Predicting Boston Marathon Finishing Times

Metis Project Luther - Regression

Mailei Vargas

January 2019

Project Overview

Road running, as of 2015, is a $1.4 billion industry. Many people run for various reasons such as physical/mental health improvements and many runners enjoy the challenge of completing a race in some goal time they have set for themselves. In particular, elite runners make a livelihood from performing well at big races such as the Boston Marathon.

There are many factors that go into the final result of a marathon such as training rigor and nutrition, but if we held these things constant, as a serious athlete would, could a race time be predicted based on prior year results?

Problem Statement

Why would you want to predict a race time? Well, large companies such as Nike and Adidas spend a lot of money in sponsorship of athletes. An algorithm that could predict a race time of a runner could aide one of these companies in selecting the ideal athelte that would maximize exposure of their product since the larger races are aired live and tend to focus camera time on the top runners of the race.

Often the finishers finish within seconds of each other, so predicting race times with such precision will be a challenge. I decided to attack this problem using 4 years (2015-2018) of Boston Marathon finishing times. The data is narrowed down to "legacy runners" (repeat runners) fed into various models using linear regression and stochastic gradient descent. Metrics of R^2 and $RMSE$ (Root Mean Squared Error) are used to measure the results.

Tools

Below is a list of the libraries, sites, and tools that aided in the completion of this project.

Websites	**Tools**	**Libraries for Webscraping**	**Libraries for Analysis/Modeling**	**Custom functions scripts**
Boston Marathon 2018 Finishers	Python	Selenium	numpy	scraper.py
Kaggle	Jupiter Notebook	Beautiful Soup	pandas	data_cleaner.py
Weather Underground	PyCharm	time	matplotlib	modeling.py
	GitHub		seaborn	
			statsmodels	
			sklearn	

Data Preprocessing

Initially there were 4 data sets, one for each year of 2015-2018, with approximately 25k runners per year or 100k runners in total. The following features were given in each set:

- *Bib* - race number determined by the qualifying race time
- *Name* - the name of each runner
- *Age* - age of the runner on the day of the race
- *M/F* - gender of the runner
- *City* - city where runner is from
- *State* - state where runner is from (optional)
- *Country* - country where runner is from
- *Citizen* - country of citizenship (optional)
- *Unnamed* - special category for runners who are visually or mobility impaired (optional)
- *5K, 10K, 15K, 20K, 25K, 30K, 35K, 40K* - the elapsed time at each 5K split of the race
- *Half* - elapsed time at the half way point
- *Pace* - overall average pace of the race
- *Official Time* - finishing time
- *Overall* - overall ranking of all runners
- *Gender* - ranking within the gender
- *Division* - ranking within the age/gender division (for example: Female ages 30-34, Male ages 45-50, etc.)

Feature Enginering and additional data

From the features above *City, State, Country, Citizen, Unnamed, Pace* were dropped because it was determined that there were either too many *NULL* values, too many unique categorical values, or highly correlated with other features (e.g. *Pace* and *Offical Time*).

The *5K, 10K, 15K, 20K, 25K, 30K, 35K, 40K, Half* elapsed times were transformed into differentials and the best fit slope (rate of change of pace) was calculated and added as a feature called *pace_rate* in place of them.

The gender feature was split into 2 variables using dummy variables (0/1) for male and female.

