Speed Ratings

John Farrell, Brown University, GitHub[1]

1. Introduction

In the early 1970s, Andrew Beyer developed the first speed ratings, the Beyer Speed Figure, for horse racing. The purpose of these figures is to normalize horse race results and compare performances among different races[2]. This comparison is helpful when betting on horses, as you can compare two horses that haven't raced head to head as if they did. In the early 2000s, Bill Meylan published his first speed ratings for high school cross country. These ratings follow a pattern similar to the Beyer Speed Figure and hold a similar purpose[3]. They are used to compare the performance of one runner to another who ran on different days, in other conditions, and on various courses. Over the last 20 years, Meylan has continually improved his race comparison statistics, becoming a standard for comparing and recruiting cross-country athletes[4].

The only issue with speed ratings is their scarcity. Because Bill Meylan is the only person trained to produce speed ratings, he must choose which races to be rated. Therefore, many high school athletes go their entire cross-country career without a speed rating, robbing them of recruitment chances. This project aims to predict all speed ratings for any cross-country result in New York State (NYS), hoping to give more high school athletes the opportunities that come with having a speed rating.

The dataset consists of all NYS speed ratings from 2014 to 2019, provided by TullyRunners[5], race-day weather data from the National Centers for Environmental Information's (NCEI) public API[6], and course distances and locations from MileSplit[7]. The compiled data set is 330108 rows × 26 features (columns). The target variable, speed rating, is a continuous numerical value, making this a machine-learning regression problem with a time dependency due to race performances happening sequentially.

2. Exploratory Data Analysis

Performing data analysis will give an insight into the underlying structure of the compiled dataset and what patterns and trends are present, enabling more informed decision-making while developing a machine learning pipeline. The scatter plot below (*Figure 1*) displays the linear relationship between time and speed rating. The formula for a speed rating is,

Speed Rating = (1560 - (actual race time in seconds) - (course correction factor)) / 3 [3]

where the *course correction factor* is a subjective number, accounting for weather, competition, course difficulty, and any other factors affecting raw performance. The plot also shows shorter races tend to have lower speed ratings than longer races for a given finishing time.

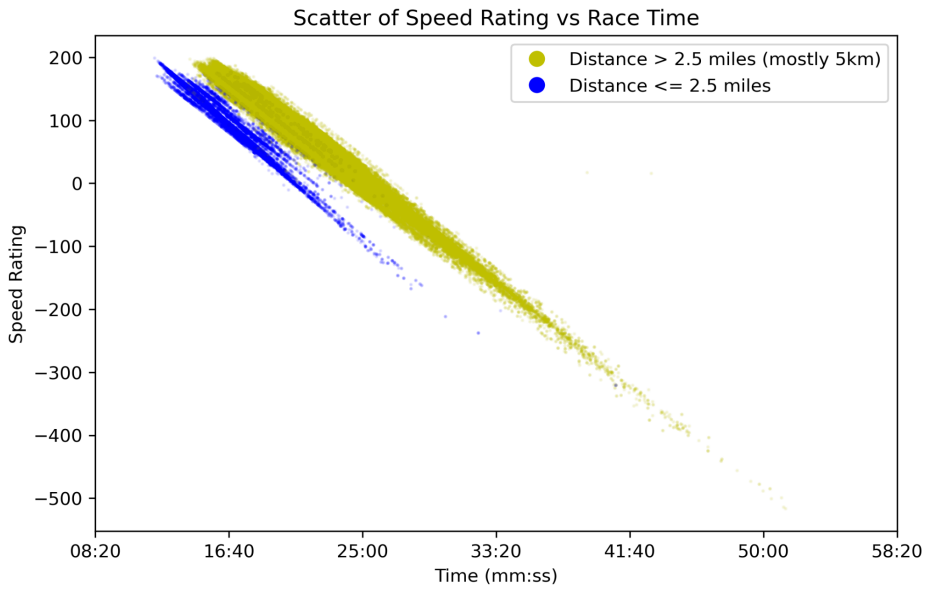

Figure 1. Scatter plot showing the linear relationship between race time and speed rating, with coloring to differentiate races shorter than 2.5 miles (blue) from those longer (yellow).

The following plot (*Figure 2*) displays the distribution of speed ratings among genders. Note that while there is a bimodal distribution, and males and females have two different average speed ratings, this is because men and women have two other average 5km race times, not because speed ratings are calculated differently per gender. The distribution of speed ratings skews left.

