | CTRY | CTZ |
|---|---|---|---|---|---|---|---|
| 14383 | Knott, Mark | 54 | M | Smithtown | NY | USA | |

good format

| 5k | 10k | 15k | 20k | Half | 25k | 30k | 35k | 40k |
|---|---|---|---|---|---|---|---|---|
| 0:24:21 | 0:48:32 | 1:12:36 | 1:37:07 | 1:42:23 | 2:01:42 | 2:26:49 | 2:52:40 | 3:18:03 |
| Finish: | | | Pace | Proj. Time | Offi. Ti..e | Overall | Gender | Division |
| | | | 0:07:59 | 3:29:06 | 3:29:06 | 9039 | 7079 | 800 |

Also, this runner's official chip time shows up as 2:58:**81** –

2015 BOSTON MARATHON RESULTS

SEARCH AGAIN  REFINE YOUR SEARCH

**All checkpoints are official times.**

bad format

| BIB | NAME | AGE | M/F | CITY | ST | CTRY | CTZ |
|---|---|---|---|---|---|---|---|
| 3820 | Wright, Brian | 27 | M | Virginia Beach | VA | USA | |

| 5k | 10k | 15k | 20k | Half | 25k | 30k | 35k | 40k |
|---|---|---|---|---|---|---|---|---|
| 0:22:47 | 0:43:17 | 1:04:51 | 1:25:30 | 1:30:05 | 1:46:28 | 2:08:27 | 2:30:35 | 2:51:49 |
| Finish: | | | Pace | Proj. Time | Offi. Time | Overall | Gender | Division |
| | | | 0:06:51 | - | 2:58:81 | 2379 | 2246 | 1689 |

We'll assume that it's supposed to be 2:59:21.

The runner fell into a small subset of runners whose official chip times were different than their unofficial chip times. As mentioned previously, these cases usually come about when the timing mat at the starting line doesn't register the chips belonging to the runners in question. Here are some examples of runners with adjusted chip times –

| year | bib | name | chip | chip_unofficial | gun |
|---|---|---|---|---|---|
| 2013 | 5599 | Ness, Erik C | 2:55:59 | 2:59:12 | 2:59:12 |
| 2013 | 4002 | Shugart, Rob C | 3:12:44 | 3:15:40 | 3:15:40 |
| 2014 | 2794 | Watson, Andy | 2:48:22 | 2:49:57 | 2:49:57 |
| 2014 | 4406 | Schakel, Christiaan | 2:57:29 | 2:59:41 | 2:59:41 |
| 2014 | 4224 | Lott, Brian | 2:57:40 | 3:00:29 | 3:00:29 |

As clearly shown, it's usually the case that these runners had the same gun time as their unofficial chip times, and their official chip times were made faster.

(The official results don't include gun times and unofficial chip times. To get gun times and unofficial chip times from the unofficial results, we need to merge the unofficial results with the official results. But before we merge the unofficial and official results, we need to fix some bib numbers in the unofficial results.)

The problem is that, **prior to 2016**, the BAA didn't adjust their split times downward in the same manner. Split times are supposed to be based on "chip time" and not "gun time". If their split times aren't adjusted, then their split paces look like the following –

| year | bib | name | chip | chip_unofficial | gun | non_missing_splitpaces_time |
|---|---|---|---|---|---|---|
| 2013 | 5599 | Ness, Erik C | 2:55:59 | 2:59:12 | 2:59:12 | [7:48, 6:43, 6:37, 6:38, 6:40, 6:38, 6:46, 6:46, 6:46, 4:25] |
| 2013 | 4002 | Shugart, Rob C | 3:12:44 | 3:15:40 | 3:15:40 | [7:53, 6:52, 6:56, 7:01, 6:58, 8:08, 7:26, 7:53, 7:46, 5:23] |
| 2014 | 2794 | Watson, Andy | 2:48:22 | 2:49:57 | 2:49:57 | [7:05, 6:30, 6:23, 6:25, 6:24, 6:21, 6:26, 6:28, 6:16, 5:09] |
| 2014 | 4406 | Schakel, Christiaan | 2:57:29 | 2:59:41 | 2:59:41 | [7:22, 6:41, 6:40, 6:48, 6:36, 6:50, 6:52, 6:53, 6:46, 5:18] |
| 2014 | 4224 | Lott, Brian | 2:57:40 | 3:00:29 | 3:00:29 | [7:43, 6:44, 6:40, 6:42, 6:40, 6:42, 6:47, 6:53, 6:55, 4:46] |

It's obvious that the first split pace (between the starting line and the 5K mark) is slow, and the last split pace (between the 40K mark and the finish line) is fast. Everything else is fine because the time deltas are based on *gun* – *gun*, not *gun* – *chip* (first split pace) or *chip* – *gun* (last split pace).

(A word on **non_missing_splitpaces_time**: for those with no missing split times, this list shows the 10 successive split paces from the start to the finish: start-5K, 5K-10K, …, 35-40K, 40K-finish. If a runner has 1 missing split, then the list contains 9 split paces; if a runner has 2 missing splits, then the list contains 8 split paces, and so forth.)

