DATA3001 FINAL REPORT

NRL MODEL GROUP 4

Table of Contents

Team Members:

Jacky Gan

Irvin Linardy

Jeffry Kwok

Haowen (Kevin) Shi

Johnny Zhou

Executive Summary

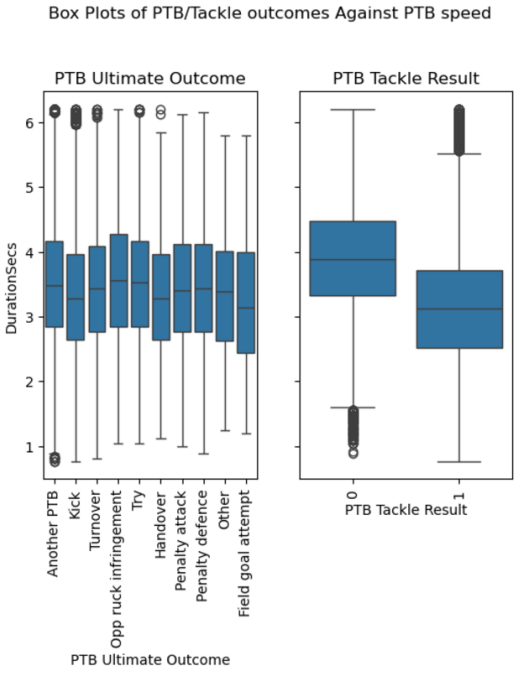
This report aims to explore the factors affecting play-the-ball (PTB) speed in National Rugby League (NRL) games, as it is found through previous work and our own analysis that altering this speed may lead to winning outcomes. Exploration of these factors is performed through predictive modelling of exact PTB speed. We have opted to start with simpler models, such as OLS and LASSO regressions, to both explore our data and act as a baseline for further models. Then, three more complex models that would be able to account for non-linear relationships with PTB speed are adopted, which include XGBoost, Random Forest and Neural Network machine learning algorithms.

From these five models, we found that the most influential factors to PTB speed were tackle related variables such as 'PTB Contest' (whether an offensive player lost a tackle, was sent to the ground, or remained on their feet after a winning tackle), 'Tackle Number' (which tackle within the set, 1–6, did the subsequent PTB occur) and 'Total Involved Tacklers', along with time related variables, such as 'Total Possession Seconds'. We also found the field position variable, 'Zone of Possession', to be highly impactful.

After identifying these key variables and analysing their relationship with PTB speed, we recommend tactics and objectives which teams can implement in order to manipulate PTB speed to their advantage. Namely, these include aggressive tackles when defending winning plays (this slows PTB speed to give defences time to recover), increasing defensive intensity earlier on in a set (as to minimise offensive momentum) and tackling with 2-3 defenders (these numbers slows PTB the most). Moreover, we recommend further analysis using models that are more focused towards finding the exact relationship of continuous variables, such as gametime. These models may include GAMs, Polynomial or Spline Regression. Overall, we have successfully investigated the factors influencing PTB speed, highlighting the competitive advantages of manipulating PTB speed and motivating further research.

Background

This project focuses on analysing the importance of the PTB speed in NRL games. PTB refers to the act of restarting play after an offensive player is tackled, which involves the tackled player kicking the ball backwards to another player. The speed of PTBs is critical in shaping the flow and outcome of games. Faster PTBs allow offensive teams to exploit gaps in the defence, maintaining momentum and potentially increasing scoring opportunities. Given its tactical importance, PTB speed has become a metric of interest among coaches and analysts in NRL, with rule changes including the '6 Again' rule aiming to influence this aspect of gameplay.[1]



Furthermore, the boxplots at the right highlight the relationship between PTB speed and game outcomes. On the left, the 'PTB

[1] GitHub. (n.d.). *NRL data set 4 - README.md*. UNSW Data3001 Repository. Retrieved November 18, 2024, from https://github.com/unsw-edu-au/data3001-data-nrl-4/blob/main/README.md; GitHub. (n.d.). NRL data

Ultimate Outcome' plot shows that shorter PTB durations are associated with more favourable outcomes, such as scoring tries or gaining handovers, whereas longer durations tend to lead to less advantageous results like turnovers or penalties. On the right, the 'PTB Tackle Result' plot indicates that faster PTBs correlate with better tackle outcomes, reflecting a more effective and dynamic game flow. Hence, these findings emphasise the importance of minimising PTB speed for the offensive team, as it enhances opportunities to secure positive outcomes by accelerating gameplay, ultimately increasing the chances of winning the game. Also, the opposite can also be said for the defending team, as they wish to maximise their opponents PTB speed to allow defensive recovery. Furthermore, a study titled 'Influence of Ball-in-Play Time on the Activity Profiles of Rugby League Match-Play' investigated how variations in ball-in-play time affect players' physical demands. The research found that longer ball-in-play periods lead to increased physical exertion, highlighting the importance of PTB speed in managing player workload and game intensity.[2]

Previous Work

Data group 5 has previously analysed PTB events to understand its influence in the NRL, identifying factors such as tackle type and field position that impact PTB speed. Moreover, existing literature on NRL performance metrics has highlighted PTB speed as an essential yet complex variable affected by game context, rule changes, and tactical adjustments.[3] Once again, our earlier box plots suggest a link between the PTB speed and game outcomes, however, this does not necessarily prove causation. Faster PTB speeds may reflect other factors, such as team strategy, player skill, or match conditions, rather than directly causing better results. Success depends on many factors, including tactics, player decisions, and external influences like rule changes. To confirm PTB speed's role, more robust analyses are needed, such as controlling for confounding factors and using advanced models. Until then, it's essential to interpret the relationship cautiously when analysing how PTB speed directly impacts winning.[4]

Despite this, the primary goal of this project is to identify key factors influencing PTB speed through predictive analysis, motivated by our finding of correlation between PTB speed and positive game outcomes. By understanding these factors, we aim to offer insights into how teams can strategically optimise PTB speed to enhance game performance, potentially suggesting new training or game-day strategies to adjust PTB speed based on in-game conditions.

set 5 - README.md. UNSW Data3001 Repository. Retrieved November 18, 2024, from https://github.com/unsw-edu-au/data3001-data-nrl-5/blob/main/README.md.

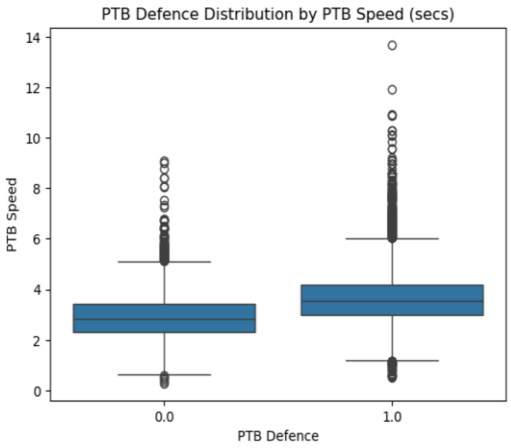
[2] Gabbett, T. J., Jenkins, D. G., & Abernethy, B. (2012). Influence of ball-in-play time on the activity profiles of rugby league match-play. *Journal of Strength and Conditioning Research, 26*(6), 1508–1516. https://doi.org/10.1519/JSC.0b013e318231a753.

[3] GitHub. (n.d.). *NRL data set 4 - README.md*. UNSW Data3001 Repository. Retrieved November 18, 2024, from https://github.com/unsw-edu-au/data3001-data-nrl-4/blob/main/README.md; GitHub. (n.d.). NRL data set 5 - README.md. UNSW Data3001 Repository. Retrieved November 18, 2024, from https://github.com/unsw-edu-au/data3001-data-nrl-5/blob/main/README.md.

[4] GitHub. (n.d.). *NRL data set 4 - README.md*. UNSW Data3001 Repository. Retrieved November 18, 2024, from https://github.com/unsw-edu-au/data3001-data-nrl-4/blob/main/README.md; GitHub. (n.d.). NRL data set 5 - README.md. UNSW Data3001 Repository. Retrieved November 18, 2024, from https://github.com/unsw-edu-au/data3001-data-nrl-5/blob/main/README.md.

Exploratory Data Analysis

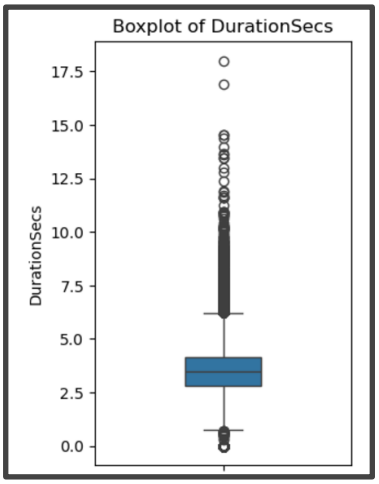
Missing values



First, upon checking for missing values, we found that the majority of 'PTB Defence' values were missing (87992 of 112027 rows) (see Figure 1 of the appendix). This is an issue, since it is intuitively expected for 'PTB Defence' (whether the defence is set or not) to meaningfully impact PTB speed (set defences can make more dominant tackles and slow PTB speed, and out of position defences incentivises faster PTB speeds in order to exploit this). In the boxplot to the right, we can clearly see the impact of PTB Defence on PTB speed, shifting the entire distribution upwards by around 0.5 seconds when defences are set (also see Figure 2 of the appendix).

However, we find that the vast missingness of our data is non-random, as we are missing data from 2021 onwards. We are unsure exactly why this is the case, but it can likely be attributed to data collection issues. Since our missingness is non-random, including 'PTB Defence' by either filling in missing values (with -1 or via a predictive model) or limiting our sample would lead to bias associated with only using data from 2020. Hence, we have chosen to remove 'PTB Defence' entirely from our models. Furthermore, we have found missing values in 'Anonymize 1PlayerId' (32) and 'Total Involved Tacklers' (36). However, since these make up a very small portion of our sample size, we simply remove these rows with missing values.

Outliers



Now that missing data is handled, we check if there are significant outliers in our data. Upon doing so, we find that our target variable, 'DurationSecs', contains 2187 outliers, which makes up approximately 2% of all rows. To observe the severity of these outliers, we inspect a boxplot for 'DurationSecs' (see Figure 3 of the appendix).

