Evaluating Test Condition Importance in Determining the Aerodynamic Profile of a Golf Ball

Group 8: Royal and Ancient

Archie Green, Oliver Sturt, Miles Weedon, and Harry Williams

March 11, 2024

Mathematical and Data Modelling 3

School of Engineering Mathematics and Technology

University of Bristol

Declaration of AI use

AI tools such as ChatGPT have helped with the development of code in some areas of this report including the generation of different visualisations. It has helped debug and provide syntax recommendations for modelling methods. It has also helped with research and has provided direction in obtaining BibTex citations.

1 Introduction

The R&A is known as a leading body within the world of golf, and is the name given to the collective group of companies which support golfing activities worldwide [1, 2]. They participate in both the testing, writing and administration of the 'Rules of Golf', which is the standard rule book denoting all golf regulations. This book outlines six rules a ball must obey, in order to conform to the rules of golf. One of these rules is the Overall Distance Standard [3], which is defined as the maximum distance a ball may travel given specific launch conditions, a key measure in preserving the integrity of the game [4].

The R&A currently test whether each golf ball conforms to the Overall Distance Standard by firstly building its aerodynamic profile. This is achieved through completing numerous setting ID tests within an Indoor Testing Range (ITR). These ITR setting ID tests correspond to unique measurements including the ball's coefficient of lift (C_L), coefficient of drag (C_D), Reynolds number (R) and spin ratio (α), which provide the parameters required to simulate the ball's trajectory. Any given simulated trajectory can then be verified against true data obtained from the Outdoor Testing Range (OTR). Once a trajectory simulator is verified, the R&A can determine the compliance of any golf ball against the Overall Distance Standard.

To accurately determine the aerodynamic profile for a given golf ball, 15 setting IDs are currently used by the R&A, which has been reduced from an original 27. Further, for each ball type, 12 balls are tested in two different orientations, giving a total of 360 tests per ball type. The R&A state that they test 900 golf balls per year [5], resulting in approximately 324,000 setting ID tests per year. This process costs the R&A valuable time and resources, which provides a strong incentive to minimise the amount of setting IDs used in testing. Therefore, the aim of this project is to reduce the number of setting ID tests whilst maintaining accuracy in the predicted trajectory of each golf ball. The results then give the R&A a suggestion as to how their organisation's testing process can be optimised and tuned for future benefit.

This project looks to replicate the existing trajectory model used by the R&A [6, 7] and aims to evaluate the accuracy of predicted distance when compared to known distances as setting IDs are removed. Section 2 investigates previous work and explores the underlying mathematical concepts governing golf ball flight. This literature is then used to build and implement a simulator to determine the trajectory of a golf ball based on its aerodynamic profile, which is explained in section 3. Methods are described in section 4 and the corresponding results are then presented in section 5, which generate optimal sequences of setting IDs to remove. This includes principal component analysis (PCA), tree-based methods, including ensembles such as eXtreme Gradient Boosting (XGBoost), and an autoencoder-based model. By comparing methods, a more holistic understanding of an optimal setting ID removal sequence is presented.

2 Literature Review

This project will build upon work done by S.J. Quintavalla in 'Golf Ball Performance Evaluation Using High-Volume Indoor Aerodynamic Testing' [6] and 'A Generally Applicable Model for the Aerodynamic Behaviour of Golf Balls' [7]. They provide a method for predicting the carry of a ball using a set of ordinary differential equations (ODE's) from given initial conditions, as given in equations 1, 2 and 3. These equations simulate the trajectory of a ball over time and provide a prediction of a ball's total carry – where carry is defined as the distance the ball travels before its first bounce.

$$\ddot{x} = -\frac{\rho A}{2m_b}|V|[\dot{x}C_D + \dot{y}C_L], \tag{1}$$

$$\ddot{y} = \frac{\rho A}{2m_b}|V|[\dot{x}C_D - \dot{y}C_L] - \bar{g_y}, \tag{2}$$

$$\dot{\omega} = \Omega\frac{\omega|V|}{r}. \tag{3}$$

Where,

$$|V| = \sqrt{V_x^2 + V_y^2}. \tag{4}$$

V is the velocity, ρ is the density of air, m_b is the mass of the golf ball and $\bar{g_y}$ is the constant acceleration due to gravity, taken to be 9.81 ms^{-2}. ω is the spin rate, with Ω being the spin rate decay factor which is equal to 2×10^{-5} [8].

Also provided in this literature are equations 5 and 6 which calculate the coefficients of lift and drag (C_L and C_D respectively) in terms of Reynolds number (R) and spin ratio (α).

$$C_L = \left[a_1 + \frac{a_2}{R^5} + \frac{a_3}{R^7}\right] + \alpha\left[b_1 + \frac{b_2 ln(R)}{R^2} + \frac{b_3}{R^2}\right], \tag{5}$$

$$C_D = \left[c_1 + \frac{c_2}{R^3} + \frac{c_3}{R^5} + \frac{c_4}{R^7}\right] + \alpha^2\left[d_1 + \frac{d_2 ln(R)}{R^2}\right], \tag{6}$$

where each of the 12 values [a$_1$, a$_2$ ... d$_2$] are parameters.

