Investigation of Methodologies for Predicting Individual Running Times

Michael Brown*

April 2018

Abstract

This project is an exploration into the methods used in predicting athletic performance in runners. Its intention is to compare predictive performance between established state-of-the-art techniques in the field and adaptations of common state-of-the-art techniques in machine learning. These adaptations provide a novel approach for prediction in the running domain. All of the methods considered are specific to the running domain, either having been created solely for or adapted to this particular task.

*Supervised by FJ Király

Contents

1 Introduction

Running calculators are a widely used tool for predicting running times for an athlete from his or her known performances. They serve a variety of purposes including helping devise training routines for competitive athletes and target setting for amateur athletes. Running calculators typically rely on prediction methods specific to the domain and can give accurate answers with relative parsimony. Whilst some calculators can be adjusted for factors such as age and gender, the prediction methods we consider in this study use only past performance as input and will be examined solely for male performance. Many of these calculators focus on middle to long distance running; we will consider all distances from sprints to marathons.

Much of the work of this project is based around Blythe and Király's *Quantification of Individual Athletic Performance in Runners*[1]. Both works make use of the same dataset, described in further detail in section *3. Data and Exploratory Analysis*, and aim to predict running performance from past performance alone. Both works explore possible applications of machine-learning methodology. The questions we intend to answer in this study are: Is there a clear and obvious best performer out of the predictive methods prevalent in the literature? Can we find new methods, drawing from techniques in machine-learning, that perform better than those prevalent in the literature?

2 Summary of the Literature

There has been much research into the relationship between performance time and distance in running over the past century. This section gives a brief overview of some of the most important themes of past research in the running calculator domain. It includes outlines of the some of the methods employing these themes and an overview of two machine-learning strategies either already introduced in the literature, specifically Blythe and Király's Local Matrix Completion scheme, or to be employed in this project.

2.1 Power Laws: Lietzke, Henry and Riegel's Formula

The presence of a power law relationship between time and distance has long been acknowledged in athletic performance. The most general formulation of the power law is as in equation 2.1.1: for distance d, time t and constants c and α the relationship is:

$$t = cd^{\alpha} \tag{2.1.1}$$

It was first acknowledged by Kennelly[10] as far back as 1906 when studying the relationship between distance and world record time in both human and horse racing; this simple model is sometimes referred to as Kennelly's model.

In their early application, power laws were thought to be restricted to optimum athletic performance and thus much of their early use was focused on modelling world record performance, where they were shown to provide good approximations. Lietzke's[9] 1954 review of the usage of power laws concluded that they were yet to have been successfully employed in the running calculator domain; whilst there was evidence of good prediction over long distances, they were less effective in predicting times over short distances. This is because distance vs. time of record performances, when viewed in log-coordinates, displays a kinked straight line, implying some form of broken power law. To address this, Lietzke's solution was to model the constants c and α in equation 2.1.1 as piecewise constants determined by separate Ordinary Least Squares regressions. This extended the useful reach of the power law to distances as short 440 yards (˜400m) but could not account for remaining inaccuracies over shorter distances.

In 1955 Henry[8] attempted to explain the kinked nature of the distance-time curve as the result of a combination of physiological processes, modelling them as a linear combination of exponentials. This was the first attempt to account for the observed broken power law using offsets; his model represents a power law with small continuous offsets over the short-middle distances.

One obvious short-coming with these methods is their intended use as models for record performances; individual athletes distance-time curves do not approximate as strongly to power law curves as they tend to specialise, both in terms of physiology and training, over a subset of running events.

In 1977 Riegel[3] extended the use of power law modelling to individual performance, proposing a parsimonious relationship dependent on an athlete specific coefficient:

$$t_2 = t_1 \times \left(\frac{d_2}{d_1}\right)^{1.06} \tag{2.1.2}$$

His equation relates time and distance of a performance to be predicted, t_2 and d_2, to time and distance of a known past performance, t_1 and d_1; this equation is often referred to as Riegel's formula. The exponent, 1.06, is the least-squares estimation from world record performances at the time of inception. As Riegel's formula has no adjustment for the observed broken power law, he later suggested[2] the formula be used only for endurance events, those ranging between 3.5 to 230 minutes, where observed performance better approximates a straight line in log coordinates. Despite this, due to its simplicity and empirical accuracy Riegel's formula has remained prevalent in athletic performance study and its variants remain among the most common methods used in simple online running calculators.

2.2 Scoring Tables and Purdy Points

Another theme central to much of the research into athletic performance is equivalent scoring. These approaches attempt to provide quantifiable and empirically derived comparison of achievement across differing distances or sports. The first known scoring table was published by the IAAF in order to compare performances in the 1912 Stockholm Olympics decathlon. Other noteworthy later additions to the field include the Portuguese Tables[6], devised by the Federacao Portuguesa de Atletismo, which are accompanied by the Portuguese Table of Speeds, a table of maximum theoretical average velocity over a given distance.

In their 1970 paper Gardner and Purdy[5] propose a method for computing equivalent scoring tables; these tables have come to be known as Purdy Point Tables. Using the Portuguese Table of Speeds as a basis, they present an algorithm for computing equivalent level performance times discounted for reaction time, acceleration time and track curvature. For a reaction time delay, C_1, constant of proportionality for the delay due to accelerating to maximum velocity, C_2, constant of proportionality for the delay due to track curvature, C_3, and fraction of distance spent running on the curved portion of the track, $f(d)$; standard time for a performance, T_s is modelled as:

$$T_s = t + C_1 + C_2 V + C_3 f(d) V^2 \tag{2.2.1}$$

where $t = d/V$ for a given distance and for the corresponding V extracted or interpolated from the Portuguese Table of Speeds. The coefficients C_1, C_2, C_3 are also inferred from the Portuguese Table of Speeds, though in a later paper Purdy notes "more study must be done to establish the best coefficients in [2.2.1]". The full paper, *Computer Generated Track and Field Scoring Tables*, reports the complete algorithm used to determine the tables.