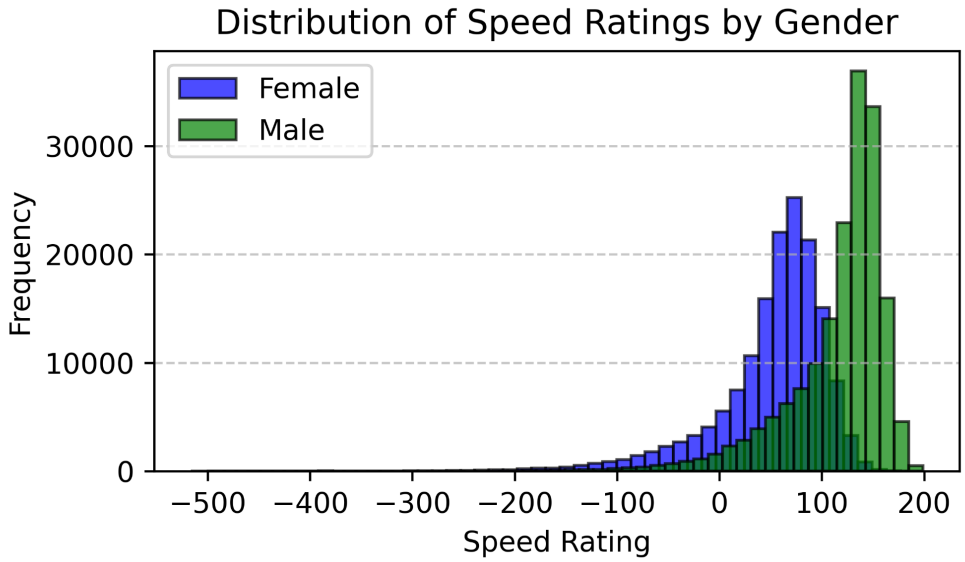

Figure 2. Histogram showing the distribution of speed ratings among female (blue) and male (green) runners, revealing a different average speed rating among genders, and therefore, a bimodal distribution of speed ratings.

The following plot (*Figure 3*) displays several years of finishing times and places for the McQuaid Invitational[8], one of the largest invites in the northeast. Speed ratings for several runners who ran the same times in different years are labeled. The 2019 race was considerably "slower"; therefore, runners who ran simultaneously in previous years got a speed rating 3 points lower. Determining what makes a race "slow" makes speed ratings hard to calculate, as it could be a combination of factors.

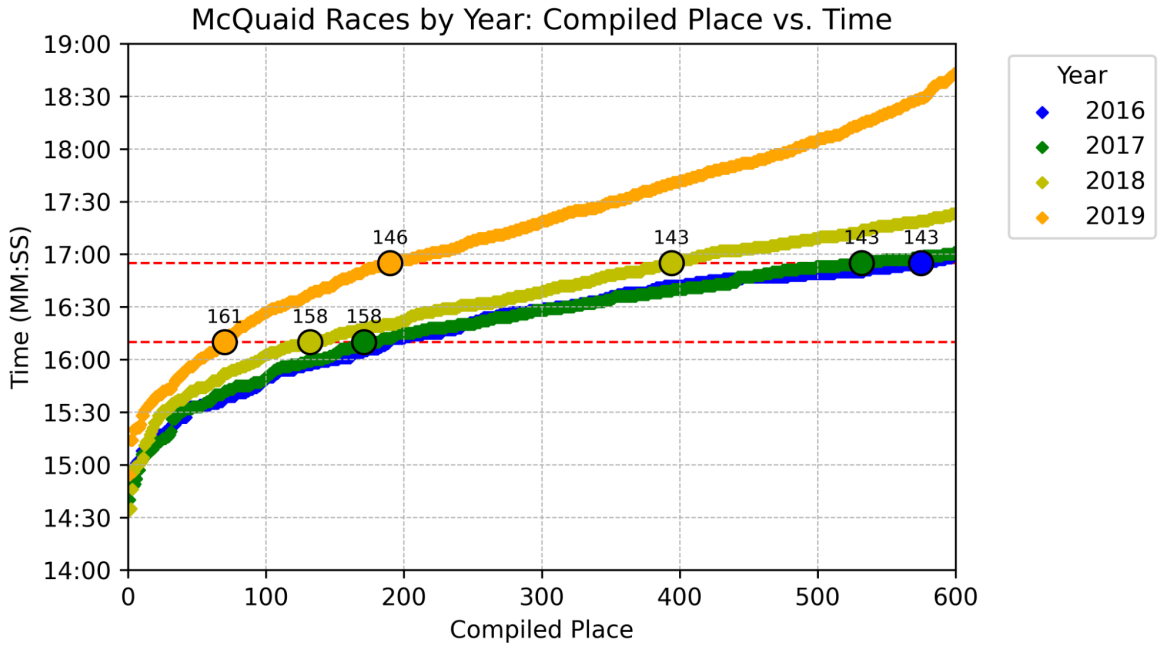

Figure 3. Scatter plot comparing the finishing time and place for runners of the McQuaid invitational, color labeling per year, where runners in 2019 got a speed rating 3 points higher due to a "slow" race.