3 Aerodynamic Profiling and Trajectory Simulation

3.1 Data

The R&A have provided two data types used in their testing process – ITR and OTR test data. There are two ITR datasets, with one corresponding to a total of 27 setting IDs, and the other corresponding to the R&A's current set of 15. Each dataset is obtained by shooting multiple golf balls out of a mechanical cannon with a particular orientation – poles horizontal (PH) or poles over pole (PP) [9]. The values of the ball's C_D, C_L, R and α are subsequently observed. Each ball is shot with an initial setting condition, corresponding to one setting ID, and this is repeated over all setting IDs in the dataset, providing all information to form the aerodynamic profile of each ball, which is described in subsection 3.2. Each ITR dataset contains a comma-separated values (csv) file for each golf ball of a particular orientation. Each entry, or row, in a golf ball's csv file holds the data captured over that corresponding setting ID. An example csv file containing the ITR setting ID data can be found in Appendix A.

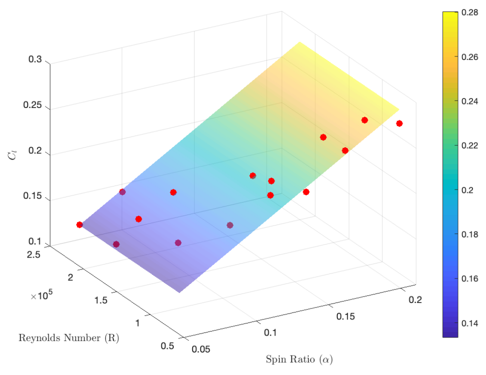
There is one OTR dataset, that contains values of true carry, time of carry and the total distance travelled of multiple different golf balls (with a variety of PH and PP orientations), hit by a golf club artificially swung by a mechanical robot. Each ball is hit with initial launch conditions, including ball speed, spin rate and launch angle. These initial conditions are passed into the ODE's described in section 2 and a trajectory is simulated. This is described further in subsections 3.2 and 3.3.

3.2 Aerodynamic Profiling

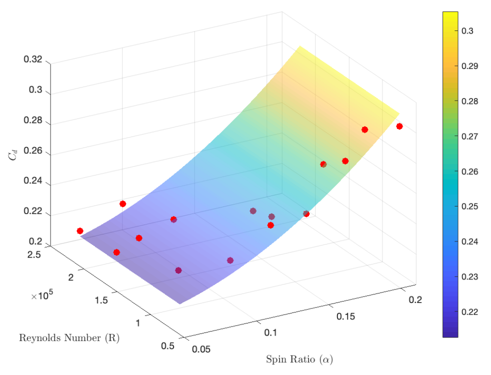
Each ball has unique measured values of C_L, C_D, α and R for each test setting ID recorded. These values are contained within both ITR datasets. Equations 5 and 6 provide functions to calculate C_L and C_D for any α and R inputted. By finding optimal parameters of $[a_1, a_2 \ldots d_2]$ that fit both functions to a unique ball's ITR setting data, as seen in Appendix A, two defined functions with known parameters are created. From this an aerodynamic profile is defined.

These optimal parameters are calculated initially through defining a residual. In this case this is the squared error between the calculated C_L and C_D from equations 5 and 6 and the observed C_L and C_D in the ITR setting ID data. By using MATLAB's 'fmincon' command found within the 'Optimization Toolbox' [10], this residual is minimised and optimal parameters for each ball are established.

The paramaterised C_L and C_D functions can then be visualised via two surface plots for a range of inputted α and R; one relating C_L, R and α and another relating C_D, R and α. Figure 1 shows an examplar aerodynamic profile for one ball, taken from the 15 setting ITR dataset.



(a) Optimised C_L surface plot



(b) Optimised C_D surface plot

Figure 1: C_L and C_D surface plots for one specific golf ball, shot from a specific orientation (labelled as 'ball 11-PH' from the '15 setting' dataset). There are a total of 15 observed values each representing a setting ID.

Later α and R are specifically calculated from the initial conditions given in the OTR dataset through equations 7 and 8, which will serve as inputs to these paramterised C_L and C_D functions:

$$\alpha = \frac{|w|r}{|V|}, \tag{7}$$

$$R = \frac{2|V|r}{v}, \tag{8}$$

It is important to note that as setting IDs are removed in line with this project's aims, the constants and by extension aerodynamic profile of each ball will change. This is because there are less datapoints and therefore less residuals for 'fmincon' to minimise. This in turn will change a ball's simulated trajectory.

3.3 Trajectory Simulation

Using the equations of motion (equations 1, 2 and 3), the trajectory of a golf ball can be simulated in Python using the 'integrate' sub-package of the 'SciPy' library [11]. Specifically, the 'odeint' function is used to integrate the equations of motion to find the position and spin of the golf ball over time. Given the ball's launch conditions provided by the OTR dataset, specific α and R are computed, via equations 7 and 8. As discussed in subsection 3.2, values of C_L and C_D are computed for the ball. These coefficients, paired with the balls initial velocity and spin rate, as well as its mass and density of the air, govern the ball's flight dynamics.

Since equations 1, 2 and 3 only describe the balls carry, the golf ball's bounce and roll isn't modelled. An example trajectory plot of a golf ball can be seen in figure 2.

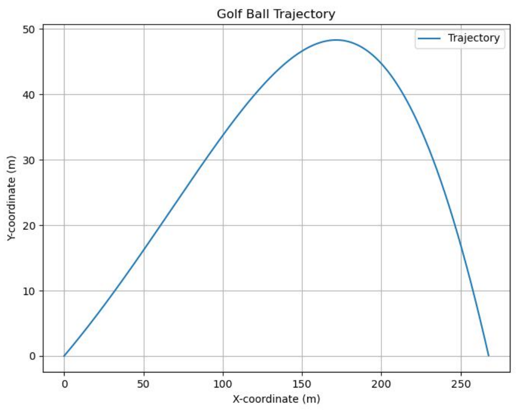


Figure 2: An example trajectory simulation for a single golf ball, launched with a specific orientation, and with its own set of initial conditions.

