Division of Computing Science and Mathematics
Faculty of Natural Sciences
University of Stirling

A Prediction of Youth Football Players' Future Sporting Success Using Neural Networks and Machine Learning

Grant T Isdale

Dissertation submitted in partial fulfilment for the degree of
Master of Science in Information Technology

September 2018

Abstract

There has been a marked increase in data collection within sports in recent years, an area of which has been the collection of performance testing data of individual athletes by sports scientist and coaches with the aim of identifying weaknesses and amplifying strengths. Performance testing data has also piqued the interest of talent identifiers as a way of identifying future talent. The current process of talent identification lacks a systematic and scientific approach. Analysis of labelled historic performance testing data will provide some insight and help form a more methodical approach that can be utilised by talent identifiers and coaches.

Through machine learning classifiers and statistics, the historic data was analysed to identify whether future success can be predicted in youth academy football players. Random Forest, Multilayer Perceptron and Naïve Bayes were implemented. The machine learning techniques were unable to provide any meaningful classification due to data imbalance and low data quality. A proof of concept was produced using a relabelled version of the historic dataset. The proof of concept confirmed that the techniques formed in this project can be utilised as tools for future prediction. A statistically significant positive relationship was identified between the length of time an athlete spent within the academy and their likelihood of future success. The study helped highlight the complexity of high-level sporting performance.

Attestation

Acknowledgements

I would like to thank my project supervisor Professor Bruce Graham for his patience and guidance throughout. I would also like to thank the client James Dugdale who, in partnership with Forth Valley Football Academy, collected the data as part of his PhD. thesis and approached me with the initial idea for this project. I would finally like to thank Dr Iain J Gallagher, who alongside James, gave me vital guidance and information about the relevance, background, and related literature to this project. And of course, I could not forget to thank my Mother, who continues to provide me with eternal support, guidance and love.

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Background and Context

The demand for an increasing level of performance in sport is never-ending. Improvement thus far in performance is a result of an advance in: effective training methods, sports science, access to sports, and technology, to name a few. The use of data has been used to improve all of these aspects in some way; but in an industry where the collection and demand for data is ever increasing [31], [41], [46], how can one capitalise on this data to improve performance going forward?

The analysis of physiological testing results, to monitor and improve athletes, is a vital aspect in improving sports performance. However, interest has piqued from a sports business perspective also, as the necessity for identification of future talent has grown over recent years [36]. Despite this, the means for gathering, interpreting and applying talent identification data is still unresolved [49].

The motivation for the project stemmed from an enquiry by a client who was faced with the task of analysing such data. The client, James Dugdale, is a PhD candidate at the University of Stirling specialising in fitness and performance testing, and strength and conditioning. James collaborates with the Forth Valley Football Academy (FVFA), an elite level football academy, providing sports science advice and services to the club.

FVFA regularly require their players to take part in performance testing sessions, which assess player's biometrics and physiological performance ability. The dataset given by the client contains the historical results of these tests, along with a label as to whether that player went on to sign a professional contract during their career (referred to as a 'success' in this project). The clients desire, therefore, was to use this data to discover how effective these testing sessions are as a means of talent identification in elite youth football players.

1.2 Scope and Objectives

An attempt to predict the future success of youth football players with the use of classifying machine learning algorithms and statistical tests was carried out. As sports science and performance analysis become more readily available, the amount of data on the performance tests of athletes is increasing rapidly. The significance of this data is increasing, and teams and athletes at all levels are beginning to invest in performance testing acquisition and its analysis. The client wanted to answer three main questions:

1. Is it possible to predict success from performance in these tests?
2. Which tests are most indicative of success?
3. Does time spent within the academy affect the likelihood of success?

It was the aim to build models that can answer these questions and to allow the findings of this research activity to be applied practically, to finetune and improve training methods for athletes and increase efficiency of testing/identification methods for coaches, athletes and other interested parties. More specifically for football, it was also the aim to be able to use the models as a first step towards the ability to input new testing data of current players and output a prediction of their likelihood of success, alongside a tool that highlights areas of performance that should be improved by players to increase this success likelihood.

The sub-phases of the project were as follows:

- Research the background and current literature on performance testing in football and talent identification
- Consultation with the client on desired project outcomes and communication through-out
- Research and select suitable machine learning algorithms and statistics for the task, considering implementations and architecture
- Explore and pre-process the dataset
- Execute proposed machine learning algorithms and statistics
- Optimise algorithms to increase predictive ability
- Evaluate the models built
- Build a proof of concept to highlight viability of models

The data was provided by the Forth Valley Football Academy, who the client works closely with as part of his PhD thesis.

1.3 Achievements

Due to a lack of prior experience handling datasets, an increase in knowledge of data handling and data processing was necessary. Additionally, due to no prior experience with Python, it was necessary to move quickly from no knowledge of Python to a good working knowledge; in particular, gaining an understanding of the theory and applications of the Panda, NumPy and Scikit-learn libraries was necessary for handling the data stored in a CSV file, performing data analysis and data mining. Additionally, a basic understanding of SPSS was gained to perform statistical analysis.

Not all outlined objectives were met. However, the project was successful in providing information for the client on the relationship between length of time spent within the academy and success. The project evaluated a number of machine learning classifier algorithms on performance testing data, providing functional insight for future analysis of this type.

With verification through a successful proof of concept, the models documented in this project can be used as tools to effectively classify athletes in similar, but more productive, datasets. Additionally, the models can be used to provide insight to coaches and athletes in other sports, by inputting their own historical data, to gain an understanding of important areas for improvement within their discipline.

1.4 Overview of Dissertation

The project report is organised into chapters documenting the analysis of the dataset and model development.

Chapter 2 – *State of the Art* outlines the background and current knowledge about machine learning classifiers, statistics and performance testing. It begins with an introduction and discussion of the supervised learning algorithms and statistics used in this paper. The chapter is concluded with a brief investigation of the importance of performance testing in sport and the use of machine learning in sport thus far.

Chapter 3 – *Design and Approach* discusses the dataset used in this project, followed by a description of the data preparation. The chapter also outlines the project methods and development methods of the project.

Chapter 4 – *Performance Test Modelling* documents and evaluates the classifiers capacity to predict future success of athletes and the most indicative tests using the performance test data.

Chapter 5 – *Length of Time in Academy* provides a statistical analysis of the relationship between the amount of time a player spends in the academy and success.

Chapter 6 – *Proof of Concept* documents the production of proof that the classifying algorithms used in this project function as tools for predicting success.

Chapter 7 – *Conclusion* provides a summary and an evaluation of the project. It also offers suggestions of future work in relation to this project.

2 State-of-The-Art

2.1 Machine Learning

Machine learning is the ability of a machine to evaluate, learn patterns and rules of a dataset and apply that knowledge to generalise new and unseen data. The computer utilises complex statistical techniques to learn how to perform a specific task without being programmed explicitly. There are 3 main types of machine learning algorithms: supervised learning, unsupervised learning and reinforcement learning. Supervised learning requires the algorithm to be presented with training data that not only includes information about each data point but also a label to which class that data point belongs to, in order to learn the patterns and produce an inferred function [2]. Falling under the category of the supervised approach, this project uses classification algorithms. The purpose of a classification task is to assign an input to one of a number of classes. The classes an input can be assigned to may be binary or numerous. The algorithm works to try and find rules and boundary conditions in the data and attribute those boundaries to a particular class; it can then assess new data against these boundaries and assign it to the class it most belongs to [2]. An example of a classification task would be for a bank to decide whether someone should be granted a loan based on a variety of factors about that person (e.g. income, employment status, credit history) with respect to the same data of people who previously either paid back the loan or did not. Decision Tree Classifiers, Multilayer Perceptron and Naïve Bayes algorithms fall into the category of supervised learning, and are the algorithms implemented in this project. Further possible classification algorithms include Gradient Boosting Machines, Learning vector Quantization, K-nearest Neighbour, Support Vector Machines and Linear Classifiers [53].

2.1.1 Decision Tree Classifier and Random Forests

A decision tree classifier categorises an input by making a sequence of binary decisions until the bottom of the tree, the leaf node, where the input is classified as one of the possible categories. A decision tree has a structure comparable to a flow chart and implements recursive partitioning (also known as 'divide and conquer') to split the data into smaller subsets [25]. Each decision node, where the binary comparison take place, uses just one variable and a condition of that variable to make the decision; each branch in the tree is the outcome of a comparison. Inputs flow through the tree according to the values of its variables.

The decision tree classifier used in this project, implemented using the scitkit-learn library, uses the CART (Classification and Regression Trees) algorithm. This model uses recursive binary splitting to create binary decision nodes that minimize the impurity at each split. The training process involves iterating through all the variables and thresholds and assessing possible splitting criterion at each node. The split that reduces impurity the most is selected. This process is repeated until a stopping rule is satisfied.

There are two measures of impurity used in the decision tree classifier in this report: Gini Impurity and Information Gain. Gini Impurity calculates the frequency that a randomly chosen element would be labelled incorrectly, if it were labelled randomly using the distribution in the subset. The algorithm seeks to reduce this measure, and the measure reaches 0 when all variables are categorised into a single category at a given node; conversely a value of 1 indicates maximum disorder.
Information Gain is built on entropy. Entropy is a measure of the lack of order in a set of data [25]; the decision tree reduces this lack of order as much as possible at each split. Entropy can be defined as:

$$Entropy(S) = \sum_{i=1}^{c} -p_i \, log_2(p_i)$$

Where *S* represents a set of data, *c* represents the number of class levels, p_i represents the proportion of values assigned to class level *i*. Information gain is a calculation of the entropy of a parent node (S_1) minus the average entropy of the child nodes (S_2) [25], defined as:

$$InfoGain\ (F) = Entropy(S_1) - Entropy(S_2)$$

After each split, the remaining data is divided into a number of partitions, the function that calculates entropy (S_2) does so by weighing the entropy of each partition by the number of variables falling into that partition. This process can be used to order the splits in a tree, highest information gain first, until the stopping criterion is reached or there is no data left to be sorted.

Decision tree algorithms are often favoured as they produce easy to comprehend visual representations and output, displaying clearly the decisions the algorithm implements and the consequences of each decision.

The model built in this project uses a random forest algorithm; which is an ensemble of the decision trees described above. During training, the random forest builds a number of decision trees and outputs the mode of the classes. An individual decision tree is susceptible to noise in the training data, however, the random forest algorithm uses a modified version of bootstrap aggregating to combat this and has the ability to decrease the variance of the model without increasing bias [6]. In short, the random forest algorithm helps prevent overfitting to the training data, something that individual decision trees can be susceptible to.

A further method of preventing overfitting is to introduce early stopping criterion to decrease complexity of the algorithm during training, with the aim of the algorithm performing better on unseen data. The algorithm has inherent stopping criterion apparent at the nodes during binary division; there are two cases [27] by which the tree can no longer split: 1) The output from a decision node is homogenous, therefore, splitting any further is unnecessary. 2) The inputs at the node are locally constant. A further set of stopping criterion can be introduced known as hyperparameters. The hyperparameters utilised in the random forest and decision tree models in this project are as follows [37]:

a) *max_depth*: The node is considered a leaf node if its depth is greater or equal to the set threshold
b) *max_features*: the maximum number of features evaluated at each split
c) *min_samples_split*: the minimum number of samples necessary to split a node
d) *min_samples_leaf*: the minimum of samples necessary in the final (leaf) node of a tree
e) *estimators*: the number of trees to be built

The hyperparameters' of a random forest can be tweaked to increase the model's performance on unseen and test data.

2.1.1.1 Feature Importance Using Random Forests

Random forests are formed with the aim of reducing impurity as much as possible at each node. Calculations as to the amount a feature reduces impurity at decision nodes across all trees in the forest can be made [17]. The features can then be ranked based on the average impurity decrease. Feature selection in this manner introduces a certain level of bias towards variables with a higher number of categories [47]. Additionally, results can be misrepresented if correlated variables are present in the data. The model will select a feature as a predictor in an arbitrary manner. Once the first of a correlated set of features is selected, the other features in that set loose importance as any impurity they can remove has already been removed by the initially selected feature [16]. This is an important issue to keep in mind when interpreting the output.

2.1.2 Artificial Neural Networks

Artificial Neural Network's (ANNs) are black box machine learning algorithms, named as such because the internal workings of the algorithm and its calculations are so complex that they are virtually impossible to interpret. These networks can learn from training data and perform tasks without explicit programming and are based on the model of a biological brain reacting to sensory stimuli. In the same way a biological brain uses a vast network of interconnected neurons to process and transmit information; ANNs use a network of interconnected nodes to compute problems.

In the biological brain, dendrites extend from the call and receive signals from neighbouring neurons axon termini [26]. Utilising biochemical mechanisms, the dendrite can sense the relative importance of the signal and convert it into an appropriately weighted electric impulse. The electrical impulse travels along to the axon hillock, where an action potential will be produced if the electric disruption is great enough to reach threshold potential [26]. The action potential will travel along the axon and out through the axon terminal, after being converted back to a chemical signal, onto another neuron or cell. Figure 1 provides a visual representation of this process.

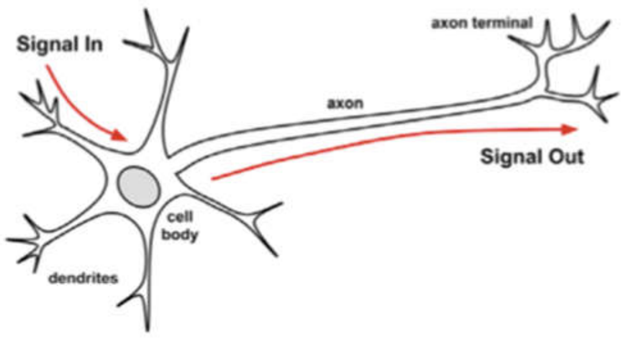

Figure 1. Signal pathway through biological neuron [25]

Similar to the neuron in the biological brain and as shown in figure 2, an artificial neuron receives in an input (*i*) and applies an appropriate weighted value (*W*) to that input, depending on importance. The inputs are converted to outputs (*o*) by the activation function (*f*). ANN's combine networks of single artificial neurons to create complex data models. ANN's have three main characteristics: The *activation function*, the *training algorithm* and the *network architecture*.