The following plot (*Figure 4*) shows a distribution of temperature labeling undefined and defined heat index values. There is a cutoff for defined heat indexes at 80°F, so assume that any undefined heat index is where the heat index does not exist. Therefore, filling missing heat indexes with the actual temperature for that point is sufficient.

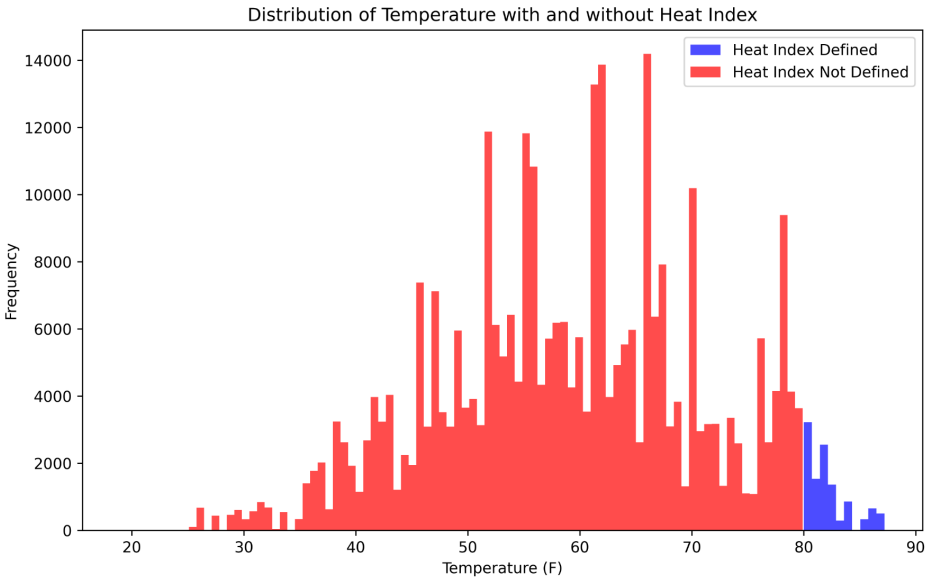

Figure 4. Distribution of temperatures, color labeling defined and undefined heat indexes.

The following plot (*Figure 5*) displays a scatter of temperatures versus wind speeds, labeling undefined and defined wind chill values. Like the heat index, conclude that all undefined wind chill values occur when there is no wind chill. Therefore, filling missing wind chills with the actual temperature for that point is sufficient.

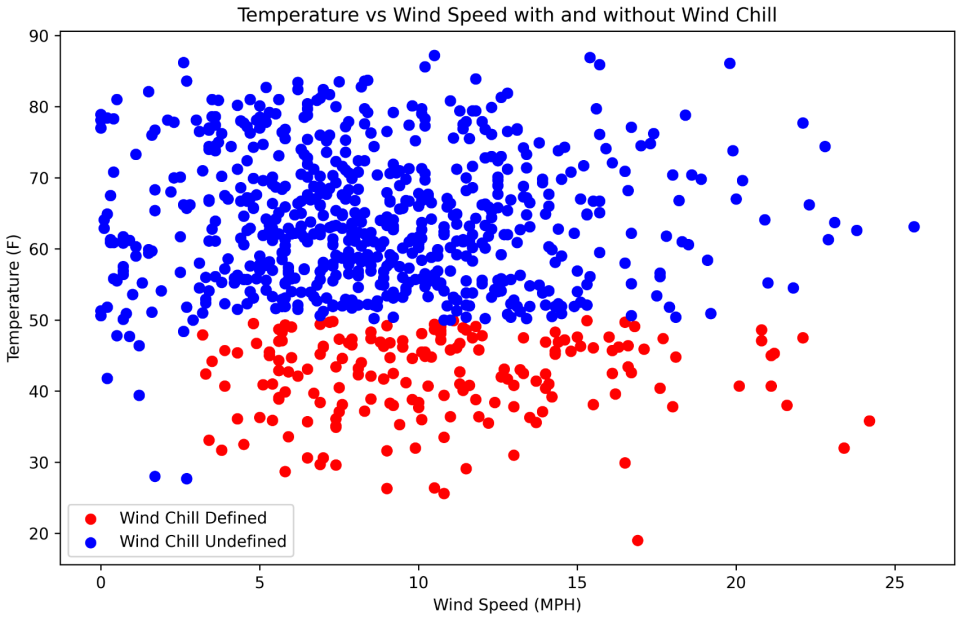

Figure 5. Scatter plot of wind speed versus temperature, color labeling defined and undefined wind chill values.