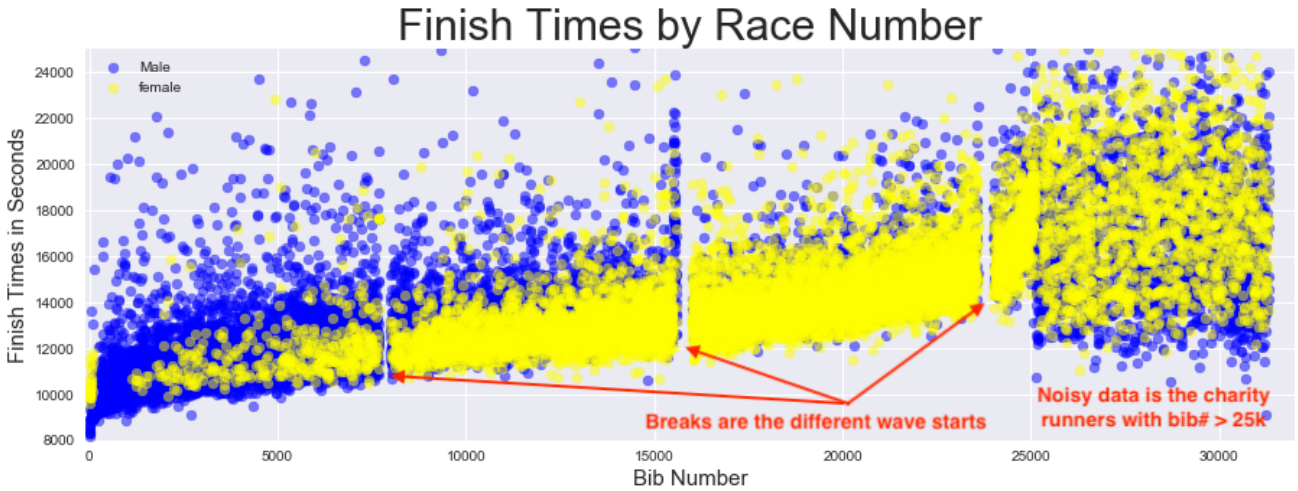
Lastly, historical weather data (*temperature, humidity, wind_speed*) from Weather Underground for each year was collected and added. This was representative of the half way point (Wellesley) and the finish point (Boston) which can vary significantly since the runners are running from inland to the coast.

The final dataset was brought down to only the legacy runners (runners who ran consecutive years) approximately 13k, and the current year finish time was replaced with the consecutive year finish time as the dependant variable and *'Bib', 'Age', 'overall_rank', 'gender_rank', 'division_rank', 'pace_rate', 'temp', 'humidity', 'wind', 'Gender_F', 'Gender_M'* as the independant variables.

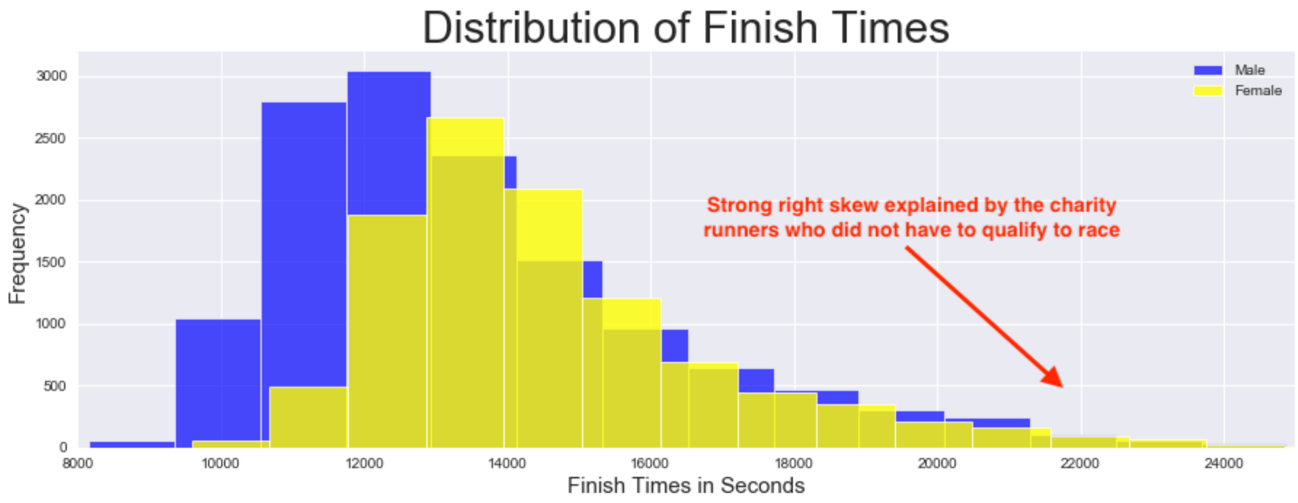
Data Exploration and Visulization

The graph below displays the spread of finish times over the given bib numbers. It makes sense that there is a linear relationship between the bib number and the finish time since the bib number is assigned based on the qualifying time from some other previous qualifying race. The breaks in the bib numbers, which is obvious on the graphs are indicative of the waves that go out at different start times. This is not unusual at large races for safty reasons.

One other interesting observation is the variance in male versus female of finish times. Men on average had a much larger variance of finish times compared to women. Meaning women stay closer to their bib qualifying time than men do. Something for exploration another time.



The graph below displays the distribution of finish times of male versus female. The distribution shape is the same for male/female and right skewed because mostly likely due to the charity runners or the older aged runners. It is interesting to see large gap between the male and female for the faster runners, but as the time gets longer eventually the distribution evens out.