Once the predicted carry is simulated an error metric can be calculated. By comparing the difference between true carry given in the OTR dataset and the predicted carry simulated, an absolute distance error metric is given. Figure 3 shows the distribution of errors for all ball flight simulations corresponding to the 15 setting ID dataset. The trajectory simulation with all 15 setting IDs used in the aerodynamic profiling, gives a mean absolute error of 1.76 yards in the OTR dataset.

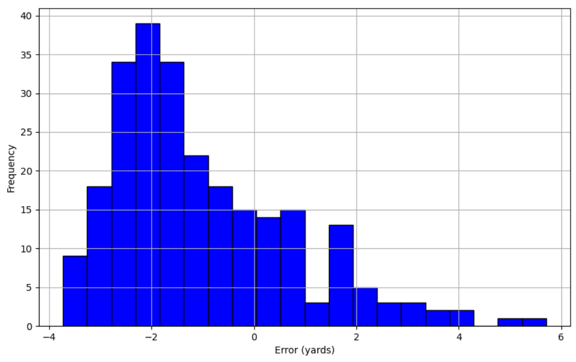


Figure 3: Distribution of absolute error between observed and predicted carry distances when using all of the 15 settings to predict C_L and C_D for each ball

4 Methods of Reducing Setting IDs

The importance of each setting ID is defined and ranked using 3 methods: PCA, XGBoost, and an autoencoder-based model. Using these importance rankings, settings will be removed sequentially and errors will be recalculated, as described in section 3. The average error against the number of setting conditions removed can then be plotted for different removal orders created by different methods. These error plots will then be compared and critically analysed.

4.1 Principal Component Analysis

Principal component analysis (PCA) is a method used to reduce the effective dimensionality of a multivariate dataset, by projecting the original components onto a linear manifold that summaries predominant patterns in the data [12, 13]. The principal components are orthonormal basis vectors, that describe the new coordinate system onto which the data is projected. With each principal component computed, comes a corresponding 'importance score', which quantifies the amount of variance it describes within the dataset [14].

The 'PCA' method in MATLAB computes a loading matrix, whose columns are loading vectors [15]. Each vector transforms all of the original dimensions onto a principal component, represented by that respective column [16]. Therefore, the magnitude of each element (or loading) within a vector gives a measure of importance for that corresponding dimension. By finding the magnitudes along each row (corresponding to one of the original dimensions), and weighting them by the variance explained by each principal component, they can be summed along all rows to give an overall measure of importance for each original dimension.

This method of measuring feature importance has been employed on the '15 setting' dataset. All csv files are firstly averaged, and a matrix is formed, whose rows are R, α, C_L and C_D, and the fifteen columns represent the averages. By calculating the importance of each column of this matrix in terms of its contribution to the variance, the settings can be ranked from least to most important. This then provides an ordered sequence that can be followed. By iteratively removing setting IDs and re-calculating the aerodynamic profile, analysis of absolute mean error in the carry of all balls from the OTR dataset is found. This process allows an assessment of the total number of setting IDs which can be removed, whilst retaining an appropriate level of trajectory accuracy. It also allows the determination of which IDs are crucial to the aerodynamic profile, and which are less significant. These results may translate to a simpler aerodynamic profile for each golf ball, but assumes that a linear relationship exists between the averages of C_L, C_D, R, and α.

4.2 XGBoost

Random forests are found by using the 'bootstrap' method on decision trees. This means that multiple decision trees with different, randomly selected hyper-parameters are sampled and averages are found [17]. However, after implementing a random forest on the '15 setting' dataset, no clear convergence to a clear ranking of feature importance was found. This led to the decision to use 'eXtreme Gradient Boosting' (commonly referred to as XGBoost), since its core algorithm is deterministic [18].

XGBoost is a supervised machine learning algorithm which can be used for many applications, including the ranking of feature importance. It uses a more advanced version of decision trees called gradient boosted trees [19]. This works by building decision trees sequentially and correcting errors in the previous trees. Once an XGBoost model is fitted and trained to its input data, the important features are interpreted as those that explain any non-linear patterns in the dataset the most. These can then be ranked, presenting an ordered list of feature importance that is reliable and non-random. The features in this case will be the setting IDs.

The XGBoost method is implemented by using the 'XGBClassifier' sub-package of the 'XG-Boost' library [20]. It takes the averaged values of the full '15 setting' dataset, in a similar fashion to the PCA method (subsection 4.1) and trains the model using the setting ID labels themselves – 1 through to 15. The settings are then ranked by using the 'feature_importances_' method on the model, which analyses how many times each of the settings are used to split the data across trees. The output of this method provides an order to remove settings from the aerodynamic profiling stage, assuming a non-linear relationship exists between the averages of C_L, C_D, R, and α.

4.3 Autoencoder

An autoencoder is an artificial neural network that consists of two sub-networks – the encoder and the decoder. The encoder reduces the dimensionality of the input data whilst capturing any non-linear relationships that exist within it. The encoded inputs are then reconstructed in the decoder to their original form, ensuring that the process conserves all information [21]. An autoencoder is therefore implemented to understand the positioning and hence importance of setting IDs in latent space.