To fix the above, we'll have to adjust the split times downward for those runners by the same amounts that their official chip times were adjusted downward (from their unofficial chip times). In other words, for runners with adjusted official chip times, their split times

need to be subtracted by the following amount: unofficial chip time – official chip time. After doing so, we get much more reasonable split paces –

| year | bib | name | chip | chip_unofficial | gun | non_missing_splitpaces_time |
|---|---|---|---|---|---|---|
| 2013 | 5599 | Ness, Erik C | 2:55:59 | 2:59:12 | 2:59:12 | [6:46, 6:43, 6:37, 6:38, 6:40, 6:38, 6:46, 6:46, 6:46, 6:47] |
| 2013 | 4002 | Shugart, Rob C | 3:12:44 | 3:15:40 | 3:15:40 | [6:56, 6:52, 6:56, 7:01, 6:58, 8:08, 7:26, 7:53, 7:46, 7:32] |
| 2014 | 2794 | Watson, Andy | 2:48:22 | 2:49:57 | 2:49:57 | [6:34, 6:30, 6:23, 6:25, 6:24, 6:21, 6:26, 6:28, 6:16, 6:18] |
| 2014 | 4406 | Schakel, Christiaan | 2:57:29 | 2:59:41 | 2:59:41 | [6:39, 6:41, 6:40, 6:48, 6:36, 6:50, 6:52, 6:53, 6:46, 6:55] |
| 2014 | 4224 | Lott, Brian | 2:57:40 | 3:00:29 | 3:00:29 | [6:49, 6:44, 6:40, 6:42, 6:40, 6:42, 6:47, 6:53, 6:55, 6:50] |

In total, there were 156 cases that had to be adjusted in the above manner – a small number, considering the total size of the dataset. There remain a handful of runners whose split times and paces still don't make sense after these adjustments. But it's a much smaller subset than previously, and we can just drop these cases.

Another aspect of data wrangling involved adding temperature data from external sources. We took temperature data from Wunderground.com, which has historical hourly temperatures from select weather stations. The unofficial results contain the time-of-day (TOD) at which runners crossed each timing mat (if the mats registered their chips). We used linear interpolation, by minute, of the temperatures; for the runners, we truncated/floored the TODs by minute. To get the temperatures when each runner crossed the starting line and halfway points, we merged the above info together.

In theory, the temperatures should be useful features because more runners slow down in hotter temperatures (and by larger amounts). But the Wunderground site didn't offer extremely localized weather info; in other words, it returned the same temperature for both Hopkinton (the location of the starting line) and Wellesley (the location of the halfway point) because the site used data from the same location (Norwood). So, while it's possible to incorporate weather info into our model, we used only starting temperatures and only in one of our two classifiers.

## Prediction baseline

Of interest to marathon stats geeks is the number of "comebacks" and "fades" at any race. A "comeback" is considered a case where a runner is behind his BQ pace at location X but speeds up after that and BQs. A "fade" is the opposite: he's ahead of BQ pace at location X but slows down and doesn't BQ. Certainly there are people who start ahead and stay ahead the whole way, and those who start behind and stay behind, but what makes this an

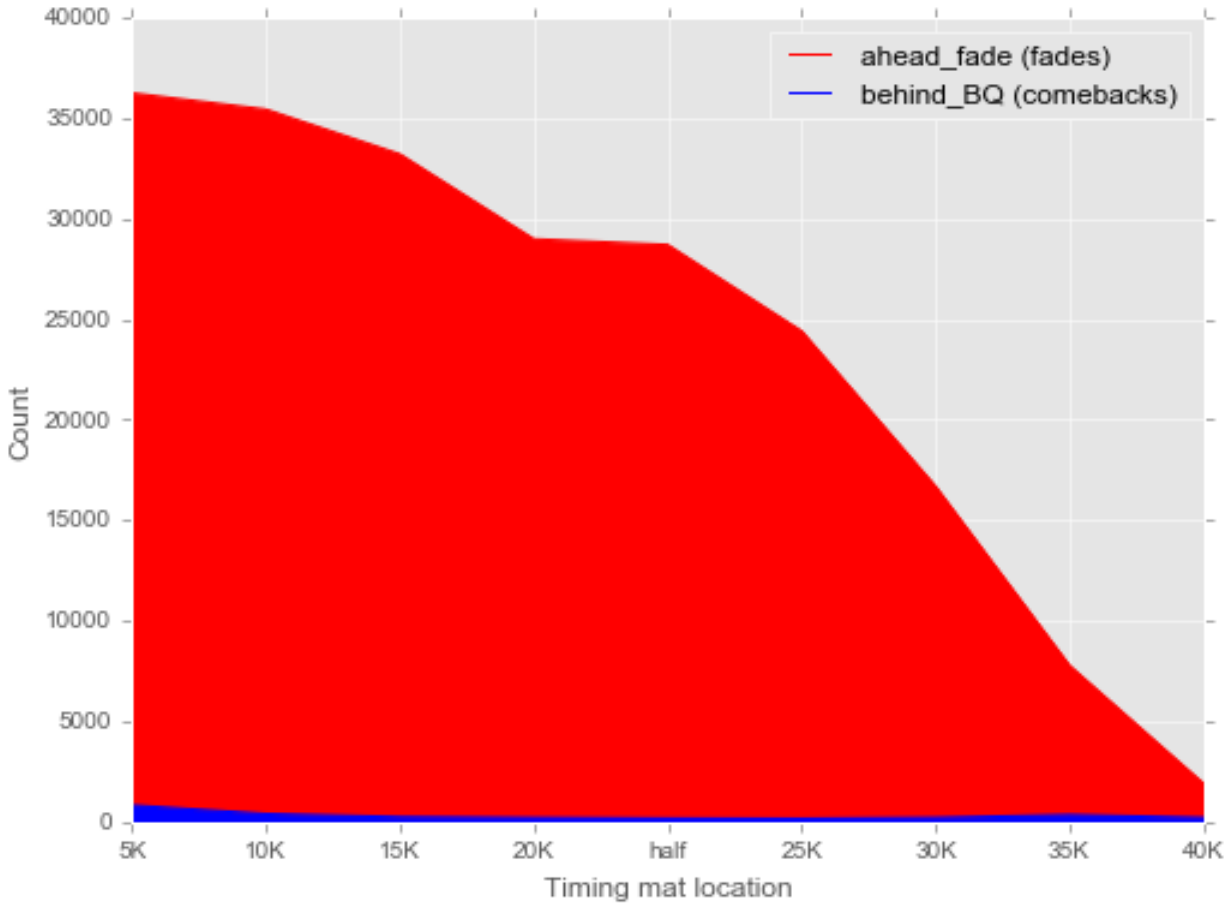
interesting machine learning problem is the effect that comebacks and fades have on the prediction process.

