Advancing Predictive Analytics in Sports: Developing a Novel Machine Learning Model for Predicting PGA TOUR Player Finishes

Sean Quinton

25725317

25725317@students.lincoln.ac.uk

School of Computer Science

College of Science

University of Lincoln

Submitted in partial fulfilment of the requirements for the

Degree of BSc(Hons) Computer Science

Supervisor: Prof. Xujiong Ye

April 2022

Acknowledgements

Firstly, I want to thank my supervisor Xujiong Ye for the constant support and help throughout the whole process of this project.

Next, I want to thank my family for always supporting me throughout my life and helping be on this journey.

Finally, I want to thank my girlfriend, Jasmine, for supporting me through this year and the stresses that come along with it.

Abstract

Predictive analytics have become an important tool for success in modern sports data analysis, especially in professional golf, where accurate predictions can be utilised by players and coaches to enhance practise and tournament strategy. This project builds and evaluates machine learning models to predict PGA TOUR player finishes through data pre-processing, feature selection, and model optimisation. This study compares the predictive performance of Linear Regression, Random Forest, Gradient Boosting, and Perceptron Neural Network models exploiting techniques such as k-fold cross-validation and grid search to optimise model performance. Each model's performance is measured using RMSE, MAE, and R^2 as well as scatter plots to visualise predictions. The results show that the Perceptron Neural Network is the best performing model on this dataset achieving the best metric results.

Table of Contents

List of Figures

List of Tables

Chapter 1 - Introduction

1.1 The role of predictive modelling in PGA TOUR success

In the competitive world of professional golf, the PGA (Professional Golf Association) TOUR has always been seen as the pinnacle of achievement and skill, showcasing the best talent in the sport. With tournament winnings ranging from 1.5 million dollars to upwards of 4 million dollars (www.pgatour.com, n.d.) and audiences growing, the pressure on players and their coaching teams to deliver consistent winning performances has been higher than ever. In this data-driven era, the reliance on advanced analytics to gain a competitive edge is essential to success in the modern game. Predictive modelling has become an invaluable tool, offering insights that go beyond traditional statistics and performance metrics. By accurately predicting outcomes, coaching teams can strategise more effectively, optimise player performance, and enhance tactical decision-making. This dissertation focuses on the integration of machine learning techniques to develop a sophisticated and robust predictive model aimed at predicting player finishes on PGA TOUR events. The model aims to improve predictions as well as deepen our understanding of the factors and statistics that most significantly impact player success, thereby contributing a novel perspective to the analytics used in golf today.

1.2 Evolution of data analytics in golf

The integration of data analytics into sports has revolutionised how athletes and teams approach training, strategy, and performance optimisation. In golf, traditionally viewed as a game of intuition and skill, the adoption of analytics started a shift towards a more empirical and data-driven approach to the game.

This evolution started simply with counting strokes and measuring driving distances but as the game has grown with technologies, this has expanded to sophisticated analyses involving biometrics, environmental factors, and equipment optimisation.

The early 2000s witnessed a pivotal change with the introduction of technologies like ShotLink. Developed by the PGA TOUR, ShotLink uses lasers and GPS to collect detailed data on every shot played during a tournament. This technology provided a large volume and variety of data, enabling deeper insights into player performance and game management. Analysts could now factually quantify areas of golf that previously were left to subjective judgement, such as shot risk assessments and the strategic value of different playing styles in differing course and environmental conditions (Shotlink.com, 2016).

Later advancements have included the use of swing analysers, ball tracking systems, and simulation models that help in understanding not just the ‘how’, but the ‘why’ behind shot outcomes. These tools provide useful feedback on swing speed, ball spin, and launch angles, which are crucial for customising training and improving player performance over time (Martinez Arastey, 2020).

Today, the use of big data and machine learning in golf extends beyond individual player performance, influencing equipment design, course architecture, and fan engagement strategies. Predictive models are now employed to predict tournament outcomes, optimise practise routines, and even tailor marketing to fan preferences, demonstrating the wide-reaching implications of data analytics in modern golf.

This rich history and ongoing innovation in golf analytics provide a solid foundation for my study, which aims to build on these technological advancements to predict PGA TOUR outcomes more accurately than before.

1.3 Challenges in current predictive models for PGA TOUR outcomes

While data-driven approaches in sports analytics have seen large advancements, current predictive models for PGA TOUR outcomes show a focus on classification and predicting solely winners.

The inspiration for this project is a similar project that aims to use historical PGA TOUR statistics to predict if a player won any tournament that year and how much money they earnt (Park, 2019). This led me to research whether there were any existing projects that use machine learning and specifically regression models to predict where every player finishes a tournament based on historical statistics. Although some similar projects do exist, for example a classification model similar to the previously mentioned project to predict if a player won in a certain year created by Daron Prater on GitHub (Daron Prater, 2018), there are no existing projects with the same objectives as my project, which I will mention in the next section.

By addressing the current lack of regression models using historical PGA TOUR data, this project aims to develop a predictive model that uses appropriate and advanced machine learning methodologies to predict PGA TOUR player outcomes for all players in a tournament with accuracy and reliability.

1.4 Implementation overview

In the initial stages of the implementation stage of this project, extensive data cleaning and organisation were essential to prepare the data for predictive modelling. Using the Python library Pandas, the dataset was first processed to make sure it was suitable for analysis. This involved extracting key information such as tournament names, player names, player statistics, and outcomes from structured data formats, demonstrating a systematic approach to managing a complex dataset.

The dataset was then structured to allow for detailed statistical analysis and machine learning modelling. This included pivoting the data table to align player-specific performance statistics with the corresponding tournament outcomes, thereby creating a dataset with ground truths allowing for the upcoming application of machine learning algorithms. Other pre-processing steps, such as normalising percentage values and converting measurements from mixed units (feet and inches to inches), were implemented to standardise the data therefore enhancing the predictive power of my predictive models.

These preliminary steps set out a good foundation for applying advanced machine learning techniques onto the dataset. The implementation of these techniques as well as further explanation of the data organisation will be discussed in the implementation section of this paper.

1.5 Structure of the dissertation

This dissertation is split into multiple chapters, each one divulging into a different part of the study. The structure of these chapters are as follows.

The literature review critically analyses existing literature on the use of predictive analytics in golf and other sports, highlighting historical developments, the current state of technology, and gaps in current research and methodologies. This chapter will also discuss how my implementation will improve on current predictive models and fill any gaps found in current research.

In the requirements analysis chapter, the specific requirements for the predictive models will be defined. The data needs will be discussed as well as the expected functionalities of the predictive models, and the criteria for evaluating their effectiveness based on the gaps discovered in the literature review.

The theoretical framework and design of the study will be outlined in the design and methodology chapter. It will also detail the choice of machine learning techniques and algorithms, the reason behind their use, and the methods used for data collection, preparation, and analysis.

The implementation chapter will describe the practical application of the techniques and algorithms discussed in the previously mentioned chapter. It will cover the entire implementation process as well as any challenges faced along the way.

In the results and discussion chapter, the findings from the project will be discussed. It will talk about the results from the predictive models and how these results compare to existing models.

The final chapter is the conclusion. This chapter evaluates the success of the project in meeting the aims and objectives that are set out below, as well as talking about areas for improvement, the limitation of the project, and suggesting gaps for future research to take place.

1.6 Aim & Objectives

1.6.1 Aim

This project aims to utilise advanced machine learning techniques to develop a robust model that predicts individual player finishes in PGA TOUR events using historical data.

1.6.2 Objectives

- To gather and pre-process PGA TOUR data from eight 2023 tournaments to create a dataset with clean, complete, and formatted data ready for analysis.
- Use scatter plots to identify relationships between statistics and how they affect player outcomes.
- Compare and evaluate at least three machine learning algorithms as well as using advanced techniques to assess their performance based on metrics

such as Mean Squared Error achieving a RMSE score of 10 or lower as a minimum.

- Implement and optimise parameter tuning techniques such as k-fold cross-validation and grid search to improve model accuracy.
- Visualise the accuracy of the models using scatter plots to compare the model's predictions and the actual outcomes.

Chapter 2 - Literature Review

2.1 Data analysis in sports

The use of data analysis in sports has gone from a growing and experimental approach to an essential strategy for success. An example of this evolution is a study conducted by Leung, C.K. and Joseph, K.W. (2014), which explores the use of data mining techniques to predict outcomes in college American football games. Using a range of machine learning algorithms, including regression models, decision trees, and neural networks, the study demonstrates that a data-driven approach can significantly outperform traditional predictive methods (Leung and Joseph, 2014).

This study shows a comprehensive approach to sports data analytics, not only applying many data mining techniques but also validating these models against real game outcomes. This validation is a crucial part of the study as it ensures the reliability of the predictions and shows the real-world utility of data-driven decision-making in sports (Leung and Joseph, 2014). Although the study effectively shows the need for predictive modelling in college American football, it raises the question of whether these models can be generalised to other sports, including golf, where different dynamics and variables will influence the accuracy of the predictions.

In golf, specifically the PGA TOUR, factors such as course conditions, weather, and player form play a significant role, the adaptation of the data mining techniques mentioned in the study could result in some more nuanced insights. By integrating both historical and real-time data, that was not utilised by Leung and Joseph, my project aims to enhance the adaptability and accuracy of predictive models. This approach promises not only to improve the accuracy of the predictions but also to make them more responsive to immediate factors that impact the outcome of matches in golf, offering a substantial improvement over existing models that are based solely on historical data.

In a study by Fried, Lambrinos, and Tyner (2004), the trio offer an insightful analysis into the performance evaluation of professional golfers across the PGA, LPGA (ladies PGA), and SPGA (senior PGA) tours. Their study identifies key performance indicators critical to success in professional golf and establishes a predictive framework exploiting these metrics (Fried, Lambrinos, and Tyner, 2004). This focused examination of golf provides an in-depth look at golf's unique dynamics, setting a standard for subsequent analytical approaches.

The authors' methodical use of historical data allows for comprehensive trend analysis, establishing a robust base for future predictive modelling. This approach effectively demonstrates how long-term performance data can be used to predict future outcomes in professional golf, providing insights that are useful for both player development and strategic planning.

However, similar to the previous study, this study primarily uses traditional statistical methods, and while effective, they may not capture the full complexity of data interactions present in today's more dynamic sporting environments. Also, the reliance on historical data, without incorporating real-time variables, potentially limits the usefulness of their predictions under varying conditions, such as changes in weather or player health etc.

Building on the work by Fried, Lambrinos, and Tyner, my project aims to integrate advanced machine learning techniques, such as random forests and gradient boosting models. These techniques are better suited to model the intricate relationships between the wide range of statistics that will be essential in creating accurate and reliable predictions, including real-time data such as weather conditions and players' physical condition.

2.2 Predictive modelling in sport

The use of video data for real-time sports analysis represents a significant advancement in sports analytics, as demonstrated by Hanson and colleagues in their 2012 study. Their research introduces an innovative method for tracking and predicting ball movements in football through a combination of video-based notational analysis and machine learning techniques. This approach not only captures the dynamic aspects of the game but also enhances prediction accuracy, which is important for effective game strategy and player training (Hanson et al., 2012).

The group's methodology excels in its detailed visual assessment capability, allowing for an analysis of nuances in ball trajectories that are often missed by traditional statistical data. By implementing machine learning algorithms, the study extends the potential of video data from just observation to predictive utility, setting a standard for future sports analytics applications.

However, the study specifically focuses on ball movements, missing out on the broader application to player performance metrics. Also, the resource-intensive nature of processing video data presents challenges in terms of scalability and real-time applicability. These areas offer room for improvement and adaptation into other sports such as golf.

Using the foundational work by Hanson et al, my project aims to adapt these video analysis techniques to golf analytics, focusing on player performance metrics such as strokes gained and driving accuracy. By implementing more efficient machine learning algorithms and optimising data processing techniques, the predictive model in my study looks to achieve greater scalability and real-time usability. This approach not only aims to improve the predictive accuracy of player performances in golf, but also ensures the model's practicality for real time application, hopefully broadening the scope and impact of predictive modelling in sport.

A study by Constantinou and Fenton (2017) develops innovative approaches to predictive accuracy in sports performance by using Bayesian networks and smart data. Their study, which focuses on long-term football team performance, is an example of the shift towards integrating multiple data sources to create robust predictive models that effectively manage uncertainties inherent in sports outcomes. This approach is notable for its comprehensive data integration, which allows for a deeper understanding of complex factors that influence team sports performances.

A strength of their study is the use of Bayesian networks to accommodate the probabilities and uncertainties that traditional predictive models often overlook. This flexibility makes their model exceptionally adaptable to the variable nature of sports, providing a framework that could be especially beneficial in predicting outcomes where conditions are continually changing.

A drawback to the study is that their model predominantly focuses on team sports, which presents a different set of variables and dynamics compared to individual sports like golf. For example, golf involves unique external factors such as course conditions and weather, which impact performance and player approach in ways that are distinct from team interactions in sports like football where all pitches are effectively identical and weather, other than extreme weather, does not impact the game to the same degree as golf.

My project will build on the study by Constantinou and Fenton by tailoring these advanced modelling techniques to golf. By adapting Bayesian networks to incorporate individual performance metrics and golf-specific external factors, the predictive accuracy for player outcomes in tournaments can potentially be enhanced. Also, integrating real-time data into this model will capture the dynamic nature of a golf tournament as well as allow for predictions that reflect the immediate conditions affecting player performance.

2.3 machine learning techniques

A study by Adam Maszczyk et al. (2014) evaluates the accuracy of neural networks and regression models in predicting sports performance, specifically focusing on javelin throwers. This research offers a detailed analysis of the effectiveness of these predictive methods and serves as a critical point of comparison for the development of similar models in golf (Maszczyk et al., 2014).

Maszczyk and his team tested non-linear regression models and multilayer perceptron neural networks to predict the performance of javelin throwers. The key predictive variables identified (cross-step time, specific power of the arms and trunk, abdominal muscle strength, and grip power) were found to significantly influence performance outcomes, with neural networks providing more accurate predictions and lower absolute errors compared to regression models.

The study's strength lies in its comprehensive use of data, which uses a broad range of performance metrics to create predictive models. This extensive data utilisation offers a holistic view of athlete capabilities, enhancing the robustness of the predictive models. Additionally, the direct comparison of different machine learning approaches under controlled conditions provides valuable insights into their respective predictive capabilities and limitations which I can utilise in my study for optimal results.

While the study provides foundational knowledge on using machine learning for sports performance prediction, its focus on javelin throwing poses challenges in direct application to golf, which involves a different set of performance metrics and physical demands. Furthermore, the lack of real-time data integration in the study highlights an area for improvement, especially for adapting these models to the dynamic conditions seen in golf tournaments.

Using Maszczyk et al.'s methodologies, this project aims to adapt and enhance these machine learning frameworks to suit the specific requirements of golf. This adaptation involves focusing on golf-centric performance metrics such as strokes gained and shot accuracy. By integrating both historical performance data and real-time conditions, such as weather data and course specifics, the models can be made more dynamic and responsive to the unique challenges of each PGA TOUR event.

Similarly, Herbert F Jelinek et al.'s (2013) study represents a significant step forward in the application of meta-regression techniques within a data mining framework, aimed at enhancing the predictive modelling of Australian football players' performance through their heart rate dynamics. This study combines physiological data with performance metrics, providing a refined approach to sports analytics (Jelinek et al., 2014).