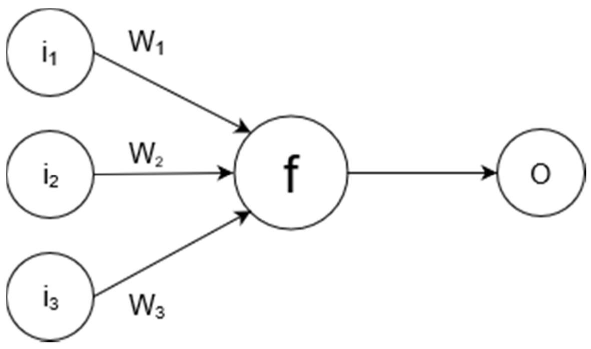

Figure 2. Single Artificial Neuron

2.1.3 Multilayer Perceptron

The multilayer perceptron (MLP) is the category of artificial neural network used in this project. The MLP is a flexible feed-forward neural network, allowing inputs to have distinct effects on any output. Most commonly, each node utilises a nonlinear activation function to determine the output given one or many inputs [42]. The ability of the MLP to use a nonlinear function enables complex problems to be solved with relatively few nodes. The two most common activation functions used by MLPs are the logistic sigmoid activation function and the hyperbolic tangent activation function. Whilst both are sigmoid functions, they do have fundamental differences. The logistic sigmoid activation function allows for output that range (0, 1), given an input of (-∞, ∞) [51]. The second, the hyperbolic tangent has outputs ranging between (-1, 1), given the same infinite inputs as above. This means that exclusively zero value inputs cause near zero outs and any strongly negative inputs will cause a negative output. As a result, the use of hyperbolic tangent activation functions can help networks overcome model parameters updating slowly [22]. With *y* representing the output of the neuron and w_i representing the weighted sum of the input connections, the two functions can be defined as:

Logistic activation function: $y(W_i) = (1 + e^{-W_i})^{-1}$

Hyperbolic tangent activation function: $y(W_i) = \tanh(W_i)$

Whilst logistic and hyperbolic tangent activation functions are often the most popular used in MLPs, due to some recent research demonstrating its ability to better train deeper networks [14], this project with use the Rectified Linear Unit (ReLU) activation function. For any positive input the function returns that value back, but returns 0 for any negative input (Figure 3); with *i* representing an input, the function can be described as:

$$f(i) = \max(0, i)$$

Despite a relatively simple function made up of two linear parts, it has the capacity to model complex, non-linear, problems well. The function itself, however, is non- linear as the slope is not constant near 0 but is a limited type of non-linearity where the slope is either 1 for positive values or 0 for negative values. Through the combination of ReLU nodes, all with different values of bias, combined functions can be produced – it is these resulting combined functions that allow for complex problems to be modelled. The ReLU activation function was found it have a 6 times improvement in convergence of stochastic gradient descent over the logistic and hyperbolic tangent activation function [24].

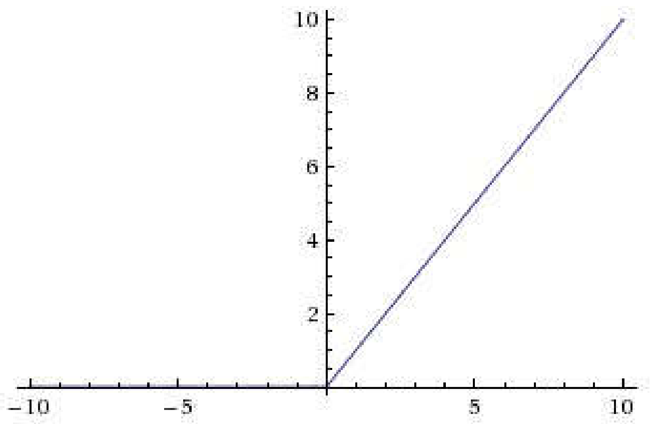

Figure 3. Graphical representation of the ReLU function

The MLP uses a cost function to determine error of the model. Gradient descent is applied to update the model parameters by stepping down the negative gradient of the function until the local minima is arrived with the aim of reducing total error. As networks become deeper, they come susceptible to the vanishing gradient problem; this is where the gradient becomes so small that the weights are unable to update, in some cases, this can cause training of the model to halt completely [15].

MLPs employ gradient descent using a supervised learning algorithm called backpropagation. Initially, all weights in the network are assigned random values; as a result, this selects a random point on the error gradient. The backpropagation algorithm repeatedly compares actual outputs to desired outputs and propagates the error back through the network to adjust the weights accordingly [13]. Initial training of the model is done using a set of training data, but the model has the ability to update online also; the initial training data contains both inputs and desired outputs for the model to learn from.

The high level MLP architecture is fairly simple (Figure 4). The main architectural concerns are; the number of layers, the number of nodes in each layer and the connections between these nodes. Each node is connected to all of the nodes in the following layer, known as being fully connected [13]. There are three categories of layers in an MLP: input layer, hidden layer and output layer. The input layer solely passes the input to the network i.e. it does not alter the inputs at all. The other nodes in the network use weights connect them and the node outputs are a sum of the inputs passed through an activation function [13]. The sum of the weighted outputs from the previous nodes gives the activation to the next node. With input represented as *X* and weight represented by *W*, activation (a_i) can be defined as:

$$a_i = \sum_{i=1}^{n} x_i w_i$$

The training process can be altered using a number of hyperparameters; MLP hyperparameters adjusted in this project are described briefly below:

a) *Learning Rate* determines the size of increments down the slope during gradient descent. Large steps down the slope may cause the global minima to be missed; however, if steps are too small then efficiency is sacrificed [34]. An adaptive learning rate keeps the learning rate constant to the initial rate, as long the training loss keeps decreasing. If two sequential training epochs do not increase the validation

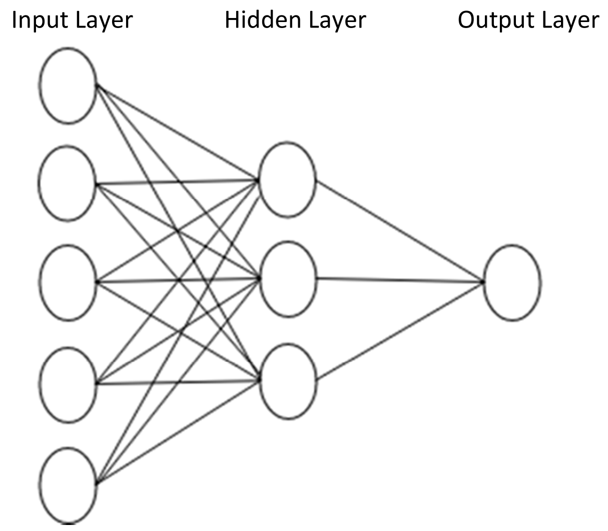

Figure 4. Example architecture of an MLP

score by at least one *Tolerance*, when *Early Stopping* is on, then the current learning rate is divided by 5 [37].

b) *Learning_Rate_Init* determines the initial learning rate (size of step). See a) above for more details

c) *Momentum* uses measures of the gradient at the current position and the running average gradient to smooth out any brief changes in gradient direction [13]. It aids the model to not get caught in a local minimum that is not the global minimum of the cost function.

d) *Early Stopping* is used to halt the training process before the training error flattens out when using an iterative learning algorithm. The model will use a prior assigned 10% of the training data to use as validation data, if the validation score does not increase enough after two consecutive epochs then training with be halted.

e) *Hidden Layers* within the model can be altered. Non-linear activation functions allow for a relatively low number of layers. If the number of layers becomes too great, the model may become susceptible to overfitting to the training data set.

f) *Maximum Iterations* for the stochastic solvers used in this project determines how many times each data point will be used, often referred to as number of epochs.

g) *Solver* refers to the weight optimisation solver. This project uses stochastic gradient descent (sgd) as its solver.

h) *Tolerance* for model optimisation. The amount the model must increase the validation score in two sequential training epochs. See *Learning Rate* above for further details

All other hyperparameters remained as default, details of which can be found on the scikit-learn documentation [37].

MLPs can be very powerful machine learning models, however in contrast to decision trees, they are hard to interpret and visualise. MLPs are not human interpretable and any visual representation cannot give a description of the process the algorithm is using nor the algorithms complexity. Further, as initial weights are assigned randomly to the model each time it is ran, this has the possibility to cause slightly different results, making it more complex to find optimal hyperparameters.

2.1.4 Naïve Bayes Classification

Naïve Bayes Classification is a further classification technique, rooted in Bayes theorem. Bayes theorem attempts to explain the probability of an event occurring, based on prior evidence that may be connected to the event. The classifier is named as Naïve as the model maintains strong assumptions of independence between the observations [18]. That is to say that each feature contributes independently to the probability of the event. The Naïve Bayes classifier can be written as *P(C|X),* a probability of an input *X* being a member of one of many classes *C*. It is relatively straightforward to take a subset of each class in the data and compute the probability distribution; then sequentially applying Bayes rule, described as:

$$P(C|X) = \frac{P(X|C)P(C)}{P(X)}$$

P(C), the probability distribution over the classes is the proportion of data belonging to each class. *P(X)* is the probability distribution of whole data set and as a result functions as a normalising constant as it does not vary. The classifier need not divide *P(X|C)* by *P(X)* as the purpose of the classifier algorithm is to only find the class where the probability of the data is highest [52]. Therefore, the classifier only has to determine *P(X|C)* for every one of the classes in C and then identifying which has the highest probability value – the class with the highest probability value is regarded as being the class to which data point belongs. The naivety of the model of the model causes an elevated model bias, however, when the observations are actually independent, the model performs strongly. This project utilises the Gaussian Naïve Bayes (GNB) Algorithm. The data in this project is continuous; it was assumed it has a Gaussian (normal) distribution [7] and that the GNB classifier algorithm was therefore appropriate.

2.2 Methods for Evaluating Machine Learning Classifiers

In order to gauge the strength of a model it is important to identify appropriate means of assessment. There are a number of means of evaluating a classifier algorithm, such as computational speed, scalability or simplicity. However, given size of the data set in this project, these are not relevant. The most relevant evaluation measures are ones pertaining to how accurate the model was at correctly labelling the data. Part of this method requires a small portion of the data to be separated from the training data before training commences, known as testing data. Towards the end of the training of the model, this testing subset is introduced to the newly built model for it to be classified; since this testing subset is also labelled, the model is able to produce an evaluation of its performance. The importance of defining and separating a subset for testing data is that a model could simply memorise the data it is training on and provide an overinflated performance evaluation.

For the purpose of this section the following abbreviations will be used:

TP: True Negative – number of negatives predicted as negative

FN: False Negative – number of positives predicted as negative

FP: False Positive – number of negatives predicted as positive

TP: True Positive – number of positive predicted as positive

2.2.1 Accuracy Rate

Accuracy describes the proportion of predictions the model made correctly. It can be written simply as:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

As the classification in this project is a binary task, the accuracy rate can also be determined with regards to positives and negatives as:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

Accuracy rate is beneficial as it provides an easily understood output; however, it is not an effective method of evaluation when there is an imbalance between the amount of positive and negative labels. To illustrate this, consider a set of data that consists of two label classes A and B. In this data set there are 90 data points labelled A and 10 data points labelled B; if the model were to predict all data points as to belonging to A, then this would produce a 90% accuracy rate. Generally, a 90% accuracy rate would indicate that the model has performed well; however, in this circumstance a score of 90% could indicate that the model has not performed prediction at all.

2.2.2 Confusion Matrix

A confusion matrix provides a tabular representation of how many times a model correctly predicted a class. The rows of a confusion matrix represent the number of actual instances in each class and the columns represent the number of predicted instances in each class.

		Predicted	
		Negative	*Positive*
Actual	*Negative*	TN	FP
	Positive	FN	TP

The output of a confusion matrix gives a more detailed look at the amount of correct classification. The significance of false cases (false positive and false negative) differ from case to case and confusion matrices work best in cases where there are fewer output classes, preferably binary classes [48].

2.3 Statistical Tests

The statistical tests in this project are implemented to determine whether a correlation or a relationship occurs between input variables and the output. More specifically, the statistical tests will be used measure the closeness of the relationship between the inputs and the class labels.

2.3.1 Correlation Coefficient

The bivariate Pearson Correlation is a popular measure of correlation between two continuous variables. However, when dealing with one continuous variable and one binary variable, a mathematically identical correlation, the Point-Biserial Correlation, can be used [23].

The Point-biserial Correlation produces a measure of the strength and direction of the relationship between a set of variables. This measure is given as a correlation coefficient, referred to as r_{pb}. The Correlation gives an indication as to whether a statistically significant linear relationship occurs between the two variables and how close the relationship is to a straight line. The hypothesis for the two-tailed significance test used in this project can be expressed as follows:

$\mathrm{H_0}: \mathrm{r_{pb}} = 0$ – correlation coefficient is 0; there is no association

$\mathrm{H_1}: \mathrm{r_{pb}} \neq 0$ - correlation coefficient is not 0; a nonzero correlation could exist

As previously mentioned, the Point-biserial Correlation is mathematically equivalent the Pearson's product-moment correlation and written as:

$$r_{pb} = \frac{M1 - M2}{S_n}\sqrt{pq}$$

Where, *M1* is the mean of the subset with the positive binary variable (i.e. 1), *M2* is the mean of the subset with the negative binary variable (i.e. 0), S_n is the standard deviation of the entire dataset, *p* is the proportion of the negative variable group and, finally, *q* is the proportion of the positive variable group. A correlation coefficient of 1 indicates a perfect positive relationship, 0 indicates no relationship, and -1 indicates a perfect negative relationship [23].