If a point level is assigned to the standard time T_s, one can use equation 2.2.2 to determine the relationship between time, T_P, and point level, P, for a different performance. Constants A and B are chosen such that there is a sliding scale between points awarded and speed relative to reference speed. Purdy and Gardner assign the 950-point level to the standard time, derived from the Portuguese Table of Speeds using equation 2.2.1. Thus equations 2.2.2 - 2.2.5 uniquely determine a point level for a given time and vice versa, and one can construct an equivalent scoring table.

$$T_P = AT_s/(P + AB) \tag{2.2.2}$$

$$0.0654–0.00258V = K \tag{2.2.3}$$

$$85 = AK \tag{2.2.4}$$

$$B = 1–950/A \tag{2.2.5}$$

Using equivalent scoring tables one can predict an individual's running performance in a distinct event as the time equivalent to the point level achieved in a known previous performance over any distance. There are some obvious shortcomings associated with these predictive methods, none more so than their failure to consider event specialisation: an individual athlete is unlikely to perform at a constant level over varying distances as runners specialise in a subset of events. Nevertheless, equivalent scoring prediction methods, and Purdy in particular, can result in very accurate predictions and thus remain state-of-the-art in individual athletic performance prediction to this day.

2.3 Prediction and Quantification of Individual Athletic Performance in Runners

Blythe and Király's paper introduces a novel method, using contemporary machine-learning techniques, for prediction of individual athletic performance. The data studied is sourced from the online database, *thepowerof10.info*, and contains records of times achieved by individual British athletes in any official UK Athletics ratified competition (for further analysis of the data, see 3. Exploratory Analysis below). The model fits the data in matrix form with columns corresponding to distinct events, rows corresponding to distinct athletes and entries corresponding to a single performance such that a given row will contain an entry for every event attempted by the athlete corresponding to that row. Events that have not been attempted by an athlete are represented by missing entries.

The model uses Local Matrix Completion, a method for completing matrices with non-uniform missing values, to impute performance times for events not previously recorded for each athlete. This process requires compiling many sub-matrices with a single missing value and solving the matrix for determinant zero. For a system of linear equations to have determinant zero means there is linear dependence between them; each row of the system is a linear combination of the system's other rows. In the case of individual athletic prediction this means that event specialisation can be accounted for. This process is repeated for many single missing entry sub-matrix formulations; the final imputation is a weighted geometric mean of these.

The low-rank model obtained from this process has rank r, with $n = r + 1$ for size n sub-matrices (thus r is a model parameter), where the rank describes the number of independent components of the model. The paper determines $r = 3$ to be the best model in this instance. The resulting model has the form:

$$\log t = \lambda_1 f_1(d) + \lambda_2 f_2(d) + \cdots + \lambda_r f_r(d) \tag{2.3.1}$$

where $\lambda_1, \lambda_2, \ldots, \lambda_s$ are runner specific coefficients (labelled the runner's 3 number summary in the case of $r = 3$) and $f_1, f_2, \ldots, f_s$ are model specific components, derived from singular value decomposition of the completed data matrix. The first component approximates to a power law

relationship ($R^2 \approx 0.99$), and thus to some extent the model can be thought of as an adjusted power law.

2.4 Random Forests

Random Forests is a machine-learning technique developed by Breiman[11], and is an ensemble of multiple simple decision trees. Whilst single decision trees have many desirable qualities, they are prone to overfitting their training set and typically result in low-bias, high-variance estimators; ensemble tree methods average over many single decision trees to give estimators with reduced variance. Each single decision tree is trained on a random with replacement subset of the training data in order to reduce correlation between trees. Breiman's Random Forests uses a random subset of features at each node split, further reducing correlation between trees and thus leading to a lower overall variance meta-estimator.

Random Forests can often result in very accurate predictive models, though this comes at the cost of limited interpretability. Given the already extensively researched application domain, we may be willing to accept this interpretability loss if it allows more accurate prediction; many of the best explanatory running calculator models are based on physiological athlete characteristics which are not widely available, potentially leaving room for a blackbox model with powerful predictive ability.

This is the first application, to knowledge, of Random Forests in the running calculator domain.

3 Data and Exploratory Analysis

The dataset used in this project is taken from *Prediction and Quantification of Individual Athletic Performance in Runners* where it was cleaned and pre-processed. It was originally sourced from *thepowerof10.info* and contains records of all individual running performances by British athletes in official UK Athletics (UKA) ratified competitions dating back to 1954. In *Prediction and Quantification of Individual Athletic Performance in runners* it was cleaned and reformatted. It contains performances over 10 distinct events; 100m, 200m, 400m, 800m, 1500m, 1 Mile, 5km, 10km, Half Marathon and Marathon, and from athletes across the spectrum; amateur to elite, young to old and both males and females.

Outliers were determined by calculating an outlier score, the difference between maximum and minimum performance percentiles for all events run by a particular athlete, and removing the athletes falling in the top five percent. The rows corresponding to athletes who had fewer than four records were also removed. We further removed the bottom 1 percent of athletes based on their lowest performance percentile. For the predictive experiments we consider only male athletes. After removal the dataset contains 53,788 male performances.