Jelinek and his team utilised meta-regression models to effectively aggregate outcomes from multiple simpler regression models, thereby enhancing the overall predictive accuracy. A key finding of their research was the significant linkage between heart rate variability and key performance metrics such as in-game endurance and effectiveness. This underscores the potential of physiological data to provide deep insights into an athlete’s fitness and game-time performance capabilities. The use of physiological data, as pioneered in this study, provides critical insights into player fitness and game-time performance, highlighting the advanced statistical techniques like meta-regression that improve the robustness and accuracy of predictions by combining various model outputs.

While Jelinek et al.'s research offers a novel approach to sports performance prediction, its focus on team sports and metrics specific to high-intensity activities poses limitations when considering its application to golf. Golf, differing significantly in its pacing and skill requirements, demands a different analytical approach that considers both physiological and non-physiological data such as psychological states and external conditions like weather, which also play crucial roles in performance outcomes.

The adaptation of meta-regression techniques for golf presents an opportunity for my study. By combining various golf-specific statistical models (predicting performance based on strokes gained, driving accuracy, and putting efficiency etc) this approach could integrate data from different tournaments and varying conditions to build a comprehensive predictive model.

Although Jelinek et al.'s research highlights the benefit of using physiological data to improve the accuracy of their meta-regression models, implementing this into my study is unrealistic as access to this data would require my own participants and access to equipment that is currently unavailable to me.

Chapter 3 - Requirements Analysis

3.1 Functional requirements

Before analysing the dataset, the program must be able to import and pre-process a cleaned historical PGA TOUR dataset which contains a wide range of useful statistics such as “Driving accuracy” and “Strokes gained” as well as the tournament, player name, and finishing position to give the models a ground truth to compare to. Data pre-processing should involve standardising features and converting values to uniform units to ensure compatibility across different models. The data is organised to provide a comprehensive look at each player’s performance.

Feature engineering and selection requires encoding relevant features and generating interaction terms to capture the nuanced relationships between stats that impact player outcomes. Scatter plots should be implemented to visualise these relationships and help to identify feature importance and relationships with player finishes.

Predictive models, including random forest, gradient boosting, linear regression, and a perceptron neural network, should be implemented and optimised using k-fold cross-validation and grid search. Model performance should be evaluated using metrics like Root Mean Squared Error (RMSE), Mean Absolute Error (MAE) and R^2 score. This will allow me to assess different aspects of the models’ performance. The program must then display a scatter plot of the model’s performance with a reference to perfect accuracy to visualise the predictive capabilities of the models.

3.2 Non-functional requirements

- **Performance and efficiency:** Data pre-processing should not exceed 7 minutes, and model training and testing should complete within 15 minutes.
- **Accuracy and reliability:** Models should achieve an RMSE of less than 10, a MAE of less than 10, and R^2 should be a positive number.
- **Usability and visualisations:** visualisations comparing actual vs predicted values should clearly represent predicted finish positions and actual finish positions.
- **Scalability and flexibility:** The model should allow for additional historical data and allow for flexibility in hyperparameter tuning.
- **Compatibility:** The program should run on Windows, MacOS and Linux requiring only standard Python libraries and machine learning frameworks.

3.2.1 Risk analysis and mitigation strategies

Table 1: Risk analysis and mitigation strategies.

Risk	**Impact**	**Likelihood**	**Mitigation Strategy**
Data Quality Issues	High	Medium	- Perform comprehensive data cleaning and standardisation.
			- Conduct exploratory data analysis (EDA) to detect anomalies.
Model Overfitting	High	High	- Implement k-fold cross-validation and grid search.
			- Regularise models using techniques like dropout.
Lack of Data Diversity	High	Medium	- Expand the dataset by incorporating historical PGA TOUR data.
			- Use at least 8 different previous tournaments for data variety.
Computational Limits	Medium	Medium	- Optimise hyperparameters using grid search.
			- Use early stopping and learning rate reduction in neural network implementation.

3.2.2 Testing and evaluation plan

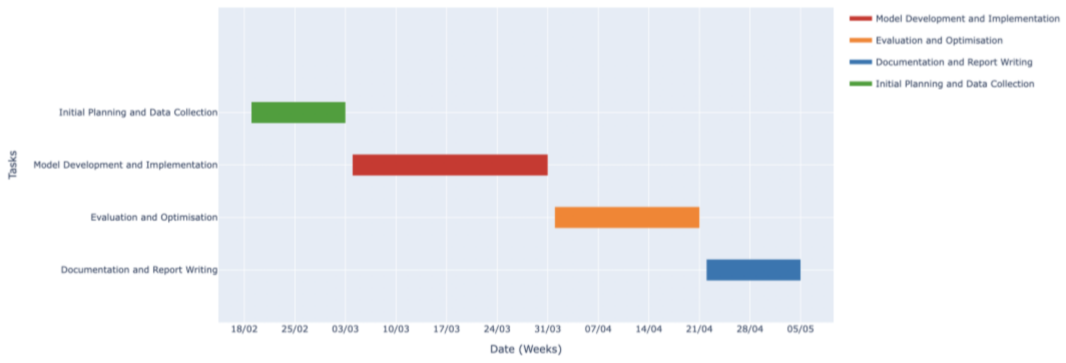
- Validate data pre-processing through quality checks.
- Utilise k-fold cross-validation and grid search for model optimisation.
- Evaluate models using RMSE, MAE, and R^2 score.
- Use visualisations to validate predictions.

Chapter 4 - Design & Methodology

4.1 Project management

The planning and time management of this project was done in the form of a Gantt chart (Figure 1) which shows each stage of the project and the projected time frame for each.

Figure 1: A Gantt chart representing the project plan.



The Gantt chart starts on the 18th of February 2024 rather than October/November 2023 (when this module began) because my project idea had to be changed. My initial project idea was to create a genetic optimisation algorithm to fit golf clubs to a player using their swing data such as clubhead speed. I was planning on getting around 10 participants, who play golf, to use their own golf clubs and take a set number of shots while I track their swing data from the launch monitor. After this I would insert their data into the algorithm and it would output optimal golf club variables such as shaft stiffness, shaft length, lie angle, grift thickness etc. however, after completing the project proposal, literature review and beginning to implement this algorithm, I found that I would need the participants to return and try new clubs with the new attributes multiple times for the genetic algorithm to work effectively. This did not fit within my time frame and therefore I had to adapt my project to my current study which, while still challenging, was more manageable for the allocated time.

From the Gantt chart we can see there are four main sections in my plan, initial planning and data collection, model development and data collection, evaluation and optimisation, and documentation and report. These main sections can be split into smaller goals for example from the 18th of February to the 25th of February the first step was researching background information and related studies to ensure there was a scientific gap for my study to sit. From the 25th of February to the 3rd of March is when I started to gather the PGA TOUR data. Each statistic from each tournament had to be downloaded separately meaning there were 23 statistics to download for each of the 8 tournaments in 2023 used. The 184 files then had to be combined, organised, and cleaned which took longer than expected but this still managed to stay within the predicted tome frame. The next step was the development process which initially started with building the models Random Forest, Gradient Boosting, and Linear regression. After implementing the models, I had to refine and optimise the models using and trying different techniques to ensure the best fit for my dataset. After this the Perceptron Neural Network was developed and refined. This whole development process was allocated 28 days from the 3rd of March to the 31st of March. The next part of the plan is the evaluation and optimisation stage which consists of implementing k-fold cross-validation, hyperparameter tuning, and comparing the models based on the evaluation metrics. This stage was planned to take from the 1st to the 21st of April. The final stage from the 22nd of April to the 5th of May was preparing and finalising the report. Fortunately, each stage of the project was completed within the allocated time which was unexpected but helped me maintain the workflow of the project.

Alongside following the plan set out by the Gantt chart, regular meetings with my supervisor, professor Xujiong Ye, took place over video call to talk about my project allowing me to ask questions and receive feedback that greatly helped in improving and refining my project.

4.2 Software development methodology

For this project, an Agile approach was chosen for the software development method because of its flexibility and iterative nature. Agile methodology is a good fit for machine learning projects as the evolving data patterns require a level of adaptability and room for iteration that agile provides (Grady, Payne and Parker, 2017).

The iterative style of Agile allowed for incremental progress of my study through sprints, each delivering a specific objective. Before each sprint, a planning process took place to define the goals of the sprint. These goals ranged from focusing on cleaning and organising the PGA TOUR data, to implementing a certain model effectively.

Using an Agile approach, I found that it allowed for proactive changes to my software which involved continuous improvement meaning that I didn't have to wait until every part of my code was implemented to go back and improve previous parts.

4.3 Software and Hardware environments

4.3.1 Software environments

The software environment for this project needed to facilitate efficient data processing, feature engineering, machine learning model development and evaluation. The program development was conducted using Jupyter Notebook which is a web-based interactive development environment that is aimed for data scientists (Jupyter, 2019). Jupyter Notebook allows for flexible and seamless integration with a range of Python libraries. It also facilitates iterative experimentation and data visualisation making it a perfect choice for my machine learning project.

For data storage, Microsoft Excel was used to hold the dataset from the initial dataset to the cleaned and organised version that was used for the final data analysis. Excel offered a convenient place to store and inspect the raw data before importing it into my program for further manipulation (Microsoft, 2023).

The programming language chosen for this project was Python 3 due to its large number of libraries tailored to data science and machine learning making it a perfect fit for this project (Python, 2019). The key libraries were managed through a Conda environment, which provided an isolated and reproducible environment that is also tailored for data science and machine learning (Conda, 2017).

The primary libraries used for this project are as follows; Pandas (v1.5.3) was used for data cleaning, manipulation, and analysis. It can seamlessly handle data using data structures called Data Frames (Pandas, 2018). NumPy (v1.23.3) was used for all numerical computing and operations done on arrays, providing a foundation for the data processing (Numpy, 2009). Scikit-Learn (v1.2.1) played a large role in implementing the initial machine learning models Random Forest, Gradient Boosting, and Linear Regression. This library has useful tools for model evaluation, feature selection, and parameter tuning that were crucial for the success of my project (Scikit-learn, 2019). TensorFlow (v2.12) was the library used for implementing and training the Perceptron Neural Network model. The library has a flexibility and scalability which made it suitable for my project (TensorFlow, 2019). Keras Tuner (v1.1.0) was used to implement the hyperparameter tuning of the neural network. It provided an easy-to-use implementation for optimising the model's performance (Team, n.d.). To create the visualisations used to see statistic relationships and model performances Matplotlib (v3.7.1) (Matplotlib, 2012) and Seaborn (v0.12.2) (seaborn, 2012) were the appropriate libraries for their ability to create scatter plots and correlation matrices.

4.3.2 Hardware environment

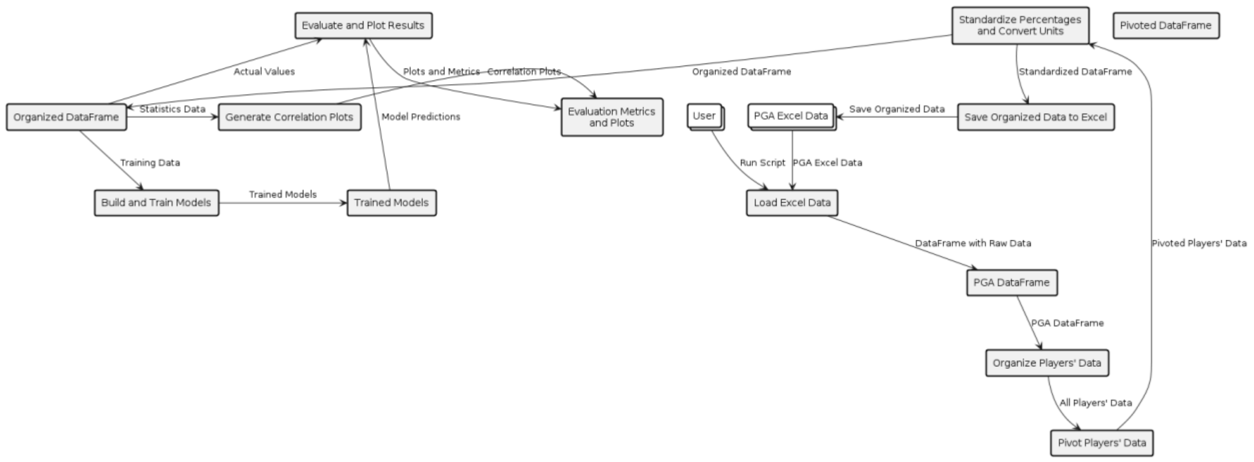
The project was developed on a 2020 MacBook Air was on M1 chip. Having an 8-core CPU, an 8-core GPU and 16GB of unified memory, this machine had ample computing power for data processing, training, and testing machine learning models. The combination of hardware and software environments provided a smooth development process enabling the experimentation and iterations seen in an Agile development methodology.

4.4 Design

4.4.1 Data flow

The data flow for this project follows a structured process that ensures efficient data handling, feature extraction, model training, and evaluation.

Figure 2: Data flow diagram.



The data flow starts with the initial PGA TOUR dataset extracted from the provided Excel file containing player statistics from eight 2023 PGA tournaments. This raw data is then loaded into a Pandas Data Frame providing a structured format for future manipulation. Following this, the Data Frame goes through the initial cleaning and formatting to ensure consistency in the dataset which is essential for accurate model predictions.

Once the raw Data frame is processed, the player statistics ate then extracted into a player-statistic structure where each row represents a unique player in a certain tournament and each column is a statistic. The data is then reorganised into a pivot table format which simplifies all the data for statistical analysis.

After the dataset is appropriately pre-processed, correlation plots are created in the form of scatter plots. This allows us to visualise the relationships that the statistics have between each other as well as between the finishing position of the player. Understanding relationships leads into the development and training of the models where the Random Forest, Gradient Boosting, Linear Regression, and Perceptron Neural Network are trained and tested on an 80/20 train-test split also utilising k-fold cross-validation.

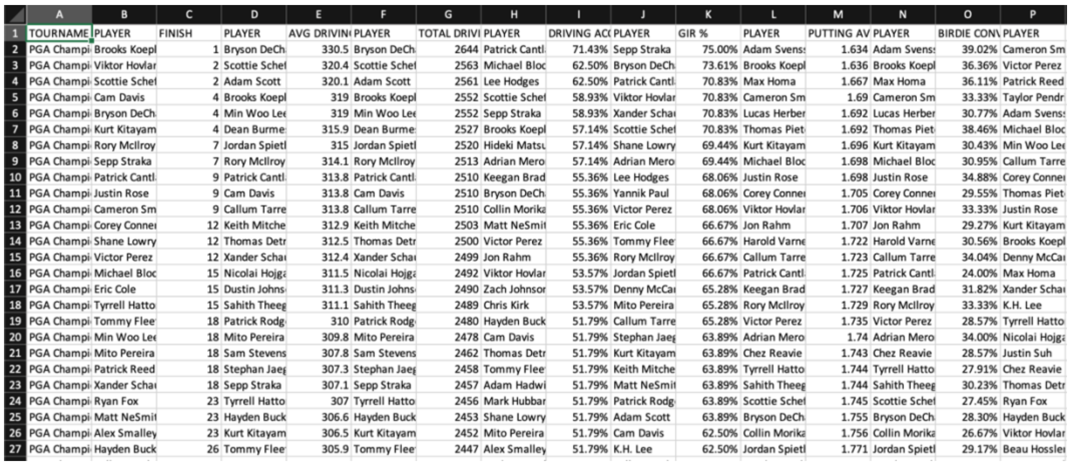
The final stage of the design of the software is the model evaluation where using the performance metrics Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R^2, the model's accuracy can be evaluated allowing for scatter plots to be created to compare the predicted vs actual outcome. This allows us to visualise the performance of each model and compare the models against each other.