Further assumptions necessary for a valid Point-biserial correlation result are: no outliers are present in the dataset; the continuous variable should be loosely normally distributed, and the continuous variable should be labelled with an equal proportion of the binary variable.

2.3.2 Pearson's Chi-Squared Test

The Pearson's Chi-Squared test analyses the relationship between two categorical classes, more accurately, it determines if any observed differences between the classes occurred by chance. Each category need have at least two variables and be expressed in a contingency table [29]. The hypothesis for the Chi-squared test is expressed as:

H_0: The variables are independent of each other

H_1: The variables are not independent of each other

With *O* as observed frequencies, *E* as expected frequencies and *i-j* representing all the cells i.e. from the first cell *i*, to the last cell *j*; mathematically, the test statistic (X^2) is written as:

$$X^2{}_{(i-j)} = \sum \frac{(O - E)^2}{E}$$

An equality of variances is not required for the Chi-squared test to be effective [29]. The test has the ability to indicate both the significance of any differences in the observed and provide information as to where the differences occur. Key to the effectiveness of the Chi-squared, it assumes that the groups are independent of each other and that all subjects only appear once in the data [29].

2.4 Significance of Performance Tests in Sport

Sports are often very complex activities, often requiring a number of skills to be performed serially and dependent on multiple facets of physical fitness. As a result, it may be advantageous to test these individual skills or facets of fitness individually. By breaking the demands of the sport down, it is possible to identify athlete's strengths and weaknesses, monitor progress and improvement, and even identify talent [36].

Talent identification has become particularly interested in performance testing recently; increased pressure for optimal returns on investment in athletes is becoming apparent throughout the sporting world. One of the greatest issues in talent identification is that an athlete's future success is reliant on a variety of complex, and sometimes unmeasurable, factors including sport psychology [1], sociology [19] and genetics. With this being said, physiological performance tests are still treated as the most important indication of possible success [36]. It is tradition to determine indicators of success by looking at the performance testing data of professional adult athletes. This method has its drawbacks - there is no evidence that the important characteristics identified in the adult athletes are also important in the prediction of future success in pre-adolescents [49]. Currently it is proposed that there are no key indicators of success until late adolescence, perhaps this is because the level of maturation is more uniform at this age. Rate of maturation is in fact suggested to be a major determinant of performance in pre-adolescents, and likely skew results [49].

Despite the importance of talent identification, there is still a lack of a scientific and methodical approach to the testing of athlete's performances; and there is certainly a lack of consensus as to which performance tests are most important to consider when identifying talent [49].

2.4.1 Discussion of Performance Tests in Project

As with most sports, football is complex sport requiring a variety of skills and activities. Tests of players' muscular power, repeated sprint ability and aerobic fitness are the most favoured by sports scientists [40]. Muscular power and acceleration are essential for effective sprinting, jumping and change of direction – all important activities in football. The ability to repetitively jump to head lofted balls is important in both attacking and defending phases of the game. Similarly, the ability to change direction is critical both for players with the ball at their feet and without - when reacting to external stimuli, most prominently when defending an opposing player. High level male football players can cover up to 13km in a single match [4], so it is clear as why aerobic fitness is important. However, most of the distance is travelled by low-intensity running and walking and it is suggested that periods of sprint are most important [4]. It was shown that international-level professional players spent 58% more time sprinting during a match, compared with lower level professional players [32]. This indicates the importance of a player's ability to repeatedly run at a high-intensity during a match. To ensure a reflection of a match, testing of repeated sprint ability must involve repeated high-intensity running, followed by brief recovery periods [40].

The performance tests in the data used in this project are as follows:

- *5m Sprint, 10m Sprint, 20m Sprint*: A maximum effort run from a static start. Given the short distance of the run, the tests measure an athlete's ability to accelerate and not maximum speed. The shorter a sprint the more important initial acceleration from rest and leg power.
- Counter Movement Jump (CMJ): A type of vertical jump test. Athletes begin from a static upright position, flex at the knees and hip to make a preliminary downward movement, and finally extend at the knees and hip to jump vertically. The static starting position helps isolate the test of leg power from other factors that can affect vertical jump in a game situation e.g. motor skills and running speed.

- *Yo-Yo*: A test of repeated sprint ability and aerobic fitness. Athletes are required to run 20m, followed by a brief rest. As the test progresses athletes must sprint faster and rest for shorter periods of time.

2.4.2 Machine Learning in Sport

To date, there has not been a great amount of machine learning in sport. Most current machine learning is from the point of view of marketing, wearable technology or match result prediction.

A Naïve Bayes Classifiers was used to predict the winner of the Cy Young Award [44] – an award for the best pitcher in Major League Baseball. The method competed against two statistical models, using data gathered about pitchers in game statistics for the year. It was highlighted that the use of a machine learning algorithm was advantageous over a static statistical method because new data could be continuously introduced to the model and its accuracy should increase each year. Model building used labelled historic data from over 40 years of baseball. It was the case that the Naive Bayes classifier correctly predicted the winner of the award when it was a starting pitcher (as opposed to a relief pitcher) 84% of the time; however, it was outdone by the two traditional static statistical methods it was assessed against.

A study [28] of the use of neural networks in talent identification in the Australian Football League (AFL) draft was conducted. The data in the study had 58 variables about each player, including data on physiological testing performance, biometrics and subjective analysis by coaches – all collected prior to a player's draft date at a draft camp. The study utilised a neural network ensemble; using player ratings, given by AFL experts, following a 3-year tenure in the AFL as a label for training the model in a supervised manner. The results of the study were that teams recruiting managers outperformed the neural networks at predicting the future success of the players in all circumstances. The study attributed this to the limitation of the neural networks to data solely from the draft camp, whereas, recruiting managers had additional data e.g. game film.

The lack of machine learning in the sphere of sport is surprising when there is a plethora of data collected daily by sporting associations, teams and players. It is likely that this will change in the near future as machine learning is made more accessible.

2.5 Programming Language, Libraries and Software used

There were a variety of possible programming languages that could have been used in this project. Python was selected as the language of choice, for two main reasons: first, the additional support of data handling, statistics and machine learning specific libraries – specifically pandas, NumPy, matplotlib and Scikit-learn. Secondly, the diversity of Python - the desirability to learn a language that has future application in not only machine learning but also a wide variety of software development projects.

Python is a high-level programming language that is human interpretable. It is a cross platform language that is widely supported and object oriented. The pandas library [30] allows for data structures to be designed, and used in a flexible manner. The library can handle data in a number of different forms, such as an SQL table or an Excel spreadsheet. The two main data structures in pandas are Series and DataFrame - both allows for the data to be labelled and indexed for later access.

The Scikit-learn library [37] is an open source package that allows data mining and data analysis to be accessible and reusable. It is built on the NumPy, SciPy and matplotlib libraries. The

NumPy library [33] is a further package used for scientific computing. It contains a multitude of high-level mathematical functions that can be applied to multi-dimensional arrays and matrices. The matplotlib library [20] allows for the creation of 2D plots in Python to be used with NumPy, creating easily interpretable data visualisations. Some visualisations of the data during data cleaning were produced in Weka [12].

Python 3.6 [39] was installed as part of Anaconda 5.2 (64-bit) [3]; Anaconda brings the advantage of installing the necessary Python libraries as part of its installation. Both Jupyter notebook 5.5.0 [38] and spyder 3.2.8 [45] were used as development environments. Both development environments provide an intuitive interface, with effective debugging features; however, Jupyter notebook was preferred towards the end of the project as it was far more reliable when handling larger tasks.

For statistical analysis, IBM SPSS was selected; a software that offers a plethora of powerful and industry standard statistical functions, via a user friendly and intuitive interface. Specifically, IBM SPS Version 25.0 was utilised in this project [21].

3 Design and Approach

This section aims to provide a concise summary of the machine learning process applied in this project, to be used as a reference or guide for future work.

3.1 Data Set Acquisition and Description

The data set was acquired by Forth Valley Football Academy (FVFA) between the dates of 2005 and 2016. The working data set contains 497 athletes, reduced from 538 athletes in the master data set; the working data set contained 1778 data points, reduced from 2258 data points in the master data set (see Chapter 3.2 *Data Preparation* below). The athletes are between the ages of 8 and 17 years old. The data represents the results of performance tests of athletes whom were members of the academy for varying lengths of time, the longest being around 6 years. There were typically 2 or 3 testing sessions held per year by the academy. At each testing session the athletes performed all of the tests in most instances, there are occasions where not all tests were completed for unknown reasons. The maximum number of testing sessions completed by an athlete was 13 (1 occurrence) followed by 11 (3 occurrences), with the mean numbers of tests performed being 3.58.

All tests that required an athlete to be timed were done so with light gates for accuracy and consistency. The Counter Movement Jump (CMJ) was measured using a 'Just Jump Mat' that provide an accurate and consistent vertical jump height.

	Features in Data	***Mean***	***Data Type***
Player Information	Player ID		Numeric - Discrete
	Date of Birth		Numeric – Discrete
	Date of Test		Numeric - Discrete
Biometrics	Age (Years)	13	Numeric - Discrete
	Height (cm)	160.6	Numeric – Continuous
Performance Tests	Weight (kg)	48.49	Numeric - Continuous
	5m Sprint (s)	1.07	Numeric - Continuous
	10m Sprint (s)	1.99	Numeric – Continuous
	20m Sprint (s)	3.33	Numeric – Continuous
	CMJ (cm)	29.16	Numeric – Continuous
	Yo-Yo (level)	17.54	Numeric – Discrete
	Success		Nominal - Binary

Table 1. Summary of features in the data

The data contains both numeric and nominal type. Numeric data can be in the form of discrete or continuous. Discrete data can only take a finite number of possible values, whereas continuous data can take any infinite number of possible values. Age has been identified as discrete as age was rounded down to the nearest lower whole number. Nominal data can be described as data that can be separated into distinct categories, in the case of this project the nominal data is a binary value of success. The data was provided in a CSV file and accessed in Python using the NumPy package; the package makes handling of large datasets fairly simple.

3.2 Data Preparation

It is the case with most machine learning projects that the data does not come in a suitable format to allow for effective model building; as a result, data cleaning must be conducted. Data cleaning deals with solving problems in the data after they have occurred. It is the case that errors in the data can be noticed by chance, however, it is suggested to attack the problem strategically [50]. It is often not always clear how errors or missing values in the data should be treated, so clear rules should be stated as to how they should be treated to enable consistency. This creates the further issue of deciding what those rules should be and where cut-offs for outliers and possible corrections should be drawn [50]. The end goal of data cleaning is to produce a dataset that has fewer errors and abnormalities so more efficient and accurate models can be built.

Initial steps taken in this project to clean the data are as follows:

1. Corrected errors in date of birth where there was more than one date of birth for a player. Fortunately, it was always the case that errors were found for athletes that had enough identical data of births that the error could be corrected e.g. A total of 6 date of birth entries for an athlete, where 5 were identical and only 1 was different – the 1 that was different was updated to match the other 5. It is worth noting that there was no error in birth year found in athletes, only day or month.
2. There were a few female athletes in the table, due to their vast minority they were removed from the data set.
3. There was inconsistency between success entries for a very small number of athletes. If there was a convincing majority of a success/non-success label then the data point was updated to match. Where it was not clear, the player ID of that data point was sent back to the client for clarification from FVFA.
4. Any data points that had missing data features were deleted from the data set.
5. The 5-0-5 performance test presented in the master data set was removed from the data as only a minority of athletes had a result for the test.
6. The original data set did not include an age column. The age was derived from the date of birth and rounded down to the nearest lower whole number (e.g. 12.74 was rounded down to 12).