The last plot (*Figure 6*) displays a scatter of temperatures versus wind speeds, labeling undefined and defined wind gust values. Because there is no clear pattern for when wind gusts are undefined, XGBoost will be used to fill the missing values.

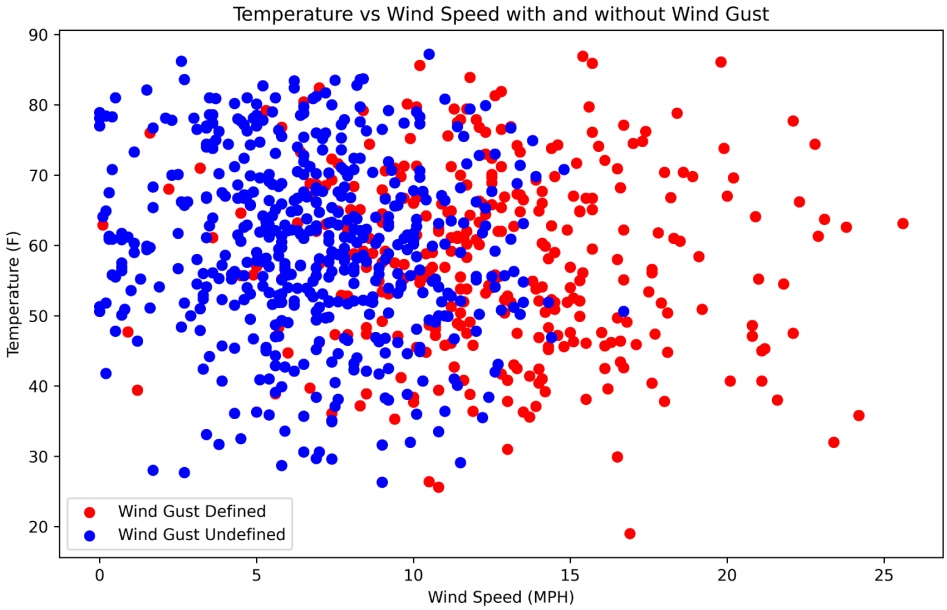

Figure 6. Scatter plot of wind speed versus temperature, color labeling defined and undefined wind gust values.

3. Methods

With the established background on the dataset, it is time to develop a pipeline that includes splitting, preprocessing, and hyperparameter tuning. The pipeline trains and evaluates the performance of four models: Elastic Net, K-Nearest Neighbors, Random Forest, and XGBoost. Since the goal is to predict speed ratings with minimized prediction error accurately and weigh larger errors, root mean squared error (RMSE) is the test statistic.

3.1 Splitting

Because the dataset is time series, the splitting strategy is to split by year. This strategy will also prevent possible data leakage, where one race ends up in multiple sets. Below (*Figure 7*) is a diagram of the four iterations of time series splits used.