4.5 Design choice justification

4.5.1 Data loading, organisation, and standardisation

The initial data loading and organisation process begins with importing the PGA TOUR dataset into a structured Pandas Data Frame. The purpose of this is to standardise the data so that it is in a consistent format. This ensures that each players statistics and tournament outcomes are properly defined. By using Pandas' versatile data manipulation capabilities, the process allows for variations in the dataset's structure. The Data Frame format facilitates efficient data manipulation, aggregation, and transformation which are essential for effective sports data analysis and therefore why it is essential to use this in my implementation. This first step ensures that all the data is in a unified format which is critical for the models to perform accurate predictions (Mckinney, 2010).

Figure 3: Initial unorganised dataset.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	TOURNAME	PLAYER	FINISH	PLAYER	AVG DRIVIN	PLAYER	TOTAL DRIVI	PLAYER	DRIVING AC	PLAYER	GIR %	PLAYER	PUTTING AV	PLAYER	BIRDIE CONV	PLAYER
2	PGA Champi	Brooks Koepl	1	Bryson DeCh	330.5	Bryson DeCh	2644	Patrick Cantl	71.43%	Sepp Straka	75.00%	Adam Svens	1.634	Adam Svens	39.02%	Cameron Sm
3	PGA Champi	Viktor Hovlar	2	Scottie Schef	320.4	Scottie Schef	2563	Michael Bloc	62.50%	Bryson DeCh	73.61%	Brooks Koepl	1.636	Brooks Koepl	36.36%	Victor Perez
4	PGA Champi	Scottie Schef	2	Adam Scott	320.1	Adam Scott	2561	Lee Hodges	62.50%	Patrick Cantl	70.83%	Max Homa	1.667	Max Homa	36.11%	Patrick Reed
5	PGA Champi	Cam Davis	4	Brooks Koepl	319	Brooks Koepl	2552	Scottie Schef	58.93%	Viktor Hovlar	70.83%	Cameron Sm	1.69	Cameron Sm	33.33%	Taylor Pendr
6	PGA Champi	Bryson DeCh	4	Min Woo Lee	319	Min Woo Lee	2552	Sepp Straka	58.93%	Xander Schau	70.83%	Lucas Herber	1.692	Lucas Herber	30.77%	Adam Svens
7	PGA Champi	Kurt Kitayam	4	Dean Burme	315.9	Dean Burme	2527	Brooks Koepl	57.14%	Scottie Schef	70.83%	Thomas Piet	1.692	Thomas Piet	38.46%	Michael Bloc
8	PGA Champi	Rory McIlroy	7	Jordan Spietl	315	Jordan Spietl	2520	Hideki Matsu	57.14%	Shane Lowry	69.44%	Kurt Kitayam	1.696	Kurt Kitayam	30.43%	Min Woo Lee
9	PGA Champi	Sepp Straka	7	Rory McIlroy	314.1	Rory McIlroy	2513	Adrian Mero	57.14%	Adrian Mero	69.44%	Michael Bloc	1.698	Michael Bloc	30.95%	Callum Tarre
10	PGA Champi	Patrick Cantl	9	Patrick Cantl	313.8	Patrick Cantl	2510	Keegan Brad	55.36%	Lee Hodges	68.06%	Justin Rose	1.698	Justin Rose	34.88%	Corey Conner
11	PGA Champi	Justin Rose	9	Cam Davis	313.8	Cam Davis	2510	Bryson DeCh	55.36%	Yannik Paul	68.06%	Corey Conner	1.705	Corey Conner	29.55%	Thomas Piet
12	PGA Champi	Cameron Sm	9	Callum Tarre	313.8	Callum Tarre	2510	Collin Morika	55.36%	Victor Perez	68.06%	Viktor Hovlar	1.706	Viktor Hovlar	33.33%	Justin Rose
13	PGA Champi	Corey Conner	12	Keith Mitche	312.9	Keith Mitche	2503	Matt NeSmit	55.36%	Eric Cole	66.67%	Jon Rahm	1.707	Jon Rahm	29.27%	Kurt Kitayam
14	PGA Champi	Shane Lowry	12	Thomas Detr	312.5	Thomas Detr	2500	Victor Perez	55.36%	Tommy Flee	66.67%	Harold Varne	1.722	Harold Varne	30.56%	Brooks Koepl
15	PGA Champi	Victor Perez	12	Xander Schau	312.4	Xander Schau	2499	Jon Rahm	55.36%	Rory McIlroy	66.67%	Callum Tarre	1.723	Callum Tarre	34.04%	Denny McCa
16	PGA Champi	Michael Bloc	15	Nicolai Hojga	311.5	Nicolai Hojga	2492	Viktor Hovlar	53.57%	Jordan Spietl	66.67%	Patrick Cantl	1.725	Patrick Cantl	24.00%	Max Homa
17	PGA Champi	Eric Cole	15	Dustin Johns	311.3	Dustin Johns	2490	Zach Johnsor	53.57%	Denny McCa	65.28%	Keegan Brad	1.727	Keegan Brad	31.82%	Xander Schau
18	PGA Champi	Tyrrell Hatto	15	Sahith Theeg	311.1	Sahith Theeg	2489	Chris Kirk	53.57%	Mito Pereira	65.28%	Rory McIlroy	1.729	Rory McIlroy	33.33%	K.H. Lee
19	PGA Champi	Tommy Flee	18	Patrick Rodg	310	Patrick Rodg	2480	Hayden Buck	51.79%	Callum Tarre	65.28%	Victor Perez	1.735	Victor Perez	28.57%	Tyrrell Hatto
20	PGA Champi	Min Woo Lee	18	Mito Pereira	309.8	Mito Pereira	2478	Cam Davis	51.79%	Stephan Jaeg	63.89%	Adrian Mero	1.74	Adrian Mero	34.00%	Nicolai Hojga
21	PGA Champi	Mito Pereira	18	Sam Stevens	307.8	Sam Stevens	2462	Thomas Detr	51.79%	Kurt Kitayam	63.89%	Chez Reavie	1.743	Chez Reavie	28.57%	Justin Suh
22	PGA Champi	Patrick Reed	18	Stephan Jaeg	307.3	Stephan Jaeg	2458	Tommy Flee	51.79%	Keith Mitche	63.89%	Tyrrell Hatto	1.744	Tyrrell Hatto	27.91%	Chez Reavie
23	PGA Champi	Xander Schau	18	Sepp Straka	307.1	Sepp Straka	2457	Adam Hadwi	51.79%	Matt NeSmit	63.89%	Sahith Theeg	1.744	Sahith Theeg	30.23%	Thomas Detr
24	PGA Champi	Ryan Fox	23	Tyrrell Hatto	307	Tyrrell Hatto	2456	Mark Hubbar	51.79%	Patrick Rodg	63.89%	Scottie Schef	1.745	Scottie Schef	27.45%	Ryan Fox
25	PGA Champi	Matt NeSmit	23	Hayden Buck	306.6	Hayden Buck	2453	Shane Lowry	51.79%	Adam Scott	63.89%	Bryson DeCh	1.755	Bryson DeCh	28.30%	Hayden Buck
26	PGA Champi	Alex Smalley	23	Kurt Kitayam	306.5	Kurt Kitayam	2452	Mito Pereira	51.79%	Cam Davis	62.50%	Collin Morika	1.756	Collin Morika	26.67%	Viktor Hovlar
27	PGA Champi	Hayden Buck	26	Tommy Flee	305.9	Tommy Flee	2447	Alex Smalley	51.79%	K.H. Lee	62.50%	Jordan Spietl	1.771	Jordan Spietl	29.17%	Beau Hossler

The initial dataset, as seen in Figure 3, is unorganised and hard for the models to predict from. The organisation of player's data involves reshaping the dataset so that each row aligns all of a player's statistics with their respective tournament and finish for that tournament. This is achieved by iterating through the Data Frame rows and collecting each player's performance data into a list, which will then be converted into a structured Data Frame. The primary reason for this step is to link each player's whole catalogue of statistics with the player's name and their tournament finish position. Organising the dataset like this ensures the models have a ground truth (Saltz and Shamshurin, 2019), in the finishing positions, while also standardising the data to allow for aggregation and analysis of relevant features (E.M.M. Alzeyani and Csaba Szabó, 2023). By normalising the dataset by tournament, it becomes possible to compare player's statistics across various events. Also, this structured data format allows for effective feature engineering in future.

Golf statistics often comes in various units or formats, for example inches and feet, which can affect a model's accuracy if these are not correctly standardised. In this study, some statistics will need to be standardised for example "AVG PROXIMITY TO HOLE" comes in feet and inches which the models will not be able to compute. To rectify this, we must convert this data to just inches for the models to understand. Standardising units like this makes sure consistent comparisons can be made across players and prevents models from potentially misinterpreting the significance of statistics because of different unit formats (Maszczyk et al., 2014). This step is crucial for identifying important patterns and relationships in the dataset.

4.5.2 Model Building and Training

Linear regression is a fundamental predictive model that establishes a linear relationship between features, which in this case are the statistics, and the target, finish positions. In golf performance predictions, linear regression is an excellent baseline model (Hastie, Tibshirani and Friedman, 2008) because of its simplicity and interpretability (James et al., 2013). The equation $y=\beta_0+\beta_1x_1+\beta_2x_2+...+\beta_nx_n$, where y represents the predicted finishing position and x_1, x_2,…, x_n are the player statistics, gives a clear insight into how each statistic impacts the model's accuracy. Despite struggling to capture non-linear relationships, linear regression is relevant to this study because of its ability to identify general trends and give an initial prediction.

Random Forest Regressor is an ensemble learning technique that builds multiple decision trees and combines their predictions for improved accuracy. Each tree is trained on a random subset of features, reducing the risk of overfitting. By averaging the predictions of many trees, the model will be able to handle complex non-linear relationships between player statistics and finishing positions. This model is a good fit for this study because of its robustness against overfitting and its ability to handle missing data, although our dataset should not have any missing data (Breiman, 2001).

Gradient Boosting Regressor (GBR) refines predictions iteratively by minimising the error of previous models. In each iteration, GBR adds a new tree that corrects errors made by prior models, leading to a more accurate overall prediction (Friedman, 2001). This model will be effective in my study because of the intricate non-linear relationship between golf statistics and player outcomes. For example, GBR can identify nuanced patterns such as the diminishing returns of long driving distances on scoring outcomes. By focusing on these errors and improving iteratively, GBR is capable of delivering very accurate predictions.

Perceptron Neural Networks (PNN) use multiple layers of interconnected nodes to learn complex patterns in the data. In this project, the network will comprise of two hidden layers, each followed by batch normalisation and dropout layers to prevent overfitting and improve generalisation (Ioffe, 2015). This format allows the model to learn intricate non-linear relationships, capturing how multiple statistics interact to predict a player's finishing position (LeCun, Bengio and Hinton, 2015). By optimising the model using hyperparameter optimisation, PNN can produce highly accurate predictions. PNN is relevant to this study because it handles high-dimensional data well, identifying complex interactions that linear and tree-based models may miss.

4.5.3 Evaluation Metrics

Root Mean Squared Error (RMSE) is a commonly used metric for regression models that measures the average magnitude of the prediction errors. This provides a good insight into the model's accuracy. RMSE calculates the square root of the mean of the squared differences between the predicted values and the actual outcomes (Chai and Draxler, 2014).

Figure 4: Root Mean Squared Error formula.

$$RMSE = \sqrt{\frac{\Sigma(y_i - \hat{y}_i)^2}{N - P}}$$

In figure 4, $\hat{y}_i$ represents the predicted value, and y_i represents the actual value. RMSE is particularly useful in this study because it punishes larger errors more heavily, which is crucial in evaluating golf performance predictions where significant deviations can largely affect decisions.

Mean Absolute Error (MAE) measures the average absolute differences between predicted and actual values. Compared to RMSE, which squares the errors, MAE treats all deviations equally, giving a direct indication of the prediction accuracy.

Figure 5: Mean Absolute Error formula.a

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N} |y_i - \hat{y}_i|$$

In figure 5, $\hat{y}_i$ represents the predicted value, and y_i represents the actual value. MAE is useful for this study for understanding the model's consistency in predicting player finishes by evaluating the average prediction error in plain units.

This metric is less sensitive, than RMSE, to outliers giving a balanced view on prediction errors (Chai and Draxler, 2014).

R-Squared (R^2), also known as the coefficient of determination, measures the proportion of variance in the dependent variable that is predictable from the independent variables.

Figure 6: R-Squared formula.

$$R^2 = 1 - \frac{\sum (y_i - \hat{y}_i)^2}{\sum (y_i - \bar{y}_i)^2}$$

Where $\bar{y}$ represents the mean of the actual values. An R^2 score of 1 indicates perfect predictions, whereas a score of 0 indicates the model does not explain any variance. This metric is useful in this study as it assesses how well the models explain the variance in the finishing positions, providing an insight into the overall explanatory power of the chosen predictive features (James et al., 2013). R^2 is also useful for comparing models to determine which one best captures the relationship between, in this case, player statistics and finish positions (Chai and Draxler, 2014).

3.5.4 Grid Search

Grid Search is a hyperparameter tuning technique that is used to optimise the performance of machine learning models by exhaustively searching for the best combination of hyperparameters. A predefined set of hyperparameter values are arranged in a grid-like structure, and each combination is evaluated through cross-validation. The best performing model configuration is then selected based on a chosen evaluation metric such as RMSE. For this study, grid search is important for optimising the machine learning models to ensure the most suitable parameters are used to maximise model performance (Bergstra, Ca and Ca, 2012).

3.5.5 K-Fold Cross-Validation

K-fold cross-validation is a robust model evaluation technique that involves partitioning the dataset into K equally sized folds. The model is trained on K – 1 folds and tested on the remaining fold, repeating the process until every fold has been used for testing. The results are averaged to provide a more reliable estimate of model performance. By reducing overfitting and providing a more accurate estimate of the models' predictive power, K-fold cross-validation is pivotal in assessing the robustness of the models. K-fold allows for a comprehensive evaluation across different data partitions, making sure that the predictive models deliver consistent and reliable results (Fushiki, 2009).

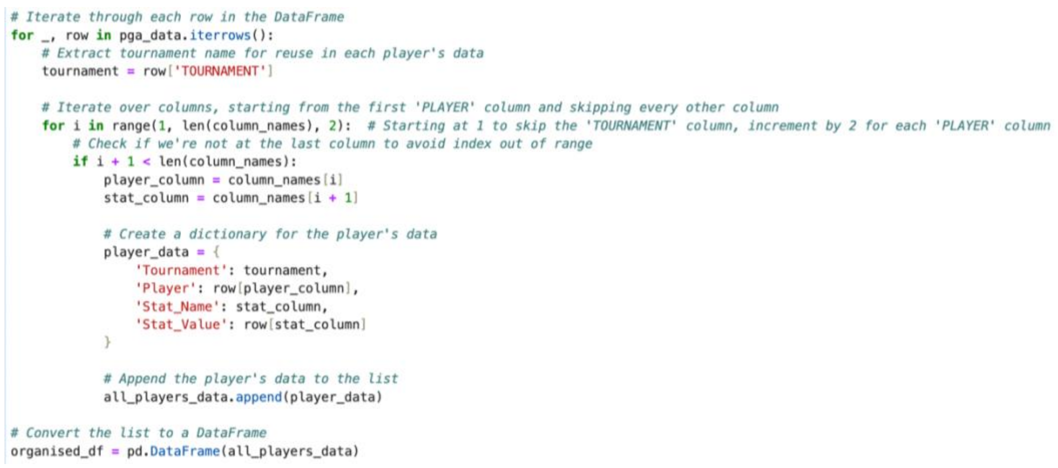
Chapter 5 – Implementation

5.1 Data Pre-processing

5.1.1 Data Organisation

Before any regression models can be trained, the initial dataset (Figure 3) must be organised in order for the features and target to be identified by the models. The initial structure of the dataset is where each row does not show a player's whole catalogue of statistics for that tournament but instead each statistic has the corresponding player associated with that stat in the column to the left of the stat.