Subsequently the data was visualised in Weka (Figure 5) to highlight any outliers in the data. Initially, Weka reported that weight and 5m sprint were “neither numeric nor nominal”, this often highlights that there is a mistyped symbol in that column and, as

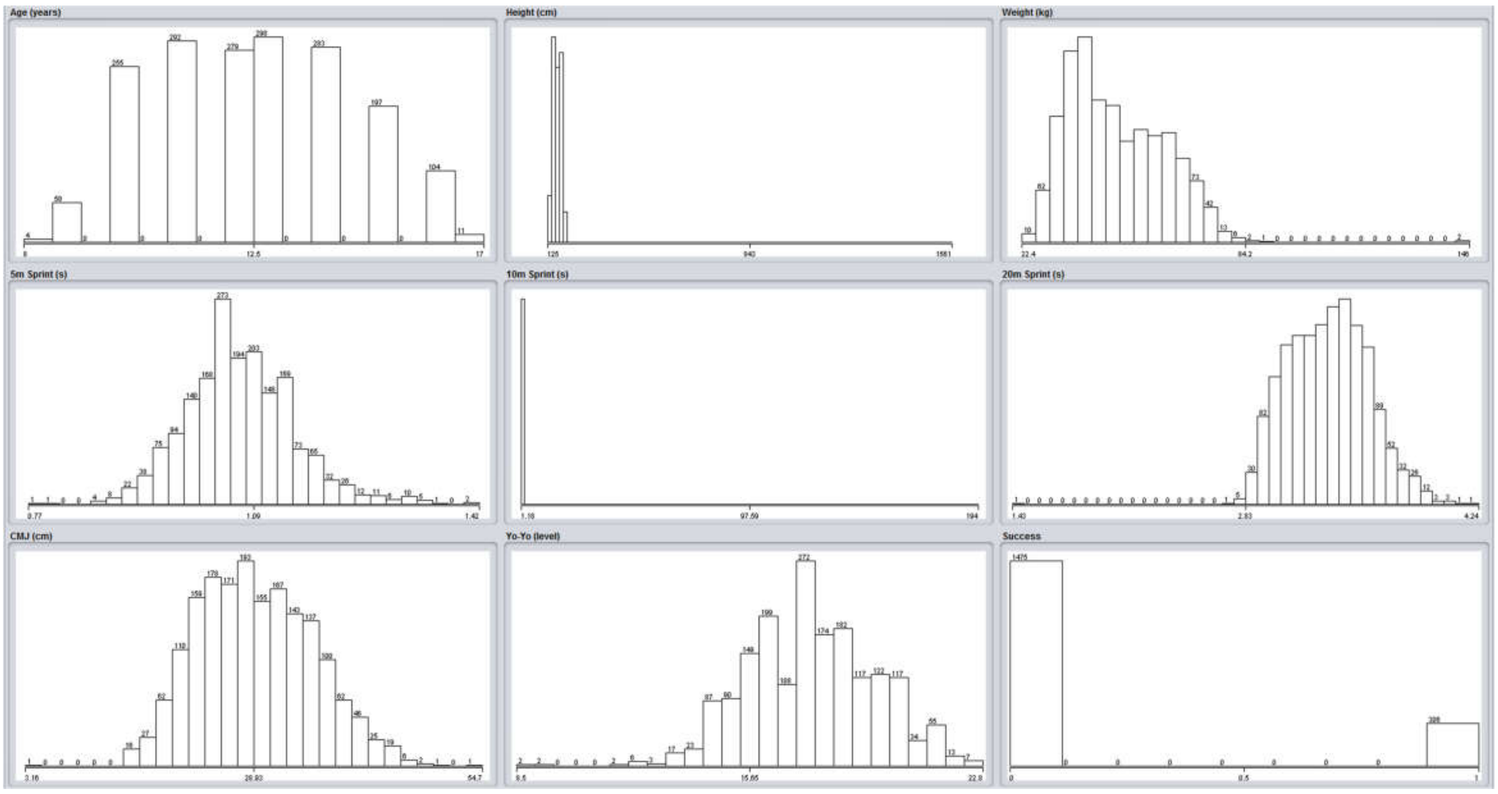

Figure 6. Weka visualisation of data during data cleaning

expected, this was the case this time. Data points with clear outliers in the data set following Weka visualisation (Figure 5) were removed or corrected, it is fortunate that in this data set outliers were very clear – indicating entry errors as opposed to athletes performing out with normal parameters. These outliers were found in height, weight, 10m sprint, 20m sprint and CMJ – data points containing these outliers were either removed from the data set as correction was not possible. It could be argued that some of the extreme performances in the 5m sprint and Yo-Yo could have been removed; however, as the project relates to sport it was decided that extreme performances are often most important and informative, so they remained in the dataset.

During data cleaning the bioinformatics (height and weight) variables, along with Player ID, were left out; as the client requested the project to solely concentrate on performance test results at this point. For any Z- score normalisation in the project, which will be clearly indicated, the data was transformed using a Z-score with data grouped by age e.g. the top age 12 performer in the 5m sprint is given the same score as the top age 15 performer in the 5m sprint. This will allow for any factors of age to be removed as maturation effects physical performance in young athletes.

For the purpose of assessing the effect length of time spent in the academy has on success, it was decided to create a new table of data based on 'number of tests performed' by each athlete; given that the academy remained consistent with the number of testing sessions it held per year, this was an adequate means of inferring time spent in the academy. This new variable table had 497 data points i.e. 497 athletes, a corresponding 'number of tests performed' and the athlete's relevant success/failure label.

There were a number of data points that contained missing variables, these data points were removed (447 data points removed - 21%). It could have been possible to replace missing values with an average or the most frequent value, however, it was decided the data set was not large enough to do this effectively.

3.3 Methodology

The project methodology followed the Cross-Industry Standard Process for Data Mining (CRISP-DM) [8], a standardised process for data mining. The process is cyclical in nature, composed of the following steps:

1) *Business Understanding*

   An understanding of the company or client's goals, business activities, and areas for innovation and development are critical. It helps define a clear objective of a project, with an outcome that is suitable and applicable. This step greatly influences machine learning tasks selected in the project. Business understanding can be found in section *1.1 Background and Context* and 2.4.1 *Discussion of Performance Tests in Project.*

2) *Data Understanding*

   The means of data acquisition must be investigated, along with an investigation of the data itself and its level of quality. This step must also consider the availability of data in the future, especially if the models developed are to be used in real-time. In combination with the business understanding step, suitable machine learning tasks can be selected. Analysis of the data understanding can be found in section *3.1 Data Set Acquisition and Description* and *3.2 Data Preparation*.

3) *Data Preparation*

   Data is rarely provided in a format that can be effectively used by a machine learning algorithm. The data often contains errors, corruptions and outliers; all of which can skew the results of an algorithm. As a result, data cleaning must be performed - details of which can be found in section *3.2 Data Preparation*.

4) *Modelling*

   It is during this phase that the models to be used are confirmed and implemented. Following an initial training period, hyperparameters can be tweaked to produce efficient models that perform effectively to unseen data – known as validation data. This validation data should be set aside to be used as an early indicator of a model's ability to do so. Details of modelling and its results can be found in chapter 4 *Performance Test Modelling* and chapter 6 *Proof of Concept*.

5) *Evaluation*

   In addition to evaluation of variations in hyperparameters and each other, the models should be evaluated within the context of the business and project goals. Following this stage, iteration of the previous steps may be applicable to meet desired aims. Project evaluation can be found in chapter 7 *Conclusion*.

6) *Deployment*

   Following a successful evaluation step, the model can be deployed – if applicable. Monitoring and maintenance must be considered if the model has to be used in real-time. The deployment phase is not applicable to this project.

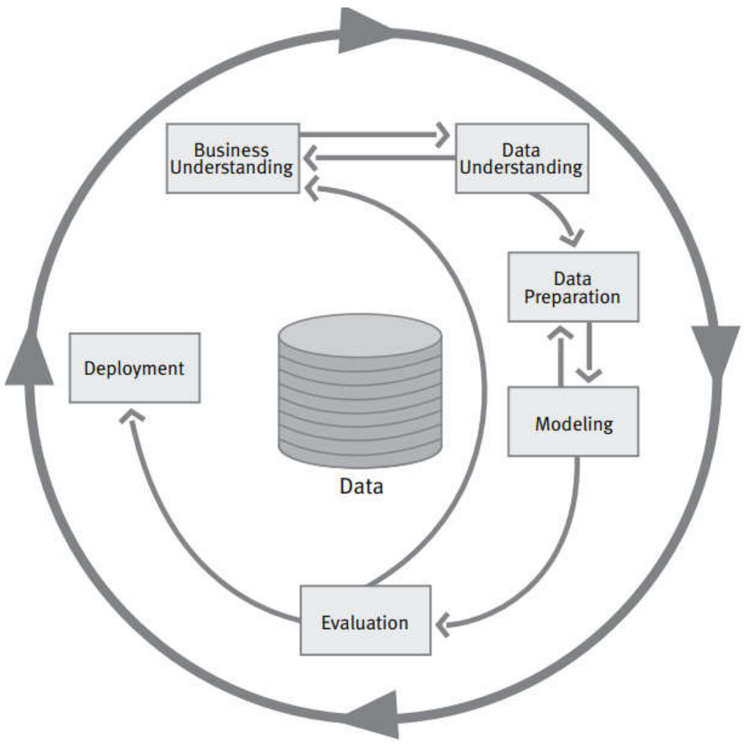

Figure 7. CRISP-DM model [8]

The data acquisition and data cleaning tasks in the project have been discussed above. Pre-processing of the data in Python was necessary to assign variables as inputs and labelled outputs, transform the success column from numeric to type category (appeared as '0' or '1' in CSV file). The data was randomised, then subsequently split into train and testing subsets in a 70% to 30% ratio, respectively. The selected models were then built in Python using the training subset and classifiers tested using the testing subset – producing an accuracy rate and confusion matrix for evaluation.

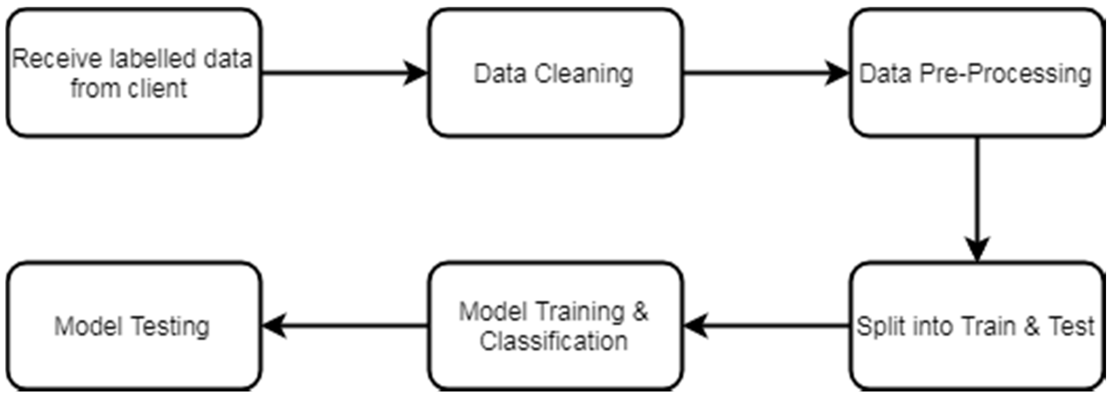

Figure 8. Development process

Random Forest, Multilayer Perceptron and Naïve Bayes models were produced to attempt prediction of success from the performance test data. To identify which tests are most indicative of performance, feature importance was extracted from a random forest model built using the performance testing data.

For the analysis of the effect time spent in the academy had on success, relationships were investigated using Point-Biserial Correlation and Pearson's Chi-squared.

During the development of the random forest classifier, the random state was set (*random_state = 42*) for reproducibility and remained the same throughout. Both random search and grid search hyperparameter optimisation was performed. A wide set of hyperparameters were decided upon and used for random search in all models to narrow down the most effective hyperparameters. The python exert in *Appendix 1.1* was used to set the random search hyperparameters throughout. Exhaustive grid search was then performed around the hyperparameters indicated as optimum by the random search to further refine and improve the model.

In both random and grid search hyperparameter optimisation, the number of fits, i.e. the total number of times the random forest learns the model, is determined by the number of cross-validation folds (folds) and number of iterations (candidates). The cross validation process is as follows: split the dataset into a number (k) of specified groups, remove one of those groups to be used as a validation group, use the remaining (k-1) groups as the training data for the model to learn, and then evaluate the model on the test set. The process is repeated k times – each group is used as the test set once. The number of iterations is defined as the number of combinations of hyperparameters to be used. In exhaustive grid search all combinations of hyperparameters are implemented. In random grid search, however, the number of iterations is specified.

The MLP is sensitive to feature magnitudes so data scaling should be performed. In preparation for the MLP model, data was scaled using a MinMaxScaler (with range (-1, 1)) as detailed in Appendix 1.2.

4 Performance Test Modelling

This section explains the output and results following the creation of models to predict the future success of the athletes and the most indicative performance tests, using the physiological performance testing data. The results shown are the outcome of model validation using a 30% subset (n = 534) of athletes, split from the original data before training commenced. The models were trained using the remaining 70% of the data. In the dataset, 82.8% of athletes are labelled as a failure (the remaining 17.2% labelled a success). Therefore, given that simply predicting all athletes as a failure would produce an accuracy rate of around 82.8%: the classifier must have an accuracy rate higher than 82.8% to be considered successful. The measures of the models are test accuracy and confusion matrices, which detail the amount of times the model correctly, predicted the athlete's outcome in the test subset.

4.1 Results

4.1.1 Random Forest

Unless stated otherwise, the parameters for the random search hyperparameter optimisation used for the random forests in this chapter are detailed in chapter *Appendix 1.1*. The results of each random forest are summarised in the relevant table; each table contains the best parameters of the model, training accuracy and test accuracy.

4.1.1.1 Without Normalisation

Random Search			
Best Parameters	**No. of Fits**	**Train Accuracy**	**Test Accuracy**
'*bootstrap*': false '*criterion*': 'entropy' '*max_depth*': 60 '*max_features*': 'sqrt' '*min_samples_leaf*': 30 '*min_samples_split*': 40 '*n_estimators*': 2000	5 folds for each of 200 candidates, totalling 1000 fits	82.7%	82.9%

Table 2. Output from random forest using random search hyperparameter optimisation

The results of the initial random search are given in Table 2. The model produced a training accuracy rate of 82.7% and a test accuracy rate of 82.9%. Using the results of the best parameters identified in the random search from, a grid search was performed to fine tune the model. The results of the following the grid search, as shown in Table 3, show a slight improvement in training and test accuracy; the model produced a training accuracy of 82.8% and a test accuracy of 83.1%.

A further attempt to optimise the random forest hyperparameters was made using a further grid search based on the results from the previous. The results, of which are detailed in table 4, show that there was no change in either the training or the testing accuracy.