Here's a table showing the number of such cases across all timing mats from 2013-17 –

| location | ahead _BQ | behind _BQ | unknown _BQ | a_fade | b_fade | u_fade | b_BQpct | a_fadepct |
|---|---|---|---|---|---|---|---|---|
| 5K | 52116 | 818 | 91 | 36230 | 39661 | 237 | 1.54% | 41.01% |
| 10K | 52583 | 390 | 52 | 35420 | 40602 | 106 | 0.74% | 40.25% |
| 15K | 52754 | 239 | 32 | 33194 | 42882 | 52 | 0.45% | 38.62% |
| 20K | 52791 | 197 | 37 | 28973 | 47072 | 83 | 0.37% | 35.44% |
| Halfway | 52814 | 170 | 41 | 28690 | 47359 | 79 | **0.32%** | **35.20%** |
| 25K | 52834 | 157 | 34 | 24396 | 51472 | 260 | 0.30% | 31.59% |
| 30K | 52775 | 201 | 49 | 16673 | 59373 | 82 | 0.38% | 24.01% |
| 35K | 52643 | 340 | 42 | 7766 | 68263 | 99 | 0.64% | 12.86% |
| 40K | 52786 | 202 | 37 | 1906 | 74116 | 106 | 0.38% | 3.49% |

The "_BQ" columns will sum up to the same number in each row, while the same is true for the "_fade" columns. It's possible for a runner to shift between the "_BQ" columns; for example, a runner might show up under "ahead_BQ" from 5K-35K, then "behind_BQ" at 40K (by slowing down between 35K-40K), but kick it into a higher gear from 40K-42.2K to BQ.

One can see from the table that the number of fades at each timing mat is much greater than the number of comebacks. Here's a plot that shows this –




We can also see that the number of "a_fades" goes down as the race progresses. "a_fadepct" is defined as follows – a_fade/(a_fade + ahead_BQ). As a concrete example, at the halfway point, we can see that 35.20% of those ahead of their BQ pace eventually faded and didn't BQ. (Analogously, "b_BQpct" is defined as the percentage of those behind who ended up with BQs. The table above shows that 0.32% of those behind at halfway finished as comeback cases.)

The baseline I will use is the following: predict "BQ" for those on or ahead of BQ pace at halfway, and "non-BQ" for those behind BQ pace at halfway. This comes out to being correct **78.50% of the time** from 2013-16, and **74.27%** in 2017.

Can a machine learning algorithm do better than this? And if so, by how much?

## Features engineering

The most important thing to know at any timing checkpoint is the following: is a runner ahead of his or her BQ pace? A runner's BQ requirement depends on two factors: age and gender. Women have 30 extra minutes to qualify compared to men of the same age; for example, a 30-year-old woman has a "BQreq" of 3:35:00 (or faster), while a 30-year-old

man needs 3:05:00 (or faster). Every 5 years, starting at age 35, a runner's BQreq goes up (gets less stringent). The entire list of qualifying times can be found here.

Recall that the Boston records include only split times. We'll have to calculate **cumulative paces** and **split paces** from the split times, as follows –

$\boldsymbol{splitX} = \frac{splitX\_time}{timingmats[X]}$ : the **cumulative pace** at the X$^{th}$ timing mat (in seconds per mile, unrounded)
$\boldsymbol{splitXY} = \frac{splitY\_time - splitX\_time}{timingmats[Y] - timingmats[X]}$ : the **split pace** between the X$^{th}$ and Y$^{th}$ timing mats (in seconds per mile, unrounded)

For more "human-readable" metrics, we define splitX_pace and splitXY_pace as splitX and splitXY converted to 'MM:SS' format. For practical purposes we need splitX and splitXY because only real numbers can be fed into machine learning applications.