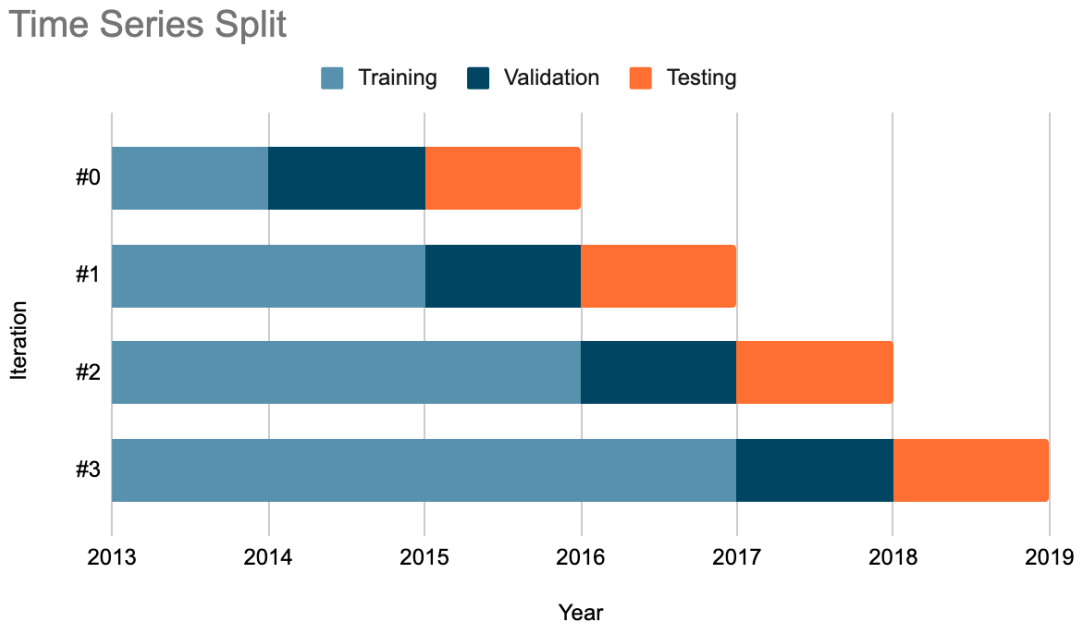

Figure 7. Time series splits for the cross validation pipeline, including 4 iterations where the training set is extended by a year each iteration.

3.2 Preprocessing

The dataset includes numerical, ordinal, and categorical features which are preprocessed with the Standard Scalar, Ordinal Encoder, and OneHot Encoder, respectively. Those column transformers handle ordinal and categorical missing values. Missing heat index and wind chill values are filled before the pipeline, with real temperatures, and missing wind gusts are filled using XGBoost. After preprocessing and feature engineering, there are 56 features.

3.3 Cross Validation and Hyperparameter Tuning

Next, the best hyperparameters are determined using cross-validation. This hyperparameter tuning aims to find the combination of parameters that maximize the model's performance. The idea is to cycle through each iteration of the time series split and determine the best parameters per iteration based on the lowest RMSE validation score of all hyperparameter combinations. The best model is the model with the lowest RMSE validation score among all iterations. Then, save the best model and corresponding RMSE test score.

3.4 Pipeline

Putting these pieces together, below (*Diagram 1*) is a visual of the pipeline. The cross-validation is performed over five random states to ensure the generalizability of results and determine uncertainties due to non-deterministic models and the time series split iterations. There will be five best models and test

scores corresponding to the five random states, of which the mean and standard deviation test scores are determined to compare different algorithms.

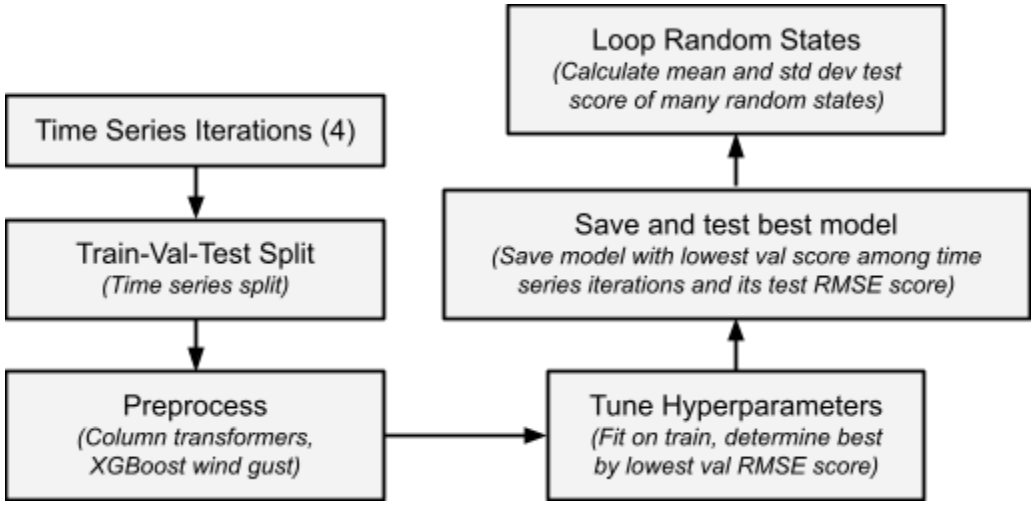

Diagram 1. Complete model training and cross validation pipeline.

The models trained and hyperparameters tuned are below (*Table 1*).

	ElasticNet	**K-NearestNeighbors**	**RandomForest**	**XGBoost**
Parameters	`alpha(logscl), l1_ratio(linscl)`	`n_neighbors(linscl), weights(uniform,distance)`	`max_depth(linscl), min_samples_split(linscl), max_features(linscl)`	`learning_rate(linscl), max_depth(linscl), min_child_weight(linscl)`
Optimal Parameters	`alpha=1.0 l1_ratio=1.0`	`n_neighbors=13 weights=distance`	`max_depth=15 min_samples_split=6`	`learning_rate=0.1 max_depth=5 min_child_weight=3`

Table 1. Models trained and hyperparameters tuned.