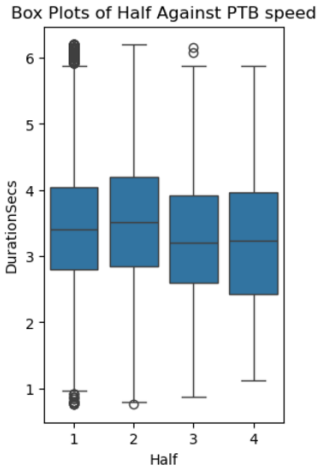
Column	No. Outliers
Away Score	1698
Home Score	1021
OppScore	899
Score	1068
CurrentMargin	7136
Total Involved Tacklers	42
Half	117

It is visible that outliers of 'DurationSecs' deviate very far from the upper whisker, even reaching up to a PTB speed of 18 seconds, which is highly unrealistic. While only representing a small proportion of our data (2%), retaining these outliers would bias the results of our linear models significantly, since their regression lines would be pulled towards these extreme values. Hence, for the interpretability and accuracy of our OLS and LASSO models, we remove these outliers (see Figures 4 and 5 of the appendix)

Moreover, we notice outliers in a few of our explanatory variables, displayed in the table above (see also Figure 5 of the appendix). These mostly occur in our score related variables and tend to make up a small proportion of row data (<2%), except for CurrentMargin (~6%). Why exactly this is the case is uncertain, but it may be related to overtime data (unlikely, due to Golden Point rule), or simply the

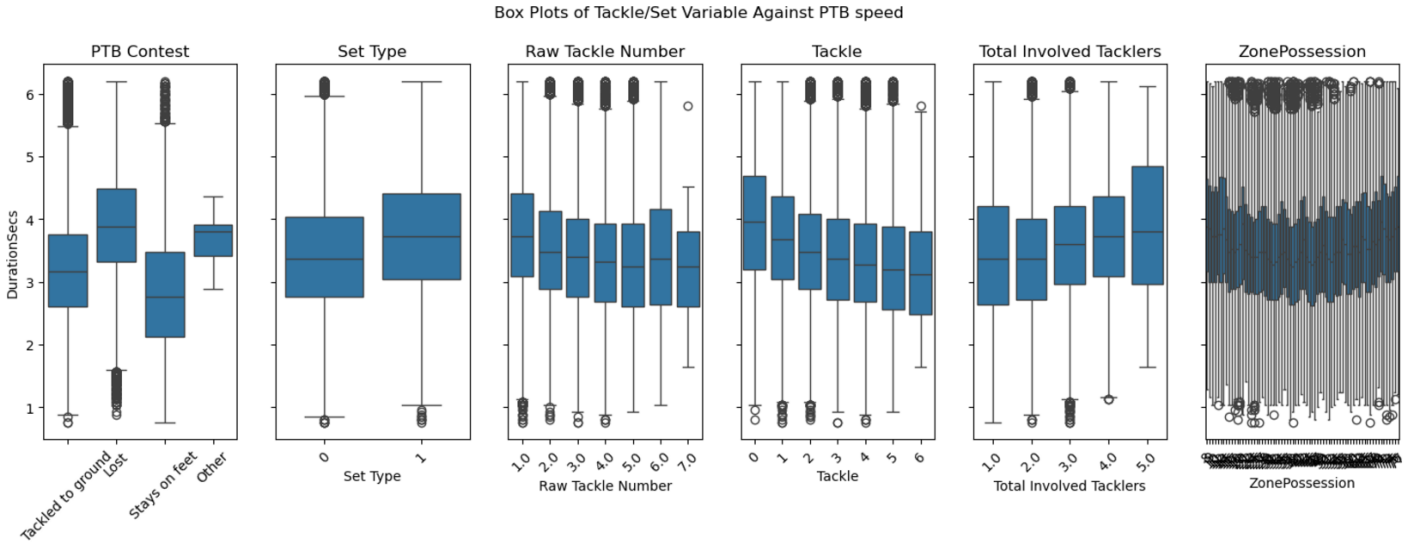
varied nature of scoring. While keeping these outliers would affect our coefficient estimates and variance for our linear models, this effect is much weaker than the outliers in 'DurationSecs', as they are our predictors. We also use non-linear models that are more robust to outliers, such as our tree based models, XGBoost and Random Forest, and our Neural Network Model. Hence, we decide to keep our outliers, as they aren't expected to have strong effects, which would only decrease through the use of models which are robust to outliers.

Variable Plots on PTB Speed

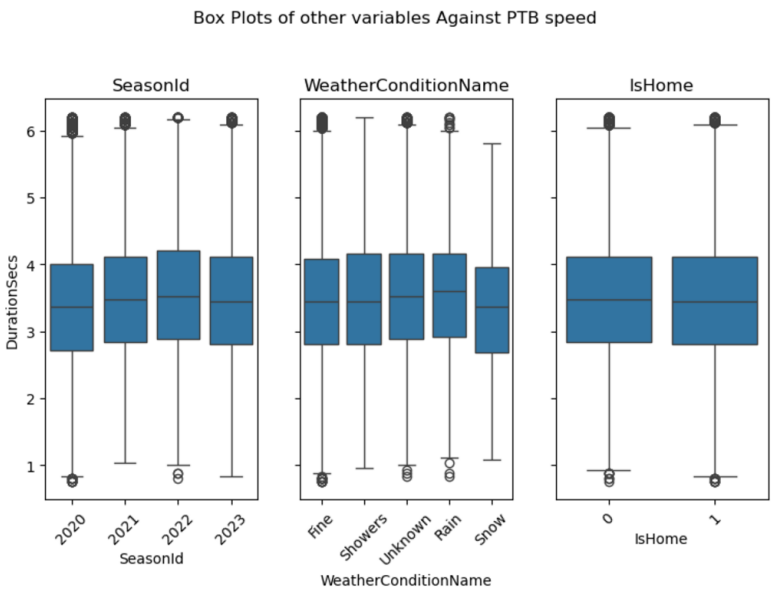


Next, we inspect scatter plots and boxplots of our explanatory variables on PTB speed in order to identify suspected key features. First, we observe scatter plots of game time related features, such as 'Score', 'PossessionSecs' and 'ElapsedTime' (see Figures 6, 7 and 8 of the appendix). Since these variables are continuous, with bunched up PTB values, plots are very difficult to make visual insights from (plots are shown in appendix), so we need to explore their relationships using different methods. However, for our simpler time variable, 'Half', we produce a boxplot that displays a clear relationship between time periods, where PTB speed between the 1st and 2nd periods, and then drops in overtime periods.

Furthermore, we observe boxplots of tackle and set features which may potentially impact PTB speed. For our boxplot of 'PTB Contest', we find that PTB speed is slowest after a lost tackle, then if they are tackled to the ground, and fastest when the tackled player stays on their feet after a tackle. These jumps in PTB speed are large (over a second difference in median between 'stays on feet' and 'Lost'), indicating its significance towards PTB Speed. Moreover we see a clear relationship with PTB speed in tackle position variables, 'Set Type' (whether possession is good ball, 1, or yardage, 0) and 'ZonePossession' (which, out of the 80 zones on the field, was the ball played, with later zones being near the endzone). We find that possessions located near either team's try line (good ball, or attacking near their own try line) have a slower PTB speed, and those near the centre of the field (yardage) are faster. Also, we notice that PTB speeds up as the tackle number within a set increases, and slows down as more tacklers are involved preceding a PTB. Hence, from our boxplots, we suspect that all of these tackle features influence PTB speed greatly, with the most severe feature being 'PTB Contest'.



Next, we observe boxplots of our other features that are not related to game time, tackles or sets, namely, 'SeasonId' (year), 'WeatherConditionName' and 'IsHome'. We find that PTB speed is slower from 2021 onwards, peaking in 2022 and speeding up a little in 2023. Also, rain is found to noticeably slow PTB speed, and on the rare occasion snow is present, PTB speed increases slightly. Finally, 'IsHome' seems to have no little to no effect on PTB speed. It should also be noted that plots of id variables, such as 'Club Id' or 'Official Id' are difficult to visually inspect, so analysis through other means are required to explore relationships with PTB speed.



Hence, from the plots we could visibly inspect, we hypothesise that tackle features influence PTB speed the most.

Correlation Matrix and Preliminary Variable Selection

A preliminary variable selection is then conducted based on an analysis of the correlation matrix and intuition. Its purpose is to eliminate a large set of mostly irrelevant variables before modelling in a highly efficient manner. Moreover, the selection is merely preliminary, meaning that several variables may be further removed based on the results of our models. The correlation matrix is attached as Figure 12 of the appendix.

The following variables are removed after the preliminary variable selection: PTB Defence, Anonymize 1PlayerId, Event Name, PTB Tackle Result, Away Score, Home Score, Match Id, Opp Possession Secs, PTB Ultimate Outcome, OppScore, Player Id, Sequence Number (SeqNumber), Set, Total Possession Secs, Tackle, Elapsed Time, ZonePhysical, Official Id and Position Id. The reasons for exclusion are outlined below.

Exclusion based on correlation matrix

- **Match Id** is excluded because it is strongly correlated with **Season Id** (correlation: 1.00), such that one of them must be removed to prevent strong multicollinearity. Season Id contains more information than Match Id as it also reflects the change of rules, which is dependent on which season.
- **Tackle** is removed because it is strongly correlated with Raw Tackle Number (correlation: 0.97). **Raw Tackle Number** is easier to interpret than Tackle because Tackle contains information about reset, which may confuse between an earlier tackle and a later tackle as both would have the same tackle number.

- **Elapsed Time** is removed due to a strong correlation with GameTime (0.99). **Game Time** will be kept because it reflects the official game time, which is more likely to affect players' decisions.
- There is a set of variables that are highly correlated with each other: **Half, Opp Possession Secs, Possession Secs, Tota lPossession Secs, Sequence Number and Set**. Only Half and Possession Secs are retained and the rest are removed. The first reason is that, except for Half, Possession Secs is strongly correlated with the rest of the variables (with a correlation over 0.9) while itself is intuitively a useful variable. Specifically, its correlations with Sequence Number and Set are over 0.95, suggesting that they represent highly similar information. The second reason is that Possession Secs, Opp Possession Secs and Total Possession Secs are perfectly correlated such that one of them must be removed.
- **Opp Score** is removed because there exists a perfect collinearity between Opp Score, Score and Current Margin such that one of them must be removed. Both Score and Current Margin are retained because it provides the information relating to the team's current performance, offering context as to players' decisions on the PTB speed.

Exclusion based on intuition

- **Home Score** and **Away Score** are removed as each variables represent the final scores of the respective teams. Each team's final score is logically incapable to impact the PTB speed during the game. Accordingly, they are removed due to a lack of causal impact. Both features are also not available to the players at the time of PTB. The same reasoning applies for the removal of **PTB Ultimate Outcome** and **Event Name**.
- **Anonymize 1PlayerId** and **Player Id** are removed because they have more than 700 distinct values, which is too computationally intensive. Moreover, the purpose of this project is to understand factors that impact the PTB speed generally. Excluding the impact of individual players and their playing style would better encapsulate the effects of other variables.
- **Official Id** is removed because they have more than 50 distinct values. Similar to the exclusion of Player Id, in order to understand features that generally impact the PTB speed, it is desirable to remove features that are overly specific..
- **Position Id** is removed due to low interpretability.
- **PTB Tackle Result** is excluded because it contains the same information as PTB Contest, while the latter provides a more contextualised situation about that particular tackle or contest. Hence, the latter is retained.
- **Zone Physical** is removed because it contains almost the same information as ZonePossession, but fails to distinguish between the offensive team and the defensive team. Accordingly, **Zone Possession** is retained and ZonePhysical is removed.