The Boston dataset contains everyone's age and gender, which are vital for computing the following metrics –

***BQreq***: 'H:MM:SS' format
***BQpace***: '[M]M:SS' format
***BQpace_sec***: BQpace in seconds per mile, unrounded

This leads to the calculation of the all-important "score" metrics –

***splitX_score*** = *splitX – BQpace_sec*: the **cumulative pace score** at the X$^{th}$ timing mat (in seconds per mile, unrounded)
***splitXY_score*** = *splitXY – BQpace_sec*: the **split pace score** between the X$^{th}$ and Y$^{th}$ timing mats (in seconds per mile, unrounded)

Think of these "score" metrics in terms of golf scores: negative values indicate being faster than BQ pace (or "under par" in golf-speak), and positive values indicate being slower. This is how we can easily determine whether a person is ahead or behind at any of the timing mats. We have "normalized" everyone's paces into scores for the purpose of seeing how they compare to their BQ pace at all the timing mats. For example, a 30-year-old man with a split time of 2:00:00 at the halfway point would be way behind his BQreq of 3:05:00, while a 60-year-old woman with the same split time at the halfway point would be way ahead of her BQreq of 4:25:00.

Some other features to note:
***bib_mod***: 1-30000+ (integer form of bib, which contains "F" for elite women)
***wave***: 0-4 (integer); the lower the wave, the faster the runner – elites are in Wave 0, and charity runners in Wave 4
***start_temp***: degrees F (float) in Hopkinton when the runner crosses the starting line

***stdpace_matX***: the standard deviation of a runner's split paces up to and including mat X (in seconds per mile) (e.g., stdpace_mat4 would be calculated as np.std([split12, split23, split34], ddof=1)

## Machine learning

To predict BQs, we'll consider two different classifiers: **gradient boosting machines** and **K-nearest neighbors**. On the spectrum of classifier complexity, these two fall toward opposite ends. Gradient boosting machines are more complex and build on the concept of decision trees and random forests. K-nearest neighbors are almost self-explanatory and are much simpler to understand.

### Classifier: Gradient Boosting Machines

Gradient boosting is an ensemble method that combines multiple decision trees. From [Müller: Introduction to Machine Learning with Python]: It works by building trees in a serial manner, where each tree tries to correct the mistakes of the previous one. By default, there is no randomization in gradient boosted regression trees; instead, strong pre-pruning is used. Gradient boosted trees often use very shallow trees, of depth one to five, which makes the model smaller in terms of memory and makes predictions faster. The main idea behind gradient boosting is to combine many simple models (in this context known as weak learners), like shallow trees. Each tree can only provide good predictions on part of the data, and so more and more trees are added to iteratively improve performance.

Gradient boosting assigns "importances" to the features that are fed into it. Because it can determine which features are more important for maximizing prediction accuracy, we'll feed all the relevant features into it that we can think of – **age, gender, bib number, wave, split paces and scores, cumulative paces and scores, starting temperature, and stdpaces** – and let the algorithm figure out how best to use them.

### Classifier: K-Nearest Neighbors

K-nearest neighbors, or KNN, is one of the simpler machine learning classifiers to understand: it makes predictions based on the "votes" of the predictor variable belonging to the K-nearest neighbors of a test point. The K value is an integer that can be tuned in order to optimize predictive performance and is usually chosen as an odd number in binary classification tasks. In some respects, KNN can be thought of as being unsophisticated because it memorizes the training set, thus acting as a "lazy learner". But its simplicity also makes it easy to comprehend and intuitive to explain.

In contrast to our methodology with the more complicated gradient boosting classifier (GBC) described above, we'll be more selective with the number of features to use for KNN.

We settle for the following: **gender, age group (via BQpace_sec), split pace scores, and cumulative pace scores**.

Why not age? We're choosing to go with BQpace_sec instead, which is a way to add age group as a feature. This way, it would be easier to compare 38-year-olds to 35-year-olds, and 18-year-olds to 33-year-olds. (It's true that men and women of different ages might have the same BQreq; for example, M60-64 and F45-49 both have 3:55:00 as their BQreq. But having gender as a feature will mostly preclude men and women from being neighbors.)

Why not split paces and cumulative paces? We're choosing to go with just the pace scores, which are just the paces offset by BQpace_sec, for simplicity.

Why not bib or wave? Based on the above, we're choosing to look for people of the same age group to compare with, and there might be people assigned to a slower wave who have improved their fitness and are capable of running as fast as others in their age group who are placed in a faster wave.

Why not starting temperature? No particular reason, other than keeping the model simple!

## Model evaluation

First, we defined base classifiers (without any hyperparameter tuning) for GBC and KNN.