Grid Search				
Parameters Available	**Best Parameters**	**No. of Fits**	**Train Accuracy**	**Test Accuracy**
'bootstrap': [false] *'criterion'*: ['entropy'] *'max_depth'*: [50, 55, 60, 65, 70] *'max_features'*: [‘sqrt’] *'min_samples_leaf'*: [20, 25, 30, 35, 40] *'min_samples_split'*: [30, 35, 40, 45, 50] *'n_estimators'*: [1900, 2000, 2100, 2200]	*'bootstrap'*: false *'criterion'*: 'entropy' *'max_depth'*: 50 *'max_features'*: ‘sqrt’ *'min_samples_leaf'*: 20 *'min_samples_split'*: 50 *'n_estimators'*: 2000	4 folds for each of 500 candidates, totalling 2000 fits	82.8%	83.1%

Table 3. Output from random forest using grid search hyperparameter optimisation

Grid Search				
Parameters Available	**Best Parameters**	**No. of Fits**	**Train Accuracy**	**Test Accuracy**
'bootstrap': [false] *'criterion'*: ['entropy'] *'max_depth'*: [40, 45, 50] *'max_features'*: [‘sqrt’] *'min_samples_leaf'*: [15, 20, 25] *'min_samples_split'*: [45, 50, 55] *'n_estimators'*: [1900, 2000, 2100]	*'bootstrap'*: false *'criterion'*: 'entropy' *'max_depth'*: 40 *'max_features'*: ‘sqrt’ *'min_samples_leaf'*: 20 *'min_samples_split'*: 55 *'n_estimators'*: 1900	4 folds for each of 500 candidates, totalling 2000 fits	82.8%	83.1%

Table 4. Output from random forest using refined grid search hyperparameter optimisation

In order to gain an understanding as to why the output from the random forest did not change between performed grid searches, a further random forest using random search was modelled to produce a confusion matrix. Table 6 displays the output from the aforementioned confusion matrix, which identifies that the model is predicting all athletes as a failure i.e. the model is not classifying the test data at all. The total of 534 were predicted a failure - 443 of those were actual failures and 91 of those were actual successes.

Random Search			
Best Parameters	**No. of Fits**	**Train Accuracy**	**Test Accuracy**
'bootstrap': false *'criterion'*: 'entropy' *'max_depth'*: 60 *'max_features'*: 'sqrt' *'min_samples_leaf'*: 30 *'min_samples_split'*: 40 *'n_estimators'*: 2000	5 folds for each of 200 candidates, totalling 1000 fits	82.7%	83.0%

Table 5. Summary of random search from random forest used to produce a confusion matrix

		Predicted	
		Failure	*Success*
Actual	*Failure*	443	0
	Success	91	0

Table 6. Confusion matrix from random forest model

4.1.1.2 Z-score Normalisation

In order to remove any age bias, the dataset was normalised using a Z-score, further details can be found in chapter *3.2 Data Preparation.*

Random Search			
Best Parameters	**No. of Fits**	**Train Accuracy**	**Test Accuracy**
'bootstrap': false *'criterion'*: 'entropy' *'max_depth'*: 20 *'max_features'*: 'auto' *'min_samples_leaf'*: 4 *'min_samples_split'*: 20 *'n_estimators'*: 400	5 folds for each of 200 candidates, totalling 1000 fits	92.1%	82.7%

Table 7. Output from random forest using random search hyperparameter optimisation, built using Z-score normalised data

The initial random search, as detailed in Table 7, shows 92.1% training accuracy and 82.7% testing accuracy. There is no improvement in test accuracy compared with the models built prior to Z-score normalisation. The best parameters of the random search, as shown in the first column of table 7, were used to attempt to optimise the model using a grid search.

The initial grid search (detailed in Table 8) and the subsequent grid search (detailed in Table 9), both produce a test accuracy of 82.7%. This test accuracy is unchanged from the initial model built using the random search method.

Grid Search				
Parameters Available	**Best Parameters**	**No. of Fits**	**Train Accuracy**	**Test Accuracy**
'bootstrap': [false] *'criterion'*: ['entropy'] *'max_depth'*: [30, 35, 40, 45, 50] *'max_features'*: auto *'min_samples_leaf'*: [2, 4, 6, 10, 15] *'min_samples_split'*: [10, 15, 20, 25, 30] *'n_estimators'*: [325, 400, 470]	*'bootstrap'*: false *'criterion'*: 'entropy' *'max_depth'*: 40 *'max_features'*: auto *'min_samples_leaf'*: 6 *'min_samples_split'*: 10 *'n_estimators'*: 470	10 folds for each of 375 candidates, totalling 3750 fits	92.3%	82.7%

Table 8. Output from random forest using grid search hyperparameter optimisation, built using Z-score normalised data

Refined Grid Search				
Parameters Available	**Best Parameters**	**No. of Fits**	**Train Accuracy**	**Test Accuracy**
'bootstrap': [false] *'criterion'*: ['entropy'] *'max_depth'*: [40] *'max_features'*: auto *'min_samples_leaf'*: [4, 6, 8] *'min_samples_split'*: [5, 10, 15, 20] *'n_estimators'*: [450, 470, 490]	*'bootstrap'*: false *'criterion'*: 'entropy' *'max_depth'*: 40 *'max_features'*: auto *'min_samples_leaf'*: 6 *'min_samples_split'*: 15 *'n_estimators'*: 450	10 folds for each of 36 candidates, totalling 360 fits	91.0%	82.7%

Table 9. Output from random forest using refined grid search hyperparameter optimisation, built using Z-score normalised data

As with the pre-normalised data, a confusion matrix was produced to further understand why there was no change in test accuracy between any of the random forests produced using the Z-score normalised data. Again, the confusion matrix in Table 11 shows that the model finds it optimal to fail athletes, it is unable to produce a classifier able to predict more accurately than this. A total of 529 athletes were predicted as a failure, 440 of those were actual failures and 89 of those were actual successes. The model does predict 4 athletes as a success, but it is only correct 1 of those 4 times.

Random Search			
Best Parameters	**No. of Fits**	**Train Accuracy**	**Test Accuracy**
'bootstrap': false *'criterion'*: 'entropy' *'max_depth'*: 60 *'max_features'*: ‘sqrt’ *'min_samples_leaf'*: 30 *'min_samples_split'*: 40 *'n_estimators'*: 2000	5 folds for each of 200 candidates, totalling 1000 fits	93.1%	82.7%

Table 10. Output from random forest using random search hyperparameter optimisation to produce a confusion matrix, built using-Z-score normalised data

		Predicted	
		Failure	*Success*
Actual	*Failure*	440	3
	Success	89	1

Table 11. Confusion matrix from random forest model of z-score normalised performance tests data

4.1.2 Random Forest Feature Importance

Using the random search method with the hyperparameters detailed in chapter *3.3 Methodology*, feature importance was abstracted from the data. When training a decision tree, the algorithm calculates the amount each feature decreases the weighted impurity. The average impurity decrease across the whole forest, measured as gini index, is then computed for each feature and features are ranked accordingly.

4.1.2.1 Without Normalisation

The feature importance of the performance tests is plotted in Figure 9 and the relevant gini index for each is found in Table 12. With a gini index of 0.328140, CMJ is computed to be the most important feature in this data. This is supported by previous findings by the client that the results from tests that measure leg power differ statistically significantly between academy players and amateur players. The second and third most important features are 20m and Yo-Yo, respectively. Due to the high correlation between the 20m and the two other sprint tests (10m and 5m) which were ranked as the least important, it is a possible that all three are roughly as important as each other. Once the algorithm selects the 20m, the 5m and 10m lose their ability to decrease impurity.

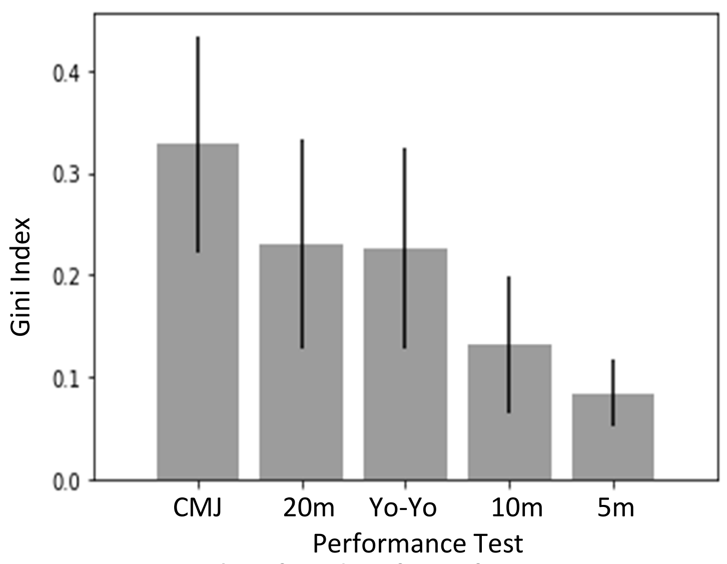

Figure 9. Plot of random forest feature importance

Performance Test	Gini Index
CMJ	0.328140
20m	0.230294
Yo-Yo	0.225988
10m	0.132032
5m	0.083547

Table 12. Table of gini index for each feature, extracted from a random forest

4.1.2.2 Z-Score Normalisation

Further analysis of feature importance was performed on the data following Z-score normalisation; the plot of which can be found in Figure 10 with the gini index for each performance test in Table 13. The features all rank reasonably similarly following Z-score normalisation. The most important feature was the Yo-Yo, closely followed by the CMJ. The Yo-Yo test is a measure of repeated sprint ability, the importance of which in high-level football is supported by the literature [32]. The similarity in importance of the three sprint performance tests (5m, 10m, and 20m) highlights the issue with correlated data in feature importance extraction suggested above in section 4.1.2.1.

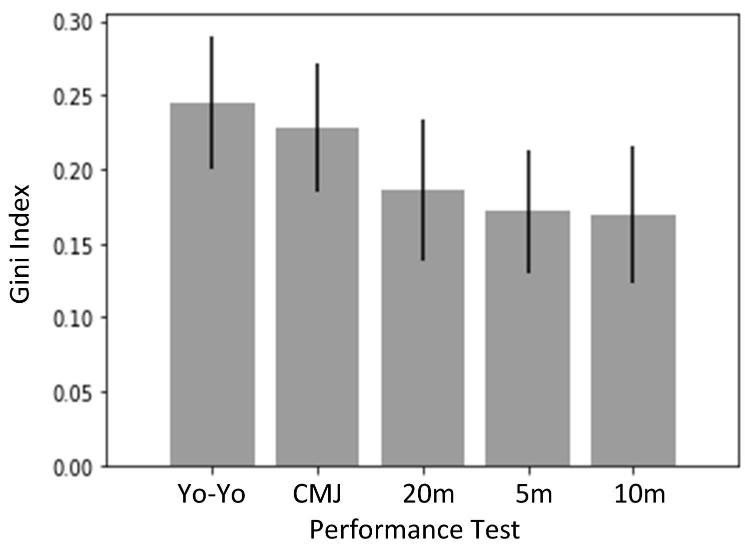

Figure 10. Plot of random forest feature importance following Z-score normalisation

Performance Test	Gini Index
Yo-Yo	0.244989
CMJ	0.228348
20m	0.186040
5m	0.171541
10m	0.169082

Table 13. Table of gini index for each feature, extracted from a random forest

4.1.3 Multilayer Perceptron

4.1.3.1 Without Normalisation

Using the parameters detailed in Table 14, the multilayer perceptron produced an 82.7% training accuracy and 82.9% test accuracy. The confusion matrix (Table 15) reports that the multilayer perceptron also predicted all athletes as a failure; 534 total athletes were predicted as a failure, 443 of those were actual failures and 91 were actual successes.

A second multilayer perceptron classifier was produced, using hyperparameters extracted from Chapter 6 *Proof of Concept* and detailed in Table 16, showing the same result – the model was not able to perform any prediction. Details can be found in Table 16 and the confusion matrix in Table 17. The hyperparameters altered were hidden_layer_sizes, learning_rate_init and momentum – the result was a simpler model.

Parameters	Train Accuracy	Test Accuracy
‘activation’: 'relu' ‘early_stopping’: True ‘hidden_layer_sizes’: (10,10,10), ‘learning_rate’: 'adaptive' ‘learning_rate_init’: 0.01 ‘max_iter’: 100 ‘momentum’: 0.9 ‘solver’='sgd' ‘tol’ = 0.0001	82.7%	82.9%

Table 14. Summary of MLP performance

		Predicted	
		Failure	*Success*
Actual	*Failure*	443	0
	Success	91	0

Table 15. MLP confusion matrix

Parameters	Train Accuracy	Test Accuracy
‘activation’: 'relu' ‘early_stopping’: True ‘hidden_layer_sizes’: (10), ‘learning_rate’: 'adaptive' ‘learning_rate_init’: 0.3 ‘max_iter’: 100 ‘momentum’: 0.2 ‘solver’='sgd' ‘tol’ = 0.0001	81.9%	84.6%

Table 16. Summary of MLP performance using altered hyperparameters

		Predicted	
		Failure	*Success*
Actual	*Failure*	452	0
	Success	82	0

Table 17. MLP confusion matrix using altered hyperparameters

4.1.3.2 Z-score Normalisation

Using data that was normalised using a Z-score within age groups, a further MLP model was produced. The parameters and results, detailed in Table 18, show no change in training or testing accuracy (82.7% and 82.9%, respectively). A confusion matrix was generated from the model (Table 19) show that the model predicts all athletes as a failure at the same rate as the pre-Z-score normalised data.

Parameters	Train Accuracy	Test Accuracy
'activation': 'relu' 'early_stopping': True 'hidden_layer_sizes': (10,10,10), 'learning_rate': 'adaptive' 'learning_rate_init': 0.01 'max_iter': 100 'momentum': 0.9 'solver'='sgd' 'tol' = 0.0001	82.7%	82.9%

Table 18. Summary of MLP performance built using Z-score normalised data

		Predicted	
		Failure	*Success*
Actual	*Failure*	443	0
	Success	91	0

Table 19. MLP confusion matrix built using Z-score normalised data

4.1.4 Naïve Bayes Classifier

The performance of the Naïve Bayes Classifier model was a *test accuracy of 77.3%.* With reference to the confusion matrix in Table 20, the model was able to correctly predict 413 out of 534 athletes. It predicted a total of 431 athletes as a failure, of which 381 were actual failures (88.4% accuracy); and 103 athletes as a success, 32 of which were actual successes (31.1% accuracy).