Figure 7: Data formatting code.



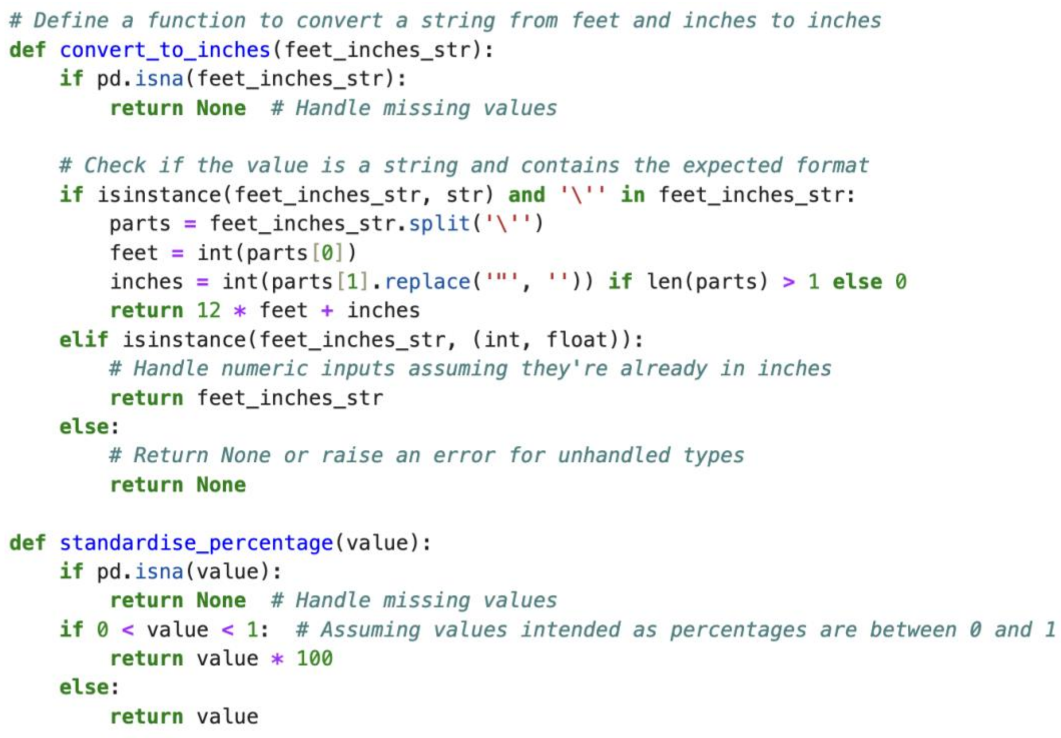
```python
# Iterate through each row in the DataFrame
for _, row in pga_data.iterrows():
    # Extract tournament name for reuse in each player's data
    tournament = row['TOURNAMENT']

    # Iterate over columns, starting from the first 'PLAYER' column and skipping every other column
    for i in range(1, len(column_names), 2):  # Starting at 1 to skip the 'TOURNAMENT' column, increment by 2 for each 'PLAYER' column
        # Check if we're not at the last column to avoid index out of range
        if i + 1 < len(column_names):
            player_column = column_names[i]
            stat_column = column_names[i + 1]

            # Create a dictionary for the player's data
            player_data = {
                'Tournament': tournament,
                'Player': row[player_column],
                'Stat_Name': stat_column,
                'Stat_Value': row[stat_column]
            }

            # Append the player's data to the list
            all_players_data.append(player_data)

# Convert the list to a DataFrame
organised_df = pd.DataFrame(all_players_data)
```

In Figure 7 we can see that the program iterates over each row of the Data Frame and takes out the tournament so it can be added to the dictionary containing all of a player’s data. Having a reference to which tournament the stats are for is crucial as many of the players have statistics in every tournament, so this is needed to differentiate between them and treat them as different players essentially. After this, the program iterates over each column starting from the first “PLAYER” column and then skipping every other column to only retrieve the “PLAYER” column for each stat. Each time it does this “i +1” checks if the next column is within range to avoid any indexing errors. The program then takes the player’s name from the column and also the corresponding stat in the column directly after. In the same loop, the player’s data is then all stored in a dictionary which includes the tournament name, player name, the name of the stat, as well as the actual stat. This is then appended to the “all_players_data” list. This is repeated until all data is mapped to the corresponding player. Once the loop ends, a new Data Frame “organised_df” is created with all player data correctly mapped and organised ready to be exported to a new and final Excel file.

5.1.2 Data Standardisation

After the dataset was correctly formatted so that each row represented a player’s whole set if stats for that tournament, the next step was standardising the data as some of the stats in the dataset had either different ways of displaying the same thing, for example in "SAND SAVE %" some of the percentages would be in the format "50.45" and other would be in the format "0.5045" both showing the same stat but in different ways. Another thing that needed to be standardised was some of the putting stats such as "AVG PROXIMITY TO HOLE" which was in feet and inches but in order for the models to understand this, the stats must be converted to just inches.

Figure 8: Statistic standardisation functions.



```
# Define a function to convert a string from feet and inches to inches
def convert_to_inches(feet_inches_str):
    if pd.isna(feet_inches_str):
        return None  # Handle missing values

    # Check if the value is a string and contains the expected format
    if isinstance(feet_inches_str, str) and '\'' in feet_inches_str:
        parts = feet_inches_str.split('\'')
        feet = int(parts[0])
        inches = int(parts[1].replace('"', '')) if len(parts) > 1 else 0
        return 12 * feet + inches
    elif isinstance(feet_inches_str, (int, float)):
        # Handle numeric inputs assuming they're already in inches
        return feet_inches_str
    else:
        # Return None or raise an error for unhandled types
        return None

def standardise_percentage(value):
    if pd.isna(value):
        return None  # Handle missing values
    if 0 < value < 1:  # Assuming values intended as percentages are between 0 and 1
        return value * 100
    else:
        return value
```

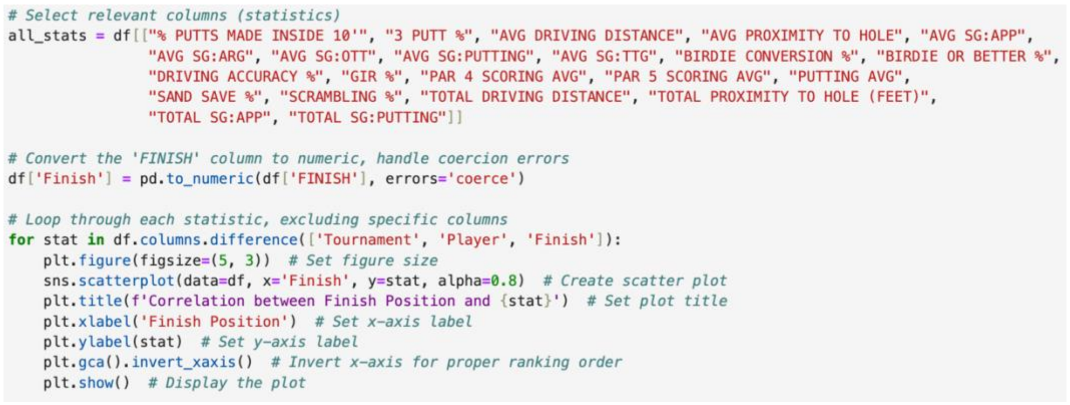
In Figure 8, we can see the two functions used for data standardisation. In the first function "convert_to_inches", we can see that the program first checks for any missing values in the data, which in our dataset there isn't. The program the proceeds to check if the input data is a string and check for the " ' " symbol which implies that the data is in feet and inches. The string is split from this symbol separating the feet and inches, the feet are then multiplied by twelve to convert it to inches, and the added back to the initial inches to give the final converted data. If the input data is already in inches, it is returned skipping the conversion phase.

The next function is used to standardise the percentages. It starts by handling any missing values which again is just a formality as there are no missing values in the dataset used. Input data is then checked if it is between 0 and 1, which would indicate that the stat is fractionalised, and if so, it is multiplied by 100 to convert it to a percentage. Finally, if the value is greater than 1, it is returned as it is already in the correct format.

5.2 Feature Correlation Visualisation

In order to get an initial look at the relationships between different stats and player finish positions, scatter plot graphs were implemented for easy visualisation of these relationships.

Figure 9: Relationship visualisation implementation.



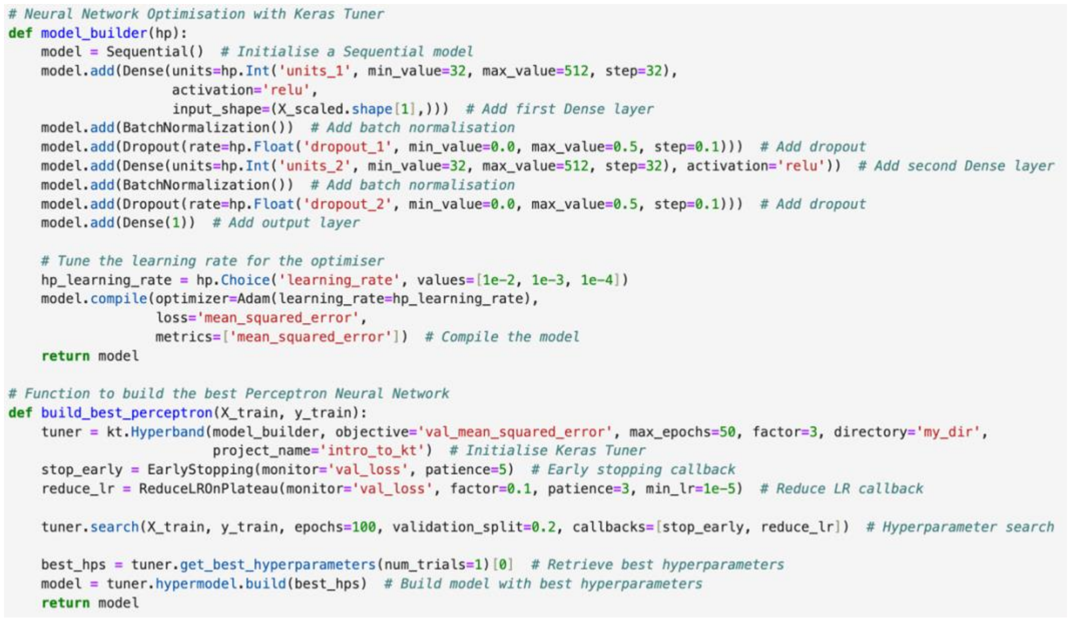
```python
# Select relevant columns (statistics)
all_stats = df[["% PUTTS MADE INSIDE 10'", "3 PUTT %", "AVG DRIVING DISTANCE", "AVG PROXIMITY TO HOLE", "AVG SG:APP",
                "AVG SG:ARG", "AVG SG:OTT", "AVG SG:PUTTING", "AVG SG:TTG", "BIRDIE CONVERSION %", "BIRDIE OR BETTER %",
                "DRIVING ACCURACY %", "GIR %", "PAR 4 SCORING AVG", "PAR 5 SCORING AVG", "PUTTING AVG",
                "SAND SAVE %", "SCRAMBLING %", "TOTAL DRIVING DISTANCE", "TOTAL PROXIMITY TO HOLE (FEET)",
                "TOTAL SG:APP", "TOTAL SG:PUTTING"]]

# Convert the 'FINISH' column to numeric, handle coercion errors
df['Finish'] = pd.to_numeric(df['FINISH'], errors='coerce')

# Loop through each statistic, excluding specific columns
for stat in df.columns.difference(['Tournament', 'Player', 'Finish']):
    plt.figure(figsize=(5, 3))  # Set figure size
    sns.scatterplot(data=df, x='Finish', y=stat, alpha=0.8)  # Create scatter plot
    plt.title(f'Correlation between Finish Position and {stat}')  # Set plot title
    plt.xlabel('Finish Position')  # Set x-axis label
    plt.ylabel(stat)  # Set y-axis label
    plt.gca().invert_xaxis()  # Invert x-axis for proper ranking order
    plt.show()  # Display the plot
```

In Figure 9, we can see that all of the statistic are taken from the Data Frame and put into a variable “all_stats”. From here, the “FINISH” stat, containing all of the player outcomes, is converted to a numeric type. This ensures that errors are coerced to “NaN” values and non-numeric data is excluded. A loop then iterates through each statistic column except “Tournament”, “Player”, and “Finish” ensuring only the statistics are used. Each stat is then plotted against the “Finish” column to visualise the relationship. “sns.scatterplot” is used to generate the plots with “Finish” on the x-axis and each statistic, on different individual scatter plots, is on the y-axis. The “alpha” parameter is used to ensure overlapping plots are still visible and the x-axis is inverted using “plt.gca().invert_xaxis()” to ensure the lower finishing positions (better ranks) are on the left.