OLS Regression

After eliminating variables during our correlation analysis, OLS regression is performed on the remaining features. We found over numerous iterations that 'SeqNumber' and 'Set' are not statistically significant (small t-stats, high p-values). This is likely due to high correlation with other time related features, such as 'PossessionSecs' and 'Half', which we see in our prior correlation analysis. Hence, we excluded these two features to reduce multicollinearity and simplify our models. We also conducted joint F-tests on our categorical features (e.g. 'WeatherConditionName', 'SeasonId', 'Half', etc) and found them all to be jointly significant to PTB speed (see Figure 9 of the appendix). Moreover, it should

also be noted that 'Current Margin' has a high P-value (0.132), which is unexpected, since point differential intuitively should have some sort of impact on PTB speed (e.g. being far behind in score may motivate faster PTBs to catch up and gain momentum). So, we have decided to keep 'Current Margin' when modelling and observe its effects. Thus, we are left with our final set of variables for modelling, consisting of 61 features (including one-hot encoded columns) and 108,805 rows (see Figure 10 of the appendix).

We then performed our simple OLS regression model on these variables to give a baseline for further models and begin exploring the most influential factors for PTB speed (see Figure 11 of the appendix). Using an 80-20 train-test split, our model has a Mean Squared Error (MSE) of 0.7212 (4dp). Similarly, when using 5-fold cross validation, an MSE of 0.7357 (4dp) is obtained. We also obtained an adjusted R-squared of 0.2362, suggesting there is still a large portion of variation in PTB speed that is not explained by our predictors. However, this is not considered too much of an issue, since we can still interpret coefficients and achieve our goal of exploring predictors.

To explore said predictors, we interpret and compare the magnitudes of our different coefficients to find which variables influence PTB speed the most in our OLS model (entire OLS output can be found at Figure 11 of the appendix). 'PTB Contest', as hypothesised in our boxplots, does seem to have the strongest effect on PTB speed, as when compared to a lost tackle, staying on your feet after a tackle leads to an average decrease in PTB speed of 1.07 seconds, ceteris paribus. Being tackled to the ground similarly leads to an average decrease of 0.69 seconds compared to a lost tackle. The next most impactful variable from our OLS model is 'Raw Tackle Number', where the 2nd tackle sees an average decrease in PTB speed of 0.26 seconds compared to the 1st, which decreases even further as tackles progress, up until the 5th tackle, which tends to be the fastest (0.45 seconds faster on average than 1st tackle). Then, PTB speed slows down a little in the 6th tackle. So, as hypothesised from our boxplots, our tackle/set features seem to have strong effects on PTB speed. However it should be noted that the reason for this could be the categorical nature of these variables. Since we weren't able to visibly inspect the box/scatter plots of many of our continuous variables, namely time related variables, along with the limitations of our OLS model (assumed linearity), the effects of these variables may not be accurately described (e.g. for 'Zone Possession', as we move closer towards the try line, PTB speed uniformly increases by 0.0041 seconds, which is not the exact relationship we found in our boxplot, where we found PTB speed to decrease near the centre of the field, and increase elsewhere), and strong influences on PTB speed may go unnoticed. Hence, we will later use models such as XGBoost, Random Forest and Neural networks to further assess feature importance, especially for variables relating to time, as 'Half' is one of the only time variables that we can accurately draw influences from in our linear OLS model. 'PossessionSecs' is another we can see as impactful, since over the course of the first two halves (80 minutes), PTB speed is estimated to increase by 0.96 seconds (80 mins for both halves * 60 sec/min * 0.0002 per PossessionSec), however, as discussed, this interpretation suffers from assumed linearity. It should also be noted that 'CurrentMargin' has a very limited impact on PTB speed, according to our OLS model, as when the current margin increases by 1 point, the average PTB speed increases by approximately 0.0003 seconds, ceteris paribus.

LASSO Regression

Methodology

After the preliminary variable selection, a LASSO regression is adopted at this stage to compare between their predictive accuracy. LASSO is selected as it is best at feature selection by shrinking the coefficients of irrelevant variables to zero. Moreover, since multicollinearity is not a strong concern, LASSO's capacity to perform regularisation is particularly suitable for high-dimensional dataset as there is a large set of predictors.

Standardisation of the dataset is essential before applying the LASSO model. Only numerical variables need to be standardised as the values of categorical variables would be highly skewed after standardisation. Hence, the following formula for standardisation is adopted:

$$\textbf{standardised variables} = (\textbf{numerical variables} - \textbf{mean})/\textbf{std}$$

Analysis

After standardisation of the dataset and application of LASSO, a lambda of 0.00099 is suggested, being so small that the LASSO model approximates to an OLS regression. The values of the coefficients of each predictor are attached as Figure 13 of the appendix. Although several coefficients are shrunk to zero exactly, those variables cannot be removed because they represent dummies of a categorical variable. These dummies, forming the categorical variable collectively, are jointly significant (see Figure 9 of the appendix).

Examination of notable variables

We will now examine the performance of notable variables from the LASSO regression. Specifically, the following variables are evaluated: Raw Tackle Number and PTB Contest.

Raw Tackle Number

Variable	Coefficient
Raw Tackle Number 2	-0.230
Raw Tackle Number 3	-0.311
Raw Tackle Number 4	-0.363
Raw Tackle Number 5	-0.415
Raw Tackle Number 6	-0.249
Raw Tackle Number 7	0.000

For Raw Tackle Number, it has six dummies to represent the categorical variable collectively. The base category is Raw Tackle Number 1, meaning the first tackle. Except for Raw Tackle Number 7, each coefficient of Raw Tackle Number 2 to 7 is negative, suggesting that the PTB speeds for later tackles are faster compared to the first tackle. There is also a decreasing trend of the coefficients from Raw Tackle Number 2 to 5, suggesting that the PTB speed increases as the tackle number goes up, ceteris paribus.

However, the PTB speed experiences a significant drop at Raw Tackle Number 6, compared to Raw Tackle Number 5. This is because each set of offence is constituted by 5 plays, so when the Raw Tackle

Number turns 6, it represents a new set of offence. It is also rare to have Raw Tackle Number 7, which has the same performance as the base category.

Hence, within a particular set of offence, the later the play, the faster the PTB speed, ceteris paribus. The fastest PTB speed generally occurs at play number 5 of a set, being the last play of a set. This conclusion is also consistent with the EDA and the findings from OLS regression.

PTB Contest

Variable	Coefficient
PTB Contest – Stays on feet	-1.043
PTB Contest – Tackled to ground	-0.684
PTB Contest -– Other	0.000

For PTB Contest, the base category is losing the PTB Contest. Both Stays on feet and Tackled to ground represent the situations of winning that particular PTB contest. PTB speed is fastest when a team wins the PTB Contest and the ball also stays on feet. It reduces 1.043 seconds compared to the losing scenario, significantly increasing the PTB speed, as the DurationSecs normally lie between 2.5 and 7.5 seconds. When a team wins the PTB Contest but the ball was tackled to ground, the PTB speed is slower than when the ball stays on feet. This is intuitively sensible as the process of grounding the ball delays the PTB. However, it is still 0.684 seconds faster than the losing scenario.

The situation labelled 'Other' is rare and has the same performance of the base category, meaning that it may be another situation of losing the PTB Contest.

In conclusion, in order to maximise the PTB speed, the team should try to both win the PTB Contest and ensure the ball stay on feet. This confirms the findings from both the EDA and OLS regression.

Evaluation of LASSO Regression

Figure 14 of the appendix shows the performance of the LASSO model. The R-squared of the model is very low, being 0.226, suggesting that the model could only explain 22.6% of the variance in the dependent variables. From the graph, most observations depart from the standard line, indicating that a linear model may be inappropriate to explain the variances. Moreover, the CV RMSE of the LASSO model is 0.8507, which is even higher than that of OLS regression, suggesting its undesirable performance in terms of predictive accuracy (see Figure 15 of the appendix).

Overall, the LASSO model is not a good fit for both training and test sets and more models need to be investigated, especially the non-linear models.

XGBoost Algorithm

XGBoost stands for Extreme Gradient Boosting. It's a powerful tool used in machine learning to help make better decisions. It's especially good because it can handle lots of data across many computers, making it appropriate for figuring out matters as to whether a particular event will happen, how much of something will happen, or putting things in order.

Supervised learning is important in XGBoost, where computers learn from examples that already have known answers, decision trees, which are similar to diagrams that help make decisions, and boosting, a method that improves decision trees by learning from mistakes.

In supervised learning, we give the computer examples with both the features and the labels. The computer then learns to recognise patterns between the information and the answers, so it can make guesses on new data it hasn't seen before.

The main purpose of XGBoost is to conduct tasks such as figuring out categories or classification, predicting amounts using regression, or organising data for ranking. It aims to achieve this by reducing mistakes during training through a method called the loss function, which helps the model make more accurate predictions.

Code Summary

The XGBoost Regressor is employed to predict the duration of events. For the model building phase, we initialise the XGBRegressor, tuning hyperparameters like n_estimators, max_depth, and learning_rate using GridSearchCV to optimise them. We train the model on 80% of the dataset to ensure that the training data accurately represents the overall data distribution. During the hyperparameter optimisation stage, we methodically explore different combinations of parameters with GridSearchCV, helping us find the optimal balance between performance and complexity, enhancing the model's accuracy by selecting the best settings.

Feature important analysis

After training, we analyse the feature importance generated by the XGBoost model, identifying which variables have the most significant impact on predictions. This analysis is crucial for understanding the factors that influence PTB speed and guide further data collection and feature engineering efforts.

Result Interpretation

In our results, the XGBoost model's efficacy for predicting the duration of events is reflected through various metrics and feature importances. The model was optimised to use learning rate of 0.1, maximum depth of 7, and the number of estimators to be 200, all determined as the best parameters. These settings are pivotal for ensuring that the model balances complexity with prediction accuracy.

The Mean Squared Error (MSE) achieved by the model is approximately 0.694, indicating the average squared difference between the estimated values and the actual value. An R^2 score of 0.253 suggests that about 25.3% of the variability in event duration is explained by the model. Although this demonstrates moderate predictive power, it also suggests room for improvement in model performance, possibly by enhancing feature engineering or by fine-tuning other model parameters.

An in-depth look at the model's feature importances reveals variables such as PossessionSecs, ZonePossession, and CurrentMargin are highly influential in predicting event durations, highlighting their critical roles in the dynamics of the events studied. Lesser, but notable influences come from specific Club_Id and Opposition_Id, which might be reflective of particular team characteristics or competitive dynamics that affect event durations.

Random Forest Algorithm

We chose the random forest method because of its robustness, its ability to efficiently handle large datasets, and its adeptness at managing both numerical and categorical data effectively. Random forest

is a popular machine learning algorithm that achieves a single outcome by aggregating the outputs from multiple decision trees. It has the ability to tackle both classification and regression tasks. The aim of the random forest algorithm is to enhance prediction accuracy while mitigating the overfitting issues common in single decision trees. It operates by constructing several decision trees during the training phase and producing either the mode of the classes for classification or the average prediction for regression from the individual trees, thereby merging the simplicity and interpretability of decision trees with increased accuracy and robustness through averaging the predictions from multiple trees.

Code summary

A Random Forest Regressor is employed to predict event durations from a sports performance dataset. The (GridSearchCV) optimises hyperparameters through cross-validation, ensuring the model's efficacy. The trained model is then evaluated on a test set, and feature importance is analysed to gauge which predictors most significantly impact event durations. This structured approach ensures robust model performance and reproducibility of results.