		Predicted	
		Failure	*Success*
Actual	*Failure*	381	71
	Success	50	32

Table 20. Naïve Bayes Confusion Matrix

4.2 Analysis

Both the random forest and the MLP were unable to effectively predict any athletes as successful. The most likely cause of this is the disparity between the number of success athletes and failure athletes in the dataset. Both the random forest and the MLP are formulated to reduce the overall error rate, therefore both focused on the prediction accuracy of the majority class, common in unbalanced datasets. In the data set there only were 306 (17.3%) success athletes, yet 1462 (82.8%) failure athletes. With 82.8% of athletes in the dataset labelled as failures, it is difficult for the model to build a classifier that can outperform simply predicting all athletes as a failure. The minor variation from 82.8% by the models can be attributed to the sampling error when randomly selecting training and testing subsets.

Two techniques to combat imbalanced data are oversampling or undersampling. Oversampling involves increasing the number of minority cases to reduce disparity, this can be done by simply duplicating a random sample of the minority class or using more complex methods such as Synthetic Minority Over-sampling Technique (SMOTE) [5]. However, as the dataset used in this project had so few data points, oversampling would likely cause overfitting. Conversely, undersampling is the process of removing instances of the majority class. Similarly, there are several techniques that can be applied to perform undersampling [10]. The removal of instances from a dataset results in valuable information being lost and the lack of information in the dataset of this project was an initial concern. Further suggestions for dealing with imbalanced datasets for random forests are given by Chen et al. 2004 [9] and given for MLPs by Zhao et al. 2014 [54].

Traditionally, both the random forest and MLP struggle with smaller datasets. Although not seen in this project, the typical problem the two models have with small datasets is the tendency to overfit. However, both artificial neural networks and random forests have been proven able to perform effectively with small training datasets [35], [43]. Therefore, it is assumed that both models would produce effective classifiers on more balanced dataset of this size.

Due to the inability of the random forest to perform classification, the random forest feature importance results are invalid. However, it is interesting to note that the results of the feature importance fall in line with what was expected from the client. Previous study by the client indicated that the 5-0-5 (a test of agility) and the CMJ were among some of the most important performance tests to discriminate between identified and unidentified youth talent. Performance in these two tests relies on leg power.

Given the confusion matrices output by the random forest models following random and grid search hyperparameter optimisation, it was decided that hyperparameter optimisation would not be performed for the MLP given its initial confusion matrices results. It was expected that hyperparameter optimisation would cause no affect. The initial hyperparameters that determined the architecture of the MLP were selected arbitrarily. A small number of hyperparameter alterations were trialled manually at random, producing no effect on model performance. The second set of hyperparameters implemented by the MLPs was decided in retrospect after producing a successful model in the proof of concept – and only used on the working dataset for completeness.

The Naïve Bayes classifier was the first model to perform prediction i.e. not predict all athletes as a failure. However, with an accuracy score of 77.3%, it performed worse than those models who did simply predict all athletes as a failure (random forest = 83.1%, MLP = 82.9%) i.e. the simple probability of an athlete being a failure in the dataset. The classifier was able to predict 88.4% of failures correctly, outperforming the random forest and MLPs method of predicting all athletes as a failure. However, it was only able to correctly classify 31.1% of the success cases. It is likely the classifier was in fact able to identify some predictors in the failure athletes, but not many. It would be interesting to ascertain which characteristics the data is picking up on in as future work.

5 Length of Time in Academy

This section details the statistical analysis of the affect length of time in the academy had on success. The length of time within the academy was based on the number of tests performed by each athlete as the academy held a consistent number of testing sessions per year. There was a total of 497 athletes. The minimum number of tests performed was 1 and the maximum was 13, with a mean number of tests performed of 3.58. Due to the nature of the Pearson's chi-squared test, the data point containing 13 tests performed was removed as there was only one member of that group.

5.1 Results

There was a statistically significant positive moderate correlation (r_{pb} = 0.303, p = 0.01) between number of tests performed (3.58 $\pm$ 2.5, n = 497) and success (0.11 $\pm$ 0.309, n = 497). Similarly, the Pearson's chi-square test found a significant relationship between number of tests performed and success, X^2 (11) = 62.2, p < 0.001. Cramér's V coefficient (φc) = 0.354, a measure of strength of relationship, supporting a moderate association.

5.2 Analysis

The Point-biserial correlation suggests a moderate correlation between the number of tests, however, the test requires the years variable to be a continuous variable and loosely normally distributed for each category – it is neither of those. There are noticeably more athletes, for both category, who completed a fewer number of tests; the number of athletes that completed 10 or more test was only 15.

An assumption of the Chi-square is that the variables within the data are categories. This is not the case in the dataset in this project. However, there are only 13 distinct values the years variable can take (i.e. 1-13 years), so this is treated as a large number of categories but which the statistic may find hard to interpret [29]. In an attempt to provide a more robust statistic, the length of time in the academy data was transformed into 4 distinct bins (shown in Table 21). The process of binning is typically implemented to reduce the noise in the data. The Chi-square statistic of the discretised data (X^2 (3) = 43.0, p < 0.001) is lower than that of the non-discretised. In other words, there is more of a statistically significant relationship following discretisation.

Length of Time in Academy (tests completed)	**Bin**
1	One-Three
2	
3	
4	Four-Six
5	
6	
7	Seven-Nine
8	
9	
10	Ten-Thirteen
11	
12	
13	

Table 21. Binning of number of tests completed to categories

The strength of the Chi-square statistic with reference to the dataset in this project is that it does not require the data to be normally distributed and allows for skewed data.
A suggested next step following a Chi-squared is to compute the strength of the association. A suggested test for this is Cramér's V coefficient [29]. Cramér's V coefficient supports the moderate relationship found in the Point-biserial correlation, however, the Cramer's V is a more valid test in this scenario given the distribution of the data.

From the statistical analysis it can be inferred that the longer an athlete spends in the academy the increased chance the athlete has of becoming a success. It could be suggested that this is due to one, or both, of the following reasons: 1) The athlete receives more valuable coaching time and more high-level training time the longer they are part of the academy, increasing the skill level and playing performance of the athlete. 2) Possible bias. The athlete will only remain in the academy if they are continually performing at a high-level relative to their peers i.e. the players who spent a number of years in the academy were playing at a relative high-level initially and perhaps found it easier to continue to do so – and continuing so as they matured towards professional age.

6 Proof of Concept

A proof of concept was produced to prove that the classifiers and models built as part of this project are valid and can be used as future tools in, or the basis of, future projects based around performance testing. The proof of concept required the data to be altered: the results of all performance testing remained the same, however, the success label of each athlete was altered based on specific criteria in each proof of concept case – details of which can be found below under each case header. The decision on which variable to select for the focus of the proof of concept were based on guidance from the client with respect to findings in previous research of the most important performance tests. The criteria cut-off point for each case were decided upon to attempt to keep the percentage of success athletes and failure athletes as close as possible to the original data set (success = 305, 20.6%; failure = 1475, 79.4%).

All three models used in *Chapter 4. Performance Testing* were implemented, applying the same methods. The same hyperparameters for random search optimisation of the random forest as found in *Appendix 1.1.* The hyperparameters used in the MLP proof of concept model are detailed in Table 16. The hyperparameters make for a simpler model, found to be effective for simpler relationships within the data in the proof of concepts. One further MLP classifier was produced for each proof of concept case using the 'adam' solver instead; all other hyperparameters remained the same.

6.1 Case 1 – CMJ

The criteria for case 1: all athletes with a CMJ more than or equal to 35.75cm were labelled as a success, the rest were labelled as a failure; the data summary for which can be found in Table 22.

Performance Test	Criteria for success	Number of success	Percentage of success
CMJ	>= 35.75cm	316	21.5%

Table 22. Data summary of case 1

6.1.1 Random Forest

As summarised in Table 21, the random forest classifier was able to correctly classify 100% of the testing data (Table 23). The confusion matrix, Table 24, shows that it correctly classified all 92 successful athletes and all 442 failure athletes.

Figure 11 displays the output of a single tree from the random forest constructed as part of training. The subset of data the algorithm used to construct this tree contained 1022 failure athletes and 224 success athletes. The algorithm only required one decision, using the CMJ performance test, to produce absolute homogeneity.

Random Search			
Best Parameters	**No. of Fits**	**Train Accuracy**	**Test Accuracy**
'bootstrap': false *'criterion'*: gini *'max_depth'*: 40 *'max_features'*: 'sqrt' *'min_samples_leaf'*: 40 *'min_samples_split'*: 60 *'n_estimators'*: 1600	5 folds for each of 200 candidates, totalling 1000 fits	100%	100%

Table 23. Summary of random forest using random search in proof of concept case 1

		Predicted	
		Failure	*Success*
Actual	*Failure*	442	0
	Success	0	92

Table 24. Confusion matrix of random forest using random search in proof of concept case 1

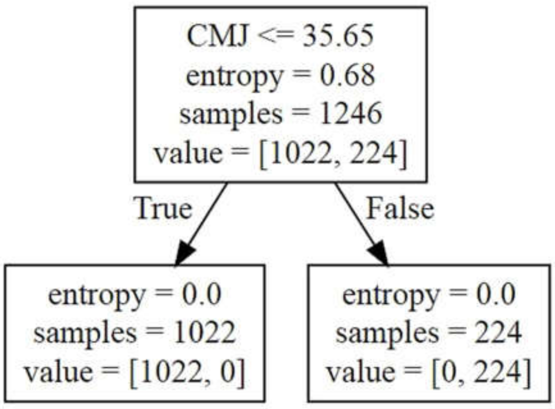

Figure 11. Single tree in the random forest of proof of concept case 1

6.1.2 Multilayer Perceptron

The multilayer perceptron was able to correctly classify 97.0% of the test dataset, as summarised in Table 25. The confusion matrix in Table 26, details that the classifier was able to correctly classify 442 failure athletes and 76 success athletes. The classifier misclassified 1 actual failure athlete and 15 success athletes.

Using the 'adam' solver, accuracy improved. The classifier was able to correctly classify 100% of the test data set (Table 27) – 104 success athletes and 430 failure athletes (Table 28).

Parameters	Train Accuracy	Test Accuracy
‘activation’: 'relu' ‘early_stopping’: True ‘hidden_layer_sizes’: (10), ‘learning_rate’: 'adaptive' ‘learning_rate_init’: 0.3 ‘max_iter’: 100 ‘momentum’: 0.2 ‘solver’='sgd' ‘tol’ = 0.0001	96.1%	97.0%

Table 25. Summary of MLP performance in proof of concept case 1

		Predicted	
		Failure	*Success*
Actual	*Failure*	442	1
	Success	15	76

Table 26. MLP confusion matrix in proof of concept case 1

Parameters	Train Accuracy	Test Accuracy
‘activation’: 'relu' ‘early_stopping’: True ‘hidden_layer_sizes’: (10), ‘learning_rate’: 'adaptive' ‘learning_rate_init’: 0.3 ‘max_iter’: 100 ‘momentum’: 0.2 ‘solver’='adam' ‘tol’ = 0.0001	99.8%	100%

Table 27. Summary of MLP performance using ‘adam’ solver in proof of concept case 1

		Predicted	
		Failure	*Success*
Actual	*Failure*	430	0
	Success	0	104

Table 28. MLP confusion matrix using ‘adam’ solver in proof of concept case 1

6.1.3 Naïve Bayes

The Naïve Bayes classifier was able to correctly classify *92.5%* of the data. The confusion matrix in Table 29 details that it correctly classified 398 failure athletes and 96 success athletes. It misclassified 1 actual success athlete and 39 actual failures.

		Predicted	
		Failure	*Success*
Actual	*Failure*	398	39
	Success	1	96

Table 29. Naïve Bayes Confusion Matrix in proof of concept case 1

6.2 Case 2 – 20m

The criteria for case 2: athletes with a 20m less than or equal to 3.07s were labelled a success, the rest were labelled as a failure. The data summary of which the can be found in Table 30.

Performance Test	Criteria for success	Number of success	Percentage of success
20m	<= 3.07s	308	20.9%

Table 30. Data summary of case 2

6.2.1 Random Forest

Table 31 shows the summary of the random forest constructed in case 2. The testing data sub-set was classified 100% correctly. A total of 445 failure and 79 success athletes were predicted correctly, as seen in the confusion matrix Table 32.

Random Search			
Best Parameters	**No. of Fits**	**Train Accuracy**	**Test Accuracy**
'bootstrap': false *'criterion'*: ‘entropy’ *'max_depth'*: 70 *'max_features'*: ‘sqrt’ *'min_samples_leaf'*: 6 *'min_samples_split'*: 35 *'n_estimators'*: 1800	5 folds for each of 200 candidates, totalling 1000 fits	100%	100%

Table 31. Summary of random forest using random search in proof of concept case 2

		Predicted	
		Failure	*Success*
Actual	*Failure*	445	0
	Success	0	79

Table 32. Confusion matrix of random forest using random search in proof of concept case 2

Figure 12 shows a single tree from the random forest constructed in this case. The training data was made up of 1060 failure athletes and 186 success athletes. The root node implemented a test on the 20m variable, again only requiring the single decision to produce complete homogeneity.