5.3 Model Building and Evaluation

5.3.1 Model Building

Figure 10: Perceptron Neural Network Build

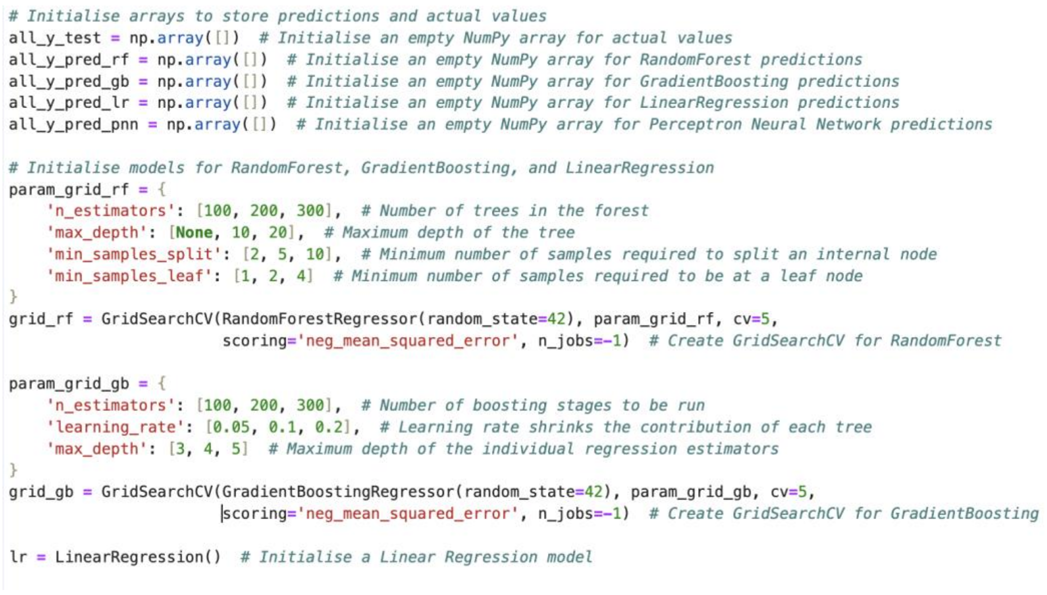


```python
# Neural Network Optimisation with Keras Tuner
def model_builder(hp):
    model = Sequential()  # Initialise a Sequential model
    model.add(Dense(units=hp.Int('units_1', min_value=32, max_value=512, step=32),
                    activation='relu',
                    input_shape=(X_scaled.shape[1],)))  # Add first Dense layer
    model.add(BatchNormalization())  # Add batch normalisation
    model.add(Dropout(rate=hp.Float('dropout_1', min_value=0.0, max_value=0.5, step=0.1)))  # Add dropout
    model.add(Dense(units=hp.Int('units_2', min_value=32, max_value=512, step=32), activation='relu'))  # Add second Dense layer
    model.add(BatchNormalization())  # Add batch normalisation
    model.add(Dropout(rate=hp.Float('dropout_2', min_value=0.0, max_value=0.5, step=0.1)))  # Add dropout
    model.add(Dense(1))  # Add output layer

    # Tune the learning rate for the optimiser
    hp_learning_rate = hp.Choice('learning_rate', values=[1e-2, 1e-3, 1e-4])
    model.compile(optimizer=Adam(learning_rate=hp_learning_rate),
                  loss='mean_squared_error',
                  metrics=['mean_squared_error'])  # Compile the model
    return model

# Function to build the best Perceptron Neural Network
def build_best_perceptron(X_train, y_train):
    tuner = kt.Hyperband(model_builder, objective='val_mean_squared_error', max_epochs=50, factor=3, directory='my_dir',
                         project_name='intro_to_kt')  # Initialise Keras Tuner
    stop_early = EarlyStopping(monitor='val_loss', patience=5)  # Early stopping callback
    reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.1, patience=3, min_lr=1e-5)  # Reduce LR callback

    tuner.search(X_train, y_train, epochs=100, validation_split=0.2, callbacks=[stop_early, reduce_lr])  # Hyperparameter search

    best_hps = tuner.get_best_hyperparameters(num_trials=1)[0]  # Retrieve best hyperparameters
    model = tuner.hypermodel.build(best_hps)  # Build model with best hyperparameters
    return model
```

In Figure 10, the “model_builder” function defines the architecture of the neural network and the hyperparameters to be optimised. A sequential model is initialised using “Sequential()” to build a linear stack of layers. Two fully connected dense layers are added with units that can be tuned ranging from 32 to 512 in steps of 32. The first dense layer is connected to the input features using “input_shape”, and both layers are activated using ReLU for non-linear decision boundaries. Batch normalisation layers are added after each dense layer to normalise activations and speed up the training process. Also, dropout layers, with a dropout rate that can be tuned from 0.0 - 0.5, are included to regularise the model and prevent overfitting. The output layer consists of a single node without an activation function to provide continuous predictions for regression. The learing rate of the Adam optimiser is also tuned using three different values: “1e-2”, “1e-3”, “1e-4”.

The “build_best_perceptron” function uses Keras Tuner’s Hyperband method for hyperparameter optimisation. “kt.Hyperband” is employed with the “model_builder” function to search for the best combination of hyperparameters, aiming to minimise validation mean squared error (“val_mean_squared_error”). The tuner is configured with “max_epochs” set to 50 and a “factor” of 3 to reduce the training time. Callbacks such as early stopping and learning rate reduction are used to prevent overfitting and to ensure the training is efficient. Early stopping stops training if the validation loss does not improve for five epochs, while “ReduceLROnPlateau” reduces the learning rate by a factor of 0.1 if the validation loss plateaus for three epochs. Both of these helps to reduce the computational strain of the program. The search process is initiated using “tuner.search”, which trains the models over 100 epochs with a 20% validation split. The search returns the best hyperparameters using “get_best_hyperparameters”, and the best model is then build using these hyperparameters.

Figure 11: Grid Search set up and model initialisation.



```
# Initialise arrays to store predictions and actual values
all_y_test = np.array([])  # Initialise an empty NumPy array for actual values
all_y_pred_rf = np.array([])  # Initialise an empty NumPy array for RandomForest predictions
all_y_pred_gb = np.array([])  # Initialise an empty NumPy array for GradientBoosting predictions
all_y_pred_lr = np.array([])  # Initialise an empty NumPy array for LinearRegression predictions
all_y_pred_pnn = np.array([])  # Initialise an empty NumPy array for Perceptron Neural Network predictions

# Initialise models for RandomForest, GradientBoosting, and LinearRegression
param_grid_rf = {
    'n_estimators': [100, 200, 300],  # Number of trees in the forest
    'max_depth': [None, 10, 20],  # Maximum depth of the tree
    'min_samples_split': [2, 5, 10],  # Minimum number of samples required to split an internal node
    'min_samples_leaf': [1, 2, 4]  # Minimum number of samples required to be at a leaf node
}
grid_rf = GridSearchCV(RandomForestRegressor(random_state=42), param_grid_rf, cv=5,
                       scoring='neg_mean_squared_error', n_jobs=-1)  # Create GridSearchCV for RandomForest

param_grid_gb = {
    'n_estimators': [100, 200, 300],  # Number of boosting stages to be run
    'learning_rate': [0.05, 0.1, 0.2],  # Learning rate shrinks the contribution of each tree
    'max_depth': [3, 4, 5]  # Maximum depth of the individual regression estimators
}
grid_gb = GridSearchCV(GradientBoostingRegressor(random_state=42), param_grid_gb, cv=5,
                       scoring='neg_mean_squared_error', n_jobs=-1)  # Create GridSearchCV for GradientBoosting

lr = LinearRegression()  # Initialise a Linear Regression model
```

In Figure 11, we can see that empty arrays are initialised to hold the predictions and actual values for each model. The arrays are created to store the true target values alongside predictions from Random Forest, Gradient Boosting, Linear Regression, and the Perceptron Neural Network models.

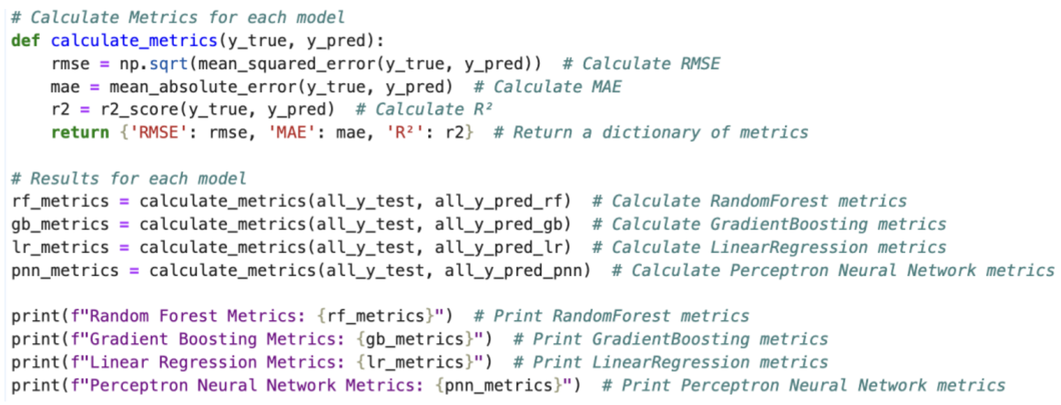
The preparation of the Random Forest and Gradient Boosting models involved specifying a set of hyperparameters that will be optimised using GridSearchCV. For Random Forest, the “param_grid_rf” dictionary defines parameters such as “n_estimators” (number of trees), “max_depth” (maximum depth of the tree), “min_samples_split” (minimum samples required to split a node), and “min_samples_leaf” (minimum samples required to be at a leaf node). these parameters control the complexity and variance of the model, directly influencing the model’s predictive power. The Random Forest model is instantiated using “RandomForestRegressor()” with a fixed random seed so the model’s predictions are reproducible.

Similarly, the Gradient Boosting model is also prepared with a parameter grid this time defined as “param_grid_gb”. The parameters in this grid include "n_estimators" (number of boosting stages), "learning_rate" (contribution of each tree), and "max_depth" (maximum depth of regression estimators). These parameters control the model's ability to capture complex interactions between features. Gradient Boosting also uses GridSearchCV to optimise these hyperparameters using "neg_mean_squared_error" as the scoring metric. The Gradient Boosting model is instantiated using "GradientBoostingRegressor()" also with a fixed random seed for reproducibility.

As well as these ensemble models, a simple Linear Regression model is used for a baseline comparison. This model is simply implemented using "LinearRegression()". This model offers a simple and effective benchmark for evaluating more complex models.

5.3.2 Model Evaluation

Figure 12: Model evaluations.

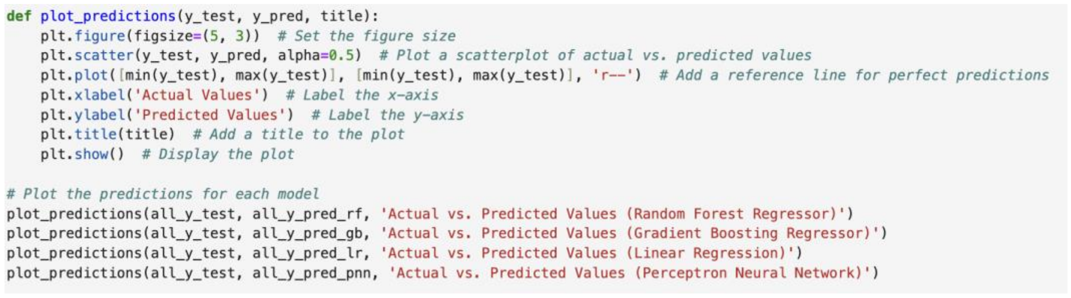


```
# Calculate Metrics for each model
def calculate_metrics(y_true, y_pred):
    rmse = np.sqrt(mean_squared_error(y_true, y_pred))  # Calculate RMSE
    mae = mean_absolute_error(y_true, y_pred)  # Calculate MAE
    r2 = r2_score(y_true, y_pred)  # Calculate R²
    return {'RMSE': rmse, 'MAE': mae, 'R²': r2}  # Return a dictionary of metrics

# Results for each model
rf_metrics = calculate_metrics(all_y_test, all_y_pred_rf)  # Calculate RandomForest metrics
gb_metrics = calculate_metrics(all_y_test, all_y_pred_gb)  # Calculate GradientBoosting metrics
lr_metrics = calculate_metrics(all_y_test, all_y_pred_lr)  # Calculate LinearRegression metrics
pnn_metrics = calculate_metrics(all_y_test, all_y_pred_pnn)  # Calculate Perceptron Neural Network metrics

print(f"Random Forest Metrics: {rf_metrics}")  # Print RandomForest metrics
print(f"Gradient Boosting Metrics: {gb_metrics}")  # Print GradientBoosting metrics
print(f"Linear Regression Metrics: {lr_metrics}")  # Print LinearRegression metrics
print(f"Perceptron Neural Network Metrics: {pnn_metrics}")  # Print Perceptron Neural Network metrics
```

In Figure 12, we can see that the "calculate_metrics" function calculates the key metrics being used: Root Mean Squared Error, Mean Absolute Error, and the Coefficient of Determination (R^2). It takes the true outcomes "y_true", and the predicted outcomes "y_pred" to calculate each of the metrics then returns a dictionary of the metrics.

Each of the model’s performance is then evaluated using the “calculate_metrics” function. The metrics are printed to the console to make it easy to compare the results of each model. Each metric allows an insight into the strengths and weaknesses of the predictive power of each model.

Figure 13: Model evaluation visualisation.



```python
def plot_predictions(y_test, y_pred, title):
    plt.figure(figsize=(5, 3))  # Set the figure size
    plt.scatter(y_test, y_pred, alpha=0.5)  # Plot a scatterplot of actual vs. predicted values
    plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], 'r--')  # Add a reference line for perfect predictions
    plt.xlabel('Actual Values')  # Label the x-axis
    plt.ylabel('Predicted Values')  # Label the y-axis
    plt.title(title)  # Add a title to the plot
    plt.show()  # Display the plot

# Plot the predictions for each model
plot_predictions(all_y_test, all_y_pred_rf, 'Actual vs. Predicted Values (Random Forest Regressor)')
plot_predictions(all_y_test, all_y_pred_gb, 'Actual vs. Predicted Values (Gradient Boosting Regressor)')
plot_predictions(all_y_test, all_y_pred_lr, 'Actual vs. Predicted Values (Linear Regression)')
plot_predictions(all_y_test, all_y_pred_pnn, 'Actual vs. Predicted Values (Perceptron Neural Network)')
```

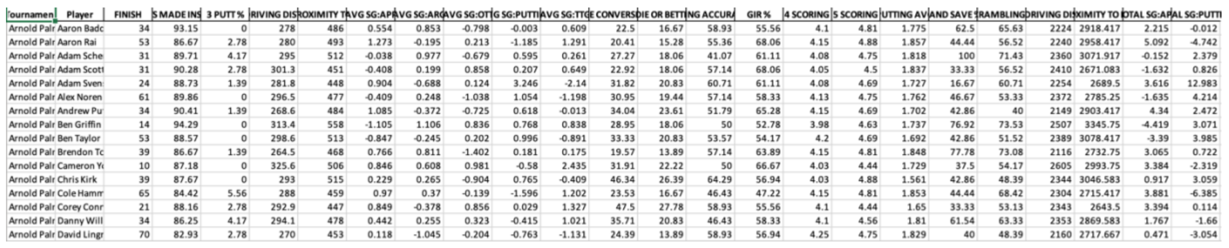
In Figure 13, we can see the “plot_predictions” function takes in “y_test”, which are the actual values, and “y_pred” which are the predicted values, as well as “title” which is the title of the visualisation for each model. The function sets out the dimensions of the graph using “figsize” and then plots a scatter plot of actual vs predicted values for each model. A reference line is added to the plot for what a perfect prediction would look like, this helps the user to understand what they are looking for and to assess the predictive power of the models. Finally, the “plot_predictions” function is called for each model providing an insightful scatter plot for each of the model’s predictions.

Chapter 6 - Results & Discussion

At the beginning of this study, a set of aims and objectives were defined in order to guide the study development. The aim was to utilise advanced machine learning techniques to develop a robust model that predicts individual player finishes in PGA TOUR events using historical data.