Model Building

In our project, the (Random Forest Regressor) is initialised with a specified random state to ensure reproducibility of results across multiple runs. (GridSearchCV) is used to optimise the random forest regressor by searching through a predefined grid of hyperparameters, including the number of trees, maximum depth of each tree, minimum number of samples required to split an internal node, and minimum number of samples required at a leaf node. This search is conducted over three folds of the dataset to ensure that the model is not only optimised for a specific subset of data but also generalises well across the entire dataset. Cross-validation scores are used to determine the best parameter combination, thus balancing model complexity with performance.

Feature important analysis

Important details regarding which features have the strongest predictive ability for the target variable are provided by the random forest regressor’s feature importance output after training. This analysis facilitates understanding the underlying dynamics of the model's judgments and can assist stakeholders in making well-informed decisions based on the model's outputs.

Result Interpretation

The optimal hyperparameters for predicting event durations have been identified as a maximum depth of 20, a minimum sample leaf of 2, a minimum sample split of 10, and the number of trees (n_estimators) in the forest set to 300. These parameters were meticulously adjusted to balance the complexity of the model with its generalisation capabilities, achieving a best cross-validation score of 0.22. This modest score indicates that while the model holds predictive power, there is potential for enhancement, possibly through more refined feature engineering or advanced ensemble methods.

The performance metrics from the test set reveal a Mean Squared Error (MSE) of 0.713 and an R^2 score of 0.241, suggesting that the model explains approximately 24.1% of the variance in event durations. This level of performance is moderate, highlighting the need for possibly incorporating additional variables or employing more sophisticated modelling techniques to boost predictive accuracy.

A detailed feature importance analysis indicates that the most significant contributors to model predictions include ‘PTB_Contest_Tackled to ground’ and ‘PossessionSecs’, with contributions of about 15.6% and 10.8% respectively. The importance of these features underscores the critical influence of physical contests and possession time on event durations. Additional significant features include ‘GameTime’, ‘PTB_Contest_Stays on feet’, and ‘ZonePossession’, among others. These features reflect the complex array of factors that impact event durations, providing valuable insights for sports analysts and coaches.

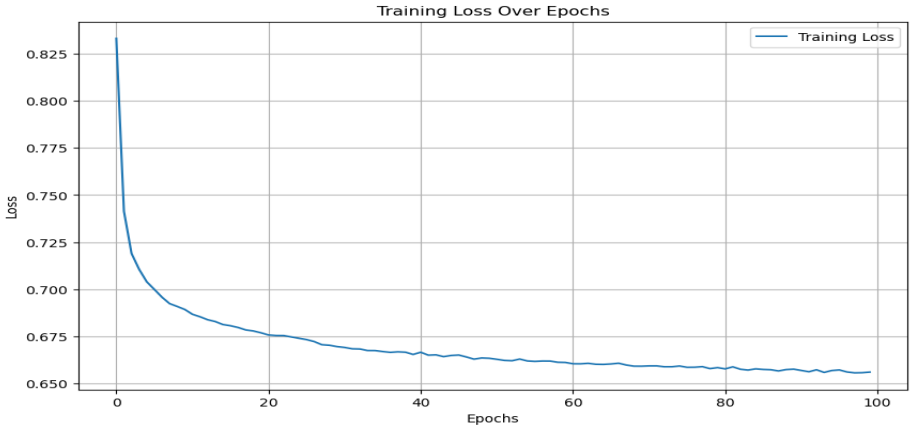
Neural Networks

The size and number variables in the dataset were a significant obstacle when deciding which models to use. Fortunately, the structure of neural networks allows for modelling of non-linear relationships and also benefits from the large dataset as it is able to generalise the data better due to its multiple layers. More commonly used for image and text data, it can also be used for tabular data, however it is considered a ‘black box’ model, meaning that it has low interpretability due to the complexity of the model. As a result, neural networks are considered as a supplementary model to back up the findings of the other models and are not considered for the final model.

Neural networks consist of three layers: an input layer, hidden layer/s, and an output layer. The input layer receives the data, the hidden layers process the data and the output layer sends the data outside the network. In the hidden layers, activation functions determine the output of the data based on the input of the data by learning the complex patterns and capturing relationships between the many variables.

In preparation for modelling the dataset was split into 80% training and 20% test. The data was then scaled using the StandardScaler function to get the data ready to be fed into the Neural Network model. In the model used for this project, two hidden layers were used because the dataset was large, and 61 units were used in these hidden layers because the number of units must be less than or equal to the number of variables, and 61 units gave the lowest MSE. Finally, the model was compiled and fit with 100 epochs and a batch size of 10. Each epoch is a complete pass through the training set, so a higher number of epochs were necessary to reach a low error, and 80 batches were used for each epoch.

From the training loss over epochs graph below, the training loss was minimised from over 0.825 to around 0.66, concluding that the model is indeed learning effectively since MSE is decreasing over time. Furthermore, the average K-fold cross validation MSE score with 5 folds was relatively low and in line with the other models at 0.706.



Cross-Validation MSE Scores: [0.7107596172390224, 0.7033375284070292, 0.7067536769833713, 0.7062760766539103, 0.701644751300968]
Average MSE: 0.7058

Saliency

Another metric we can use when modelling with NN is Saliency, which allows us to understand which features influence PTB Speed the most. Once again, due to the overall complexity of the model, it should only be used as a guide rather than a definite explanation as it cannot tell us why these features are important. Higher Salience means PTB speed is influenced more by that feature, and from the Salience Table containing the top 5 variables with the highest average saliency we can observe that RoundId has the highest salience at around 0.1 with a p-value of almost 0, making it statistically significant, with score and SeasonId_2021 not far behind.

RoundId was interesting and unexpected; nothing from the other models suggested that the round was very relevant to PTB speed. In the EDA section, a box-plot of RoundId against PTB Speed showed that PTB Speed had no discernable pattern or trend with respect to any specific round. Ultimately, it could be attributed to the downside of Neural Networks which is low interpretability; it does not tell us why certain variables are important. It could also be considered that there may have been some error when building the model, however as the other four of the top five highest average saliency are consistent with the results with the other models and can be explained very easily, so the possibility of a faulty model can be ruled out.

Otherwise, Score and SeasonId_2021 can be very easily explained as PTB Speed is likely to be slower when a team has more points compared to when a team has less points, as explained previously in the LASSO section of the report, and 2021 was the first full season after the six again rule was introduced, allowing for a more free-flowing game leading to faster PTB speeds.

Feature	Average Saliency	Regression Coefficient (scaled)	Regression p-values
RoundId	0.0968	0.0376	0.0
Score	0.0863	0.0809	0.0
Set_Type	0.0862	0.0546	0.0
SeasonId_2021	0.0726	0.1105	0.0
SeasonId_2022	0.0629	0.0848	0.0

Overall, the neural networks model provided a solid understanding of several important variables which mostly aligned with the results of the LASSO model. However, due to the low interpretability of the model, neural networks are not recommended as the ideal model for tabular data even though it has a low cross validation MSE as the activation functions within the hidden layer are extremely complex and difficult to understand.

Conclusion/ Recommendations

Model	MSE
OLS	0.736
LASSO	0.727
XGBoost	0.694
Random Forest	0.713
Neural Network	0.706

First, out of the 5 models that have chosen to implement, from our table of results, we find that our XGBoost algorithm gives the lowest 5 fold cross validation MSE, indicating that it is the best for predicting PTB speed. However, we find that the cross validation MSE of all of our models do not vary too much. Namely, our simple models that assume linearity (OLS and LASSO) did not perform too poorly compared to our more complex models which addressed this assumption. This could suggest that the relationships between our PTB speed and our predictors are likely approximately linear. This could be an interesting topic for future research, as we did not find the exact functional form of the relationship between each predictor and PTB speed. Potential models that can explore this include Generalised Additive Models (GAMs), Polynomial Regression or Spline Regression.

Moreover, let's discuss which features have been identified as most influential to PTB speed for each model. First, from our OLS and LASSO models, we find that tackle/set features such as 'PTB Contest', 'Raw Tackle Number' and 'Total Involved Tacklers' have the most impactful regression coefficients. Then, our XGBoost model found 'PossessionSecs' to be impactful, along with 'Zone Possession', which we were not able to accurately assess through our limited linear models. This is further supported by our Random Forest model, which found 'PTB contest', 'PossessionSecs' and 'Zone Possession' as its most influential factors. However, our Neural Network model provides slightly different results than the rest of our models, as 'RoundId' is considered to be the most influential. While this deviates from our other models, we do not ignore these findings, as there are still other features that overlap in significance with other models, such as 'Score', 'Set type' and 'SeasonId'. Potentially more research should be done on why this is the case.

Now that these influential features have been identified, let's explore how teams should approach these factors. For '**PTB Contest**', defensive teams should always try to tackle players into the ground when possible, as this will not only fatigue the offensive player, but force them to take time to get to their feet before playing the ball, slowing down their PTB and giving time for defences to get onside. This is especially the case if the offence won the tackle, as defensive players are usually behind the play as a result. Moreover, from '**Raw Tackle Numbe**r', defensive teams may wish to increase defensive intensity more early into a set, as we find that PTB tends to get faster as the tackle number increases (i.e. stifle offensive momentum early). Next, from '**Total involved tacklers**' we suggest that defensive teams try to use more than 1 tackler, as we find that this slows down PTB speed. We also found that going from 2 to 3 involved tacklers increases PTB duration the most (from boxplot in EDA), however, it should be noted that attempting to tackle with more players opens up passing opportunities, so caution is advised. From '**PossessionSecs**', we notice that PTB speed slows down as more gametime passes, likely due to fatigue. Management strategies through rotations and timeouts are likely to help with this, but fatigue is inevitable. Also, '**Half**' tells us that PTB tends to be faster in overtime periods, likely due to the extra effort for the Golden Point. Moreover, '**ZonePossession**' indicates that PTB speed is faster towards the middle of the field, compared to near end zones, as this area tends to be associated with yardage, where momentum and fast PTB is emphasised. So, when in this middle area, defensive players should rush quickly to the defensive line after a tackle, as the offence prioritises speed in this area.

Thus, through predictive modelling, we have successfully explored how PTB impacts NRL games, identifying which game factors most heavily influence its speed and highlighting the potential of data science in future decision making for NRL teams.

Appendix

1.