After fitting the base classifiers on the entire training set, we get the following performance on the test set –

GBC:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| non-BQ | 0.951 | 0.911 | 0.931 | 17892 |
| BQ | 0.827 | 0.900 | 0.862 | 8408 |
| avg / total | 0.911 | 0.908 | 0.909 | 26300 |

KNN:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| non-BQ | 0.943 | 0.901 | 0.921 | 17892 |
| BQ | 0.807 | 0.884 | 0.844 | 8408 |
| avg / total | 0.899 | 0.895 | 0.897 | 26300 |

Not surprisingly, GBC did better than KNN by achieving higher accuracy and F1 scores.

This didn't change even after using grid search with cross validation. For KNN, in keeping with the "simple is best" approach, we limited the grid search on n_neighbors to not exceed 71. Here is the performance on the test set with tuned hyperparameters –

GBC:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| non-BQ | 0.951 | 0.911 | 0.931 | 17892 |
| BQ | 0.827 | 0.900 | 0.862 | 8408 |
| avg / total | 0.911 | 0.908 | 0.909 | 26300 |

KNN:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| non-BQ | 0.954 | 0.902 | 0.927 | 17892 |
| BQ | 0.812 | 0.907 | 0.857 | 8408 |
| avg / total | 0.909 | 0.903 | 0.905 | 26300 |

We can see that KNN's performance improved, but not nearly to the level of GBC.

## Benefits of KNN

Even though it's a simpler model that doesn't achieve the highest accuracy, KNN offers the very big bonus of seeing the closest neighbors to (1) hypothetical feature values or (2) someone from the test set – to see which training data influenced a prediction. GBC can make predictions, too, of course, but its output is limited to a prediction value (0 or 1) and a *predict_proba* value as a measure of its certainty.

We defined the problem as a classification issue – does a runner BQ or not? – but by using KNN, we can see the neighbors and how they ended up: their paces in the second half as a whole, their paces in specific parts of the second half, and their final paces. We can easily turn it into a regression problem and predict a runner's final pace based on her nearest neighbors. For example, let's say we wanted to see how a 50-year-old man would fare if he stuck closely to his BQpace of 8:01/mi for the entirety of the first half –

```
Halfway pace: 8:01/mi

Prediction (KNN): 0 (probability: 89.72%)
Best expected: 7:55/mi
Middle 50% of neighbors: 8:09-8:23/mi
Median value: 8:22/mi
```

| BQ | year | bib | name | sex | age | chip | pace | non_missing_splitpaces_time | p_1st | p_2nd |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2013 | 12873 | Forbes, Steven J | M | 50 | 3:47:22 | 8:40 | [8:01, 8:02, 7:59, 8:04, 8:00, 8:11, 8:40, 9:23, 10:16, 10:31] | 8:01 | 9:19 |
| 1 | 2013 | 13778 | Santoro, David P | M | 50 | 3:27:41 | 7:55 | [8:03, 8:02, 7:58, 8:02, 8:01, 7:59, 7:48, 7:45, 7:40, 8:05] | 8:01 | 7:49 |
| 0 | 2013 | 13126 | Petrolino, Steven D | M | 51 | 3:31:28 | 8:04 | [7:59, 8:01, 7:59, 8:00, 8:00, 8:04, 8:11, 8:14, 8:04, 8:07] | 8:00 | 8:08 |
| 0 | 2015 | 14146 | Brouch, Robert F | M | 53 | 3:38:11 | 8:19 | [8:00, 7:59, 8:00, 8:04, 8:01, 8:13, 8:34, 8:49, 8:45, 8:53] | 8:00 | 8:38 |
| 1 | 2015 | 14386 | Reazin, Troy A | M | 52 | 3:28:25 | 7:57 | [8:00, 7:59, 7:58, 8:01, 8:04, 8:00, 8:03, 7:58, 7:49, 7:28] | 8:00 | 7:54 |

(table truncated)

By looking up his nearest neighbors, we can see that his fastest neighbor from 2013-6 ended up averaging 7:55, and that the 25-75$^{th}$ percentiles of his nearest neighbors showed a final pace in the 8:09-8:23 range. It's possible that he exceeds this "best expected" pace, or that he completely falls off the pace in his own race. But the point is that KNN allows a user to gain deeper insights beyond a prediction of 1 or 0.

## Model Usage

To use the KNN model, one needs to supply the following inputs –

- *age*: age (integer) on the day of the Boston race
- *sex_int*: 0 for M, 1 for F
- *target_cpaces*: a list, containing either (1) 1 item representing the planned pace that one intends to keep throughout the first half, or (2) 5 items representing the planned cumulative paces at 5K, 10K, 15K, 20K, and halfway (in 'MM:SS' format)

These get fed into the *getPrediction* function –
*getPrediction(age, sex_int, target_cpaces)*

The function calculates the required feature values that are then input into the KNN model.