3.1 Exploration of the Features

The exploratory analysis for this project was exported to R. The dataset contains information on 7 variables: athlete ID, sex, date of birth, a numeric code corresponding to the event run, distance of the event, the recorded time, the date of the record. Athlete ID is a unique number, ranging from 1 to 25,515, prescribed to each unique athlete in the database. Before error and outlier removal the dataset contains 31,452 and 58,564 female and male records respectively. The date of birth variable ranges from 21/06/1913 to 03/08/2000 and contains 01/01/1901 entries as place-holders for missing values. Excluding the 1 Mile with only 1,585 records, there are a similar number of records for each event, ranging from 7,866 for the 100m to 13,113 for the 10,000m. Distance is measured in metres and ranges from 100 to 42,195. Time is measured in seconds and ranges from 10.12 in the 100m to 30,623 in the marathon; its distribution is discussed in greater detail in 3.2 Exploration of Time. Date refers to date of recording and ranges from 1959 to 2014, though its distribution is significantly skewed towards recent recordings with a median in 2011 and lower quartile in 2009. It is perhaps worth considering this skew further if we further reduce the data by performance level.

3.2 Exploration of Time

As time is the variable of interest in this study, this section details a more in depth exploration of its distribution and characteristics. Figures 3.2 and 3.3 show scaled and unscaled distributions of time for both male and female athletes. When considered by event separately, time shows an approximate positive skew normal distribution. This skew is possibly explained by the existence of a lower boundary imposed by the limits of human physiology (not to mention the strictly positive domain of possible values). A Shapiro test was conducted on log transformed time for each Sex, Event subset to test for log-normality: each rejected the null with at least *2e-08* significance. The data show rough agreement with Riegel's hypothesised 1.06 power law (minor discrepancy may be attributed to changes to the distributions of performances over time, Riegel's exponent was calculated in 1977) when plotted in log coordinates; median and record times are plotted with a line of slope 1.06 in figure 3.1.

3.3 Male vs. Female Performance

Mann-Whitney U tests were employed to investigate whether or not male and female performance varied significantly. The test is non-parametric and has a null hypothesis that the two samples, male and female, are collected from the same population. The test was rejected at all 10 events with a significance greater than *2e-16*. Perhaps of more interest is whether the distributions of times are comparable when standardised. When the Mann-Whitney U tests are repeated with scaling (subtract the sample mean, divide by sample standard deviation), the tests are rejected only at the 100m and 200m events. Thus for the remaining 8 events scaled male and female performance is comparable. This means methods considered here for male athletes could be easily generalised to female athletes in most cases, though this would be true in most cases anyway as there are few distributional assumptions made.

Figure 3.1: Riegel's exponent in log-coordinates

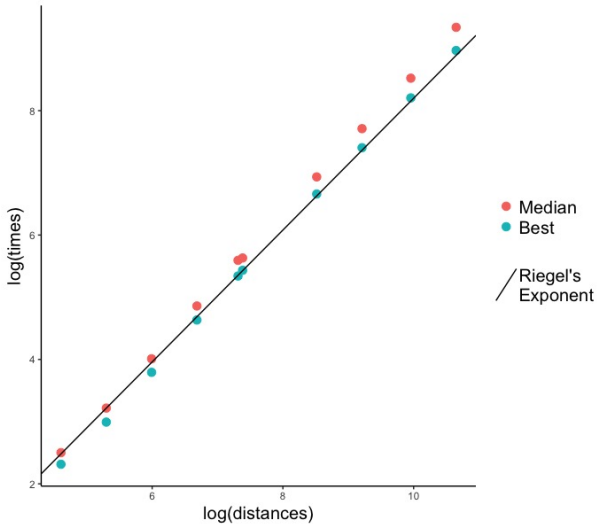
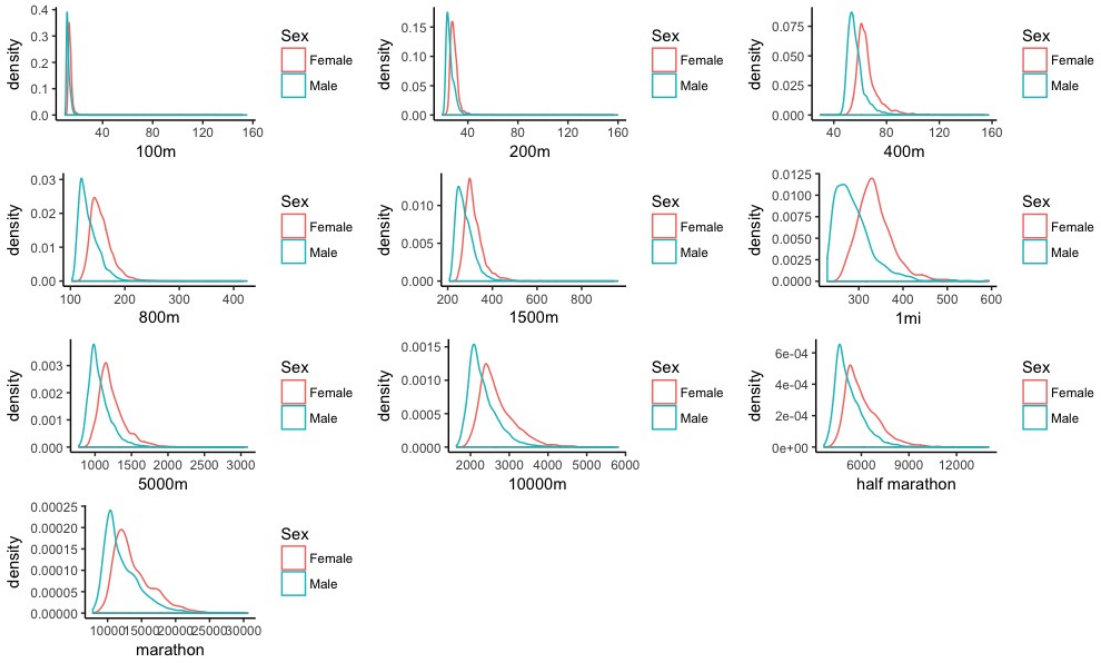