Checking For Missing Values:

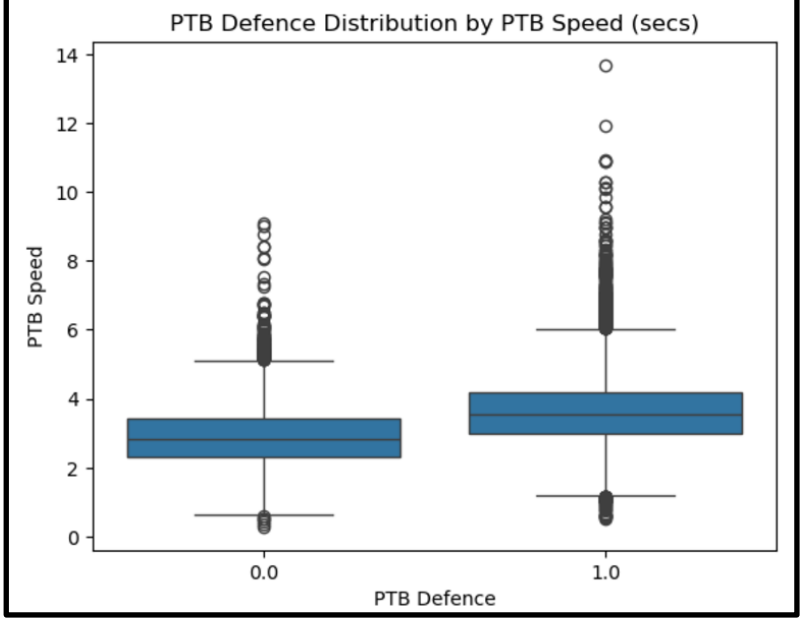
```
ptb_data = pd.read_csv('ptb_data.csv')
missing_values = ptb_data.isnull().sum()
features_with_missing_values = missing_values[missing_values > 0]
features_with_missing_values
```

```
Anonymize 1PlayerId           32
Total Involved Tacklers       36
PTB Defence                87992
Raw Tackle Number              1
dtype: int64
```

2.

Missing Values: Boxplot of PTB defence ¶

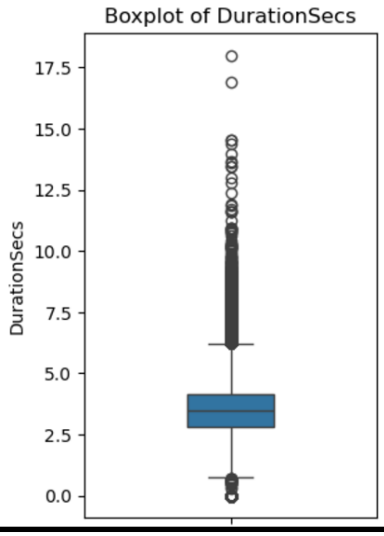
```
sns.boxplot(x='PTB Defence', y='DurationSecs', data=ptb_data)
plt.title('PTB Defence Distribution by PTB Speed (secs)')
plt.xlabel('PTB Defence')
plt.ylabel('PTB Speed')
plt.show()
```



3.

Outliers: PTB Speed ¶

```
plt.figure(figsize=(3, 5))
sns.boxplot(data=ptb_data, y='DurationSecs', width=0.3)
plt.title('Boxplot of DurationSecs')
plt.show()
```



4.

Removing Outliers in PTB Speed

```
Q1 = ptb_data['DurationSecs'].quantile(0.25)
Q3 = ptb_data['DurationSecs'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
ptb_data = ptb_data[(ptb_data['DurationSecs'] >= lower_bound) & (ptb_data['DurationSecs'] <= upper_bound)]
```

5.

Checking For Outliers:

```
outlier_counts = {}

for column in ptb_data.columns:
    if ptb_data[column].dtype in ['float64', 'int64']:  # Only check numeric columns

        # To Skip binary and ID variables
        unique_values = ptb_data[column].dropna().unique()
        if len(unique_values) == 2 and set(unique_values).issubset({0, 1}):
            continue
        elif 'Id' in column:
            continue

        Q1 = ptb_data[column].quantile(0.25)
        Q3 = ptb_data[column].quantile(0.75)
        IQR = Q3 - Q1

        # outlier bounds:
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR

        # Identifying outleirs:
        outliers = ptb_data[(ptb_data[column] < lower_bound) | (ptb_data[column] > upper_bound)]

        # Count of outliers
        num_outliers = outliers.shape[0]
        if num_outliers > 0:
            outlier_counts[column] = num_outliers  # Store the count in dict

# Print outlier counts for each feature
for col, count in outlier_counts.items():
    print(f"Outliers for {col}: {count}")
```

```
Outliers for Away Score: 1698
Outliers for DurationSecs: 2187
Outliers for Half: 117
Outliers for Home Score: 1021
Outliers for OppScore: 899
Outliers for Total Involved Tacklers: 42
Outliers for Score: 1068
Outliers for CurrentMargin: 7136
```

6.

Time variable Boxplot:

```
y = ptb_data['DurationSecs']
X = ptb_data[['Half']]

# Set up the figure with a grid of subplots
num_vars = X.shape[1]
fig, axes = plt.subplots(1, num_vars, figsize=(num_vars * 3, 5), sharey=True)

# Loop over each variable in X and create a scatter plot

sns.boxplot(x=X[column], y=y)
plt.title('Box Plots of Half Against PTB speed')
plt.show()
```

7.

Tackle/set feature Boxplots:

```
y = ptb_data['DurationSecs']

tackleSetFeatures = ['PTB Contest', 'Set Type', 'Raw Tackle Number', 'Tackle', 'Total Involved Tacklers', 'ZonePossession']

X = ptb_data[tackleSetFeatures]

# Set up the figure with a grid of subplots
num_vars = X.shape[1]
fig, axes = plt.subplots(1, num_vars, figsize=(num_vars * 3, 5), sharey=True)

# Loop over each variable in X and create a scatter plot
for i, column in enumerate(X.columns):
    sns.boxplot(x=X[column], y=y, ax=axes[i])
    axes[i].set_title(column)
    axes[i].set_xlabel(column)
    axes[i].tick_params(axis='x', rotation=45)

# Add a main title and adjust layout
fig.suptitle('Box Plots of Tackle/Set Variable Against PTB speed', y=1.0)
plt.show()
```

8.

Other Box/Scatter Plots

```
y = ptb_data['DurationSecs']

other_features = ['SeasonId', 'WeatherConditionName', 'IsHome']

# 'SeasonId', 'WeatherConditionName', 'IsHome',

X = ptb_data[other_features]

# Set up the figure with a grid of subplots
num_vars = X.shape[1]
fig, axes = plt.subplots(1, num_vars, figsize=(num_vars * 3, 5), sharey=True)

# Loop over each variable in X and create a scatter plot
for i, column in enumerate(X.columns):
    sns.boxplot(x=X[column], y=y, ax=axes[i])
    axes[i].set_title(column)
    axes[i].set_xlabel(column)
    axes[i].tick_params(axis='x', rotation=45)

# Add a main title and adjust layout
fig.suptitle('Scatter Plots of Each Time Variable Against PTB speed', y=1.05)
plt.show()
```

9.

JOINT F TESTS FOR CATEGORICAL FEATURES:

```
# Add a constant to the independent variables
X_full = sm.add_constant(X)
X_restricted = sm.add_constant(X.loc[:, ~X.columns.str.startswith('PTB Contest')])

# Fit the modelshow c
model_full = sm.OLS(df_encoded['DurationSecs'], X_full).fit()
model_restricted = sm.OLS(df_encoded['DurationSecs'], X_restricted).fit()

f_test = model_full.compare_f_test(model_restricted)

# Results
print("F-statistic:", f_test[0])
print("p-value:", f_test[1])
```

```
F-statistic: 6713.090087258544
p-value: 0.0
```

10.

Final data setup ¶

```
df_encoded = ptb_data.drop(columns = ['Anonymize 1PlayerId', 'Player Id', 'ZonePhysical', 'EventName', 'MatchId', 'Tackle',
                                      'PositionId', 'OppScore', 'Away Score', 'Home Score', 'PTB Tackle Result', 'OfficialId', 'OppPossessionSecs',
                                      'TotalPossessionSecs', 'ElapsedTime', 'PTB Ultimate Outcome', 'SeqNumber', 'Set', 'PTB Defence'])
# OppPossessionSecs  Official ID

df_encoded = pd.get_dummies(df_encoded, columns=['WeatherConditionName', 'Club Id', 'SeasonId', 'Opposition Id', 'Half', 'PTB Contest',
                                                 'Raw Tackle Number'], drop_first=True)


df_encoded = df_encoded.dropna()

df_encoded[df_encoded.select_dtypes(include=['bool']).columns] = df_encoded.select_dtypes(include=['bool']).astype(int)

df_encoded.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 109805 entries, 0 to 109804
Columns: 223 entries, DurationSecs to Raw Tackle Number_7.0
dtypes: float64(6), int32(211), int64(6)
memory usage: 91.4 MB
```

#	Column	Non-Null Count	Dtype
0	DurationSecs	109805 non-null	float64
1	PossessionSecs	109805 non-null	float64
2	Set Type	109805 non-null	int64
3	Total Involved Tacklers	109805 non-null	float64
4	RoundId	109805 non-null	int64
5	RunOn	109805 non-null	int64
6	Score	109805 non-null	float64
7	ZonePossession	109805 non-null	int64
8	GameTime	109805 non-null	float64
9	IsHome	109805 non-null	int64
10	CurrentMargin	109805 non-null	float64
11	WeatherConditionName_Rain	109805 non-null	int32
12	WeatherConditionName_Showers	109805 non-null	int32
13	WeatherConditionName_Snow	109805 non-null	int32
14	WeatherConditionName_Unknown	109805 non-null	int32
15	Club Id_1d6cd83892ee4afdcd8ccd94f817b4a6	109805 non-null	int32
16	Club Id_1d6cd83892ee4afdcd8ccd94f81ftnjhl3s	109805 non-null	int32
17	Club Id_367ef61d2bc259e608027a8d349c933e	109805 non-null	int32
18	Club Id_3b26834df063f9d51de216a07ec36929	109805 non-null	int32
19	Club Id_58485e3acf60682c8fc37d9d521b3019	109805 non-null	int32
20	Club Id_5e03a19f4d014a2220665cfd56522d35	109805 non-null	int32
21	Club Id_837e03d56b4dba3b8a4a5425c0420abd	109805 non-null	int32
22	Club Id_980c9c368ae4f1129ea0a6fdd711fa8f	109805 non-null	int32
23	Club Id_a73752d38e4a78e3e14917f5435ffb6d	109805 non-null	int32
24	Club Id_b53920c88e4eebf2faa9f4fb43b8944a	109805 non-null	int32
25	Club Id_c03196722c1a837b39f79f1714db475d	109805 non-null	int32
26	Club Id_c14e0139ad91a9741a5731a596aa6549	109805 non-null	int32
27	Club Id_d3ac47d424b41fd738ec9500dbda2d59	109805 non-null	int32
28	Club Id_dc3c7bd8148814b7c4105841baa68e23	109805 non-null	int32
29	Club Id_f38f7f087f646c38c0207f1b2af32f12	109805 non-null	int32
30	Club Id_fdfcde48e2cbf12cc4710a2644b86d85	109805 non-null	int32
31	SeasonId_2021	109805 non-null	int32
32	SeasonId_2022	109805 non-null	int32
33	SeasonId_2023	109805 non-null	int32
34	Opposition Id_1d6cd83892ee4afdcd8ccd94f817b4a6	109805 non-null	int32
35	Opposition Id_1d6cd83892ee4afdcd8ccd94f81ftnjhl3s	109805 non-null	int32
36	Opposition Id_367ef61d2bc259e608027a8d349c933e	109805 non-null	int32
37	Opposition Id_3b26834df063f9d51de216a07ec36929	109805 non-null	int32
38	Opposition Id_58485e3acf60682c8fc37d9d521b3019	109805 non-null	int32
39	Opposition Id_5e03a19f4d014a2220665cfd56522d35	109805 non-null	int32
40	Opposition Id_837e03d56b4dba3b8a4a5425c0420abd	109805 non-null	int32
41	Opposition Id_980c9c368ae4f1129ea0a6fdd711fa8f	109805 non-null	int32
42	Opposition Id_a73752d38e4a78e3e14917f5435ffb6d	109805 non-null	int32
43	Opposition Id_b53920c88e4eebf2faa9f4fb43b8944a	109805 non-null	int32
44	Opposition Id_c03196722c1a837b39f79f1714db475d	109805 non-null	int32
45	Opposition Id_c14e0139ad91a9741a5731a596aa6549	109805 non-null	int32
46	Opposition Id_d3ac47d424b41fd738ec9500dbda2d59	109805 non-null	int32
47	Opposition Id_dc3c7bd8148814b7c4105841baa68e23	109805 non-null	int32
48	Opposition Id_f38f7f087f646c38c0207f1b2af32f12	109805 non-null	int32
49	Opposition Id_fdfcde48e2cbf12cc4710a2644b86d85	109805 non-null	int32
50	Half_2	109805 non-null	int32
51	Half_3	109805 non-null	int32
52	Half_4	109805 non-null	int32
53	PTB Contest_Other	109805 non-null	int32
54	PTB Contest_Stays on feet	109805 non-null	int32
55	PTB Contest_Tackled to ground	109805 non-null	int32
56	Raw Tackle Number_2.0	109805 non-null	int32
57	Raw Tackle Number_3.0	109805 non-null	int32
58	Raw Tackle Number_4.0	109805 non-null	int32
59	Raw Tackle Number_5.0	109805 non-null	int32
60	Raw Tackle Number_6.0	109805 non-null	int32
61	Raw Tackle Number_7.0	109805 non-null	int32

