Expected Goals in Rocket League

Axel Bartsch
abartsch@oxy.edu
Occidental College

1 Introduction and Problem Context

In the last 10 years the use of advanced statistics for sports analysis has increased dramatically, for professional teams looking to gain an advantage over their opponents and for hardcore fans looking to improve their understanding of the sport, forecast the trajectory of their favorite team's performances and even improve their betting odds. The data analytics industry in sports has exploded in this time [8], making it another aspect of our lives where computer science has become vital to the success of any business or team that wants to compete at the highest level.

While they are currently much smaller than the global sports industry, E-sports are growing faster, and the financial incentives for winning are the same [23]. Yet data analysis and advanced statistics are barely used in E-sports, with teams and pundits relying on more traditional forms of analysis which are largely opinionated and can be heavily biased. I am a fan of both physical sports and E-sports, primarily watching soccer and rocket league. After a soccer game is finished I will often look at advanced statistics to get a better understanding of what happened, to see which players performed well, who 'deserved' to win and if what I remember from the game lines up with the statistically important events. The main statistic that I use for soccer is the expected goals metric (xG), which is used to quantify the probability that each shot taken in a match will result in a goal. Soccer is a game that heavily involves luck, and teams often win games based off of a single good shot from a mediocre chance despite being under pressure for most of the game [3]. The expected goals statistic shows which team deserved to win based off of the chances they created and what players performed well. When watching e-sports I often find myself disappointed that there aren't any statistics like this available. Rocket league is a very fast paced game, which can make it very hard to determine how good a chance was in real time. Additionally, a good scoring opportunity is much more dependant on the position of defenders than the distance from goal, so a simple stat like 'shots taken inside the box' which is used often in soccer is not very applicable to rocket league. As a result the majority of analysis on player performance is opinion and observation based with limited data available to provide depth or proof for an argument.

In an attempt to solve this problem, I will be creating an expected goals model for rocket league. As mentioned above, an expected goals model uses information about a shot, primarily distance and angle to goal, the type of shot and positioning of defenders, to determine the likelihood of that shot resulting in a goal. Despite what the name 'expected' goals may suggest, it is not a predictive statistic. Rather, xG is used to evaluate team and player performance after a game is over. The value in expected goals is the consistency it adds to analysis. If a team or player is scoring a lot of goals every game while generating few expected goals it means that they are on a hot (lucky) streak and their form will soon return to a more normal level [22]. While scoring above xG rates can be attribute to individual skill, it is very rare for a player to over-perform their expected goals season on season. xG models are built using data from thousands of shots taken over seasons of competition and machine learning algorithms which will be expanded upon in the next section. This means that the probability value placed on a shot for it to result in a goal is based on league average finishing ability. Our intuition would then tell us that the best strikers would consistently over-perform their expected goals as they are above average finishers. This however is not the case, and expected goals shows us that the skill of great strikers is consistently finding space to take shots from good positions, not their 'finishing ability' and the execution of said shots.

The consistency of expected goals is why I think it will translate so well to rocket league. The most important matches in competitive rocket league are played in best of seven series where by the end, even the most attentive viewer will only remember glimpses of early the games. As viewers we tend to interpret the quality of a chance as how close the ball was to going in the goal, whether it was a narrow miss or if the goalkeeper had to make a good save. Expected goals tells us how good the chance was not based on what the attacker did with the ball, but the position and scenario they took the shot in which makes for much more reproducible and applicable analysis [13].

This project will require extensive data collection and pre-processing before any analysis or machine learning takes place, which will make it much more difficult than a basic implementation of a machine learning model. I will be testing the effectiveness of multiple algorithms for the

model. Additionally there are numerous ways to keep expanding and perfecting the project if data processing does not take as long as expected to make the it more difficult if need be.

2 Technical Background

The vast majority of expected goals models use some combination of statistical analysis and machine learning algorithms to derive the most accurate model possible. Part of my project will to decide what algorithm will be the best for rocket league. As a result, I will be detailing some methods of prior work that will be part of my decision.

Four algorithms tested by Egels H.P.H in an expected goals modeling project are Logisitic Regression, Decision Tree, Random Forest and Ada Boost. These algorithms are outlined as described by Egels below [10].

- Logistic Regression: Estimation of the probability of occurrence of an event as a function of a relatively larger number of variables.
- Decision Tree: A classifier expressed as a recursive partition of the instance space.
- Random Forest: A set of multiple decision trees where the predicted class is determined from the model of the classes. Random forests are able to correct for the poor generalization of decisions trees.
- Ada Boost: Starts by fitting a classifier on the original data. Then, additional copies of the classifier are fitted on the data where the weights of incorrectly classified instances such that the new classifiers focus on more difficult cases. Decision trees can be used as the underlying classifier.