4. Results

Once the best models and test scores for all random states are determined, the next step is to calculate all test scores' mean and standard deviation. Below (*Figure 8*) is a boxplot comparison of RMSE scores for all models. All models performed much better than the baseline RMSE of 68.31, and random forest performed the best with the lowest mean RMSE of 7.47. Therefore, random-forest is the best model.

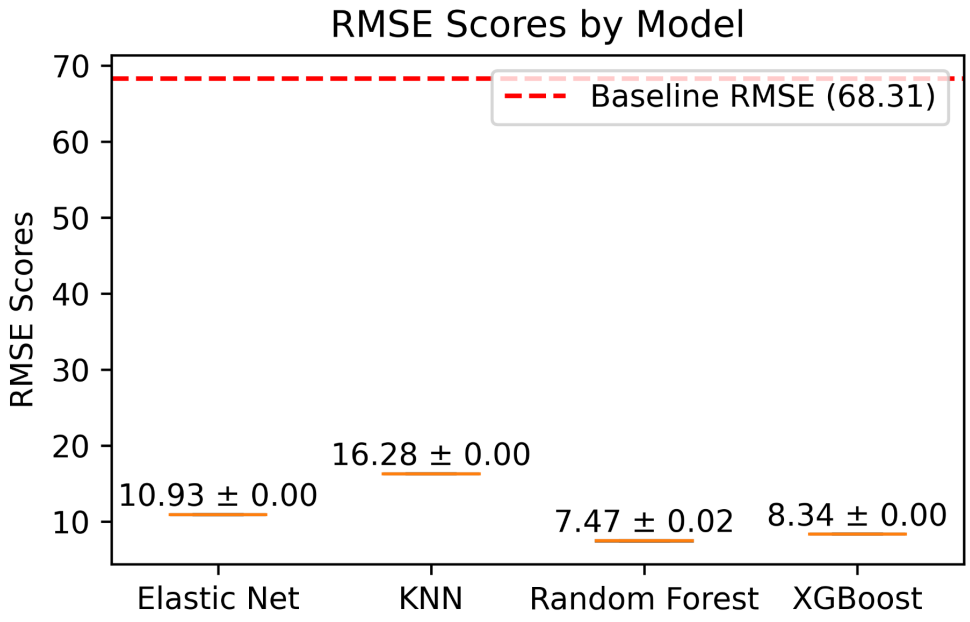

Figure 8. Boxplot comparison of RMSE scores for models and baseline.

4.1 Global Feature Importance

Global feature importance can be used to gauge features impact on model's predictions. These three importance metrics are performed on the random forest model:

1. Permutation feature importance: estimates influence of features based on decreased model performance due the features random perturbation.
2. Random forest feature importance: are obtained directly from the model, based on how much each feature decreases impurity.
3. Shap values for global feature importance: quantifies feature importance using game theory principles.

Below (*Figure 9*) are the top 10 features according to permutation feature importance. Unsurprisingly, time and speed are the most important factors, with various feature engineered versions of time and speed following.

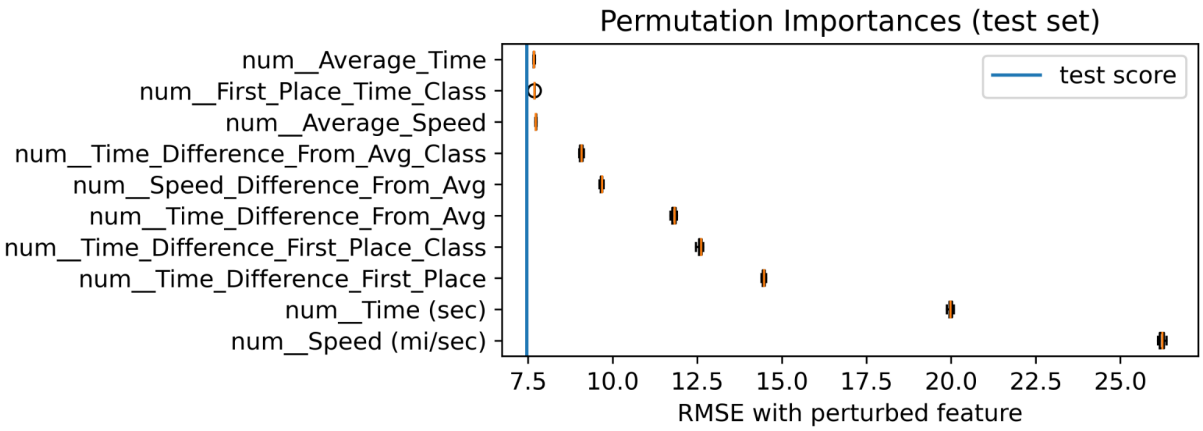

Figure 9. Top 10 features according to permutation feature importance.