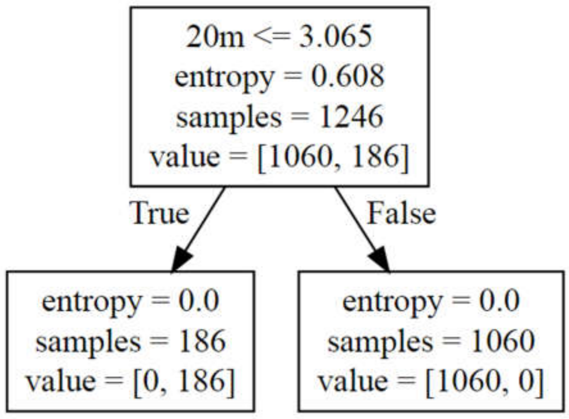

Figure 12. Single tree in the random forest of proof of concept case 2

6.2.2 Multilayer Perceptron

As summarised in Table 33, the MLP classifier was able to correctly classify 95.1% of the test subset in case 2. The confusion matrix (Table 34) details that it correctly classified 52 success athletes and 456 failure athletes. It misclassified 13 actual success athletes and 13 actual failures.

Using the 'adam' solver, testing accuracy improved to 99.3% (Table 35). The confusion matrix in Table 36 indicates that the classifier was able to correctly classify 75 success and 455 failure athletes. It only misclassified 1 actual success athlete and 3 actual failure athletes.

Parameters	Train Accuracy	Test Accuracy
‘activation’: 'relu' ‘early_stopping’: True ‘hidden_layer_sizes’: (10), ‘learning_rate’: 'adaptive' ‘learning_rate_init’: 0.3 ‘max_iter’: 100 ‘momentum’: 0.2 ‘solver’='sgd' ‘tol’ = 0.0001	94.2%	95.1%

Table 33. Summary of MLP performance in proof of concept case 2

		Predicted	
		Failure	*Success*
Actual	*Failure*	456	13
	Success	13	52

Table 34. MLP confusion matrix in proof of concept case 2

Parameters	Train Accuracy	Test Accuracy
‘activation’: 'relu' ‘early_stopping’: True ‘hidden_layer_sizes’: (10), ‘learning_rate’: 'adaptive' ‘learning_rate_init’: 0.3 ‘max_iter’: 100 ‘momentum’: 0.2 ‘solver’='adam' ‘tol’ = 0.0001	99.2%	99.3%

Table 35. Summary of MLP performance using ‘adam’ solver in proof of concept case 2

		Predicted	
		Failure	*Success*
Actual	*Failure*	455	3
	Success	1	75

Table 36. MLP confusion matrix using ‘adam’ solver in proof of concept case 2

6.2.3 Naïve Bayes

The Naïve Bayes classifier had an accuracy of *94.4%.*The confusion matrix in Table 37 indicates that the model correctly classified 78 success athletes and 426 failure athletes. However, it did misclassify 6 actual success athletes and 24 actual failure athletes.

		Predicted	
		Failure	*Success*
Actual	*Failure*	426	24
	Success	6	78

Table 37. Naïve Bayes Confusion Matrix in proof of concept case 2

6.3 Case 3 – CMJ and 20m

Criteria for success in case 3 involved a composite of two performance tests: athletes with a CMJ greater than or equal to 33.2cm and a 20m less than 3.14s were labelled a success, all athletes who did not meet these criteria were labelled as a failure. The data summary of case 3 can be found in Table 38.

Performance Test	Criteria for success	Number of success	Percentage of success
CMJ	>= 33.2cm	319	21.8%
20m	< 3.14s		

Table 38. Data summary of case 3

6.3.1 Random Forest

The random forest classifier constructed in case 3 was able to classify 100% of the testing subset correctly, details of which can be found in Table 39. The confusion matrix output from this ensemble, Table 40, shows that the model correctly classified 440 failure athletes and 94 success athletes.

Random Search			
Best Parameters	**No. of Fits**	**Train Accuracy**	**Test Accuracy**
'bootstrap': false *'criterion'*: 'gini *'max_depth'*: 10 *'max_features'*: 'auto' *'min_samples_leaf'*: 8 *'min_samples_split'*: 5 *'n_estimators'*: 600	5 folds for each of 200 candidates, totalling 1000 fits	100%	100%

Table 39. Summary of random forest using random search in proof of concept case 3

		Predicted	
		Failure	*Success*
Actual	*Failure*	440	0
	Success	0	94

Table 40. Confusion matrix of random forest using random search in proof of concept case 3

Figure 13 show the output of a single tree from the random forest constructed as part of training. The tree begins with 1018 failure athletes and 228 success athletes and required only two series of decisions to classify the data. The root node used the CMJ variable as the basis of the first test. In the second layer of decision nodes, the tree utilised the 20m performance test, as well as the CMJ, to further split the data. This particular tree in the random forest was unable to produce complete homogeneity within the final nodes; one of the final nodes contains 8 failure and 3 success athletes. The other final nodes, however, are completely homogenous.

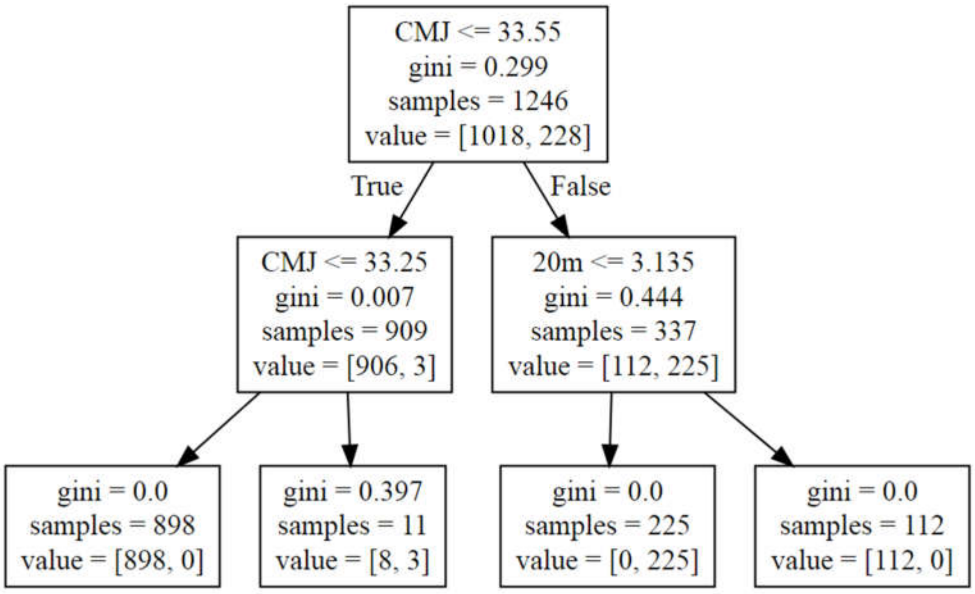

Figure 13. Single tree in the random forest in proof of concept case 3

6.3.2 Multilayer Perceptron

Table 41 shows that in case 3, the multilayer perceptron classifier was produced 94.0% test accuracy. As detailed in the confusion matrix in Table 42, this was broken down into: 71 success and 431 failure athletes' correctly classified; and 17 success and 15 failure athletes misclassified.

Parameters	Train Accuracy	Test Accuracy
‘activation’: 'relu' ‘early_stopping’: True ‘hidden_layer_sizes’: (10), ‘learning_rate’: 'adaptive' ‘learning_rate_init’: 0.3 ‘max_iter’: 100 ‘momentum’: 0.2 ‘solver’='sgd' ‘tol’ = 0.0001	94.5%	94.0%

Table 41. Summary of MLP performance in proof of concept case 3

		Predicted	
		Failure	*Success*
Actual	*Failure*	431	15
	Success	17	71

Table 42. MLP confusion matrix in proof of concept case 3

The solver hyperparameter was set to ‘adam’. Classification accuracy increased to 97.6% in the testing data, as shown in Table 43. The confusion matrix (Table 44) displays that, in the testing subset, the classifier was able to correctly classify 89 success athletes and 435 failure athletes. However, it misclassified 1 success athlete and 28 failure athletes.

Parameters	Train Accuracy	Test Accuracy
‘activation’: 'relu' ‘early_stopping’: True ‘hidden_layer_sizes’: (10), ‘learning_rate’: 'adaptive' ‘learning_rate_init’: 0.3 ‘max_iter’: 100 ‘momentum’: 0.2 ‘solver’='adam' ‘tol’ = 0.0001	97.4%	97.6%

Table 43. Summary of MLP performance using ‘adam’ solver in proof of concept case 3

		Predicted	
		Failure	*Success*
Actual	*Failure*	435	5
	Success	8	86

Table 44. MLP confusion matrix using ‘adam’ solver in proof of concept case 3

6.3.3 Naïve Bayes

The Naïve Bayes classifier was able to correctly classify 94.6% of athletes in case 3. As seen in the confusion matrix in Table 45, it was able to correctly predict 89 success athletes and 416 failure athletes. The classifier misclassified 1 actual success athlete and 28 actual failure athletes.

		Predicted	
		Failure	*Success*
Actual	*Failure*	416	28
	Success	1	89

Table 45. Naïve Bayes Confusion Matrix in proof of concept case 3

6.4 Analysis

The proof of concept proves that machine learning is an effective and valid method of analysis that can be used as a valid tool of prediction and talent spotting of young athletes based on performance testing results. Specifically, the models used in this project, and their relevant parameters, can be used as effective tools in other projects of this nature. The proof of concept also helps to confirm the suspicion that the data provided for this project lacked the necessary quality and quantity needed to perform machine learning analysis.

In each random forest, the root node selected by the model was the variable on which the criteria for the case was based on; this was to be expected as the most information can be gained from this variable. This further proves that the random forests in this project models are viable. The inability of the single tree output from case 3 (Figure 13) to provide homogeneity in the final node, yet the model itself still able to produce a 100% classification accuracy on the test set, exhibits the ability of the random forest model to average the multiple decision trees in the ensemble to reduce variance – producing a classifier that will generalise effectively to unseen data.

The multilayer perceptron produced some mixed results. Initially, the hyperparameters used were those in table 14. This produced the same result as in Chapter 4 *Performance Test Modelling* – the classifier predicted all athletes as a failure. However, with hyperparameter optimisation (hyperparameters in Table 16) the classifier managed to perform highly accurately. The hyperparameters made for a simpler model; MLPs possess the propensity to overcomplicate problems if given unsuitable hyperparameters for the task at hand. Most notable, the number of hidden layers were reduced from *10, 10, 10*, to *10*; reducing the total number of functions applied to the inputs, therefore reducing complexity. The simpler model was discovered to be effective during the proof of concept stage. It was added into the performance modelling stage in retrospect, to determine whether this improved model may be able to classify the data given by the client. A number of hyperparameter were altered in an attempt to optimise the MLP proof of concept, however, the most significant improvement was observed when using 'adam' as the solver hyperparameter. The Adam optimisation algorithm is an augmentation of stochastic gradient descent. Stochastic gradient descent uses a constant learning rate for all weight updates throughout training, whereas, Adam maintains an individual learning rate for each network weight and updates each separately as learning develops.

The Naïve Bayes is known to perform well on simple problems; therefore, one would expect the proof of concept to be an optimal dataset for the classifier. It is perhaps its independence assumption that caused minor difficulty. It could be argued that all of the performance tests are linked somewhat, but the relationship between the 5m, 10m and 20m is incontrovertible.

7 Conclusion

7.1 Summary

With machine learning classifiers and statistics, the project attempted to determine whether success can be predicted from performance test results of FVFA players and provide insight into the most indicative performance tests. At the outset, the project required a study of the Python programming language and its Pandas, NumPy and Scikit-learn libraries to be able to efficiently implement the necessary machine learning algorithms.

The project was unable to build a sufficient classifier with the historic dataset as too much class imbalance occurred. It was initially thought that the machine learning classifiers were producing relatively high classification accuracy; however, following analysis of the confusion matrices of each, it transpired that the classifiers were predicting all athletes as failures. Due to the extent of the imbalance in the data this gave a false sense of classifier effectiveness. An investigation of the most indicative performance tests was also attempted using feature importance extraction from the random forest. Initially it was thought that the CMJ was candidate for the most indicative performance test in this data set; however, following the discovery of the random forests propensity to fail all athletes in this dataset, this finding was established as invalid. The Naïve Bayes classifier did produce a model that performed classification, with an accuracy rate of 77.3%. Again, due to the imbalance in the data, the classifier was unable to outperform a random guess.

Despite the performance of the classifiers, statistical analysis was able to prove a significant positive correlation between time spent in the academy by an athlete and success. It was suggested that this could be due to an increased exposure to high-level coaching and training, players who last longer in the academy have always played at a high-level relative to peers, or a combination of both.

A proof of concept was formulated to assess the viability of the machine learning algorithms used in this project. The proof of concept proved that the algorithms researched and employed, with their associated hyperparameters, are efficient tools capable of classifying data of this nature given a more satisfactory dataset. These tools can be utilised going forward on more superior datasets to provide the desired insight into test performance and, eventually, produce a model capable of effectively classifying vast amounts of unseen data that can be used by talent identifiers to help identify talent.

7.2 Evaluation

The project was successful in providing tools for future use in machine learning with sports performance testing data. It was also successful in correlating success with the length of time in the academy. The statistics used to prove this were simple and easy to implement given sufficient software. Initially accuracy score was implemented as the sole measure of model evaluation, however, this project highlights importance of using confusion matrices. The accuracy scores for all the models were all around 82-83%, and it happened to be the case that there were 82.8% of athletes were labelled as a failure in the data. It was not until confusion matrices were produced was it discovered the classifiers were predicting all athletes as failures. As a result, the project was unable to fulfil the goals of the client wholly. The inability to produce an effective classifier can be attributed to the several issues in the dataset provided:

1) The dataset had high class imbalance, although that is the general nature of success vs. failure in sport, this caused the random forest and multilayer perceptron classifiers to find it more efficient to predict all athletes as a failure than to perform classification.