dtypes: float64(6), int32(51), int64(5)
memory usage: 31.4 MB

11.

OLS

```
# RUNNING OLS WITH TRAIN TEST SPLIT
from sklearn.model_selection import train_test_split
import statsmodels.api as sm
from sklearn.metrics import mean_squared_error, r2_score

Y = df_encoded['DurationSecs']
X = df_encoded.drop(['DurationSecs'],axis = 1)
X = sm.add_constant(X)

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=37)

OLS_model = sm.OLS(Y, X).fit()

y_pred = OLS_model.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error: {mse}" )

r_squared = r2_score(y_test, y_pred)
print(f"R-squared: {r_squared}")
n = X_test.shape[0]  # Number of observations
p = X_test.shape[1]  # Number of features (predictors)
adjusted_r_squared = 1 - (1 - r_squared) * (n - 1) / (n - p - 1)
print(f"Adjusted_r_squared: {adjusted_r_squared}")

print(OLS_model.summary())
```

Mean Squared Error: 0.7240801151127758
R-squared: 0.2287697544764945
Adjusted_r_squared: 0.22658616349912408

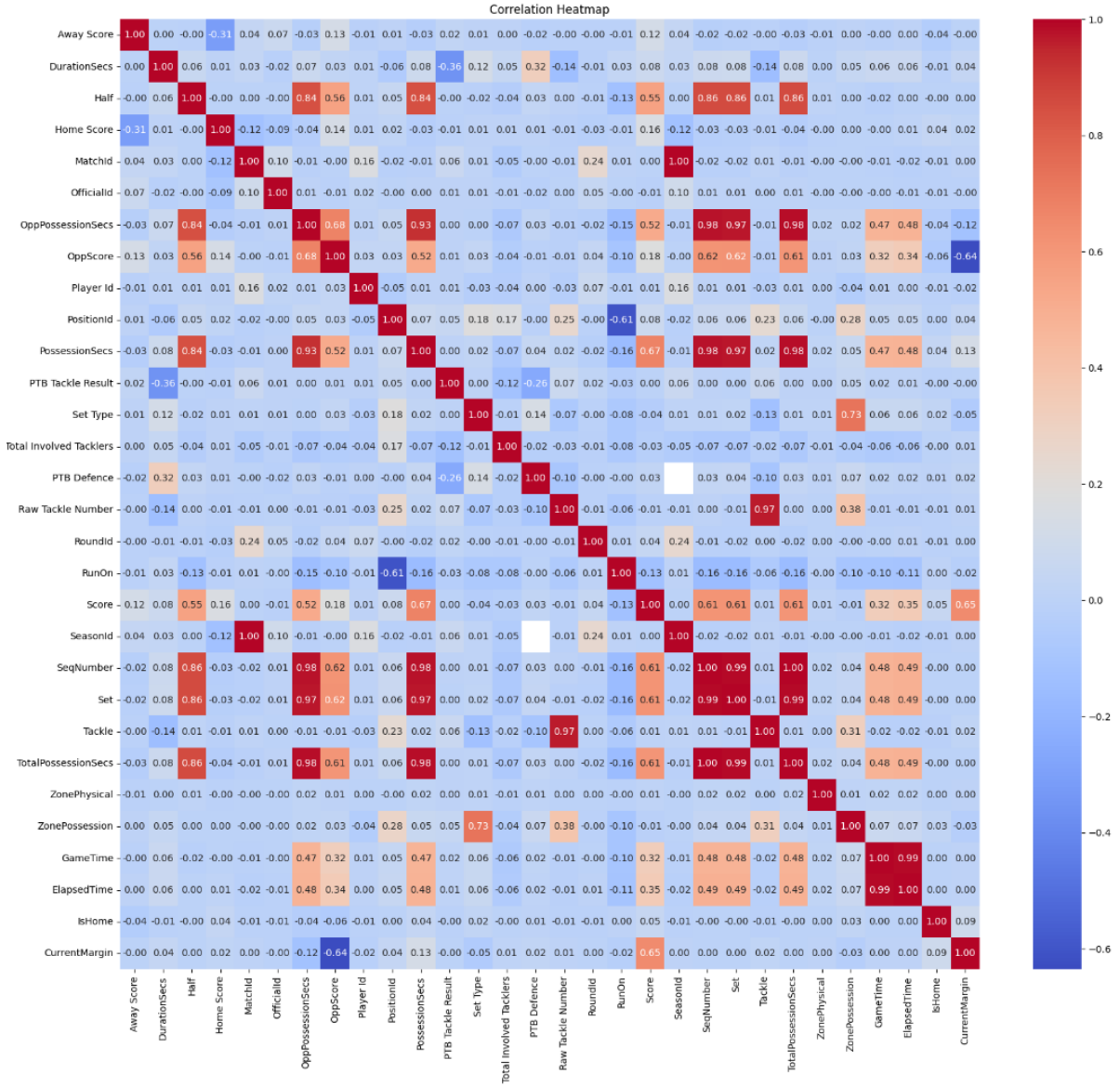
OLS Regression Results

Dep. Variable:	DurationSecs	R-squared:	0.220
Model:	OLS	Adj. R-squared:	0.220
Method:	Least Squares	F-statistic:	508.3
Date:	Wed, 20 Nov 2024	Prob (F-statistic):	0.00
Time:	17:03:55	Log-Likelihood:	-1.3793e+05
No. Observations:	109805	AIC:	2.760e+05
Df Residuals:	109743	BIC:	2.766e+05
Df Model:	61		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	3.5281	0.023	151.974	0.000	3.483	3.574
PossessionSecs	0.0001	3.19e-05	3.674	0.000	5.47e-05	0.000
Set Type	0.1720	0.010	16.542	0.000	0.152	0.192
Total Involved Tacklers	0.0745	0.004	20.269	0.000	0.067	0.082
RoundId	-0.0023	0.000	-6.153	0.000	-0.003	-0.002
RunOn	0.0996	0.007	14.287	0.000	0.086	0.113
Score	0.0027	0.001	3.717	0.000	0.001	0.004
ZonePossession	0.0041	0.000	15.793	0.000	0.004	0.005
GameTime	0.0011	0.001	1.352	0.176	-0.000	0.003
IsHome	-0.0268	0.005	-5.085	0.000	-0.037	-0.016
CurrentMargin	0.0008	0.000	1.526	0.127	-0.000	0.002
WeatherConditionName_Rain	0.0879	0.019	4.658	0.000	0.051	0.125
WeatherConditionName_Showers	0.0070	0.018	0.385	0.700	-0.029	0.043
WeatherConditionName_Snow	-0.0911	0.037	-2.438	0.015	-0.164	-0.018
WeatherConditionName_Unknown	0.0261	0.007	3.617	0.000	0.012	0.040
Club Id_1d6cd83892ee4afdcd8ccd94f817b4a6	0.0076	0.016	0.486	0.627	-0.023	0.038
Club Id_1d6cd83892ee4afdcd8ccd94f81ftnjhl3s	-0.0534	0.026	-2.081	0.037	-0.104	-0.003
Club Id_367ef61d2bc259e608027a8d349c933e	-0.1597	0.016	-10.198	0.000	-0.190	-0.129
Club Id_3b26834df063f9d51de216a07ec36929	0.0358	0.015	2.374	0.018	0.006	0.065
Club Id_58485e3acf60682c8fc37d9d521b3019	0.0521	0.016	3.261	0.001	0.021	0.083
Club Id_5e03a19f4d014a2220665cfd56522d35	-0.0184	0.016	-1.177	0.239	-0.049	0.012
Club Id_837e03d56b4dba3b8a4a5425c0420abd	-0.0468	0.015	-3.026	0.002	-0.077	-0.016
Club Id_980c9c368ae4f1129ea0a6fdd711fa8f	-0.0550	0.016	-3.530	0.000	-0.086	-0.024
Club Id_a73752d38e4a78e3e14917f5435ffb6d	0.0819	0.015	5.440	0.000	0.052	0.111
Club Id_b53920c88e4eebf2faa9f4fb43b8944a	0.0641	0.015	4.146	0.000	0.034	0.094
Club Id_c03196722c1a837b39f79f1714db475d	-0.0873	0.015	-5.682	0.000	-0.117	-0.057
Club Id_c14e0139ad91a9741a5731a596aa6549	0.0746	0.016	4.748	0.000	0.044	0.105
Club Id_d3ac47d424b41fd738ec9500dbda2d59	-0.0318	0.015	-2.082	0.037	-0.062	-0.002
Club Id_dc3c7bd8148814b7c4105841baa68e23	-0.0782	0.015	-5.104	0.000	-0.108	-0.048
Club Id_f38f7f087f646c38c0207f1b2af32f12	-0.1096	0.015	-7.121	0.000	-0.140	-0.079
Club Id_fdfcde48e2cbf12cc4710a2644b86d85	-0.0445	0.016	-2.792	0.005	-0.076	-0.013
SeasonId_2021	0.1948	0.008	25.263	0.000	0.180	0.210
SeasonId_2022	0.2479	0.008	29.362	0.000	0.231	0.264
SeasonId_2023	0.1887	0.008	22.807	0.000	0.172	0.205
Opposition Id_1d6cd83892ee4afdcd8ccd94f817b4a6	0.0037	0.016	0.240	0.810	-0.027	0.034
Opposition Id_1d6cd83892ee4afdcd8ccd94f81ftnjhl3s	-0.3064	0.025	-12.133	0.000	-0.356	-0.257
Opposition Id_367ef61d2bc259e608027a8d349c933e	-0.0656	0.016	-4.196	0.000	-0.096	-0.035
Opposition Id_3b26834df063f9d51de216a07ec36929	-0.1187	0.015	-7.815	0.000	-0.148	-0.089
Opposition Id_58485e3acf60682c8fc37d9d521b3019	-0.0368	0.016	-2.307	0.021	-0.068	-0.006
Opposition Id_5e03a19f4d014a2220665cfd56522d35	-0.0914	0.015	-5.912	0.000	-0.122	-0.061
Opposition Id_837e03d56b4dba3b8a4a5425c0420abd	-0.1262	0.015	-8.186	0.000	-0.156	-0.096
Opposition Id_980c9c368ae4f1129ea0a6fdd711fa8f	-0.0990	0.016	-6.367	0.000	-0.129	-0.069
Opposition Id_a73752d38e4a78e3e14917f5435ffb6d	-0.1264	0.015	-8.347	0.000	-0.156	-0.097
Opposition Id_b53920c88e4eebf2faa9f4fb43b8944a	-0.0301	0.015	-1.956	0.050	-0.060	5.8e-05
Opposition Id_c03196722c1a837b39f79f1714db475d	-0.1210	0.015	-7.817	0.000	-0.151	-0.091
Opposition Id_c14e0139ad91a9741a5731a596aa6549	0.0175	0.016	1.100	0.271	-0.014	0.049
Opposition Id_d3ac47d424b41fd738ec9500dbda2d59	-0.0783	0.016	-5.021	0.000	-0.109	-0.048
Opposition Id_dc3c7bd8148814b7c4105841baa68e23	-0.0102	0.015	-0.669	0.503	-0.040	0.020
Opposition Id_f38f7f087f646c38c0207f1b2af32f12	-0.0119	0.015	-0.770	0.441	-0.042	0.018
Opposition Id_fdfcde48e2cbf12cc4710a2644b86d85	-0.1000	0.016	-6.333	0.000	-0.131	-0.069
Half 2	-0.0159	0.031	-0.514	0.607	-0.077	0.045
Half_3	-0.2956	0.084	-3.528	0.000	-0.460	-0.131
Half_4	-0.2972	0.104	-2.858	0.004	-0.501	-0.093
PTB Contest_Other	-0.4691	0.347	-1.351	0.177	-1.149	0.211
PTB Contest_Stays on feet	-1.0728	0.012	-92.056	0.000	-1.096	-1.050
PTB Contest_Tackled to ground	-0.6907	0.005	-127.513	0.000	-0.701	-0.680
Raw Tackle Number_2.0	-0.2591	0.008	-34.012	0.000	-0.274	-0.244
Raw Tackle Number_3.0	-0.3450	0.008	-42.000	0.000	-0.361	-0.329
Raw Tackle Number_4.0	-0.3978	0.009	-43.155	0.000	-0.416	-0.380
Raw Tackle Number_5.0	-0.4529	0.011	-42.320	0.000	-0.474	-0.432
Raw Tackle Number_6.0	-0.3720	0.024	-15.421	0.000	-0.419	-0.325
Raw Tackle Number_7.0	-0.4225	0.207	-2.046	0.041	-0.827	-0.018