To make use of these algorithms, one of course needs a large data set to work with. In soccer the variables commonly used to define a shot distance and angle to goal, speed of the player when they take the shot, what kind of shot they took, how the ball came to them before the shot, the number of defenders between the shooter and the goal, and the positioning of the goalkeeper [4]. Many of these, such as type of shot and how the ball came to the shooter are categorical variables. For rocket league the important variables will be shot position defined by the distance and angle from goal, the speed and angle of the ball at the time of the shot (how the ball came to the shooter), speed of the shooter and the positioning of all players on the field as well as the direction and speed of their movement. In rocket league players on both teams, not just the defenders must be accounted for as a common tactic is for attackers to drive into defenders to bump them out of the way so that the shooter has a clear path to goal. Two shots that are otherwise equal will have very different probabilities of resulting in a goal if a defender is being bumped, or forced to avoid a bump.

Variables like how the ball came to the shooter will not be categorical in rocket league, as the position of the ball is constantly recorded in game data, contrary to soccer where an accurate numerical position of the ball is nearly impossible to consistently calculate with current technologies. Other potentially influential variables to include are the spin of the ball, a categorical variable of whether or not the shooter used a flip to shoot, which greatly increases the power they are able to generate and another categorical variable for bumps. The bump variable would likely determine if a defender had been bumped in a short period of time before and after the shot was taken, as it greatly impacts their ability to save a shot and is not represented clearly enough by player position data.

Due to the rarity of goals in soccer, one of the biggest challenges that expected goals models face is the major imbalance between shots and shots that result in goals in data sets. To counter this, a technique called under-sampling is commonly used, simply removing shot data points at random from the majority set, so that the data set is more balanced between shots and goals. An improvement to this technique which improves expected goals models is SMOTE, or Synthetic Minority Over-sampling Technique, which uses attributes of the minority class to create 'fake' or synthetic data points to over sample the minority data class and achieve better classifier performance in analysis [17].

The machine learning portion of this project will be implemented using the python Scikit-learn library also known as Sklearn. Sklearn is a machine learning library that has various classification, regression and clustering algorithms including logarithmic regression, random forests and gradient boost. The variety of features it includes will be very useful to this project, as I will be testing various algorithms to determine which is the most accurate for the expected goals model.

3 Prior Work

As I outlined in the introduction there has been limited work done on advanced data analysis in the world of esports. However the expected goals metric has become very popular in professional soccer and ice hockey in recent years, so I will also focus on those here. Since it has been popularised the majority of scientific work in the field is dedicated to improving the accuracy of the expected goals model, which involves a combination of technological advances and honing the variables a algorithms used to generate the model. In their 2021 project, Anzer and Bauer utilized an advanced synchronization algorithm to improve the effectiveness of their tracking data. An issue that most expected goals models face is that event data, which in this case are player actions classified as shots, is still manually recorded as it is very difficult for current camera tracking

technology to accurately follow the ball. As a result event data and player tracking data are de-synchronized making tracking data very difficult to use. With their synchronization algorithm they were able to include variables in the machine learning algorithm such as 'Speed of player taking the shot', 'Defenders in line of the shot' and 'Pressure on the player taking the shot' which is a combination of metrics to give a numerical value to the defensive pressure on the player. The resulting expected goals model has a ranked probability score (RPS) of 0.197 which is "more accurate than any model previously published" [4].

The success of a model with more advanced player tracking data than previously implemented is important because it shows the high potential for success that an expected goals model in rocket league has. Since it is a video game, data such as the position and speed of all players on the field, the position and speed of the ball as well as other variables to do with game state are all known by the game engine at all times and recorded in reply files. With this intrinsic accuracy in the data, if implemented properly the rocket league xG model could even show what levels of accuracy are possible for real world sports with the improvement of data collection technology.

There is one existing expected goals model in rocket league that was created in 2019 using a neural net to determine the quality of chances. While the neural net does have its strengths for analysis, it has the draw back of the user not having complete control over what attributes are being focused on, which is why the more established expected goals models in soccer use other methods. The neural net was fed the position and velocity of the ball and all players as well as the time remaining in the game a couple of frames (.1-.2s) before the shot was taken [1]. This method also struggles with the imbalance in data between shots and goals and as a result, predicted roughly 1.2 goals for every actual goal that was scored in the data set.