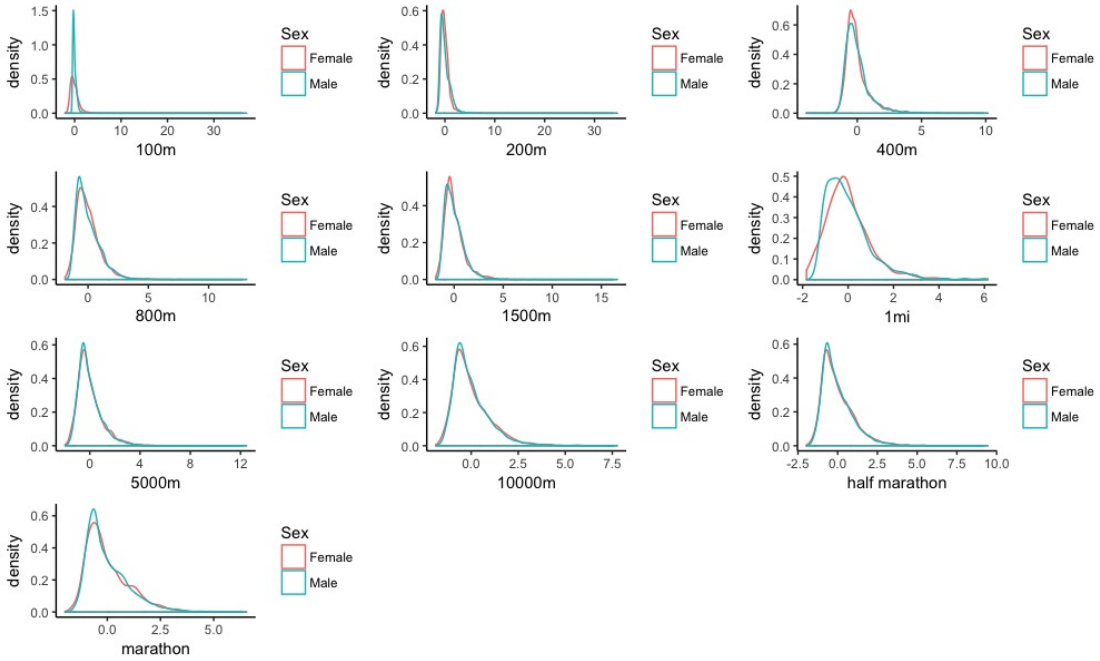
Figure 3.2: Distributions of time for male and female performances



3.4 Errors and Changes

As the dataset was pre-cleaned by Prediction and Quantification of Individual Athletic Performance, there were few errors. There are 3 times recorded faster than the current official UKA records. 2 of these entries are only marginally faster than their respective records (and 1 of

Figure 3.3: Scaled distributions of time for male and female performances



these cases included Paula Radcliffe recorded as faster than her own UKA record), the other was significantly faster and appears to be a data entry error. All 3 were removed from the dataset. As previously mentioned, there are a number of placeholder "1901-01-01" values in the Date of Birth variable; these were replaced with NaNs. Finally, there was an error regarding the ordering of the distances: the events 1 Mile, 5,000m and 10,000m were mis-assigned resulting in times faster for the 10,000m than the 1,609m long 1 Mile. These were reassigned.

4 Methods

The general mathematical setting of this prediction task can be thought of in a number of ways. The formulation we describe here considers features ID and Distance, $\boldsymbol{Z}$ and $\boldsymbol{D}$, and labels Time, $\boldsymbol{T}$:

$$\boldsymbol{X} = [\boldsymbol{Z}, \boldsymbol{D}], \quad \boldsymbol{y} = \boldsymbol{T} \quad \text{with } \boldsymbol{X} \in \mathbb{R}^{N\times 2}, \boldsymbol{y} \in \mathbb{R}^N$$

$$\begin{aligned} \text{and} \quad & \boldsymbol{Z} \text{ t.v. in } \{1, \ldots, \mathcal{N}\}^N \\ & \boldsymbol{D} \text{ t.v. in } \{100, 200, \ldots, 42194\}^N \\ & \boldsymbol{T} \text{ t.v. in } \mathbb{R}_+^N \end{aligned}$$

We can assume reasonably that $(X_i, y_i) \overset{i.i.d.}{\sim} (X, y)$.

We can also consider the data collated into one wide matrix with missing entries, $\mathbf{M}_{(\mathcal{N}\times 10)}$. The rows of $\mathbf{M}$ correspond to distinct athletes, the columns to distinct events and the entries the personal best of a particular athlete in a particular event, that is $\mathbf{M}_{i,j}$ corresponds to athlete i's personal best in event j. Here, N is the number of non-empty entries of matrix $\mathbf{M}$, such that the total number of missing entries $= \mathcal{N} \times 10 - N$.

For convenience we also assign event codes, E_i, to the distances, D_i with E_i t.v. in $\{1, \ldots, 10\}$ i.e. $D_i = 100 \rightarrow E_i = 1, \quad D_i = 200 \rightarrow E_i = 2, \ldots$.