Omnibus:	780.427	Durbin-Watson:	1.848
Prob(Omnibus):	0.000	Jarque-Bera (JB):	811.132
Skew:	0.193	Prob(JB):	7.33e-177

12. Correlation matrix

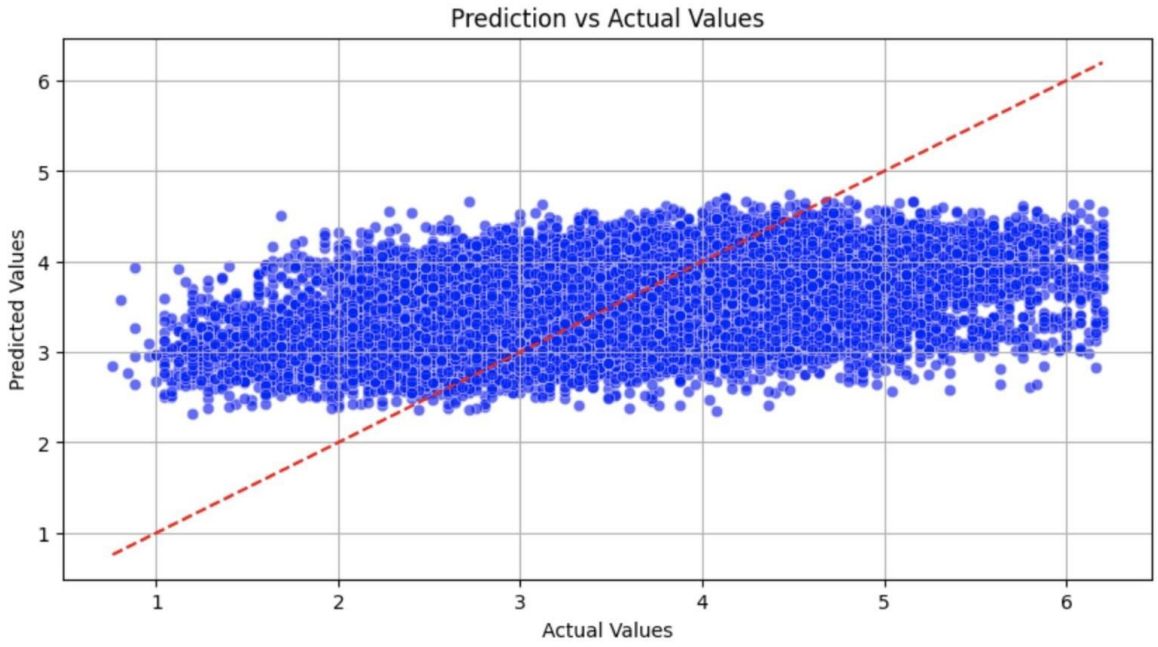


13. Results for LASSO Regression

	Coefficient
Intercept	3.726976
PossessionSecs	0.025040
Set_Type	0.177140
Total_Involved_Tacklers	0.052206
RoundId	-0.002044
RunOn	0.092324
Score	0.022605
ZonePossession	0.003772
GameTime	0.026926
IsHome	-0.020581
CurrentMargin	0.015904
WeatherConditionName_Rain	0.036077
WeatherConditionName_Showers	0.000000
WeatherConditionName_Snow	-0.000000
WeatherConditionName_Unknown	0.024976
Raw_Tackle_Number_2.0	-0.229501
Raw_Tackle_Number_3.0	-0.311402
Raw_Tackle_Number_4.0	-0.362677
Raw_Tackle_Number_5.0	-0.414903
Raw_Tackle_Number_6.0	-0.249407
Raw_Tackle_Number_7.0	-0.000000
Half_2	0.037055
Half_3	-0.000000
Half_4	-0.000000
PTB_Contest_Other	-0.000000
PTB_Contest_Stays on feet	-1.043039
PTB_Contest_Tackled to ground	-0.683620
Club_Id_1d6cd83892ee4afdcd8ccd94f817b4a6	0.014402
Club_Id_1d6cd83892ee4afdcd8ccd94f81ftnjhl3s	-0.000000
Club_Id_367ef61d2bc259e608027a8d349c933e	-0.114280
Club_Id_3b26834df063f9d51de216a07ec36929	0.048667
Club_Id_58485e3acf60682c8fc37d9d521b3019	0.060931
Club_Id_5e03a19f4d014a2220665cfd56522d35	0.000000
Club_Id_837e03d56b4dba3b8a4a5425c0420abd	-0.005530
Club_Id_980c9c368ae4f1129ea0a6fdd711fa8f	-0.008514
Club_Id_a73752d38e4a78e3e14917f5435ffb6d	0.096282
Club_Id_b53920c88e4eebf2faa9f4fb43b8944a	0.080931
Club_Id_c03196722c1a837b39f79f1714db475d	-0.044245
Club_Id_c14e0139ad91a9741a5731a596aa6549	0.086715
Club_Id_d3ac47d424b41fd738ec9500dbda2d59	-0.000000
Club_Id_dc3c7bd8148814b7c4105841baa68e23	-0.037801
Club_Id_f38f7f087f646c38c0207f1b2af32f12	-0.067807
Club_Id_fdfcde48e2cbf12cc4710a2644b86d85	-0.000000
Opposition_Id_1d6cd83892ee4afdcd8ccd94f817b4a6	0.041198
Opposition_Id_1d6cd83892ee4afdcd8ccd94f81ftnjhl3s	-0.168430
Opposition_Id_367ef61d2bc259e608027a8d349c933e	0.000000
Opposition_Id_3b26834df063f9d51de216a07ec36929	-0.037511
Opposition_Id_58485e3acf60682c8fc37d9d521b3019	0.000000
Opposition_Id_5e03a19f4d014a2220665cfd56522d35	-0.020747
Opposition_Id_837e03d56b4dba3b8a4a5425c0420abd	-0.053289
Opposition_Id_980c9c368ae4f1129ea0a6fdd711fa8f	-0.027555
Opposition_Id_a73752d38e4a78e3e14917f5435ffb6d	-0.051731
Opposition_Id_b53920c88e4eebf2faa9f4fb43b8944a	0.013425
Opposition_Id_c03196722c1a837b39f79f1714db475d	-0.047743
Opposition_Id_c14e0139ad91a9741a5731a596aa6549	0.049034
Opposition_Id_d3ac47d424b41fd738ec9500dbda2d59	-0.001286
Opposition_Id_dc3c7bd8148814b7c4105841baa68e23	0.032378
Opposition_Id_f38f7f087f646c38c0207f1b2af32f12	0.029888
Opposition_Id_fdfcde48e2cbf12cc4710a2644b86d85	-0.016040
SeasonId_2021	0.175553
SeasonId_2022	0.225726
SeasonId_2023	0.163216

14. Evaluation of the performance of LASSO Model

```
Test Error: 0.7265
R^2: 0.2261723721987955
Adjusted R^2: 0.2240
```



15. CV RMSE of the LASSO regression

```
[ ] from sklearn.model_selection import cross_val_score
    from sklearn.metrics import mean_squared_error
    import numpy as np

    # Perform 5-fold cross-validation
    cv_scores = cross_val_score(
        lasso,
        X_train,
        Y_train,
        cv=5,
        scoring='neg_mean_squared_error'
    )

    # Convert negative MSE to RMSE
    cv_rmse = np.sqrt(-cv_scores)

    # Calculate average RMSE across folds
    mean_cv_rmse = np.mean(cv_rmse)

    # Display results
    print("Cross-Validation RMSE for each fold:", cv_rmse)
    print(f"Average 5-Fold CV RMSE: {mean_cv_rmse}")
```

```
Cross-Validation RMSE for each fold: [0.84895677 0.85158854 0.85277328 0.85229755 0.84794807]
Average 5-Fold CV RMSE: 0.8507128412533381
```