While this project certainly was not as robust as much of the work from the world of soccer that I have referenced, it is the only existing model in rocket league, and will therefore provide an important benchmark for comparison of the model I create, even if the methodologies are very different. One very important tool that this model make use of, is its own algorithm to determine what is a shot and what isn't, as the game engine is notoriously poor a this especially for the high speed shots that often occur in professional play. While an implementation of such an algorithm could be a difficult addition to the project, similarly to the synchronization algorithm in Anzer and Bauer's work it could make a major difference for the accuracy of the model.

4 Methods

The goal of this project is to create a functional expected goals model in rocket league. The first step to achieving this is collecting enough data to properly train the model. For my data I used publicly available replays from professional level games on ballchasing.com. Ballchasing is a popular website in the rocket league community, where players of all levels upload replays of their own games for the websites analysis features and for public viewing. It also has a large catalogue of replays from professional competition which is what I focused on for this project. I will expand on the decision to focus on professional game play in the ethical considerations section but in short, the value of expected goals is the analysis of player and team effectiveness on the field which lends it to the professional scene where there are high stakes and high rewards for improving efficiency.

A challenge that is unique to this project compared to creating an expected goals model for soccer is that there are no publicly available data sets with the detailed shot information required to train the model. This is simply because such kinds of data analysis are not yet as popular in the world of video games. To gather shooting data I manually downloaded replay files and parsed them into json files using the carball plugin [20]. The raw replay files from a single game contain data on the ball and player positions roughly every 10th of a second. The replay analysis tool of carball sorts the replay by frame, with key information about every touch of the ball. This included some but not all of the information needed for the model features, so I used both the analysed and raw files to create a data set. The analysed features include location of the touch, distance from the goal and information regarding the previous and subsequent touch. The raw replays were used to gather information about the other players on the field besides the one who took the shot. To compile the replays into json files and generate the data set I wrote a python script. I chose to write in python to stay consistent with the machine learning later in the process and because carball is also written in python. I decided not to implement my own algorithm to determine what a shot is and what isn't. As mentioned in the previous section, the rocket league engine is not perfect at determining this and such an algorithm would certainly improve the accuracy of the model. However as I became more familiar with the replay files and the architecture of full game analysis tools like carball, it was clear that this was outside the scope of what was possible for this project.

The expected goals models that I have reviewed in preparation for this project have used anywhere between 1000 [2] and 100,000 [4] to train their machine learning algorithms. My pessimistic goal before starting was to have a data set of around 5000 shots, which is a realistic number based on replay availability and potential limitations in

my data processing, but still enough to sufficiently train the model. I was able to generate a final data set of 16,333 shots which exceeded this goal and is competitive with many of the models that I discussed in the prior work section. The limiting factor in the end was the time it took my script to convert replay files into the analysed and raw json files and the time to generate the data set from these files. I used a combination of replays from the past three seasons of professional rocket league, being rlcs season X, season 20-21 and 22-23. There is enough data between these seasons to create a much larger data set which is a potential future expansion for this project.

With the data collected and processed the next step was be to start creating the model. The core data variables used were:

- Distance to Goal: Calculated using x, y and z coordinates of the shot position
- Angle to Goal: Angle created between the location of the shot and the two goal posts. This is preferred in expected goals models to the angle created between the center of the goal and the ball because it is a better representation of the angle seen by the shooter, and the options they have of where in the goal to aim.
- Height of the ball. This is not a common variable in models for soccer and hockey as they are played on a largely two dimensional plane. The ball does go in the air a lot in soccer, but it is difficult to accurately track height with current technology. This is not an issue in rocket league and height is a very important feature of a shot.
- Shot type. This variable describes how the ball came to the shooter, whether they received a pass, were dribbling before shooting, or intercepted the ball. In addition it defines if the player was in the air or on the ground as there is a major difference in how they are able to hit the ball. This is a categorical variable, with the four categories mentioned as well as the combination between 'aerial' and 'pass' and 'aerial' and 'dribble' as both are possible simultaneously.
- Number of Defenders between the shooter and the goal. This is the number of defenders in front of the shooter, defined as the rectangle created by drawing a line from the shot location straight forward to the end line and a line from the far goal post to the y coordinate of the shot (y being the end to end axis of the field and x the side to side). While a drawing a cone from the shot location to frame of the goal would intuitively be more accurate, cars in rocket league move very fast compared to the dimensions of the field and it was important to capture a larger section of the field for this feature to be effective. The z coordinate of the shot and the defenders are not taken into account for similar reasons, as cars can cover the entire height of the field very quickly especially when coming from above so any player in front of the shooter is considered a defender.
- Under Pressure is a categorical variable that determines whether the shooter was under direct pressure from a defender. The defender could be applying pressure from any direction which is what makes this feature different from and useful along side number of defenders. Additionally defenders could all be near the goal line or close to the shooter which the number of defenders variable does nothing to describe. The shooter is determined to be under pressure if there is a defender within 150 in game units from them, which is one car length. I initially experimented with making this 300 units but it was far too common for a player to be under pressure for the feature to have any impact on the model.