6.1 Data Pre-processing Results

Figure 14: Formatted, cleaned, and standardised dataset.



ournamen	Player	FINISH	S MADE INS	3 PUTT %	RIVING DIS	ROXIMITY T	AVG SG:AP	AVG SG:ARG	AVG SG:OT	G SG:PUTTI	AVG SG:TTG	E CONVERS	DIE OR BETT	NG ACCURA	GIR %	4 SCORING	5 SCORING	UTTING AV	AND SAVE	RAMBLING	DRIVING DIS	XIMITY TO	OTAL SG:AP	AL SG:PUTTI
Arnold Palr	Aaron Badc	34	93.15	0	278	486	0.554	0.853	-0.798	-0.003	0.609	22.5	16.67	58.93	55.56	4.1	4.81	1.775	62.5	65.63	2224	2918.417	2.215	-0.012
Arnold Palr	Aaron Rai	53	86.67	2.78	280	493	1.273	-0.195	0.213	-1.185	1.291	20.41	15.28	55.36	68.06	4.15	4.88	1.857	44.44	56.52	2240	2958.417	5.092	-4.742
Arnold Palr	Adam Sche	31	89.71	4.17	295	512	-0.038	0.977	-0.679	0.595	0.261	27.27	18.06	41.07	61.11	4.08	4.75	1.818	100	71.43	2360	3071.917	-0.152	2.379
Arnold Palr	Adam Scott	31	90.28	2.78	301.3	451	-0.408	0.199	0.858	0.207	0.649	22.92	18.06	57.14	68.06	4.05	4.5	1.837	33.33	56.52	2410	2671.083	-1.632	0.826
Arnold Palr	Adam Sven	24	88.73	1.39	281.8	448	0.904	-0.688	0.124	3.246	-2.14	31.82	20.83	60.71	61.11	4.08	4.69	1.727	16.67	60.71	2254	2689.5	3.616	12.983
Arnold Palr	Alex Noren	61	89.86	0	296.5	477	-0.409	0.248	-1.038	1.054	-1.198	30.95	19.44	57.14	58.33	4.13	4.75	1.762	46.67	53.33	2372	2785.25	-1.635	4.214
Arnold Palr	Andrew Pu	34	90.41	1.39	268.6	484	1.085	-0.372	-0.725	0.618	-0.013	34.04	23.61	51.79	65.28	4.15	4.69	1.702	42.86	40	2149	2903.417	4.34	2.472
Arnold Palr	Ben Griffin	14	94.29	0	313.4	558	-1.105	1.106	0.836	0.768	0.838	28.95	18.06	50	52.78	3.98	4.63	1.737	76.92	73.53	2507	3345.75	-4.419	3.071
Arnold Palr	Ben Taylor	53	88.57	0	298.6	513	-0.847	-0.245	0.202	0.996	-0.891	33.33	20.83	53.57	54.17	4.2	4.69	1.692	42.86	51.52	2389	3078.417	-3.39	3.985
Arnold Palr	Brendon Tc	39	86.67	1.39	264.5	468	0.766	0.811	-1.402	0.181	0.175	19.57	13.89	57.14	63.89	4.15	4.81	1.848	77.78	73.08	2116	2732.75	3.065	0.722
Arnold Palr	Cameron Y	10	87.18	0	325.6	506	0.846	0.608	0.981	-0.58	2.435	31.91	22.22	50	66.67	4.03	4.44	1.729	37.5	54.17	2605	2993.75	3.384	-2.319
Arnold Palr	Chris Kirk	39	87.67	0	293	515	0.229	0.265	-0.904	0.765	-0.409	46.34	26.39	64.29	56.94	4.03	4.88	1.561	42.86	48.39	2344	3046.583	0.917	3.059
Arnold Palr	Cole Hamm	65	84.42	5.56	288	459	0.97	0.37	-0.139	-1.596	1.202	23.53	16.67	46.43	47.22	4.15	4.81	1.853	44.44	68.42	2304	2715.417	3.881	-6.385
Arnold Palr	Corey Conr	21	88.16	2.78	292.9	447	0.849	-0.378	0.856	0.029	1.327	47.5	27.78	58.93	55.56	4.1	4.44	1.65	33.33	53.13	2343	2643.5	3.394	0.114
Arnold Palr	Danny Will	34	86.25	4.17	294.1	478	0.442	0.255	0.323	-0.415	1.021	35.71	20.83	46.43	58.33	4.1	4.56	1.81	61.54	63.33	2353	2869.583	1.767	-1.66
Arnold Palr	David Lingr	70	82.93	2.78	270	453	0.118	-1.045	-0.204	-0.763	-1.131	24.39	13.89	58.93	56.94	4.25	4.75	1.829	40	48.39	2160	2717.667	0.471	-3.054

In Figure 14, we can see a small section of the dataset results after pre-processing. The figure shows that each columns represents either the tournament name, the player’s name, where the player finished in that tournament, and every statistic that the models are trained on. We can also see that each row correctly depicts all the stats for a certain player and all of the stats are standardised with no variations in how the data is formatted. This shows that our dataset pre-processing code is successful in cleaning, standardising, and organising the initial dataset we had.

6.2 Statistic Relationship Visualisation

Figure 15: Correlation between finish position and % of putts made inside 10'.

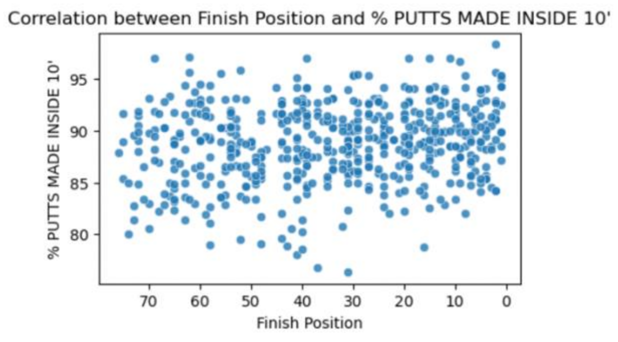


Figure 16: Correlation between finish position and 3 putt %.

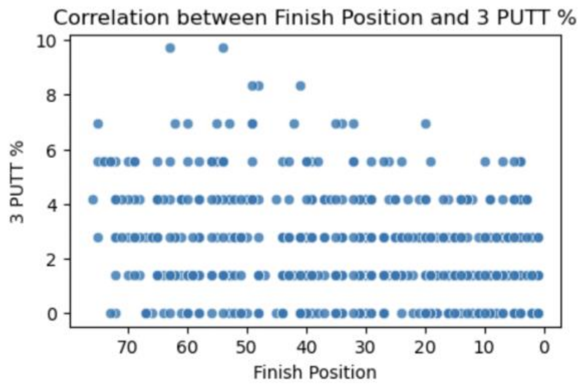


Figure 17: Correlation between finish position and average driving distance.

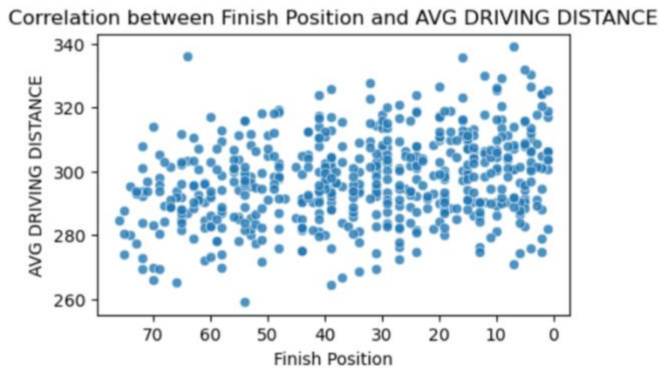


Figure 18: Correlation between finish position and average proximity to the hole.

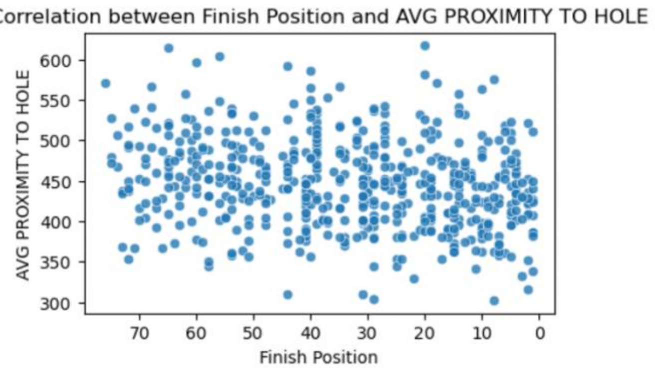


Figure 19: Correlation between finish position and average strokes gained: approach to the green.

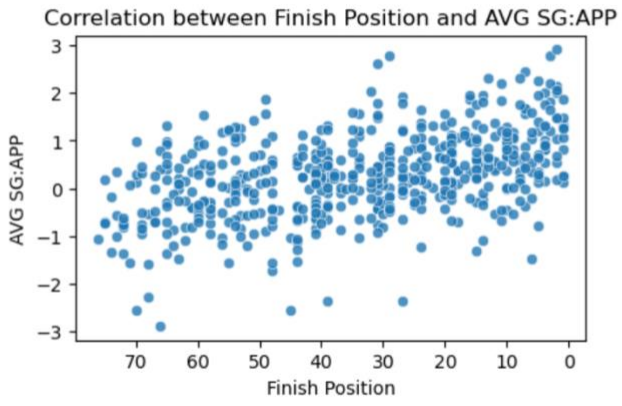


Figure 20: Correlation between finish position and average strokes gained: around the green.

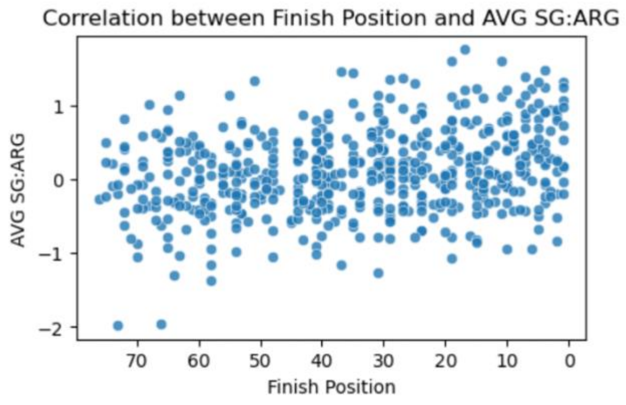


Figure 21: Correlation between finish position and average strokes gained: off the tee.

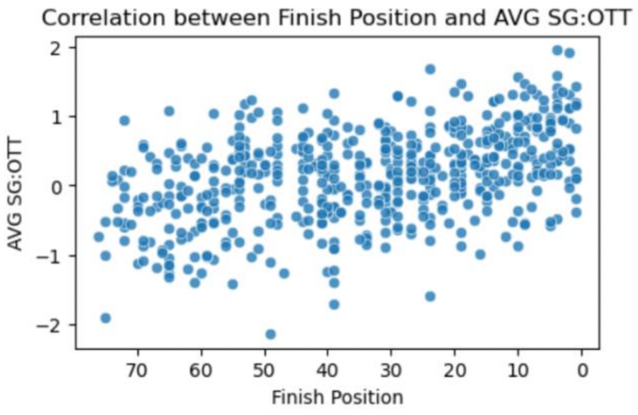


Figure 22: Correlation between finish position and average strokes gained: putting.

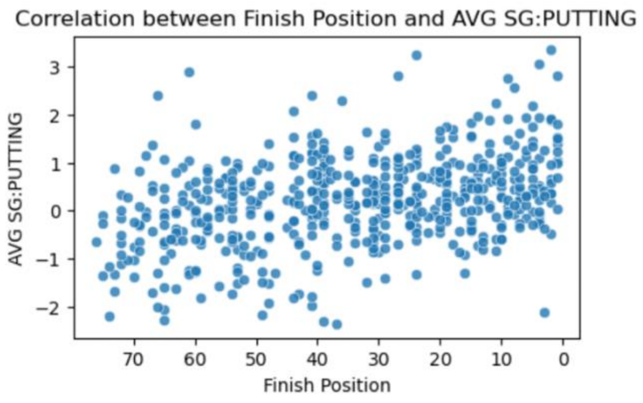


Figure 23: Correlation between finish position and average strokes gained: tee to green.

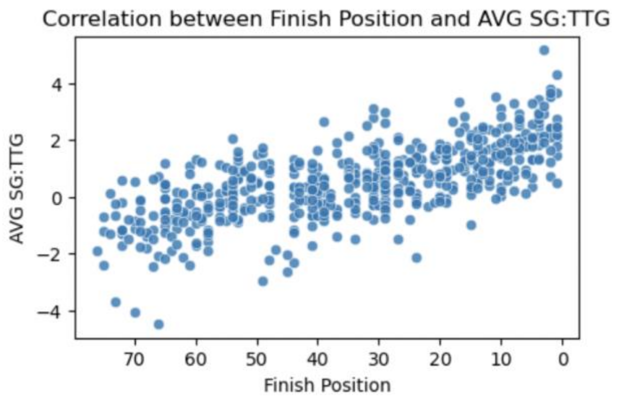


Figure 24: Correlation between finish position and birdie conversion %.

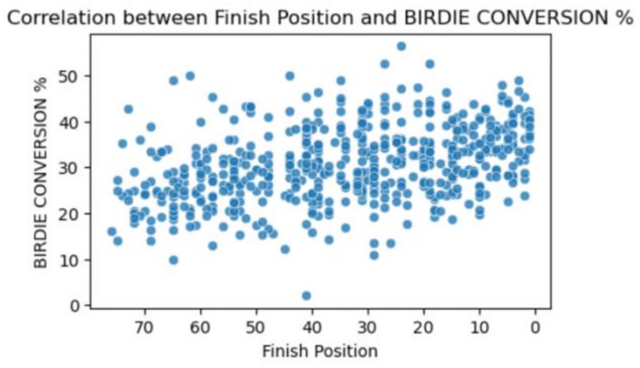


Figure 25: Correlation between finish position and birdie or better %.

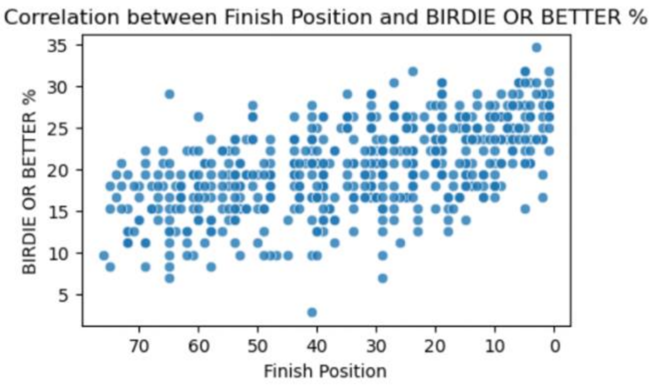


Figure 26: Correlation between finish position and driving accuracy %.

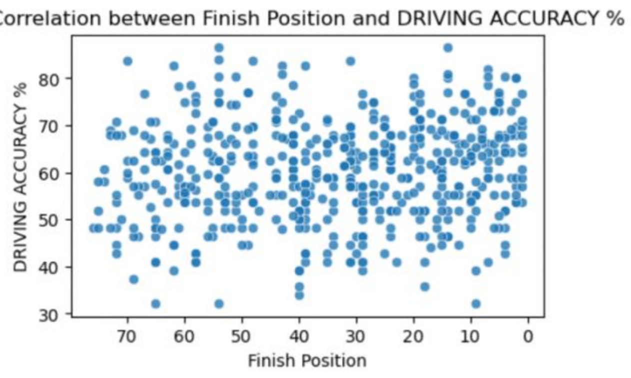


Figure 27: Correlation between finish position and greens in regulation %.

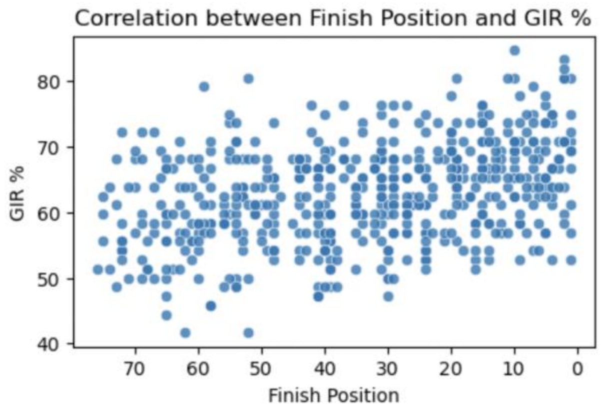


Figure 28: Correlation between finish position and par 4 scoring average.