We denote data seen by the model with $'$, i.e. $\boldsymbol{X}', \boldsymbol{Y}'$ and $\boldsymbol{M}'$

The methods considered by this project will be run using *scikit-learn*[cite], a widely used Python machine-learning toolbox. It provides implementations of a range of common machine-learning strategies, as well as a number of other modelling functionalities such as sampling methods, data processing and cross-validation methods. We can thus make use of such functionalities by defining estimators compatible with the *scikit-learn* interface.

Scikit-learn uses object orientation to define estimators; we must define our own estimators as objects that fit within its object inheritance hierarchy. This means our estimators inherit from the *scikit-learn* objects `BaseEstimator` and `RegressorMixin`.

When implementing estimators, all model hyper-parameters must be passed in during initialisation. The estimator objects must define fit and predict methods to fit training data to the model and then predict targets from test data. All other methods and attributes can be inherited from parent classes.

4.1 Riegel

This subsection will describe the predictive methodology and implementation of the `RiegelRegressor` using pseudo-code. Full code, along with a detailed description, is available in Appendix A.2.1. The prediction function f for prediction via Riegel's formula is defined as follows for the mathematical setting described above:

$$f : X_i \longmapsto y'_j \times \left(\frac{D_i}{D'_j}\right)^{1.06} \qquad \text{for } y'_j \in \boldsymbol{y}'$$
$$\text{where} \quad j = \text{argmin}\{\text{abs}(E_i - E'_j)\} \quad \text{s.t. } Z_i = Z'_j$$

This is the same as Riegel's formula as defined in section 2.1, with the additional specification of a method for selecting which past record, y'_j, we use for calculating the coefficient. We also consider selecting y'_j, and the corresponding distance D_j, randomly, that is:

$$y'_j \sim \text{Unif}\{y'_j : Z_i = Z'_j\}$$

Whilst there are no explicit hyper-parameters needed for estimation by Riegel's formula, this selection method constitutes an implicit hyper-parameter we must specify upon initialisation. We pass these in during the object's initialisation as either `random` or `nearest`. We also allow specification of the exponent used, defaulting to 1.06 as calculated in Riegel's study, for potential further investigation. Lastly, we use a flag to indicate whether or not the input data has been log-scaled.

Next we must define a fit method, taking as arguments the feature and target training sets. This method assigns the training data as object variables and, depending on `log`, scales the targets back to standard linear scale. The justification for this approach, resorting back to standard linear scale, is that applying Riegel's Formula on standard data results in the same predictions as log-adjusting Riegel's Formula for application on log-scaled data. Thus we adopt this approach for ease of implementation.

The predict method takes as an argument the feature test set. The prediction algorithm selects a past record from the object's fitted data to use for prediction using the selection strategy specified upon initialisation. We use the time and distance corresponding to the selected training data point, t_1 and d_1, and the test point distance, d_2, as inputs into Riegel's formula. We return this result, scaled back into log-space if required. In pseudo-code the algorithm is as follows:

with training data $\mathbf{X}'$, $\mathbf{y}'$ and test data $\mathbf{X}$;
instantiate $\mathbf{y}$ as an empty vector;
for X_i in $\mathbf{X}$:
    select X'_j, y'_j from $\mathbf{X}'$, $\mathbf{y}'$ using specified selection method;
    find D'_j from X'_j and D_i from X_i;
    calculate y_i as $y'_j \times \left(D_i / D'_j\right)^{1.06}$;
    add y_i to $\mathbf{y}$;
return $\mathbf{y}$

4.2 Purdy

The prediction function f for Purdy points equivalent scoring, in the mathematical setting described above, is as follows:

$$f : X_i \longmapsto \frac{A(D_i)T_{950}}{P(X'_j) + A(D_i)B(D_i)} \quad \text{for } X'_j \in \boldsymbol{X}'$$
$$\text{and } j = \text{argmin}\{\text{abs}(E_i - E'_j)\} \quad \text{s.t. } Z_i = Z'_j$$

$$\text{where} \quad A : D_i \longmapsto \frac{85}{0.0654 - 0.00258V(D_i)}$$
$$B : D_i \longmapsto 1 - \frac{950}{A(D_i)}$$
$$P : X'_j, y'_j \longmapsto A(D'_j)\left(\frac{T_{950}(D'_j)}{y'_j} - B(D'_j)\right)$$

where $V(D_i)$ is velocity extracted or interpolated from the Portuguese Table of Speeds as described in section 2.2 and A and B are defined as in equations 2.2.2-5. $P(X'_j)$ represents the Purdy point level for a performance X'_j. $T_{950}(D_j)$ represents the standard time for D'_j as described in equation 2.2.1 (referred to there as T_s).

Full code for the implementation of the `PurdyRegressor`, a *scikit-learn* compatible Purdy points equivalent scoring estimator, is available in Appendix A.2.2. This section describes the algorithm and gives a pseudo-code implementation.

Again, there are no explicit hyper-parameters, though initialisation accepts two arguments: `select_method` and `log`. These are used as in `RiegelRegressor`.

The fit method is defined identically to `RiegelRegressor.fit`, taking the training and test data as `X` and `y`.

Before we define the predict method we need to define some helper functions. These include: `v(d)` for extracting, or interpolating if necessary, a speed from the Portuguese Table of Speeds as a function of d; `f(d)` for calculating the fraction of an event spent running on the curved portion of the track; `t950(d)` for calculating the standard time as in 2.2.1; `a(v)` and `b(v)` as in equations 2.2.2-2.2.4 and in the mathematical description of the predictive method above.