Below (*Figure 10*) are the top 10 features according to random forest feature importance. Again, speed and time are the most important, with variations of speed and time following. Interestingly, gender holds high importance for this metric.

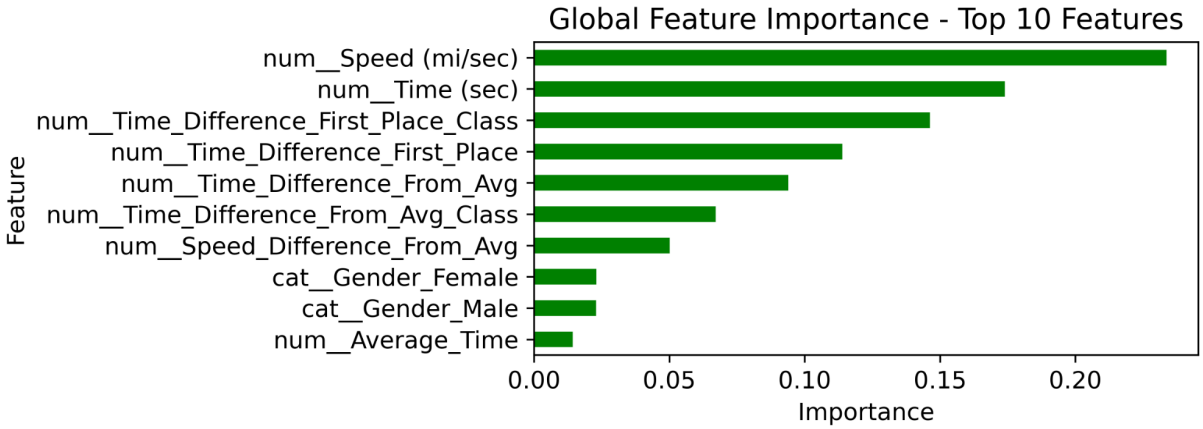

Figure 10. Top 10 features according to random forest feature importance.

Below (*Figure 11*) are the top 10 features according to calculated Shap values. The Shap global importance plot and random forest feature importance look similar, with many of the same top features in a somewhat shuffled order. Again, gender holds high importance for this metric.