The model uses the '15 setting' dataset and aims to visualise these settings in latent space. The data is then partitioned into a test and train set in an 80/20 random split. The input size of the autoencoder is the number of features passed to it. In this case the input is the features of each setting ID, which is of size four (α, R, C_L, C_D). It is important to note that a setting ID is an input catagorised by a set of four features. Through a known ordering and not explicitly passing setting ID to the autoencoder, each input can be encoded into latent space with a known setting ID. This can then simply be visualised in two dimensions.

'Experimentally, deep autoencoders yield much better compression than corresponding shallow or linear autoencoders' [22]. Using this knowledge, the specific deep structure of the autoencoder is formed and can be seen in figure 4.

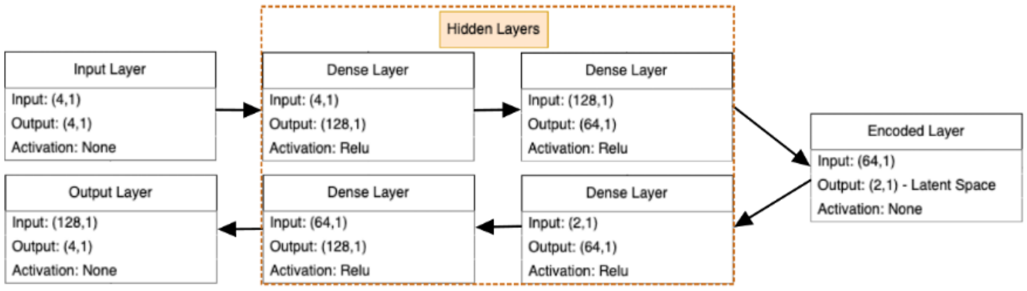


Figure 4: Auto-Encoder architecture

The autoencoder is then trained by reducing the loss defined as the mean squared error (MSE) between the input and the reconstructed output. Using the 'Pytorch' library the backward pass is computed and the gradients are updated at every step. By running the model without setting a random seed a stochastic element is introduced. Reproducing sets of results with different splits and weight intialisations validates the model and ensures convergence to a single solution.

5 Results

5.1 PCA and XGBoost

After ranking the importance of each setting ID via the methods discussed in subsections 4.1 and 4.2, it is found that both PCA and XGBoost rank the settings with the same order of importance. This order descends numerically from 15 through to 1, with increasing importance toward 1. Figure 5 displays the mean absolute error in the carry of all golf balls in the OTR dataset, as setting IDs are sequentially removed (in ascending order of importance) from each aerodynamic profile calculation.

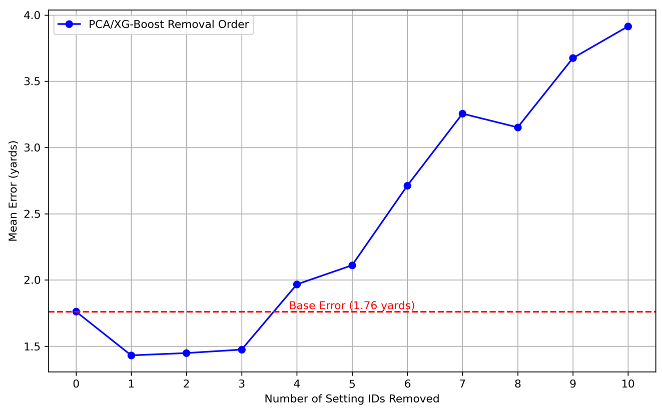


Figure 5: Mean absolute error in the OTR dataset, as setting IDs are removed from each aerodynamic profile calculation, in a removal order produced by PCA/XGBoost

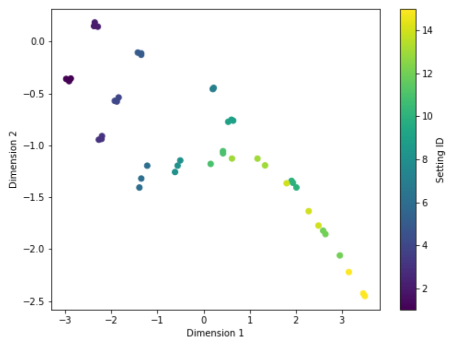
As expected, the mean absolute error tends to increase as setting IDs are removed. However, what is interesting is it only increases past the original absolute error of 1.76 yards after the first four IDs have been removed from the ordered list. This implies that the aerodynamic profile may have been over-fitted to the setting ID data and so the removal of these points have allowed the profile to be more general and more accurate. However, this is only a proposition – further analysis is required to assess any potential over-fitting, such as a proposal and assessment of a simpler model to equations 5 and 6.

5.2 Autoencoder

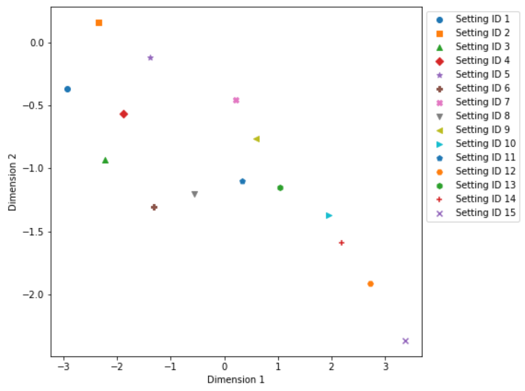
Using the test dataset the resulting encoded data points are visualised in figure 6(a). By reproducing results with different random seeds, the model is seen to converge to the same set of solutions visualised in figure 6(a), further validating this model.