Algorithms and Techniques

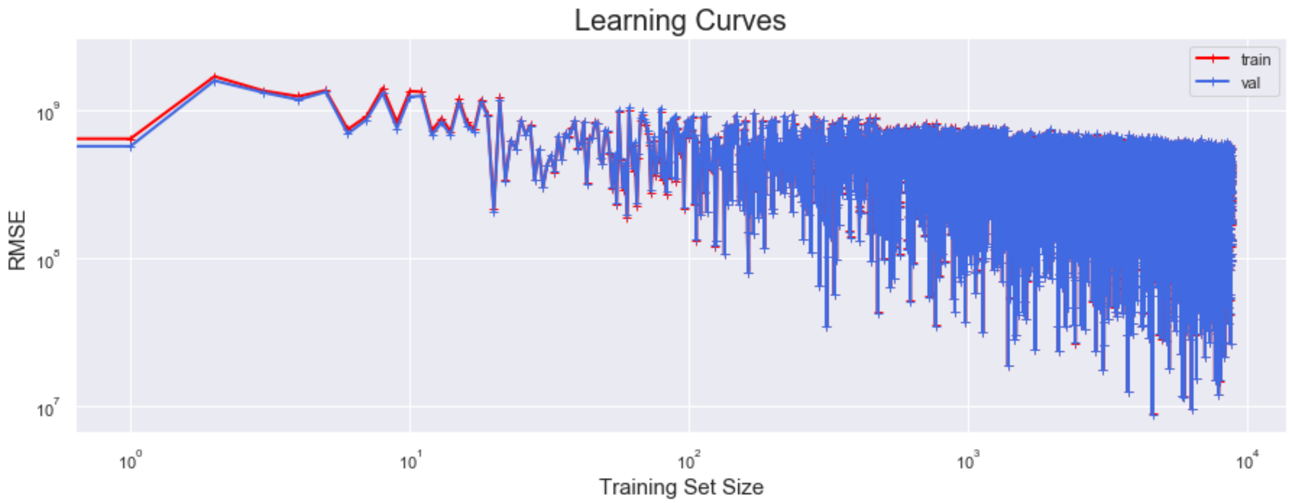
The algorithms that were most used were **Linear Regression** and **Stochastic Gradient Descent**.

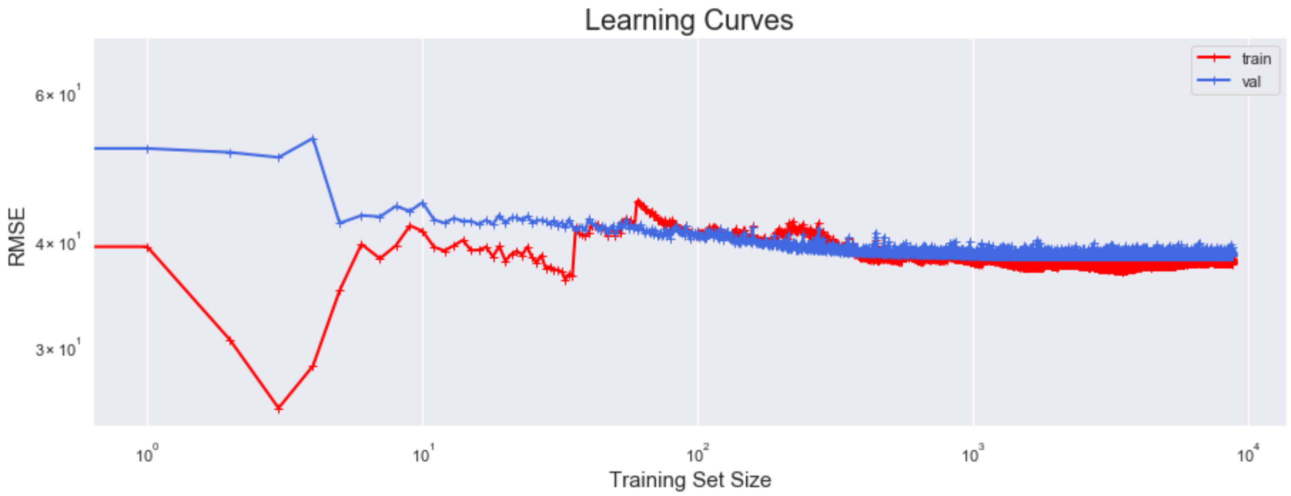
A pipeline was built that took in the features and data and split the data into X/y train/validation/test sets. The train and validation sets were fit onto a model and predictions were made. The predictions were then used to measure the results in terms of R^2 and $RMSE$. Learning curves and residual plots were made for each model in order to visualize the process.