The predict method accepts `X`, the test data features, and computes and stores the selected training data points in the same way as the `RiegelRegressor`. We store t_1 and d_1 from the selected training data points and d_2 from the test data. We then calculate the Purdy points associated with these training data points, using a series of helper functions defined in the module. From this we use can calculate the equivalent scoring time for our desired distance, d_2. We return this results, again scaled back into log-space if required.

This predictive algorithm is implemented as follows:

with training data $\mathbf{X}'$, $\mathbf{y}'$ and test data $\mathbf{X}$;
instantiate $\mathbf{y}$ as an empty vector;
for X_i in $\mathbf{X}$:
    select X'_j, y'_j from $\mathbf{X}'$, $\mathbf{y}'$ using specified selection method;
    find D'_j from X'_j;
    calculate Purdy points for y'_j, D'_j as described in mathematical setting above;
    calculate prediction y_i as described in mathematical setting above;
    add y_i to $\mathbf{y}$;
return $\mathbf{y}$

4.3 Random Forests

In order to apply the model we must transform the data to allow application of a Random Forest regression. Starting with the matrices $\mathbf{X}$, $\mathbf{y}$ we transform as follows:

$$T : \mathbf{X}, \mathbf{y} \longmapsto \mathbf{M}, \quad U : \mathbf{M}, k \longmapsto \boldsymbol{M}_k, \boldsymbol{y}_k$$

where $\boldsymbol{M}_k$ is $\boldsymbol{M}$ with column k removed and $\boldsymbol{y}_k$ is column k. This means we have effectively

subset the data into 10 distinct groups dependent on which event we want to calculate a prediction for. We fit 10 distinct Random Forest regressions on the 10 ($\boldsymbol{M_k}$, $\boldsymbol{y_k}$) feature-label splits, defining the prediction function for each feature-label split k, f_k, as the average of B individual learning tree predictions, each with fitted prediction function $f'_{k,b}$:

$$f_k : m_{k,i} \longmapsto \frac{1}{B} \sum_{b=1}^{B} f'_{k,b}(m_{k,i})$$

Thus the top-level prediction function, f, is the piece-wise combination of f_k for $k = 1, \ldots, 10$:

$$f : X_i \longmapsto f_k(m'_{k,Z_i}) \quad \text{for } k = E_i$$

$$\begin{aligned} \text{where} \quad & m'_{k,Z_i} = Z_i^{th} \text{ row of } U(T([\boldsymbol{X'}, X_i], [\boldsymbol{y'}, *]), k) \\ \text{and} \quad & * \text{ to be treated as a missing value} \end{aligned}$$

The implementation of a *scikit-learn* compatible estimator initialises with the same parameters a standard *scikit-learn* `RandomForestRegressor`. The fit method combines the feature and target set, X and y, into wide format using the `ConvertLong2Wide` transform. This constitutes the transform T described above.

The predict method fits a standard Random Forest model for each feature-label split, training on the data stored by the fit method. The model fitting is done in the predict method rather than the fit method for computational efficiency; we only need to fit models for the feature-label splits required by the test data. Unfortunately, despite Random Forests being robust to missing features in theory, the *scikit-learn* implementation requires a complete input training dataset. In his original paper Breiman suggests model performance is reasonably well preserved when using median imputation for missing values; it is unlikely however that Breiman was considering problems with as many missing features as are present in our problem. Nevertheless, this is the approach we adopt for filling in missing values. It would be interesting to explore the performance of Random Forests with other imputation strategies, though for now this has not been considered. We then feed the test data into the appropriate sub-models and make predictions.

with training data $\mathbf{X}'$, $\mathbf{y}'$ and test data $\boldsymbol{X}$;
transform $\mathbf{X}'$, $\mathbf{y}'$ into $\boldsymbol{M'}$ using T described above;
calculate *medians* as column medians of $\boldsymbol{M'}$;
instantiate $\mathbf{y}$ as an empty vector;
for k in the distinct set of events in $\boldsymbol{X}$:
    transform $\boldsymbol{M'}$ into $\boldsymbol{M'_k}$, $\boldsymbol{y'_k}$ using $U(\boldsymbol{M'}, k)$;
    remove rows of $\boldsymbol{M'_k}$, $\boldsymbol{y}_k$ for which $\boldsymbol{y}_k$ is empty;
    fill missing values of $\boldsymbol{M'_k}$ with *medians*;

    find $\boldsymbol{X}_k$ as subset of $\boldsymbol{X}$ with Event= k;
    find $\boldsymbol{M}$ as rows of $\boldsymbol{M'}$ indexed by IDs in $\boldsymbol{X}_k$;
    find $\boldsymbol{M}_k$ by removing column k from $\boldsymbol{M}$;
    fill missing values of $\boldsymbol{M}_k$ with *medians*;

fit Model$_k$ on $\boldsymbol{M}'_k$, $\boldsymbol{y}'_k$;
predict for $\boldsymbol{M}_k$ using Model$_k$, add predictions to $\mathbf{y}$;

return y

4.4 Baselines

The experiment also considers two baseline methods. The first of these is a simple event mean estimator, defined in mathematical notation as:

$$f : X_i \longmapsto \frac{1}{N_{D_i}} \sum_{\{j:D_i=D'_j\}} y'_j \quad \text{for } y'_j \in \boldsymbol{y}'$$

$$\text{where} \quad N_{D_i} = n\{X'_j : D_i = D'_j\} \text{ for } X'_j \in \boldsymbol{X}'$$
$$\text{i.e. number of fitted observations where Distance} = D_i$$

The implementation of this strategy uses a one-of-k encoding for the distance variable, using *scikit-learn*'s built-in `OneHotEncoder` preprocessing transform, and fits an Ordinary Least Squares linear model to these ten dummy covariates. A *scikit-learn* pipeline encapsulates these as 3 distinct steps; selecting only the distance column, encoding as dummy covariates and fitting a `LinearRegression`.