To interpret the results in latent space additional methodology is required. Firstly, by taking the average of each grouping of setting ID, a single point can represent where each setting ID falls in latent space as seen in figure 6(b). This allows a comparison and analysis of the importance of each setting ID average against one another. A position vector can then be computed as the average position of each setting ID in latent space. This can then be used to determine the importance of each setting using a K-Nearest-Neighbour (KNN) algorithm.

A cosine similarity is used which ranks similarity based on alignment and orientation. It focuses on the angle between the positional vectors of setting IDs in latent space rather than by their



(a) Setting ID groupings visualised in latent space from testing dataset



(b) Average position of setting ID groupings visualised in latent space from testing dataset

Figure 6: Outputs from the encoded layer on the test dataset in latent space

distance apart. If a setting ID is similar in orientation to another it is deemed insignificant as it shares similar characteristics with one or more setting IDs – with the number of setting IDs depending on the value of k in the KNN algorithm. The opposite is the case for setting IDs in different orientations from others.

However, a value of k needs to be determined as optimal. This optimal k will create a removal order that produces the least mean carry error as setting IDs are removed, compared to other k values. By calculating removal orders through the KNN algorithm using multiple values of k, an opportunity to compare these k values is presented.

By comparing mean carry error created by each k's removal ordering an optimal k can be found. As seen in figure 7, as k increases the associated removal orderings produce an increased mean carry error. Interestingly, for large k the initial error decreases the most but increases at a higher rate over additional removals, damaging the performance of these orderings.

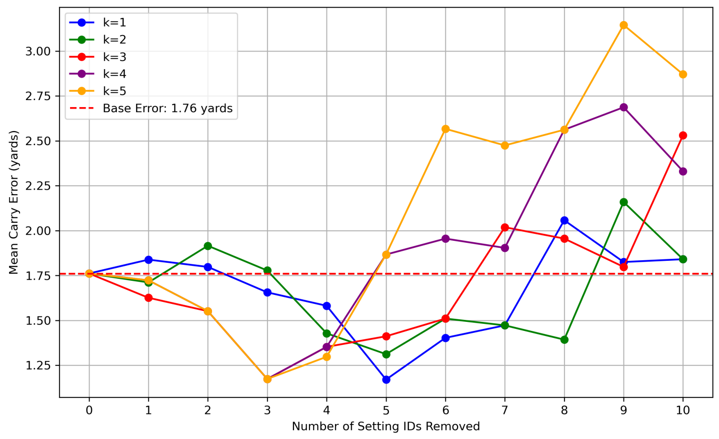


Figure 7: Mean carry error for different removal orderings created through different values of k used in the KNN algorithm

To achieve consistently low error, even after 7 or 8 removals, k must equal 1 or 2. The mean error does not surpass the base mean error until 8 or 9 setting IDs are removed, for $k = 1$ and $k = 2$ respectively. These orderings show similar error curves, however $k = 2$ stays below the base mean error for an extra removal step. Therefore, this may be considered the optimal ordering when attempting to remove as many setting IDs as possible.

List 9 gives the ranking of settings from insignificant to important according KNN algorithm, for $k = 2$.

$$\text{Setting IDs ranked for removal} = [10, 14, 3, 4, 1, 2, 12, 15, 9, 11, 5, 7, 6, 13, 8] \quad (9)$$

5.3 Comparison of Results

To further analyse whether PCA and XGBoost generate similar results, both have been applied to the 27 ITR setting dataset. This will help to assess if any discrepancies exist between the two methods when determining results.

Figure 8 shows the application of these models to the 27 ITR dataset. The XGBoost ordering tends to have a lower mean carry error as higher numbers of setting IDs are removed. However, for the first 7 setting removals, the PCA ordering shows lower mean error when compared to XGBoost. This suggests if smaller numbers of setting IDs are to be removed from the original 27 setting ID dataset, the PCA ordering would be more optimal.

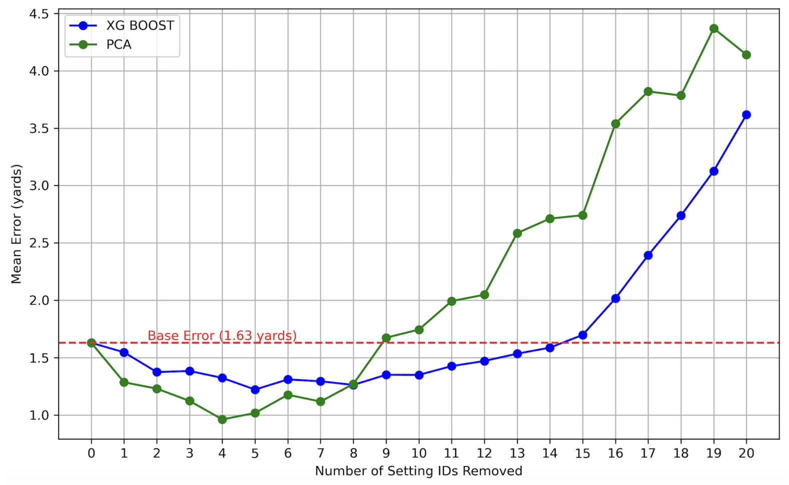


Figure 8: Comparison of cumulative error when setting IDs are removed from 27 ITR dataset according to PCA and XGBoost orderings

A comparison of the 15 setting ID removal orders has been undertaken across all methods, as seen in figure 9. For 4 or fewer setting ID removals, PCA and XGBoost orderings show a lower mean error than that produced by the autoencoder ordering. However as more settings are removed, the setting ID sequences produced by PCA and XGBoost significantly increase the mean error. This is where the autoencoder shows a greater reduction in mean error over more setting removals.