Here's the expected output for a 26-year-old male who plans to stay close to 6:44/mi for the entire first half –

```
Halfway pace: 6:44/mi

Prediction (KNN): 1 (probability: 75.57%)
Best expected: 6:42/mi
Middle 50% of neighbors: 6:47-7:03/mi
Median value: 6:51/mi
```

| BQ | year | bib | name | sex | age | chip | pace | non_missing_splitpaces_time | p_1st | p_2nd |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2013 | 1323 | Branch, Russell B | M | 33 | 2:58:25 | 6:48 | [6:44, 6:44, 6:43, 6:45, 6:43, 6:44, 6:46, 6:53, 6:56, 7:13] | 6:44 | 6:52 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2013 | 2196 | Lloyd, Jeremy | M | 33 | 3:08:59 | 7:12 | [6:44, 6:44, 6:43, 6:45, 6:43, 6:43, 6:46, 7:16, 8:47, 9:53] | 6:44 | 7:41 |
| 1 | 2014 | 4546 | Olson, Jeremy | M | 30 | 2:57:19 | 6:46 | [6:44, 6:43, 6:45, 6:43, 6:43, 6:44, 6:56, 6:55, 6:45, 6:26] | 6:44 | 6:48 |
| 0 | 2014 | 3725 | Wilson, Mitch | M | 23 | 3:13:16 | 7:22 | [6:44, 6:45, 6:44, 6:45, 6:42, 6:46, 7:06, 8:21, 9:10, 8:50] | 6:44 | 8:00 |
| 1 | 2015 | 3188 | Hadro, Andrew | M | 29 | 2:57:34 | 6:46 | [6:45, 6:43, 6:43, 6:43, 6:43, 6:42, 6:45, 6:50, 6:48, 7:13] | 6:44 | 6:49 |

(table truncated)

As we can see from the *p_1st* and *non_missing_splitpaces_time* columns, these runners kept their overall first-half pace around 6:44/mi with their fairly consistent split paces between the early timing mats.

## Limitations

By all indications, we have enough training data because the cross validation scores show very little variance (which means that the model generalizes well) and the accuracy on the training and test sets are comparable (which means that we're not overfitting on the training data).

The training set data show us how previous runners ran. However, the data don't tell us the following about the runners' –

- physical condition at the time of the race
- recent training levels
- hill-running ability
- race goals

We can make rough guesses as to a runner's goal based on how he paces in the early going, but if he slows down, we don't know whether he did so due to injury, overzealousness, or ineptitude going uphill. (The Newton Hills, covering Miles 17-21, will greatly impact those who train primarily on flat terrain.) These are unknowns that will impact the accuracy of our model.

As we're using data up to halfway, there are guaranteed to be cases where runners look good for the first half, and even most of the race, before their paces fall off a cliff, as in the following examples –

| BQ | year | bib | name | sex | age | chip | pace | non_missing_splitpaces_time | p_1st | p_2nd |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2017 | 269 | Mosey, Nick | M | 30 | 3:16:51 | 7:30 | [6:08, 6:14, 6:02, 6:06, 6:02, 6:25, 7:02, 9:26, 10:51, 11:53] | 6:07 | 8:54 |
| 0 | 2017 | 228 | Mendoza, Maximo C Sr | M | 28 | 3:12:22 | 7:20 | [5:38, 5:31, 5:42, 5:52, 5:56, 6:21, 7:25, 9:05, 10:40, 13:10] | 5:42 | 8:59 |
| 0 | 2017 | 3908 | Tookey, | M | 48 | 3:28:13 | 7:56 | [7:07, 6:56, 7:06, 6:58, 6:58, | 7:02 | 8:51 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | Charles R | | | | | 7:25, 8:14, 8:51, 9:31, 11:20] | | |
| 0 | 2017 | 575 | Jaedtke, Dylan D | M | 29 | 3:21:13 | 7:40 | [6:08, 6:14, 6:02, 6:05, 6:02, 6:25, 7:38, 10:16, 12:14, 8:43] | 6:07 | 9:14 |
| 0 | 2017 | 2010 | Kline, Jonathan A | M | 56 | 3:50:01 | 8:46 | [6:18, 6:31, 6:25, 6:33, 6:33, 6:45, 7:20, 8:47, 17:10, 18:51] | 6:27 | 11:06 |

The K-nearest neighbors model predicted BQs for all of the above runners. There is just no way to know who will implode that like. Certainly these are extreme cases of hitting the wall, but there are enough data points of gradual fades that will bias the KNN model into making more conservative estimates; that is, looking at the nearest neighbors for any given set of inputs will probably reveal far more people who positive-split (by running slower in the second half) than negative-split (by running faster in the second half).

If you're a runner who's planning to race Boston someday, then there are some things to keep in mind. When trying out the model to predict your own race, it's important to put in a realistic pacing plan for your current state of fitness. For example, if you're barely able to average seven-minute miles for a half marathon right now, it means you should *not* try to go out at 7:00 pace for a full marathon – that will inevitably end in disaster. Just because the model predicts, say, a 60% BQ probability based on a 7:00 pace, combined with your age group info, that doesn't mean that you have a favorable chance to do so, given your current running ability. For exploratory purposes it's fine to enter 7:00s as the prospective paces. But for planning purposes, it's best to put in a conservative value and see how previous runners have fared, and not put so much weight on the single prediction probability.