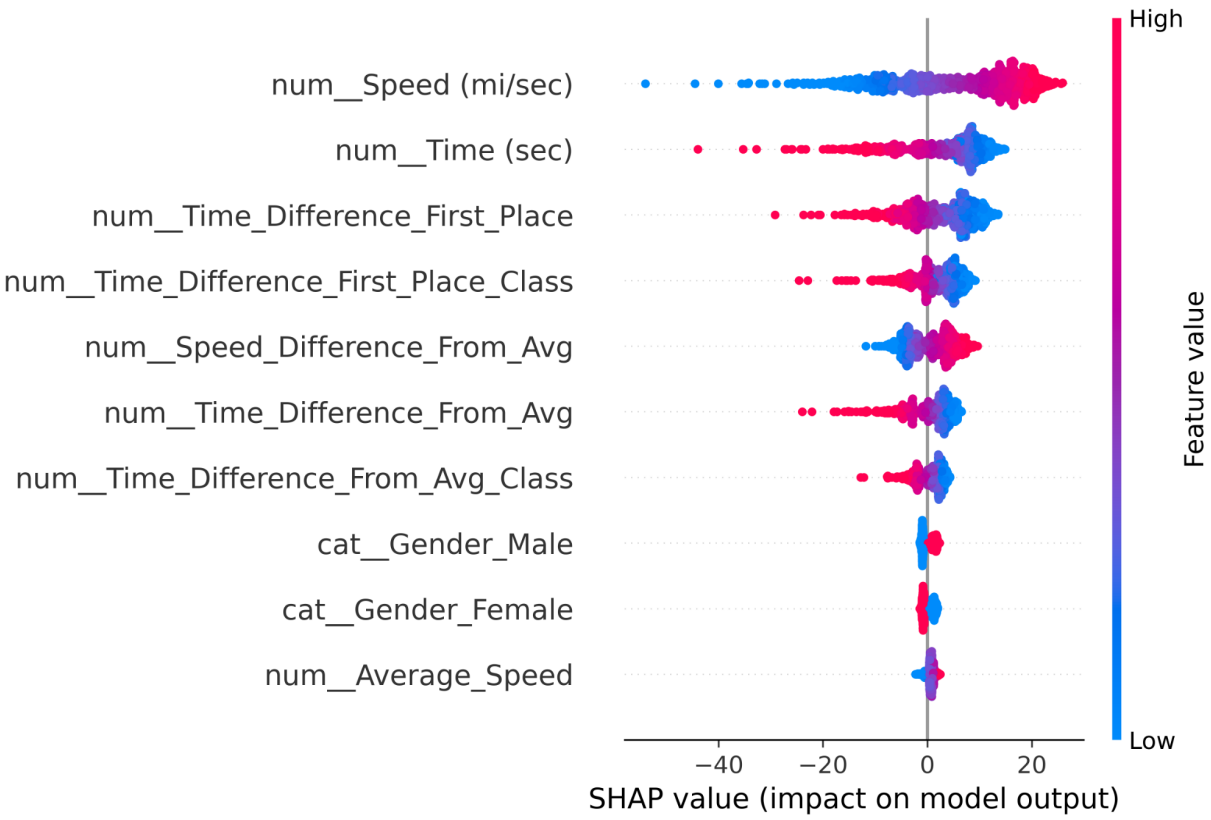

Figure 11. Top 10 features according to calculated Shap values.

Key Observations:

- Top features were similar among all three metrics, including speed, time and their feature engineered variations.
- For random forest feature importance and shap values, gender also played a significant role in predicting a speed rating.

4.2 Local Feature Importance

Local feature importance metrics measure the contribution of features to a specific data point. Below are Shap force plots for two different speed ratings. For the low predicted speed rating (*Figure 12*), speed and time were the most essential features for decreasing from the base value. For the high predicted speed rating (*Figure 13*), the difference in speed from the average runner in the race was the most crucial feature for decreasing the base value. Dew point × temperature and the race month were the two most essential features for increasing from the base value. The key observations from local feature importance are that:

- Time, speed, and their variations are the most important features for decreasing from the base value speed rating

- Many features account for increasing from the base value, including date and weather variables such as month and dew point × temperature. These features aren't displayed within the top 10 global importance feature metrics.

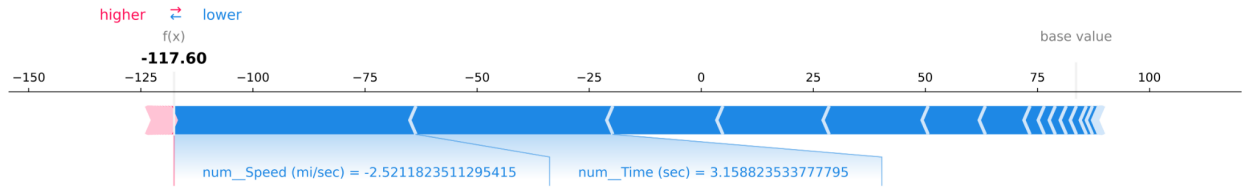

Figure 12. Local feature importance shap values for a predicted speed rating of -117.60.

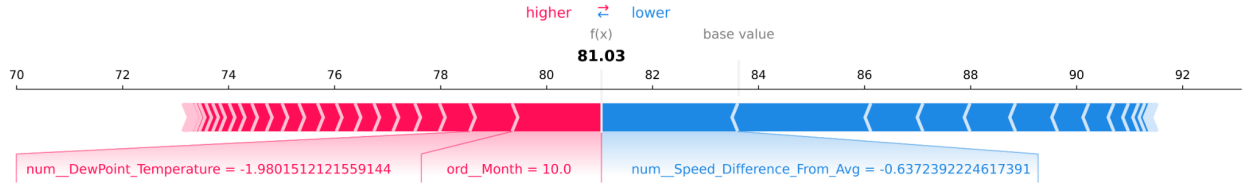

Figure 13. Local feature importance shap values for a predicted speed rating of 81.03.

5. Outlook

Further steps that could be taken to improve the model's predictive power and interpretability:

- Train the models on unique racer and course IDs. These IDs would improve the model's ability to predict a speed rating by identifying courses and racers on their previous data, also following sequential data modeling and improving the model's predictive power.
- Improve feature engineering to include more combinations and variations of speed and time. Speed, time, and their variations were consistently the most important global features, so including more variations could improve the model's predictive power.
- To improve interpretability, evaluate feature importances after dropping highly correlated features to evaluate feature interaction.

6. References

[1] GitHub repository – https://github.com/johnsfarrell/Speed-Ratings-DATA1030

[2] Figure Makers: An Open Mind Rules The Beyer Method
Scott Jagow

[3] Standardized Race Time Ratings
Bill Meylan

[4] Running the Numbers
James Costanzo

[5] TullyRunners - https://tullyrunners.com/

[6] NCEI API - https://www.ncdc.noaa.gov/cdo-web/webservices/v2

[7] MileSplit - https://www.milesplit.com/

[8] McQuaid XC Invitational https://mcquaid-xc-invitational.runnerspace.com/