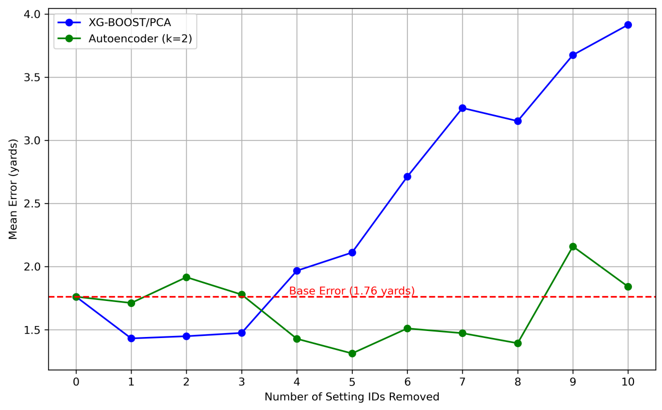


Figure 9: Comparison of mean error when setting IDs are removed from the 15 ITR dataset according to PCA/XGBoost and autoencoder ($k = 2$) orderings

6 Discussion

6.1 Interpretation of Results

The result produced by the autoencoder from the '15 setting' dataset (when interpreted using KNN) appears optimal as this produces the least mean absolute error on the OTR dataset for a large number of setting IDs removed. The resulting mean error, even up to 8 removals, is below the base error. Therefore the autoencoder model could be deemed the best data compression method compared to the other presented models. However the models shown have been tested with a small dataset, so there may be some unknown variance in the results found. The OTR dataset contains only 7 balls, launched in two different orientations, each hit with 18 unique initial launch conditions – defined by initial ball speed, launch angle and spin rate. This is a relatively small amount of OTR data in which the predicted trajectories are measured against. This could explain why the error does not always increase as setting conditions are removed. Alternatively, as suggested in section 5.1, the aerodynamic profile may just simply be over-fitted for each golf ball. This would explain why the error decreases initially and accuracy in the mean carry improves. To test this proposition, a simplification to equations 5 and 6 should be implemented and analysed. Currently this is a seventh order model, which may be overly complex to describe the trends seen.

Compared to the autoencoder-based model, the results of both PCA and XGBoost appear to be worse for removals of 4 or more setting IDs, but are better for 3 or less. It is important to note that both PCA and XGBoost perform their calculations using the averages of C_L, C_D, R and α, whilst the autoencoder method does not, which may explain the large variation in results. The fact that PCA and XGBoost order the IDs identically does however reinforce a sense of validity of both methods.

Since XGBoost is an ensemble of gradient boosted decision trees, it finds non-linear trends between the averages of C_L, C_D, R, and α. However, PCA is a method that describes only linear trends between these averages. This indicates that the trends in the data are linear, since accounting for non-linearity has no effect on the result. Further, this supports the argument that equations 5 and 6 are overly complex for the aerodynamic profile calculation – these equations fit a non-linear manifold to a trend that could potentially be described linearly. Therefore, a simplified set of equations describing the aerodynamic profiling should be proposed, which is an area to further research.

The specific ordering created by XGBoost and PCA, suggest the R&A should remove test setting IDs which relate to lower initial ball speed. This is because high setting ID values, descending from 15, correspond to lower initial ball speeds and hence relatively smaller values of Reynolds number. This result could be because only larger initial ball speeds (254 and 270 ft/s) are being tested in the OTR. Smaller initial ball speeds are tested within the ITR (down to 93 ft/s) which is a large misalignment. Therefore, these setting IDs may not be needed to determine the aerodynamic profile of golf balls tested under the current initial conditions used in the OTR. The autoencoder provides a more scattered approach to removing setting ID's, which suggests the R&A could reduce setting IDs in a more conservative manor whilst preserving the overall structure of the current testing process.

6.2 PCA and XGBoost models

The PCA model gave a useful result for removing a small number of setting IDs but does have some limitations associated with it. PCA can be ineffective when more feature dimensions than observations are being considered. The PCA model only considers 4 observations with 15 feature dimensions (setting IDs), hence the natural structure of this dataset may lead to errors in the method's result.

PCA is also known to be sensitive to outliers. This is due to the fact that PCA aims to capture variance across principal components, which outliers would skew. However, this data does not include any known outliers. It is important to note that if this method was adopted by the R&A, ball data must first be checked for outliers in order for the continued validity of this model.

The biggest limitation in this context is that PCA assumes linearity in order to restrict the set of potential bases that best describe the original data [14]. This model's result could therefore be substantially inaccurate if the data is non-linear, as PCA is not able to capture non-linear trends. Both XGBoost and the autoencoder-based model do not suffer from this problem and may be a better solution for the R&A if their data is non-linear. However, as described in subsection 6.1, there is strong evidence to suggest that the trends in the test setting data is in-fact linear.

6.3 Autoencoder model

Although the autoencoder model does create a valid non-linear compression of setting IDs in latent space, the model relies on the assumption that this compressed space is in a euclidean basis. Commonly, an autoencoder outputs a warped non-linear latent space with no guarantee that the spatial dimensions will be independent [23]. This makes the euclidean cosine similarity metric used within the KNN algorithm imperfect and fairly illogical which is a limitation to this model.