As a benchmark, I first ran a simple linear regression model using all of the independant variables and recorded the results. The goal was to improve the results by reducing the $RMSE$ value and increasing the R^2 value with subsequent models. Those subsequent models inolved cross validation using KFolds and Stochastic Gradient Descent with various combinations of regularization (LASSO and RIDGE), tuning of hyper-parameters, and omitting various independant variables.

Results

In the process of viewing the results of each model a few things are worth pointing out in the graphs below. The 2 graphs are representative of learning curves one which converges the other does not. Both graphs are for SGD and it is interesting to see the behavior when it behaves badly. After scaling the data by using the `StandardScaler()` function in `sklearn` the learning curve behaved better and it convereged.

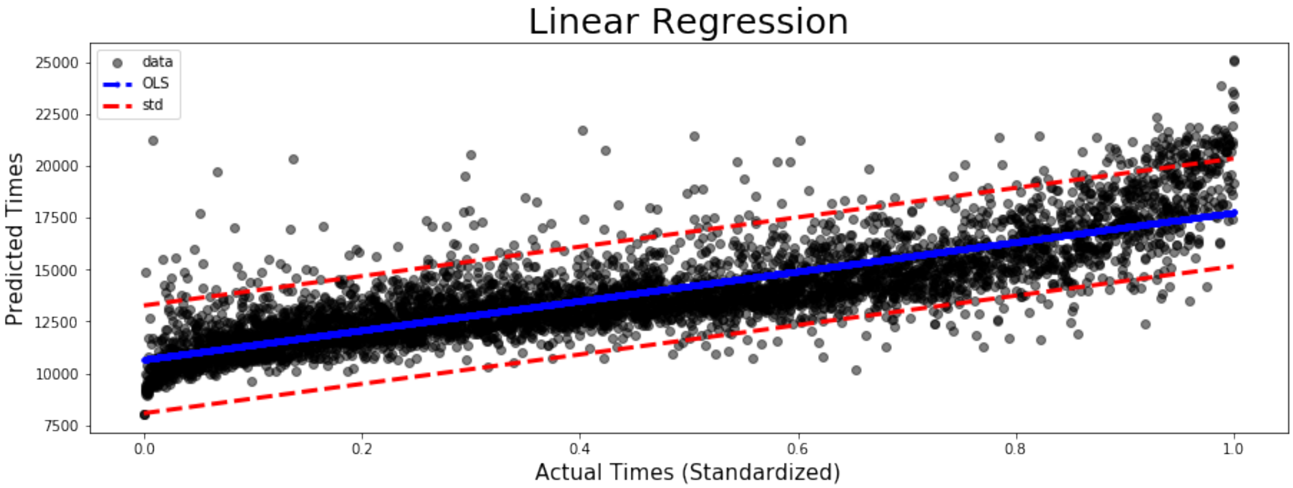




Below is a table of the results of each model:

Model	R^2	$RMSE$	**Minutes conversion**
Linear Regression	0.66	1426.35	23.77
Linear Regression w/CV KFolds=5	**0.69**	**1353.81**	**22.56**
SGDRegressor (No Penalty)	-1.828x10^27	1.05x10^17	1.75x10^15
SGDRegressor - selective X features (No Penalty)	0.63	1486.79	24.78
SGDRegressor - Scaled (No Penalty)	0.65	1458.81	24.31
SGDRegressor w/LASSO regularization	0.65	1463.54	24.39
SGDRegressor w/RIDGE regularization	0.31	2035.24	33.92

Below is the results of Linear Regression actual Vs. predicted values. Notice how most of the points fall within one standard deviation (red dashed lines) of the regression line (in blue).



Conclusions

Given more time...

If there were more time, it would be interesting to explore some other avenues of the dataset. These are things I think of in hind-sight reflecting back...

- Focus the dataset on only the elite runners to see wht kind of predictive power that gets.
- Train the model seperately on male versus female.
- Remove the charity runners since their data is less linear and more "noisy" compared to the qualified runners.
- Add more data fields such as the elevation and population of the city a person is from.
- Add interactive terms between gender and ranks.
- Take the time to explore and remove possible outliers seen in residual plots.
- Add more years to the dataset going back to 2000.