There are a number of features that I considered or planned to use before starting work on the model and they are included below along with reasons for excluding them from the final model.

- Player position and velocity: All players except the shooter. This will be split into 5 variables, one for each player. This feature was attractive because this data is very difficult to obtain in real world sports due to the limitations in tracking technology outline earlier. Since we have the exact position of all players on the field in rocket league I thought it would make more sense to use that instead of calculating the number of defenders in the play. However as I learned more about machine learning and this project, it became clear that feeding a model a bunch of x, y and z coordinates didn't provide enough context of the situation and a simple variable of the number of defenders in the way was much more efficient.
- Shooter velocity: This and the three following variables would have been much more difficult to calculate than initially expected and could be included in a future expansion of the project. I determined them to be out of the scope of this project, especially as they were expected to be minor features in the model.
- Player boost amount: Split into 6 variables to record all player's boost amount at the time of the shot. Boost greatly affects a players speed and range of motion.
- Flip: Categorical variable if shooter used a flip to shoot. Using a flip increases potential power and angle change of the shot.
- Bump: Categorical variable if any defender was bumped or demolished in a short time frame before and after the shot. This could be updated to a numerical value of the number of players bumped, or even a value of how much defenders were affected by

the bump (either distance bumped or time taken to recover). However this would be a complicated change to implement and would likely have little impact on the overall model as it is specific to a niche type of shot.

The data set was be divided in a typical 80-20% split between training and testing data sets, then split again with the same ratio for a training, validation and testing set. As will be expanded upon in the evaluation section, an important aspect of analysing an expected goals model is testing how well it reproduces reality when predicting the total number of goals scored over a large sample size. Because of this it was important to keep a section of the data (the testing set) separate from all of the model training to be sure that the model is not over-fitting on training data.

An important aspect of this project was to determine the best algorithm to use for the model. As outlined in the technical background section, common algorithms used in expected goals models are Logistic Regression, Decision Tree, Random Forest and Ada-Boost and Gradient Boosting[10]. In the study performed by Egels, Random Forest performed the best, however Anzer and Bauer also tested different algorithms with their model and a customized extreme gradient boost performed the best there [4]. From these contrasting results and other works it is clear that the algorithms should be tested for each model and there is no distinct favorite. While much of my decision making on what algorithms to use was based on prior work, the five algorithms I chose to focus on were Logistic Regression, Support Vector Machine, Random Forest, Ada-Boost and Gradient Boost. I chose not to use a decision tree algorithm because in all prior works evaluated it performed worse than the random forest algorithm which has very similar, but more complex structure. I addition Ada-boost is an ensemble model and I used a decision tree as the base estimator, so the algorithm still has representation in the project. I replaced it with a support vector machine algorithm, which on the face of it, is not a great fit for the problem. This is because probability prediction is not automatically built in for support vector classifiers and it performs better with high dimensional data [11] which my data is not. My decision to include it was therefore based curiosity and an attempt to test something different from other expected goals models, even if it was not expected to be the best performing algorithm.

With the models selected, a large part of the project went into training algorithms, evaluating results then tuning the models to improve results. The evaluation of the models will be discussed in the next section. One of the difficulties with creating an accurate expected goals model is dealing with the class imbalance between the goals and shots in the data set. As mentioned earlier, the common way to handle such imbalance is with some combination of over and or under-sampling. As the number of goals in the data and overall size of the data set were potential limiting factors of model success, I chose to stick with oversampling, and implement SMOTE. The default way to implement is to create enough synthetic samples that the minority and majority classes have an equal number of data points. As will be discussed further in the results, this can result in the model vastly over-compensating when it comes to making predictions on the validation or testing set, which remain imbalanced, as they are the real data. Thus an important aspect of tuning models when working with oversampling is finding the correct rate to over-sample the minority class at.

If at this point my model has similar levels of accuracy it will be complete and I will move onto the best ways to visualize and present data so that it can be beneficial for public use. If it is not, which is the most likely outcome for the first few (or many times) running it, I will return to my variables to see what details can be added or are unnecessary, and tweak parameters of the machine learning algorithm.