However the cosine similarity does reduce the impact of other problems created by the autoencoder. Inadequate autoencoder architecture is known to produce variance in magnitude between visualised datapoints [22]. As the correct neural network architecture for this problem is ambiguous and unknown, insufficient architecture is likely used in this model creating variance in the magnitude of datapoints. However, cosine similarity unlike euclidean distance is invariant to alterations in magnitude, which provides a more reliable and robust metric than other euclidean alternatives.

Training an autoencoder, like all neural networks, takes large amounts of data. The ITR datasets given by the R&A only contain 210 and 162 observational tests for the 15 and 27 setting datasets respectively. This is structured such that there are 14 tests per setting ID in the 15 setting dataset and only 6 tests for each setting ID in the 27 setting dataset. This sparse amount of data per setting ID, especially in the 27 setting dataset, introduces potential overfitting as the model simply has less to learn. In these cases, autoencoders may not generalise well to unseen data (test data) and therefore could give inaccurate results displayed in latent space. Cross-validation would prove this hypothesis and is an area to further research.

7 Conclusion

This paper evaluates the effectiveness of 3 methods in reducing the number of setting ID tests required to accurately profile a golf ball's aerodynamics, whilst still accurately predicting the carry of golf ball shots. Removal orderings are compared, discussed and evaluated and an optimal ordering is presented.

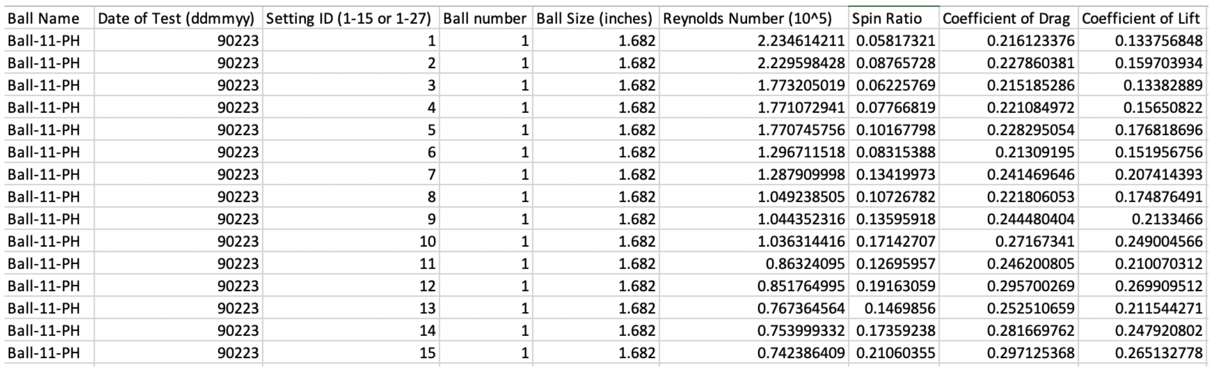
The findings show that PCA and XGBoost give the same result, which is better than the autoencoder-based method when removing small numbers of setting IDs. However the autoencoder-based method provides far better results after removing larger numbers of setting IDs, giving all methods merit. All methods show that setting ID removal is possible whilst conserving accuracy, giving direct solutions to the project aim.

An optimal ordering created by the autoencoder method is able to reduce the 15 initial setting conditions down to 7 with minimal loss in accuracy. This is a very promising result, as it would almost half the number of golf ball shots required on the ITR, saving a large amount of time and money for the R&A. However, this result is somewhat unreliable and would need to be tested on a larger dataset. By testing on more golf balls, the true strength of each method could be tested and therefore a more reliable solution could be found which would be of more use to the R&A.

The results would also be more useful if the bounce and roll were included in the ball trajectory model. This is because the bounce and roll are included in the definition of the Overall Distance Standard. With more time the bounce and roll would be modelled which would give a more meaningful result to the R&A.

Other areas of further research include other non-linear dimensionality reduction tools which include t-distributed stochastic embedding (t-SNE) and Uniform Manifold Approximation and Projection (UMAP), which do not require training like an autoencoder and are also better at preserving data structure in the reduced dimensional space [24]. Exploring these methods may provide other results which can give more intuition into an optimal setting ID removal order.

A Appendix



Ball Name	Date of Test (ddmmyy)	Setting ID (1-15 or 1-27)	Ball number	Ball Size (inches)	Reynolds Number (10^5)	Spin Ratio	Coefficient of Drag	Coefficient of Lift
Ball-11-PH	90223	1	1	1.682	2.234614211	0.05817321	0.216123376	0.133756848
Ball-11-PH	90223	2	1	1.682	2.229598428	0.08765728	0.227860381	0.159703934
Ball-11-PH	90223	3	1	1.682	1.773205019	0.06225769	0.215185286	0.13382889
Ball-11-PH	90223	4	1	1.682	1.771072941	0.07766819	0.221084972	0.15650822
Ball-11-PH	90223	5	1	1.682	1.770745756	0.10167798	0.228295054	0.176818696
Ball-11-PH	90223	6	1	1.682	1.296711518	0.08315388	0.21309195	0.151956756
Ball-11-PH	90223	7	1	1.682	1.287909998	0.13419973	0.241469646	0.207414393
Ball-11-PH	90223	8	1	1.682	1.049238505	0.10726782	0.221806053	0.174876491
Ball-11-PH	90223	9	1	1.682	1.044352316	0.13595918	0.244480404	0.2133466
Ball-11-PH	90223	10	1	1.682	1.036314416	0.17142707	0.27167341	0.249004566
Ball-11-PH	90223	11	1	1.682	0.86324095	0.12695957	0.246200805	0.210070312
Ball-11-PH	90223	12	1	1.682	0.851764995	0.19163059	0.295700269	0.269909512
Ball-11-PH	90223	13	1	1.682	0.767364564	0.1469856	0.252510659	0.211544271
Ball-11-PH	90223	14	1	1.682	0.753999332	0.17359238	0.281669762	0.247920802
Ball-11-PH	90223	15	1	1.682	0.742386409	0.21060355	0.297125368	0.265132778