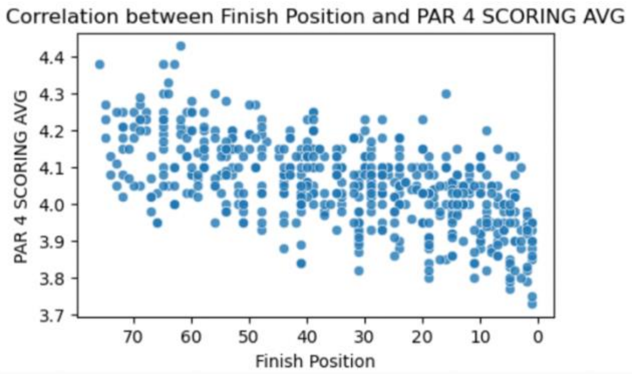


Figure 29: Correlation between finish position and par 5 scoring average.

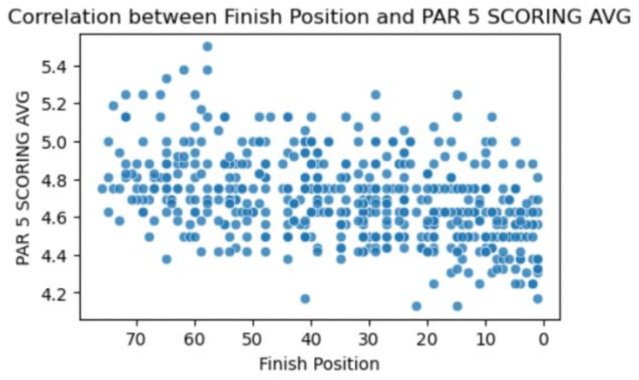


Figure 30: Correlation between finish position and putting average.

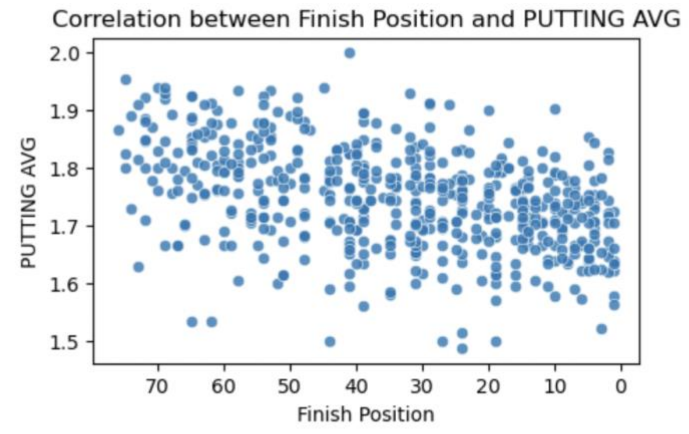


Figure 31: Correlation between finish position and sand save %.

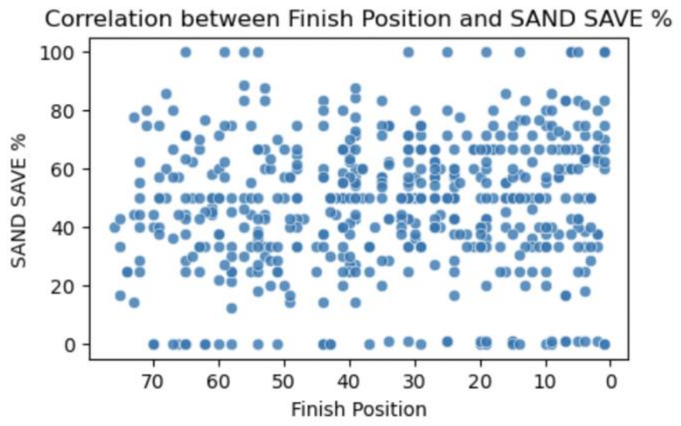


Figure 32: Correlation between finish position and scrambling %.

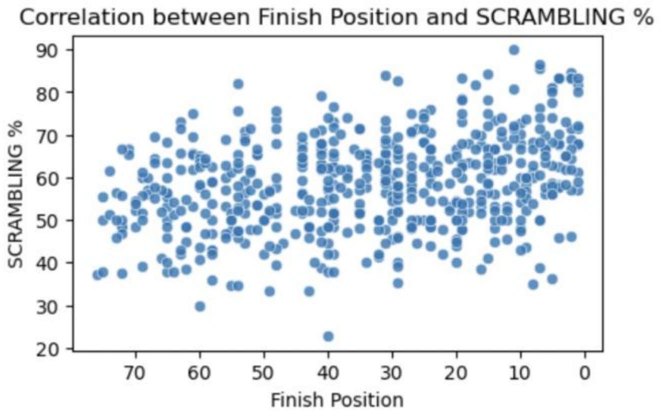


Figure 33: Correlation between finish position and total driving distance.

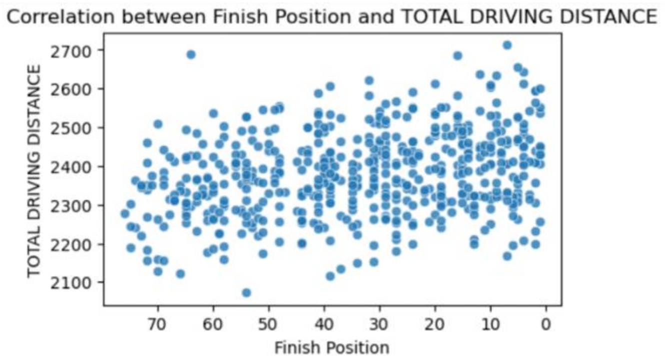


Figure 34: Correlation between finish position and total proximity to the hole.

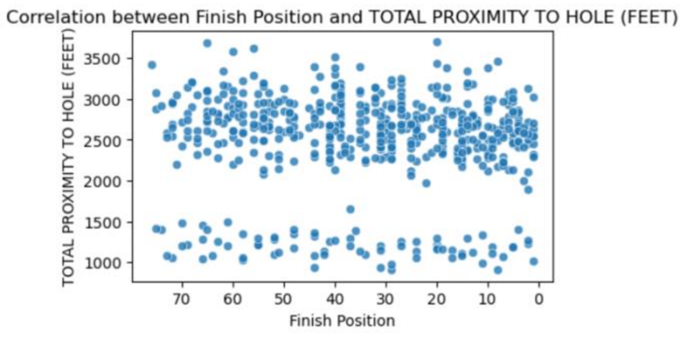


Figure 35: Correlation between finish position and total strokes gained: approach to the green.

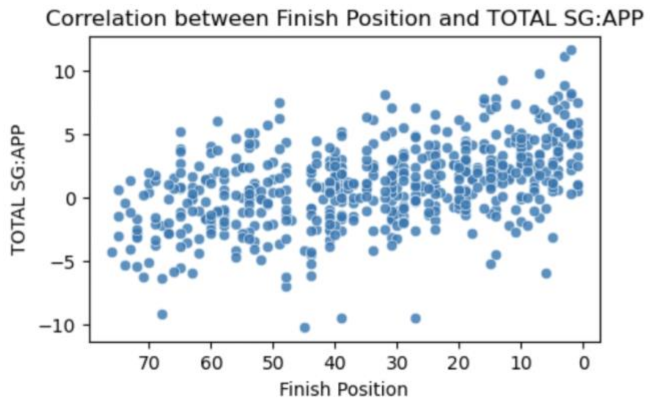
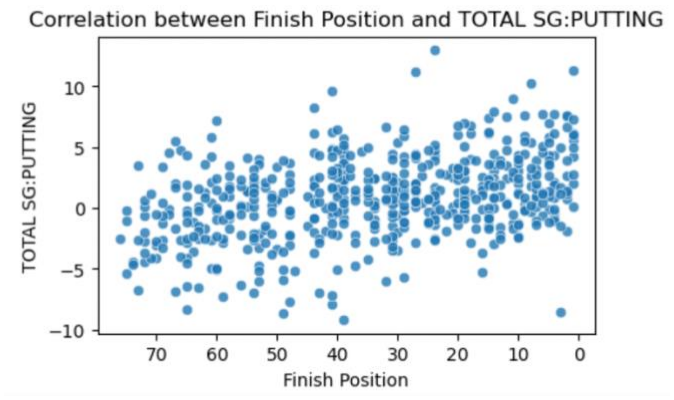


Figure 36: Correlation between finish position and total strokes gained: putting.



The above figures allowed me to get an insight into the importance of statistics and their correlation with finish position. For example, we can see that driving accuracy % has a weak correlation in that when the accuracy percentage goes up, the player's finish isn't necessarily higher. On the other hand, birdie or better % has a high correlation in that when the percentage is higher, the finish position is likely lower.

6.3 Model Performance

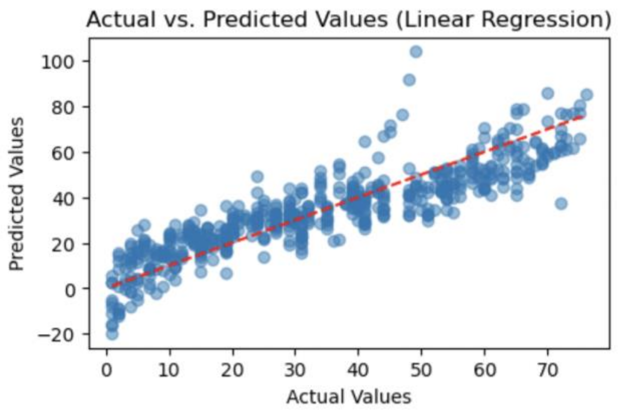
6.3.1 Linear Regression

Figure 37: Linear Regression performance evaluation.

```
Linear Regression Metrics: {'RMSE': 9.857814919987382, 'MAE': 7.8019767155233355, 'R²': 0.7717208661993014}
```

Figure 37 shows that Linear Regression met the objective of having an RMSE of below 10. The RMSE of the model was 9.8578, indicating that, on average, the predicted finish positions deviate by approximately 9.86 positions from the actual finish positions. The MAE for this model is 7.8019, this suggests that the mean absolute error in predictions is about 7.8 positions on average. Linear Regression achieved and R^2 score of 0.7717, meaning it can explain 77.17% of the variance in the finish positions.

Figure 38: Linear regression performance visualisation.



In Figure 38 we can see how a large proportion of the predictions follow the red reference line showing the actual outcomes. This reinforces the relatively good accuracy of the model. However, we can see there are some outliers and some of them are very inaccurate highlighting the room for improvement in this model. This visualisation gives us a better understanding of the model's accuracy and the reason for some of the metric results.

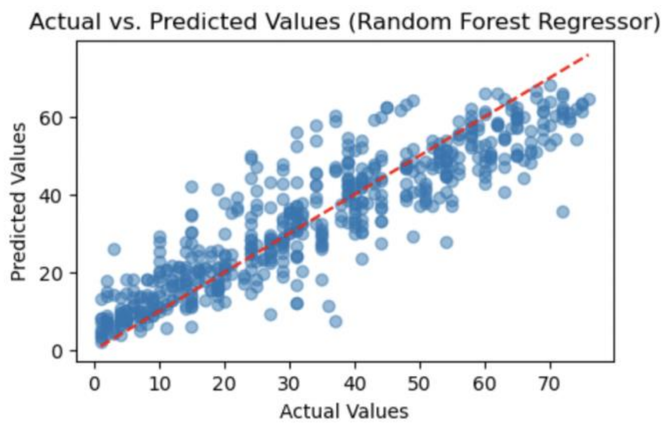
6.3.2 Random Forest

Figure 39: Random Forest performance evaluation.

```
Random Forest Metrics: {'RMSE': 9.110455138390554, 'MAE': 6.972404639583047, 'R²': 0.8050222548043111}
```

Figure 38 shows that Random Forest also meets the objective of an RMSE under 10. It achieves an RMSE of 9.1104 meaning, on average, the predicted finish is around 9.1104 positions away from the actual outcome. An MAE of 6.9724 suggests the mean absolute error in predictions is around 6.9 positions on average. The model also achieves an R^2 score of around 0.8050, this means it can explain 80.50% of the variance in the finish positions. From these metrics, we can see that Random Forest performs slightly better in each metric compared to Linear Regression meaning that it is a more accurate model for this dataset.

Figure 40: Random Forest performance visualisation.



Looking at Figure 40, we can see that Random Forest has many predictions along the red line implying a good level of accuracy. however, although the model is good at predicting the lower finishing positions, as the finishing positions increase, we see a larger level of dispersion compared to Linear Regression. This shows the importance of visualising the predictions and not only using evaluation metrics.

6.3.3 Gradient Boosting

Figure 41: Gradient Boosting performance evaluation.

```
Gradient Boosting Metrics: {'RMSE': 9.217751355749684, 'MAE': 7.127384258756923, 'R²': 0.8004026026408735}
```

Using Figure 39, we can see that Gradient Boosting was also successful in achieving an RMSE of lower than 10. This model got an RMSE of 9.2177 which means that on average, the predicted finish was 9.2 positions away from the actual outcome. This model got an MAE of 7.1273 meaning that the mean absolute error in predictions is around 7.1 positions on average. The R^2 score for this model was around 0.8004. This means that it can explain 80.04% of the variance in the finish positions. From these results, we can see that the Gradient Boosting model performed better than Linear Regression in every metric but slightly worse than Random Forest in every metric meaning that although it is more accurate than Linear Regression, Random Forest is a better choice for this dataset.

Figure 42: Gradient Boosting performance visualisation.

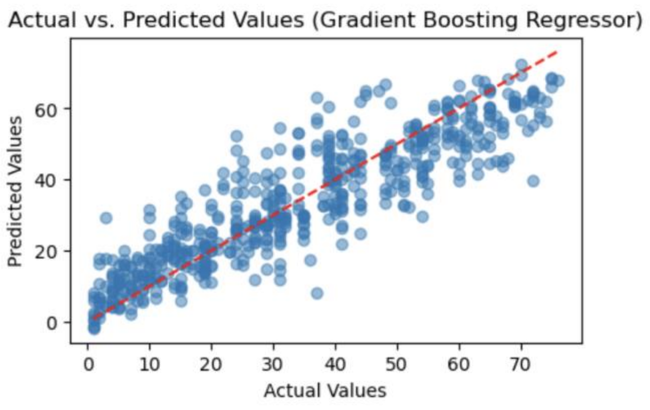
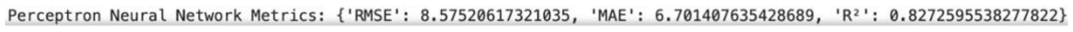


Figure 42 shows us that the visualisation of Gradient Boosting is very similar to that of Random Forest. This helps use to visualise the similarity in evaluation metric results showing that the lower finishing positions are predicted quite accurately whereas dispersion can be seen as the finish positions get higher.

6.3.4 Perceptron Neural Network

Figure 43: Perceptron Neural Network performance evaluation.



```
Perceptron Neural Network Metrics: {'RMSE': 8.57520617321035, 'MAE': 6.701407635428689, 'R²': 0.8272595538277822}
```

From Figure 40 we can see that the Perceptron Neural Network also meets the objective of having an RMSE lower than 10. This model has an RMSE of 8.5752 meaning that the predicted finish was 8.5 positions away from the actual finishes on average. With an MAE of 6.7014, the mean absolute error of predictions made by this model was around 6.7 positions on average. The R^2 score of this model was 0.8272 meaning that the model can explain 82.72% of the variance in finish positions. Looking at all of the models' performance, we can see that the Perceptron Neural Network was the best performing in every metric meaning that it is the best option for this dataset.