2) Given the data imbalance, the relationship between the performance testing results and success was not strong enough. However, this reflects the complexity of the interaction of components that make up high-level performance i.e. it is unlikely that there will be one or two clear factors that make for high-level performance.
3) The dataset was not large enough. Predominantly this meant that methods to balance the data could not be applied.
4) The dataset had too few features. Again, given the complexity of the building blocks underlying sporting success, having only a few features likely does not provide enough information.

To highlight issues 2 and 4 above, the performance test results (Z-scored) were extracted and labelled a success or failure at random with a roughly 50:50 ratio. The random forest algorithm was still unsuccessful at effective classification. In other words, nothing can be learned from this dataset by the algorithm. See appendix 2 for details.

It was overlooked that some of the tests were completed so recently that the athlete is not old enough to sign a professional contract, therefore, some of the athletes labelled as a failure now would have been labelled a success if this project was conducted later. This causes the obvious issue of performance test results that belong to (future) successful athletes are learned as unsuccessful results by the algorithms in this project. The data issues identified in this dataset would need to be absent in future dataset if effective classifiers are to be built.

In hindsight, the machine learning classifiers selected still seem a sound choice. Commonly, the Naïve Bayes assumes features are independent of each other, which could be a criticism of selecting this classifier. Yet, it has been seen to still achieve strong classification when this assumption is violated as discussed mathematically by Domingos and Pazzani [11]. Similarly, random forest and (especially) MLP classifiers are traditionally thought of as unsuitable for smaller datasets. However, as previously mentioned, it has been shown that they can still perform surprisingly well [35,43].

The project draws attention to the complexity of success in sport. It is the combination of the interaction between many factors, and not just performance testing. In an ideal situation, a dataset would contain extensive information about athletes addressing as many as these factors as possible. This would be the ‘gold standard’ of dataset to produce classifier that could be used by talent identifiers. Given its complexity, machine learning is an ideal method for analysis of sports performance going forward.

7.3 Future Work

Whilst this project is a good initial step into using machine learning on sports performance testing results, there is still a way to go before a workable model can be produced to be used in talent identification.

A possible next step using this dataset would be to implement oversampling techniques, such as SMOTE, then reinvestigate the classifier results. An enquiry into why the Naïve Bayes classifier in this project produced the results it did would also be a reasonable avenue of next study.

Perhaps a more superior next step would be to implement the techniques used in this project with a more viable dataset. An ideal candidate dataset would include vastly more features and more data points, on more factors contributing to success than just performance testing data. Given the proof that the models selected in this project have the ability to produce classification on this type of data, it is hoped that work on a more desirable dataset would provide practical insight. It would then be possible to suitably assess the models and perform constructive hyperparameter optimisation. This practical insight would provide knowledge to coaches and athletes on aspects to address to improve performance and increase chance of future success. It would also highlight which performance tests and results are most important when assessing players for talent identification. Perhaps an end goal of this line of study would be to develop a programme running on a classifier that would allow talent identifiers and coaches to input an athlete’s results, and output that athlete’s likelihood of future success and areas for improvement.

References

[1] A. Abbott and D. Collins, "Eliminating the dichotomy between theory and practice in talent identification and development: considering the role of psychology," J. Sports. Sci., vol. 22, (5), pp. 395, 2004.

[2] C. C. Aggarwal and C. Zhai, "A survey of text classification algorithms," in Mining Text Data. ggarwal C., Zhai C. Springer, 2012, pp. 163.

[3] Anaconda Inc., "Anaconda Documentation," 2018.

[4] J. Bangsbo, "Physiological demands of Football," Sports. Sci. Exch., vol. 27, pp. 1, 2014.

[5] K. W. Bowyer et al, "SMOTE: Synthetic Minority Over-sampling Technique," J. Artif. Intell. Re., vol. 16, 2002.

[6] L. Breiman, "Machine Learning," vol. 24, (2), pp. 123, 1996.

[7] C. Bustamante, L. Garrido and R. Soto, "Comparing Fuzzy Naive Bayes and Gaussian Naive Bayes for Decision Making in RoboCup 3D," Micai, vol. 4293, pp. 237, 2006.

[8] P. Chapman, J. Clinton and R. e. a. Kerber, CRISP DM 1.0: Step-by-Step Data Mining Guide. USA: SPSS, 2000.

[9] C. Chen, A. Liaw and L. Breiman, "Using Random Forest to Learn Imbalanced Data," University of California, Berkeley.

[10] M. C. P. de Souto, V. G. Bittencourt and J. A. F. Costa, "An empirical analysis of undersampling techniques to balance a protein structural class dataset," in King I., Wang J., Chan LW., Wang D. (Eds) Neural Information Processing. ICONIP 2006. Lecture Notes in Computer Science, Vol 4234Anonymous Berlin, Heidelberg: Springer, 2006,

[11] P. Domingos and M. Pazzani, "On the Optimality of the Simple Bayesian Classifier under Zero-One Loss," Mach. Learn., vol. 29, pp. 103, 1997.

[12] E. Frank, M. A. Hall and I. H. Witten, "The WEKA Workbench. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques"," Morgan Kaufmann, vol. Fourth Edition, 2016.

[13] M. W. Gardner and S. R. Dorling, "Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences," Atmos. Environ., vol. 32, (14), pp. 14, 1998.

[14] X. Glorot, A. Bordes and Y. Bengio, "Deep sparse rectifier neural networks," Aistats, vol. 15, pp. 315, 2011.

[15] F. Godin et al, "Dual Rectified Linear Units (DReLUs): A Replacement for Tanh Activation Functions in Quasi-Recurrent Neural Networks," Arxiv:1707. 08214v2.

[16] B. Gregorutti, B. Michel and P. Saint-Pierre, "Correlation and variable importance in random forests," Arxiv:1310. 5726v5, 2016.

[17] M. J. Hallett et al, Stat. Model., vol. 14, (6), pp. 523, 2014.

[18] D. J. Hand and K. Yo, "Idiot's Bayes-Not So Stupid After All?" International Statistics Review, vol. 69, (3), pp. 385, 2001.

[19] N. L. Holt and D. Morley, "Gender differences in psychosocial factors associated with athletic success during childhood," Sport. Psychol., vol. 18, (2), pp. 138, 2004.

[20] J. D. Hunter, "Matplotlib: A 2D Graphics Environment," Comput. Sci. Eng., vol. 9, pp. 90, 2007.

[21] IBM Corp, "IBM SPSS Statistics for Windows," vol. 25.0, 2017.

[22] M. I. Jordan, "Why the logistic function? A tutorial discussion on probabilities and neural networks." Computational Cognitive Science Technical Report 9503, Massachusetts Institute of Technology, 1995.

[23] D. Kornbrot, "Point biserial correlation," Wiley StatsRef Statistics Reference Online, 2014.

[24] A. Krizhevsky, I. Sutskever and G. Hinton, "ImageNet classification with deep convolutional neural networks," Nips, vol. 1, pp. 1097, 2012.

[25] B. Lantz, Machine Learning with R. Birmingham, UK: Packt Publishing, 2013.

[26] H. Lodish, A. Berk and S. L. e. a. Zipursky, "Section 21.1, overview of neuron structure and function," in Molecular Cell Biology, 4th Edition ed. Anonymous New York: W. H. Freeman, 2000, .

[27] G. Louppe, "Understanding Random Forests.", University of Liege, Liege, 2015.

[28] J. McCullagh, "Data Mining in Sport: A Neural Network Approach," International Journal of Sports Science and Engineering, vol. 04, (03), pp. 1750, 2010.

[29] M. L. McHugh, "The Chi-square test of independence," Biochemia. Medica., vol. 23, (2), pp. 143, 2013.

[30] W. McKinney, "Data Structures for Statistical Computing in Python," Proceedings of the 9th Python in Science Conference, pp. 51, 2010.

[31] B. Millington and R. Millington, "'The Datafication of Everything': Toward a Sociology of Sport and Big Data," Sociol. Sport J., vol. 32, (2), pp. 140, 2015.

[32] M. Mohr, P. Krustrup and J. Bangsbo, "Match performance of high-standard soccer players with special reference to development of fatigue," J. Sports. Sci., vol. 21, pp. 439, 2003.

[33] T. E. Oliphant, A Guide to NumPy. USA: Trelgol Publishing, 2006.

[34] A. G. Parlos et al, "An accelerated learning algorithm for multilayer perceptron network," IEEE Transactions on Neural Networks, vol. 5, (3), pp. 493, 1994.

[35] A. Pasini, "Artificial neural networks for small dataset analysis," J. Thorac. Dis., vol. 7, (8), pp. 953, 2015.

[36] D. T. Pearson, G. A. Naighton and M. Torode, "Predictability of physiological testing and the role of maturation in talent identification for adolescent team sports," J. Sc. i Med. Sport, vol. 9, (4), pp. 277, 2006.

[37] Pedregosa et al., "Scitkit-learn: Machine Learning in Python," Jmlr, vol. 12, pp. 2825, 2011.

[38] Project Jupyter, "The Jupyter Notebook," 2015.

[39] Python.org, "Python 3.6.6 documentation," 2016.

[40] E. Rampinini, D. Bishop and S. e. a. Marcora, "Validity of simple field tests as indicators of match-related physical performance in top-level professional soccer players," Int. J. Sports. Med., vol. 28, (3), pp. 228, 2007.

[41] R. Rein and D. Memmert, "Big data and tactical analysis in elite soccer: future challenges and opportunities for sports science," Springerplus, vol. 5, (1), pp. 1410, 2016.

[42] D. W. Ruck et al, "The multilayer perceptron as an approximation to a Bayes optimal discriminant function," IEEE Transactions on Neural Networks, vol. 1, (4), pp. 296, 1990.

[43] T. Shaikhina, D. Lowe and S. e. a. Daga, "Decision tree and random forest models for outcome prediction in antibody incompatible kidney transplantation," Biomed Signal Process Control, 2017 Feb 09.

[44] L. Smith, B. Lipscomb and A. Simkins, "Data Mining in Sports: Predicting Cy Young Award " Jcsc, vol. 22, (4), pp. 115, 2007.

[45] Spyder project contributors, "Spyder - Documentation," 2018.

[46] M. Stein, H. Janetzko and D. e. a. Seebacher, "How to Make Sense of Team Sport Data: From Acquisition to Data Modeling and Research Aspects," Data, vol. 2, (1), 2017.

[47] C. Strobl, A. L. Boulesteix and A. e. a. Zeileis, "Bias in random forest variable importance measures: Illustrations, sources and a solution," BMC Bioinformatics, vol. 8, (25), 2007.

[48] R. Susmaga, "Confusion matrix visualization," in Kłopotek, M.A.; Wierzchoń, S.T.; Trojanowski, K. (Eds) Intelligent Information Processing and Web MiningAnonymous Springer, 2004, pp. 107.

[49] R. Vaeyens et al, "Talent Identification and Development Programmes in Sport," Sports Med., vol. 38, (9), pp. 703, 2008.

[50] J. Van den Broeck et al, "Data Cleaning: Detecting, Diagnosing, and Editing Data Abnormalities," PLoS Med., vol. 2, (10), 2005.

[51] A. Vehbi Olgac and B. Karlik, "Performance Analysis of Various Activation Functions in Generalized MLP Architectures of Neural Networks," IJAE, vol. 1, pp. 111, 2011.

[52] B. Vikramkumar and T. Vijaykumar, "Bayes and Naive Bayes Classifier," Arxiv, 2014.

[53] J. Wainer, "Comparison of 14 different families of classification algorithms on 115 binary datasets," Arxiv: 1606.00930, 2016.

[54] Z. Zhao et al, "Investigation of Multilayer Perceptron and Class Imbalance Problems for Credit Rating," Ijcsit, vol. 3, (4), pp. 805, 2014.

Appendix 1

Appendix 1.1

```
criterion = ['gini', 'entropy']
n_estimators = [int(x) for x in np.linspace(start = 200, stop = 2000, num =
10)]
max_features = ['auto', 'sqrt']
max_depth = [int(x) for x in np.linspace(10, 110, num = 11)]
max_depth.append(None)
min_samples_split = [2, 5, 10, 15, 20, 25, 30, 35, 40, 50, 60]
min_samples_leaf = [1, 2, 4, 6, 8, 9, 12, 15, 20, 25, 30, 35, 40,]
bootstrap = [True, False]
```

Appendix 1.2

```
from sklearn.preprocessing import MinMaxScaler
 min_max_scaler = preprocessing.MinMaxScaler(feature_range=(-1, 1))
X_scaled = min_max_scaler.fit_transform(X)
X_normalized = pd.DataFrame(X_scaled)
```

Appendix 2

Appendix 2 provides further detail on the analysis of identifying whether a model can learn from the dataset in the project.

	Success	**Failure**
Number of instances	916	859
Percentage	51.6%	48.4%

Table 46. Ratio of success and failure for evaluation of strength in relationships in the data

Random Search			
Best Parameters	**No. of Fits**	**Train Accuracy**	**Test Accuracy**
'bootstrap': false *'criterion'*: gini *'max_depth'*: 40 *'max_features'*: 'sqrt' *'min_samples_leaf'*: 40 *'min_samples_split'*: 60 *'n_estimators'*: 1600	5 folds for each of 200 candidates, totalling 1000 fits	68.5%	52.2%

Table 47. Random Forest summary of the evaluation of strength in relationships in the data

		Predicted	
		Failure	*Success*
Actual	*Failure*	142	103
	Success	152	136

Table 48. Confusion matrix output to evaluate strength in relationships in the data