Figure 10: An example csv containing the ITR setting ID data for a single ball

References

[1] T. Breitbarth, S. Kaiser-Jovy, and G. Dickson. *Golf Business and Management: A Global Introduction*. Taylor & Francis, 2017. ISBN 9781317351580. URL https://books.google.co.uk/books?id=gp0uDwAAQBAJ.

[2] The RA. About the RA. https://www.randa.org/en/about-the-r-a, . [Accessed: February 13, 2024].

[3] The R&A — randa.org. https://www.randa.org/roe/the-rules-of-equipment/part-4-conformance-of-balls. [Accessed 02-02-2024].

[4] The RA. Decision to Revise Golf Ball Testing Conditions Beginning in 2028 Announced. https://www.randa.org/en/articles/decision-to-revise-golf-ball-testing-conditions-beginning-in-2028-announced, . [Accessed: February 13, 2024].

[5] USGA. Usga research and test center primer. https://www.usga.org/equipment-standards/research-and-test-center-primer.html. [Accessed 09-02-2024].

[6] Steven Quintavalla. Golf ball performance evaluation using high-volume indoor aerodynamic testing. 12 2001.

[7] Eric Thain. *Science and Golf IV*, chapter 30 (S.J.Quintavalla). Routledge, November 2012. ISBN 9780203715000. doi: 10.4324/9780203715000. URL http://dx.doi.org/10.4324/9780203715000.

[8] *Science and Golf II*, chapter A new aerodynamic model of a golf ball in flight (A.J. Smits, D.R. Smith). Taylor Francis, September 2002. ISBN 9781135825324. doi: 10.4324/9780203474709. URL http://dx.doi.org/10.4324/9780203474709.

[9] Actual launch conditions overall distance and symmetry test procedure (phase ii), 2011. [Accessed: February 14, 2024].

[10] MathWorks. fmincon - mathworks documentation, 2024. URL https://uk.mathworks.com/help/optim/ug/fmincon.html. [Online; accessed 10-February-2024].

[11] SciPy Contributors. Scipy: Open source scientific tools for python, 2024. URL http://www.scipy.org/. [Online; accessed 10-02-2024].

[12] Pedro R. Peres-Neto, Donald A. Jackson, and Keith M. Somers. Giving meaningful interpretation to ordination axes: Assessing loading significance in principal component analysis. *Ecology*, 84(9):2347–2363, 2003. doi: https://doi.org/10.1890/00-0634. URL https://esajournals.onlinelibrary.wiley.com/doi/abs/10.1890/00-0634.

[13] Geoffrey Hinton. Lecture 15.1 — from pca to autoencoders — [deep learning — geoffrey hinton — uoft]. https://www.youtube.com/watch?v=PSOt7u8u23w, 2017. Accessed: 5 February 2024.

[14] Jonathon Shlens. A tutorial on principal component analysis, 2014.

[15] MathWorks. Principal component analysis (pca), n.d. URL https://uk.mathworks.com/help/stats/pca.html. Accessed: February 15, 2024.

[16] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 2(1–3):37–52, August 1987. ISSN 0169-7439. doi: 10.1016/0169-7439(87)80084-9.

[17] Steven J. Rigatti. Random forest. *Journal of Insurance Medicine*, 47(1):31–39, January 2017. ISSN 0743-6661. doi: 10.17849/insm-47-01-31-39.1. URL http://dx.doi.org/10.17849/insm-47-01-31-39.1.

[18] Stats Stack Exchange Community. Why is xgboost deterministic? https://stats.stackexchange.com/questions/523904/why-is-xgboost-deterministic#:~:text=According%20to%20the%20XGBoost%20parameter,at%20each%20iteration%2D%2Ddeterministic, 2024.

[19] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16. ACM, August 2016. doi: 10.1145/2939672.2939785. URL http://dx.doi.org/10.1145/2939672.2939785.

[20] XGBoost Developers. XGBoost documentation, n.d. URL https://xgboost.readthedocs.io/en/stable/. Accessed: February 15, 2024.

[21] Saïd Ladjal, Alasdair Newson, and Chi-Hieu Pham. A pca-like autoencoder, 2019.

[22] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, July 2006. ISSN 1095-9203. doi: 10.1126/science.1127647. URL http://dx.doi.org/10.1126/science.1127647.

[23] Saïd Ladjal, Alasdair Newson, and Chi-Hieu Pham. A pca-like autoencoder. *CoRR*, abs/1904.01277, 2019. URL http://arxiv.org/abs/1904.01277.

[24] Nikolay Oskolkov. tSNE vs. UMAP: Global Structure — towardsdatascience.com. https://towardsdatascience.com/tsne-vs-umap-global-structure-4d8045acba17. [Accessed 15-02-2024].