The model doesn't have as many data points for older runners as it does for people of ages 30-50. Older runners tend to be even less predictable than younger runners, as their stdpace values tend to be much higher, so it's certainly a weakness of our model that would be difficult to overcome even if we had more data on the older folks.

## Future considerations

One major factor to further incorporate into the model would be weather. We spent some effort incorporating the starting temperatures for each runner as they crossed the starting line in Hopkinton, but those temperature values were (linearly) interpolated from hourly temperatures at a local airport that's located close to Hopkinton, but still a few miles away. I ended up choosing not to use the temperatures at halfway (Wellesley), since the data from wunderground.com were from the same airport as used for Hopkinton, and the values for 2016 don't appear accurate, based on personal experience. (I participated in 2015-6 Boston.) In theory, having accurate weather along the course should help the prediction process, since runners will be more likely to slow down in the latter half under hotter conditions.

Another consideration would be to build multiple models and implement a voting classifier to make predictions based on more than one model. For example, in our KNN classifier, we're currently using age groups as a feature (through BQpace_sec), as well as gender (sex_int). By doing this, we're comparing, say, a 30-year-old male to other males in the same age group (18-34). We could build separate models that use other features, so that a 30-year-old male can be compared to, say, a 45-year-old female going for the same projected finish time based on their early paces. This might adversely affect BQ prediction accuracy, since a 45-year-old woman could fade much more than a younger man but still BQ; however, this would allow users to find more neighbors and see a wider range of runners who tried to pace for a certain time, regardless of age or gender.

## Web application

Prior to 2018 Boston (April 16), we were able to deploy our model as a web application on PythonAnywhere.com. There are online services that can host web apps, free of charge, for indefinite periods of time. PythonAnywhere was a good choice due to its simplicity. We used the Flask framework to put our model online and display our site, which can be found at http://dunkemo.pythonanywhere.com

(Attribution: we followed the directions in Chapter 9 (Embedding a Machine Learning Model into a Web Application) of the book Python Machine Learning by Sebastian Raschka, which walked us through the steps to take the project online.)

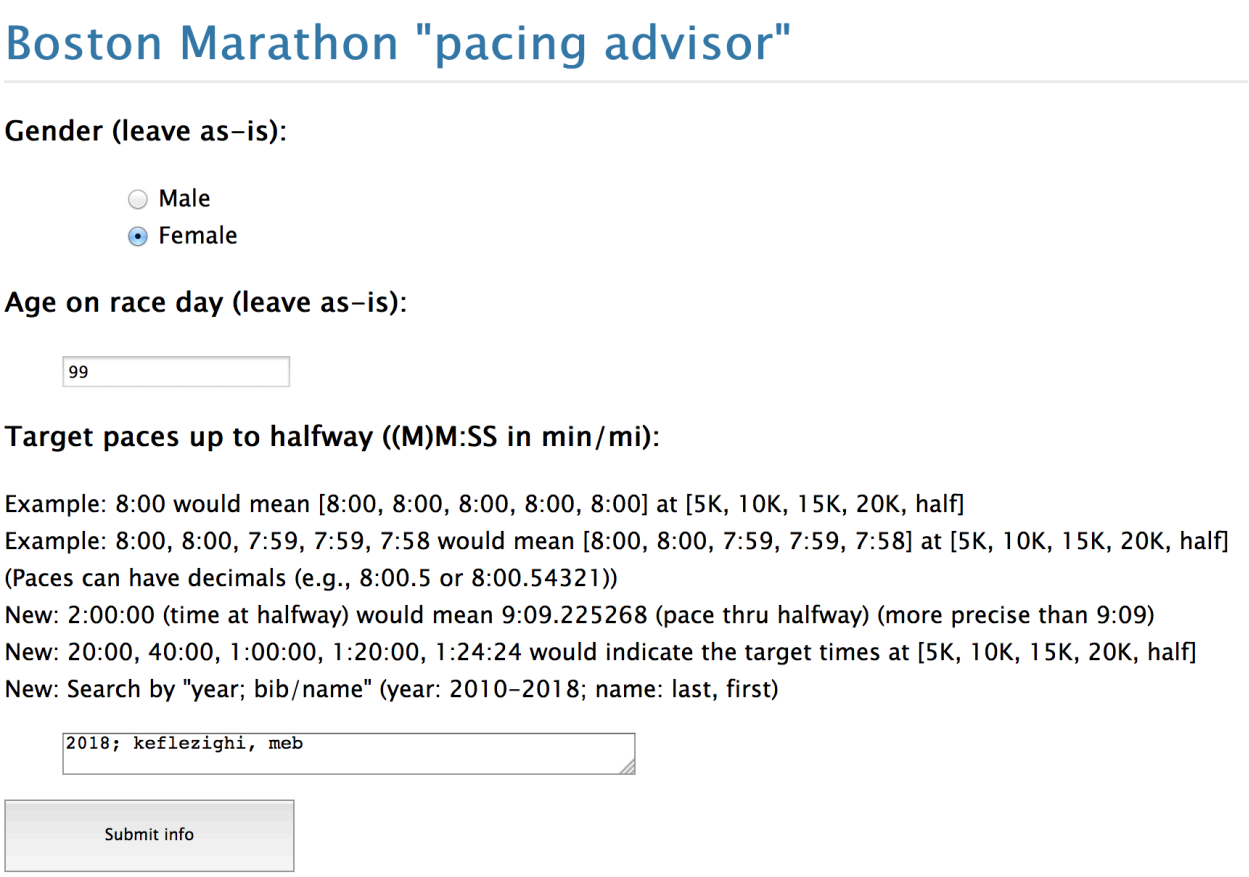
Why did we bother? The goal has always been to help runners form a plan for their race. By allowing them to see what others have done before them, they can tell from the possible outcomes whether their plan can help them reach their goals.