The second of these is a two-factor regression, using both the distance and the athlete ID.

$$\mathbf{X} \to \mathbf{X}_{OOK} \quad \text{where} \quad \mathbf{X}_{OOK} = [\mathbf{D}_{OOK}, \mathbf{Z}_{OOK}]$$

where $\mathbf{D}_{OOK}, \mathbf{Z}_{OOK}$ are one-of-k encodings of the distance and ID column vectors. We then fit an Ordinary Least Squares (OLS) regression of $\mathbf{X}_{OOK}$ on $\mathbf{y}$, giving the usual prediction function:

$$f : X_{i,OOK} \longmapsto \boldsymbol{\beta}^T X_{i,OOK}$$

for $\boldsymbol{\beta}$ defined as the standard OLS regression coefficients and $X_{i,OOK}$ the one-of-k encoding of X_i.

The implementation of this is similar to that of the event mean estimator; a pipeline encapsulates selection of the ID and distance columns, encoding as binary factor covariates and fitting a `LinearRegression`. The algorithm for prediction via the two-factor regression is:

with training data $\mathbf{X}'$, $\mathbf{y}'$ and test data $\mathbf{X}$;
fit an OLS regression of $\mathbf{X}'_{OOK}$ on $\mathbf{y}'$;
find regression coefficients $\boldsymbol{\beta}$;

calculate $\boldsymbol{X}_{OOK}$ as one-of-k encoding of $\boldsymbol{X}$;
calculate $\mathbf{y}$ as $\boldsymbol{X}_{OOK}^{T}\boldsymbol{\beta}$;
return $\boldsymbol{y}$

5 Experiment

There are two main stages to the experiment process; parameter tuning and model validation. In order to give an unbiased estimate of generalisation error, it is important to ensure test data used in the validation process is unseen during tuning and fitting; using *scikit-learn*'s `GridSearchCV` guarantees this. For this two-stage process we use two distinct cross validation set-ups. We will refer to these as the inner and outer set-up for the tuning and validation steps respectively.

All procedures in the experiment use log-transformed targets. Without any scaling, the largest time values are orders of magnitude higher and thus prediction accuracy is skewed in favour of better predictions for the longer distances. Log-transformation is the solution adopted in this project though other possibilities, including bivariate transformations like average speed, could be explored.

5.1 Parameter Tuning

Of the approaches considered in the project, all but the baselines require some degree of parameter tuning. *Scikit-learn* provides functionality for doing so using grid search cross-validation. This requires passing in the parameter options to be considered and, using a specified cross-validation set-up, scores predictive performance for each parameter combination by a specified scoring criteria.

The inner cross validation set-up we use is a custom built 1000 randomly sampled iterations of leave-one-out validation. This means selecting one observation at random as the test set and the remaining observations as the training set, repeated (without replacement) 1000 times. There are requirements dictated by the format of our problem that prevent us from using typical validation set-ups. For instance, one common option is to use K-fold validation: this strategy splits the dataset into K equal subsets (with or without shuffling) and runs each permutation of one as the test set and the remaining $K-1$ subsets as the training set. This will likely cause errors when run on our dataset as the two-factor, Purdy and Riegel estimators demand that the models have been trained on at least one record per athlete in the test set. The most straightforward solution to this potential issue is to ensure that the test set is relatively small compared to the train set. Leave-one-out validation guarantees this requirement holds, though it will however result in as many iterations as there are observations. Our custom solution retains these guarantees without being too computationally expensive. The defintion of `InnerCVSetup` is given in Appendix A.3.

As our scoring criterion we adopt Mean Squared Error (MSE). MSE is the average squared difference between predicted and true value, defined as:

$$MSE = \frac{1}{N}\sum(\hat{f}_i - y_i)^2 \tag{5.1.1}$$

for predicted values $\hat{f}_i$, true values y_i in a sample size N.

5.1.1 RiegelRegressor

There is one hyper-parameter, `select_method`, with two factor options, `nearest` and `random`, to be tuned for the `RiegelRegressor`. The model is tuned over the whole dataset. Perhaps surprisingly, `random` selection results in higher neg. MSE and thus our final `RiegelRegressor` will use this selection strategy. [include exact results]

5.1.2 PurdyRegressor

As with the `RiegelRegressor`, we must tune only for the `select_method`. We find similar results and thus will proceed with the `random` selection strategy.

5.1.3 RandomForestsWideRegressor

For this study we will consider tuning only `n_estimators`, the number of trees in the forest, and `max_features`, the number of features to consider at each split. For `n_estimators` we will consider using the range 6 to 16 and for `max_features` we will consider three popular approaches: `n_features`, using all the features at each split; `sqrt`, using $\sqrt{n\_features}$; `log2`, using $\log_2(n\_features)$ where `n_features` is 9 in our case.

This will result in a grid of 48 outer iterations, one for each parameter combination, with each outer iteration being run 100 times by our cross-validation strategy.

The grid search suggests using 10 trees and all the features at each split as the maximum neg. MSE regressor.

5.2 Model Validation

The second stage of the experiment is to provide quantitative comparison of the different prediction strategies. As is standard practice in statistics we use both MSE of prediction to make this comparison. This is thus an estimator of the generalisation squared error, that is the expected error of prediction.

The outer cross validation set-up we use is custom variant of K-fold. In order to address the issue raised above, i.e. the requirement that models must have been trained on at least one record per athlete in the test set, we limit the test set in each fold to a maximum of one record per athlete. We use 6 folds. The definition of `OuterCVSetup` is given in Appendix A.3.