Figure 44: Perceptron Neural Network performance visualisation.

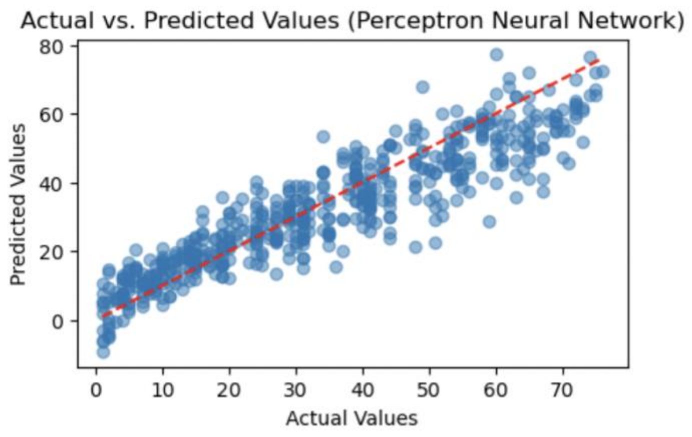


Figure 44 shows us the predictive power of the Perceptron Neural Network as the predictions closely follow the ground truth line. This model doesn't follow as closely as Linear Regression does however, showing how large deviations can affect the evaluation metrics as Linear Regression had the worst metric results but the best-looking visualisation but with the largest outliers.

In summary, the evaluation metrics gave us a numeric comparison of the models' performance allowing us to compare different parts of the models' predictive power. Also, using scatter plots to visualise the predictions gave us an alternative view of the models' accuracy and how the level and severity of dispersion can affect evaluation metric outcomes.

6.5 Objectives

At the beginning of this study, I set out five objectives that would act as milestones for the project. Throughout the project I was able to use these objectives as a guide for how successful the project was at each stage.

6.5.1 Objective 1

To gather and pre-process PGA TOUR data from eight 2023 tournaments to create a dataset with clean, complete, and formatted data ready for analysis.

This objective was met in the first stage of the implementation where the initial dataset containing statistics from eight different tournaments was correctly organised, cleaned, and standardised. This can be seen in section 6.1.

6.5.2 Objective 2

Use scatter plots to identify relationships between statistics and how they affect player outcomes.

This objective was met in the stage after the data pre-processing where the statistic relationships were visualised using scatter plots. Look at section 6.2 for the completion of this objective.

6.5.3 Objective 3

Compare and evaluate at least three machine learning algorithms as well as using advanced techniques to assess their performance based on metrics such as Mean Squared Error achieving a RMSE score of 10 or lower as a minimum.

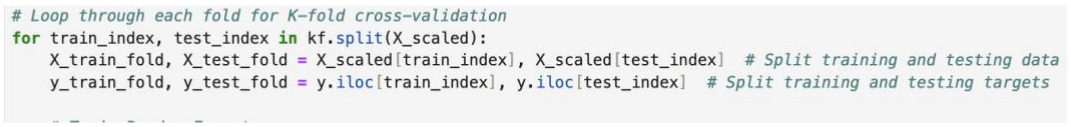
This objective was successfully completed and can be seen in section 6.3 where four models were successfully trained and tested, and all of the models achieved an RMSE less than 10.

6.5.4 Objective 4

Implement and optimise parameter tuning techniques such as k-fold cross-validation and grid search to improve model accuracy.

This objective was successfully met and can be seen in section 5.3 as well as Figure 11. The hyperparameter tuning played a big impact in achieving more accurate model predictions. K-fold cross-validation was not included in the implementation section but was a vital part of my code as seen in Figure 45.

Figure 45: K-fold cross-validation implementation.



```
# Loop through each fold for K-fold cross-validation
for train_index, test_index in kf.split(X_scaled):
    X_train_fold, X_test_fold = X_scaled[train_index], X_scaled[test_index]  # Split training and testing data
    y_train_fold, y_test_fold = y.iloc[train_index], y.iloc[test_index]  # Split training and testing targets
```

6.5.5 Objective 5

Visualise the accuracy of the models using scatter plots to compare the model's predictions and the actual outcomes.

This objective was successfully complete and can be seen in section 6.3. The use of scatter plots to visualise the models' accuracy gives an alternative and insightful view of the models' predictions.

6.6 Requirements Evaluation

6.6.1 Functional Requirements

The project started with importing the relevant PGA TOUR statistics such as "strokes gained" and "driving accuracy". This dataset was the most important part of the project as it contained all information needed by the models which were the statistics and the ground truth. The data was then cleaned, organised, and standardised to ensure it was in its best form to ensure the best possible predictions can be made by each model. This part of the requirements analysis was successfully met.

Feature engineering was an important part of the project, where relevant features were encoded. This process was essential for capturing the complex relationships between various statistic and player outcomes. To help in understanding these relationships, scatter plots were also employed for each stat to gain a full understanding of every relationship in the dataset. This part of the requirement analysis was met; however, more feature engineering could've taken place in order for even better results. For example, combining features with a high correlation could've improved the value of the dataset leading to better accuracy.

The core of the project involved the implementation of various predictive models such as Random Forest, Gradient Boosting, Linear Regression, and a Perceptron Neural Network. Each model was tuned using advanced techniques like k-fold cross-validation and grid search, these were essential for optimising model performance and reducing the risk of overfitting. Each model was then evaluated on three metrics RMSE, MAE, and R^2, as well as visualised using scatter plots providing a clear understanding of each model's accuracy and efficiency. This part of the study successfully followed the requirement analysis.

6.6.2 Non-functional Requirements

- **Performance and efficiency:** Data processing and model training were optimised to complete withing the time frame that was set out, ensuring the program was efficient and able to handle the computations within the 15-minute limit.
- **Accuracy and reliability:** The models achieved the target metrics with RMSE and MAE under 10 and an R^2 greater than 0 for all models, indicating high reliability and accuracy in predictions.
- **Usability and flexibility:** the program provided clear and insightful visualisations of both relationships in features, and for model evaluations.
- **Scalability and flexibility:** The design allows for the use of additional historical data to be added, and the models were built with flexibility in hyperparameter tuning to adapt to different data conditions.
- **Compatibility:** the software will be compatible with all major operating systems and hardware using standard Python libraries and machine learning frameworks.

6.6.3 Risk Management

Potential risks such as data quality issues and model overfitting were managed through comprehensive data cleaning and standardisation. The use of regularisation techniques like dropout in neural network training also helped managed risks. Computational limits were addressed with efficient parameter tuner and early stopping in the neural network.

Chapter 7 - Conclusion

7.1 Successes

This Study was successful in developing a novel machine learning model for predicting PGA TOUR player finishes. The results of this project show successes in data pre-processing, model building, and evaluation. By adhering to the projects aims, objectives, and requirements, the study was able to implement a Linear Regression model, Random Forest model, Gradient Boosting mode, and a Perceptron Neural Network with a relatively high level of accuracy and reliability seen through the use of evaluation metrics such as RMSE, MAE, and R^2. The models were successfully optimised using k-fold cross-validation and Grid Search and visualisations of both feature relationships and each model's predictive accuracy were successfully implemented.

7.2 Limitations

Despite having many successes, this study does have some limitations. Firstly, the size of the dataset was relatively small. This was due to the extremely time-consuming process of downloading over 180 stats and then collaborating these into one dataset. Only using eight tournaments from one year may not fully capture the diversity of playing conditions and player performance variations. The limited dataset size could have affected the models' ability to generalise across broader scenarios. Also, although the models achieved relatively good predictive accuracy, some relationships between player statistics and player outcomes remained complex and challenging to fully capture. The Perceptron Neural Network sometimes showed overfitting even after hyperparameter optimisation due to the dataset size. Finally, real-time data was something that I wanted to implement after seeing the gap for it in my literature review. Real-time data such as weather conditions, course conditions, and player form would've improved the model's ability to dynamically adjust predictions however, obtaining this data was extremely difficult with golf being a very old-fashioned sport.

7.3 Future work

In the future, expanding the dataset would significantly improve the predictive accuracy of the models. Implementing real-time data would also improve the accuracy of the models as well as the dynamic predictive ability of the models. Exploring more complex machine learning techniques such as deep learning models like Long Shot-Term Memory (LSTM) networks or Convolutional Neural Networks (CNN) could provide better handling of complex relationships.

References

Bergstra, J., Ca, J. and Ca, Y. (2012). Random Search for Hyper-Parameter Optimization Yoshua Bengio. *Journal of Machine Learning Research*, [online] 13, pp.281–305. Available at: https://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf.

Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1), pp.5–32. doi:https://doi.org/10.1023/a:1010933404324.

Chai, T. and Draxler, R.R. (2014). Root mean square error (RMSE) or mean absolute error (MAE)? – Arguments against avoiding RMSE in the literature. *Geoscientific Model Development*, 7(3), pp.1247–1250. doi:https://doi.org/10.5194/gmd-7-1247-2014.

Conda (2017). *Conda — Conda documentation*. [online] docs.conda.io. Available at: https://docs.conda.io/en/latest/.

Constantinou, A. and Fenton, N. (2017). Towards smart-data: Improving predictive accuracy in long-term football team performance. *Knowledge-Based Systems*, 124, pp.93–104. doi:https://doi.org/10.1016/j.knosys.2017.03.005.

Daron Prater (2018). *PGA-Tour-Data-Science-Project/PGA Tour Machine Learning Project - Classification.ipynb at master · daronprater/PGA-Tour-Data-Science-Project*. [online] GitHub. Available at: https://github.com/daronprater/PGA-Tour-Data-Science-Project/blob/master/PGA%20Tour%20Machine%20Learning%20Project%20-%20Classification.ipynb [Accessed 29 Apr. 2024].

E.M.M. Alzeyani and Csaba Szabó (2023). A Study on the Effectiveness of Agile Methodology Using a Dataset. *Acta electrotechnica et informatica*, 23(1), pp.3–10. doi:https://doi.org/10.2478/aei-2023-0001.

Fried, H.O., Lambrinos, J. and Tyner, J. (2004). Evaluating the performance of professional golfers on the PGA, LPGA and SPGA tours. *European Journal of Operational Research*, 154(2), pp.548–561. doi:https://doi.org/10.1016/s0377-2217(03)00188-7.

Friedman, J.H. (2001). Greedy Function Approximation: A Gradient Boosting Machine. *The Annals of Statistics*, [online] 29(5), pp.1189–1232. Available at: https://jstor.org/stable/2699986 [Accessed 7 May 2024].

Fushiki, T. (2009). Estimation of prediction error by using K-fold cross-validation. *Statistics and Computing*, 21(2), pp.137–146. doi:https://doi.org/10.1007/s11222-009-9153-8.

Grady, N., Payne, J. and Parker, H. (2017). *Agile Big Data Analytics AnalyticsOps for Data Science*. [online] Available at: http://www.midp-info.org/uploads/1/0/6/5/10650753/s02208_2693.pdf.

Hanson, H., Harland, A., Holmes, C. and Lucas, T. (2012). Method for understanding football ball motions using video based notational analysis. *Procedia Engineering*, 34, pp.164–169. doi:https://doi.org/10.1016/j.proeng.2012.04.029.

Hastie, T., Tibshirani, R. and Friedman, J. (2008). *Springer Series in Statistics The Elements of Statistical Learning Data Mining, Inference, and Prediction Second Edition*. [online] Available at: https://hastie.su.domains/Papers/ESLII.pdf.

Ioffe, S. (2015). *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. [online] Available at: https://arxiv.org/pdf/1502.03167.

James, G., Witten, D., Hastie, T. and Tibshirani, R. (2013). *An Introduction to Statistical Learning*. *Springer Texts in Statistics*. New York, NY: Springer New York. doi:https://doi.org/10.1007/978-1-4614-7138-7.

Jelinek, H.F., Kelarev, A.V., Robinson, D.M., Stranieri, A. and Cornforth, D. (2014). Using meta-regression data mining to improve predictions of performance based on heart rate dynamics for Australian football. *Applied Soft Computing*, 14, pp.81–87. doi:https://doi.org/10.1016/j.asoc.2013.08.010.

Jupyter (2019). *Project Jupyter*. [online] Jupyter.org. Available at: https://jupyter.org/.

LeCun, Y., Bengio, Y. and Hinton, G. (2015). Deep Learning. *Nature*, [online] 521(7553), pp.436–444. doi:https://doi.org/10.1038/nature14539.

Leung, C.K. and Joseph, K.W. (2014). Sports Data Mining: Predicting Results for the College Football Games. *Procedia Computer Science*, 35, pp.710–719. doi:https://doi.org/10.1016/j.procs.2014.08.153.

Martinez Arastey, G. (2020). *The Increasing Presence Of Data Analytics In Golf*. [online] Sport Performance Analysis. Available at: https://www.sportperformanceanalysis.com/article/increasing-presence-of-data-analytics-in-golf.

Maszczyk, A., Gołaś, A., Pietraszewski, P., Roczniok, R., Zając, A. and Stanula, A. (2014). Application of Neural and Regression Models in Sports Results Prediction. *Procedia - Social and Behavioral Sciences*, 117, pp.482–487. doi:https://doi.org/10.1016/j.sbspro.2014.02.249.

Matplotlib (2012). *Matplotlib: Python plotting — Matplotlib 3.1.1 documentation*. [online] Matplotlib.org. Available at: https://matplotlib.org/.

Mckinney, W. (2010). *Data Structures for Statistical Computing in Python*. [online] Available at: https://pdfs.semanticscholar.org/ef4e/f7f38bb907e5d7b4df3e6ff1db269d4970f5.pdf?_gl=1.

Microsoft (2023). *Microsoft Excel, Spreadsheet Software*. [online] www.microsoft.com. Available at: https://www.microsoft.com/en-gb/microsoft-365/excel.

Numpy (2009). *NumPy*. [online] Numpy.org. Available at: https://numpy.org/.

Pandas (2018). *Python Data Analysis Library*. [online] Pydata.org. Available at: https://pandas.pydata.org/.

park, J. (2019). *PGA Tour Machine Learning Project*. [online] kaggle.com. Available at: https://www.kaggle.com/code/jmpark746/pga-tour-machine-learning-project [Accessed 29 Apr. 2024].

Python (2019). *Python*. [online] Python.org. Available at: https://www.python.org/.

Saltz, J.S. and Shamshurin, I. (2019). *Achieving Agile Big Data Science: The Evolution of a Team's Agile Process Methodology*. [online] IEEE Xplore. doi:https://doi.org/10.1109/BigData47090.2019.9005493.

Scikit-learn (2019). *scikit-learn: machine learning in Python*. [online] Scikit-learn.org. Available at: https://scikit-learn.org/stable/.

seaborn (2012). *seaborn: statistical data visualization — seaborn 0.9.0 documentation*. [online] Pydata.org. Available at: https://seaborn.pydata.org/.

Shotlink.com. (2016). *ShotLink: History*. [online] Available at: https://shotlink.com/about/history.

Team, K. (n.d.). *Keras documentation: KerasTuner*. [online] keras.io. Available at: https://keras.io/keras_tuner/.

TensorFlow (2019). *TensorFlow*. [online] TensorFlow. Available at: https://www.tensorflow.org/.

www.pgatour.com. (n.d.). *Money/Finishes | Categories | PGA TOUR Stats*. [online] Available at: https://www.pgatour.com/stats/money-finishes.