Users would input the required information like age, gender, planned paces, and the model would show them the prediction, prediction probabilities, and nearest neighbors, as we have shown above. Recall that we built the model by using 2013-16 data as the training set and the 2017 data as the test set. Ideally we would have re-fit the model using 2013-17 data as the training set, but for simplicity we kept the model as-is.

After the 2018 race was over, we updated the model. Before the race the goal was to help runners plan, whereas after the race the goal is to allow runners to compare their 2018 result with those who ran the most similar to them. We simplified the model a little by not using age and gender as features; only the pacing metrics up until halfway were used. (What this means is that the search results will include runners from different age groups.) We enlarged the dataset by fitting the model with 2010-18 data.

Those who ran in 2018 can search for their closest matches by entering their name or bib number. In fact, one can now search for the pacing profiles of any finisher from 2010-18. In this way, the web app also doubles as a database of 2010-18 results.

Here's a screenshot of the input page:



And here's a screenshot of the resulting search:

## Input: 2018; keflezighi, meb

Halfway pace: 6:55/mi
Best expected: 6:52/mi
Middle 50% of neighbors: 7:13–7:32/mi
Median value: 7:20/mi

### Ordered by similarity

| | BQ | year | bib | name | sex | age | chip | pace | non_missing_split_paces_time | mm | p_1st | p_2nd | 1sthalf | 2ndhalf | splits |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2018 | 8984 | Keflezighi, Meb | M | 42 | 3:00:13 | 6:52 | [6:42, 6:50, 7:05, 7:12, 6:08, 6:34, 7:15, 6:34, 6:52, 6:56] | [] | 6:55 | 6:50 | 1:30:35 | 1:29:38 | -0:57 |
| 1 | 1 | 2011 | 1571 | Bruning, Paul B | M | 44 | 3:12:08 | 7:20 | [6:43, 6:49, 7:02, 7:09, 6:51, 7:18, 7:41, 7:43, 7:54, 8:19] | [] | 6:55 | 7:44 | 1:30:46 | 1:41:22 | 10:36 |
| 2 | 0 | 2012 | 4880 | Sadrian, Luke A | M | 42 | 3:16:22 | 7:29 | [6:42, 6:53, 6:59, 7:12, 6:59, 7:20, 7:43, 8:07, 8:23, 9:02] | [] | 6:57 | 8:02 | 1:31:02 | 1:45:20 | 14:18 |
| 3 | 1 | 2011 | 2827 | Lawrence, Daniel R | M | 35 | 3:15:09 | 7:27 | [6:43, 6:50, 7:04, 7:11, 6:51, 7:21, 7:50, 8:38, 8:00, 7:32] | [] | 6:57 | 7:56 | 1:31:03 | 1:44:06 | 13:03 |
| 4 | 0 | 2018 | 7392 | Gendron, Sebastien | M | 40 | 3:17:45 | 7:33 | [6:40, 6:51, 7:02, 7:11, 7:07, 7:22, 7:43, 8:05, 8:28, 9:54] | [] | 6:57 | 8:09 | 1:31:01 | 1:46:44 | 15:43 |
| 5 | 1 | 2017 | 3190 | Henderson, Pablo | M | 43 | 3:09:17 | 7:13 | [6:42, 6:47, 7:05, 7:15, 6:51, 7:06, 7:16, 7:45, 7:45, 7:31] | [] | 6:57 | 7:29 | 1:31:05 | 1:38:12 | 7:07 |
| 6 | 0 | 2014 | 1435 | Durkin, Kevin M | M | 33 | 3:12:31 | 7:21 | [6:40, 6:55, 7:03, 7:05, 6:54, 7:14, 7:43, 8:02, 7:59, 7:38] | [] | 6:56 | 7:46 | 1:30:48 | 1:41:43 | 10:55 |

The results will show up twice in 2 tables: ordered by similarity (as shown above), and ordered by fastest (cut off from the image). The runner being searched for will appear on the top row. (If he ran with someone else for the first half, his partner might take the top row, since their pacing profiles might be identical.)

To summarize: we turned our capstone project into a web application for users to do the following –

- See what their pacing plan up until halfway can result in for the full distance
- Encourage runners to start more conservatively by seeing how slowly they can go and still have a chance to reach their goals
- Search for finishers from 2010-18 Boston and find their closest matches

As the years go by, we'll have access to more race data with which to update our model and capture more and more pacing profiles and outcomes.