The models considered are: 1.1 group-mean estimator; 1.2 two-factor regressor; 2.1 Riegel estimator; 2.2 Purdy estimator; 3 Random Forests. We calculate the MSE for the test set in each of the 6 iterations in our cross-validation set-up. We then pool these 6 values to form the validation samples. Our final MSE for each method is the mean of the validation samples. In

order to compare the MSEs from each model we need to estimate the MSE standard error. It is statistically invalid to use the sample standard deviation of the MSEs from the validation samples due to correlation; instead we will compute the sample standard deviation of the empirical losses in each cross-validation fold and take their mean; this may results in an estimator with a slight upward bias but will suffice for our purposes.

We also calculate Root MSE (RMSE) to give errors on a more interpretable scale. To do this we take the square root of our MSEs; to estimate their standard error we use the Taylor series approximation for the second moment of a function of a random variable.

6 Results and Discussion

Results for the comparison of models were obtained via the experimental procedures described in section 5. Table 6.1 shows the RMSEs of prediction for both the top 25 and 95 percentiles of the data.

Table 6.1: RMSE of prediction

Percentiles	**Mean** 1.1	**Two Factor** 1.2	**Riegel** 2.1	**Purdy** 2.2	**RF** 3
95	0.1288 ±0.0167	0.0554 ±0.0089	0.0914 ±0.0151	0.0616 ±0.0104	0.0489 ±0.0085
25	0.1201 ±0.0183	0.0549 ±0.0082	0.0691 ±0.0096	0.0603 ±0.0102	0.0455 ±0.0081

In order to determine whether or not the mean RMSEs are significantly different, we employ a series of Wilcoxon signed-rank tests. These hypothesis tests compare two samples to determine whether or not they have equal mean without making any parametric assumptions. The p-values of the tests are displayed in table 6.2. A low p-value implies low probability the two validation samples are drawn from the same population.

Due to small sample sizes, 6 for 6-fold cross-validation, we obtain identical p-values for the test combinations. This value, 0.0277, represents the smallest possible p-value obtainable from the test in this case; there is no overlap between the ranges of any of the validation samples.

Table 6.2: Wilcoxon Signed-Rank Tests $\alpha = 95\%$ for top $0 - 95\%$

	Mean 1.1	**Two Factor** 1.2	**Riegel** 2.1	**Purdy** 2.2	**RF** 3
Mean	1	0.0277	0.0277	0.0277	0.0277
Two Factor	0.0277	1	0.0277	0.0277	0.0277
Riegel	0.0277	0.0277	1	0.0277	0.0277
Purdy	0.0277	0.0277	0.0277	1	0.0277
RF	0.0277	0.0277	0.0277	0.0277	1

We can also test the significance of RMSE differences by constructing confidence intervals using the Standard Error (SE) estimates. These are displayed in table 6.3. As there is no

overlap between any of the ranges, all differences are significant at the $\alpha = 95\%$ level. Model 3, the Random Forests model, is the best performing estimator over both the top 25 and 95 percentiles.

Table 6.3: $\alpha = 95\%$ confidence intervals for top $0 - 95\%$

	Confidence Interval
Mean	(0.1267, 0.1309)
Two Factor	(0.0540, 0.0568)
Riegel	(0.0889, 0.0938)
Purdy	(0.0598, 0.0633)
RF	(0.0474, 0.0504)

Model 3, the Random Forests model, therefore provides a valuable contribution to the running calculator domain. As mentioned before, its lack of interpretability may be acceptable in simple cases where only best predictive performance is desired. These cases include online running calculators where athletes may want to predict their times in new events. The model does not provide any insight into the exact physiological processes determining running performance; typically models with strong explanatory capabilities require much more information than the relatively limited input to our model. In conclusion, we have addressed both of the questions we raised at the beginning of the project. We have determined the Purdy prediction scheme to be the best predictor of the considered prevailing methods. We have also introduced a new method with better predictive power than the considered prevailing methods; this project has proven our adaptation of Random Forests to be a successful application of machine-learning techniques to the running calculator domain.

References

[1] Blythe DAJ, Király FJ (2016) *Prediction and Quantification of Individual Athletic Performance of Runners*. PLoS ONE 11(6): e0157257. doi:10.1371/ journal.pone.0157257

[2] Riegel PS (1980) *Athletic records and human endurance*. American Scientist.

[3] Riegel PS (1977) *Time predicting*. Runner's World Magazine.

[4] Gardner JB, Purdy JG (1970) *Computer Generated Track and Field Scoring Tables I*. Medicine and science In sports, Vol.2, No.3, pp.152-161.

[5] Purdy JG (1970) *Computer generated track and field scoring tables: II. Theoretical foundation and development of a model*. Medicine and science in sports. Vol.7, No.3, pp.111–115.

[6] Federacao Portuguesa de Atletismo (1962) *Système Rationnel Pour Classer Les Performances Athletiqués*, Lisboa, Portugal.

[7] Katz L, Katz JS (1994) *Fractal (power-law) analysis of athletic performance*. Research in Sports Medicine: An International Journal.

[8] Henry FM (1955) *Prediction of world records in running sixty yards to twenty-six miles*. Research Quarterly American Association for Health, Physical Education and Recreation.

[9] Lietzke MH (1954) *An analytical study of world and olympic racing records*. Science.

[10] Kennelly AE (1906) *An approximate law of fatigue in the speeds of racing animals*. Proceedings of the American Academy of Arts and Sciences.

[11] Breiman L (2001) *Random Forests* Machine Learning, 45, 5–32.