16. Lambda of the LASSO regression

```
# Fit the LassoCV model
lasso = LassoCV(cv=5)
lasso.fit(X_train, np.ravel(Y_train))
print(f'LASSO Lambda: {lasso.alpha_}')
```

```
LASSO Lambda: 0.0009714873348748116
```

17. Standardisation of numerical variables (including their means and standard deviation)

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

X_train[num_var] = scaler.fit_transform(X_train[num_var])
X_test[num_var] = scaler.transform(X_test[num_var])
```

```
X_train[num_var].mean().head()
```

	0
PossessionSecs	2.788980e-16
Score	-1.058000e-16
CurrentMargin	1.294190e-18
Total_Involved_Tacklers	-8.946089e-17
GameTime	3.458723e-16

dtype: float64

```
X_train[num_var].std().head()
```

	0
PossessionSecs	1.000006
Score	1.000006
CurrentMargin	1.000006
Total_Involved_Tacklers	1.000006
GameTime	1.000006

dtype: float64

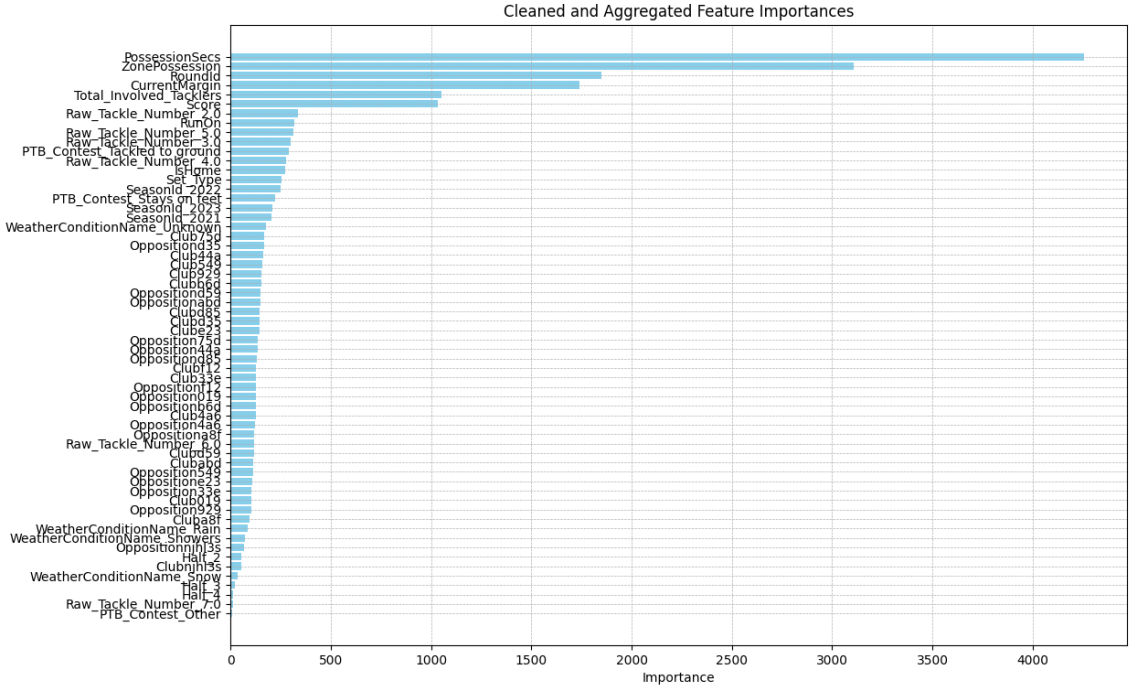
18. XGboost model preparation code

```
# Initialize and train XGBoost Model with grid search for basic hyperparameter tuning
model = xgb.XGBRegressor(objective='reg:squarederror', random_state=42)
parameters = {
    'n_estimators': [50, 100, 200],
    'max_depth': [3, 5, 7],
    'learning_rate': [0.05, 0.1, 0.2]
}
grid_search = GridSearchCV(estimator=model, param_grid=parameters, cv=3, scoring='neg_mean_squared_error')
grid_search.fit(X_train, y_train)
best_model = grid_search.best_estimator_
```

19. Code for XGboost Feature important analysis

```
# feature importance
importance = best_model.get_booster().get_score(importance_type='weight')
sorted_importance = sorted(importance.items(), key=lambda x: x[1], reverse=True)
for feature, importance in sorted_importance:
    print(f'{feature}: {importance}')
```

20. Results for XGboost



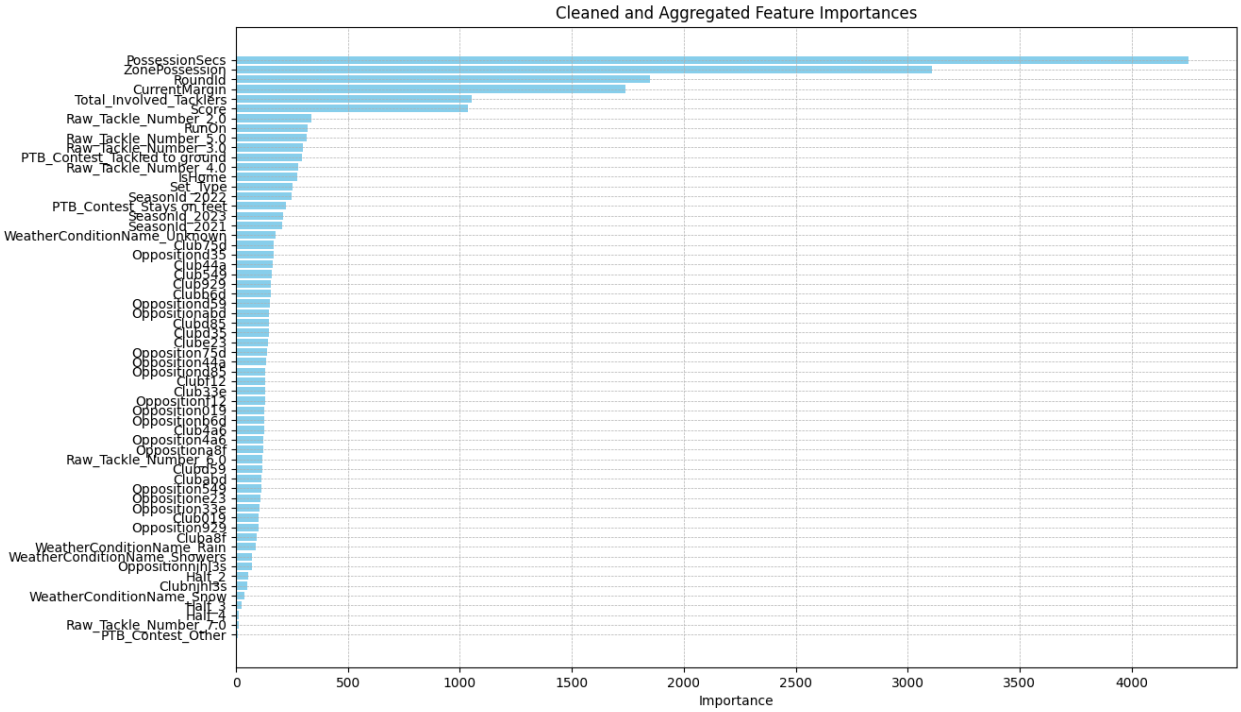
21. Code for Random Forest model building

```
# Setup the grid search
grid_search = GridSearchCV(pipeline, param_grid, cv=3, verbose=2, n_jobs=-1)

# Split the data into training and testing sets
X = df_encoded.drop(['DurationSecs'], axis=1, errors='ignore')  # Adjust the column names based on your needs
y = df_encoded['DurationSecs']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=37)

# Fit the grid search to the training data
grid_search.fit(X_train, y_train)

# Output the best parameters and best score
print("Best parameters found: ", grid_search.best_params_)
print("Best cross-validation score: {:.2f}".format(grid_search.best_score_))

# Evaluate on the test set
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

22. Code for Random Forest feature analysis

```
# Feature importances
importances = best_model.named_steps['regressor'].feature_importances_
indices = np.argsort(importances)[::-1]
features = X.columns

# Print the feature importances
print("Feature ranking:")
for i in range(len(features)):
    print(f"{i + 1}. feature {features[indices[i]]} ({importances[indices[i]]:.3f})")
```

23. Results from Random Forest



24. Building Neural Networks Model

```
model = Sequential()


model.add(Input(shape=(num_features,))) # the input layer where the shape of inputs is specified, for now it is just the number of features (more advanced inputs like imag
model.add(Dense(61, activation='relu'))
model.add(Dense(61, kernel_regularizer=regularizers.l2(0.001), activation='relu'))
model.add(Dense(1, activation='linear')) # the output layer has 1 unit, the linear activation is used as this is for regression
```

```
model.compile(loss='MSE', optimizer='adam')
```

```
modelout = model.fit(X_train, y_train, epochs=100, batch_size=10, validation_data=(X_test, y_test))
```

25. Building Saliency Tables

```
def compute_saliency(model, input_data):
    input_data = tf.convert_to_tensor(input_data)  # Ensure input data is a tensor
    with tf.GradientTape() as tape:
        tape.watch(input_data)
        prediction = model(input_data)
    # Get the gradients of the output with respect to the input
    saliency = tape.gradient(prediction, input_data)
    return saliency.numpy()

saliency_map = compute_saliency(model, X_train)

# Convert list to array and compute mean across samples
all_saliencies = np.array(saliency_map).squeeze()  # Remove extra dimensions if necessary
average_saliency = np.mean(all_saliencies, axis=0)

#compare with regression coefficients and add names of the features
feature_importance = pd.DataFrame({
    "Feature": nrl_train.columns[:-1],
    "Average Saliency": average_saliency,
    "Regression Coefficient (scaled)": lm_scaled.params[1:],
    "Regression p-values": lm_scaled.pvalues[1:].round(5)
})
feature_importance = feature_importance[feature_importance["Feature"] != "DurationSecs"]

feature_importance
```

```
top_features = feature_importance.sort_values(by="Average Saliency", ascending=False).head(5)

# Display top 5 most important features
top_features
```

References

[1] National Rugby League. (2022, April 28). *Arm wrestle returns to NRL but dangerous Sharks buck trend*. Retrieved from https://www.nrl.com/news/2022/04/28/arm-wrestle-returns-to-nrl-but-dangerous-sharks-buck-trend/

[2] GitHub. (n.d.). *NRL data set 4 - README.md*. UNSW Data3001 Repository. Retrieved November 18, 2024, from https://github.com/unsw-edu-au/data3001-data-nrl-4/blob/main/README.md

[3] GitHub. (n.d.). *NRL data set 5 - README.md*. UNSW Data3001 Repository. Retrieved November 18, 2024, from https://github.com/unsw-edu-au/data3001-data-nrl-5/blob/main/README.md

[4] Gabbett, T. J., Jenkins, D. G., & Abernethy, B. (2012). Influence of ball-in-play time on the activity profiles of rugby league match-play. *Journal of Strength and Conditioning Research, 26*(6), 1508–1516. https://doi.org/10.1519/JSC.0b013e318231a753