5 Evaluation Metrics

Compared to many other machine learning models, which have many common and very relevant evaluation metrics, the quality of an expected goals can be difficult to quantify. This is because an xG model lies somewhere in between a classification and a regression problem. The intended result is the probability of a chance resulting in a goal, which is a regression result. However the ground truth data that this result is being compared to is the binary classification of goal (1) or no goal (0). As a result models are typically evaluated using typical classification evaluation metrics, such as precision, recall and auc score. The results of these metrics are important, but must be evaluated with the context that the primary goal of the model is not to accurately classify a individual shot as a goal or miss, but to produce an accurate probability of the chance resulting in a goal. Thus, the most important form of evaluation for an expected goals model is to compare it to real world results, as that is what we are trying to replicate. For a single game the xG values may be very different than the actual game score as there is a large element of luck and randomness in each result. Players will execute some shots better than others, and while some of this has to do with characteristics of the shot that we can capture in variables like number of defenders, there is still a lot of un-captured randomness in each shot[22]. However over an entire data set the model should output a very similar value of expected goals to the total number of goals in the data set, meaning that the sum of all the predicted probabilities should be very similar to the total number of goals scored. For example after completing their model, Anzer and Bauer tested its accuracy on a small data set of the first 54 matches of the 2020/21 Bundesliga season. 150 goals were scored in the 54 games, and

their model returned a value of 151.6 expected goals [4].

In terms of typical metrics to evaluate the model there is some disagreement in the literature. As mentioned the basic metrics of precision, recall and auc are commonly used, but are limited in the use for model evaluation. Anzer and Bauer propose Ranked Probability Score (RPS) as an improvement on typical metrics because it is a measure of how good forecasts are at matching the observed outcomes [9]. Ranked Probability Score was developed for evaluating weather prediction models and specializes in comparing such probabilities to actual outcomes. While their implementation of the statistic makes it seem to be a very attractive metric for evaluation of an expected goals model, I chose not to use RPS because this was the only work that used the metric in this context. While Anzer and Bauer did create an industry leading expected goals model, making it a very important source for this project, my implementation of RPS would only be useful in comparison to their model. In creating their expected goals model, Cavus and Biecek set out to improve the metrics used to evaluate xG models and simultaneously compare their results to other popular models. They agree that precision and recall are not all that valuable to an expected goals model, and propose the use of The Matthews correlation coefficient (MCC), Brier Score and Log Loss due to their functionality with imbalanced data. The Matthews correlation coefficient is similar to precision and recall, but it only returns a good result (the closer to 1 the better) the model performs well in all four confusion matrix categories [7]. This makes it especially good for this project as an expected goals model can have a solid precision score by accurately classifying most non goals correctly. Brier score is essentially an adaptation of mean squared error for probability forecast [24] which makes it perfect for evaluating expected goals models. Finally log loss is a common evaluation metric for binary classification problems as it compares the predicted the probability to the actual result, and penalizes 'confidently incorrect' predictions [21]. I will be using the three metrics outlined above along with precision, recall, auc and the sum of predictions to evaluate my expected goals models.

6 Results and Discussion

The 16233 shots in my data set were split into a training, validation and testing set so that algorithms could be trained on the data and evaluated on the validation data, without ever over-fitting on the test set, which the final best performing algorithms would be evaluated on.The initial results of each algorithm trained on the raw training set are shown below. It is important to note that I did end up training each model as a classifier which can be seen in the first column. This makes these binary classification metrics more applicable, and additionally standardized the predicted probability results, as there were some algorithms that produced results slightly below zero or above one when using the regressor versions of the models. In the sklearn libraries, are built the same way, and the probability values that are used as expected goal values are accessed using the models predict-proba() method, which returns the same values that the regressor predicts.

Algorithm	Precision	Recall	AUC	MCC	Log Loss	Brier Score	Sum of xG
LogisticRegression	0.59	0.10	0.75	0.18	0.45	0.14	548.22
GradientBoostingClassifier	0.62	0.19	0.76	0.26	0.44	0.14	553.50
AdaBoostClassifier	0.55	0.26	0.74	0.28	0.66	0.23	1244.34
SVC	0.60	0.07	0.68	0.16	0.49	0.16	553.08
RandomForestClassifier	0.63	0.36	0.77	0.38	0.66	0.13	566.73

Figure 1: Model Results with Raw Data

We can see from the precision and recall scores that the gradient boosting, ada-boost and random forest algorithms are the best performers, with Random Forest producing both the best precision and recall scores of the whole group. The auc scores are comparable for every algorithm and while they are respectable values, they do as much to show that the auc is not a very effective metric for this model, as they demonstrate any of the algorithms accuracy. The Matthew's correlation coefficient back up the the precision and recall scores that random forest is the best model, followed by ada-boost and gradient boost, although none of the scores are particularly high. The log loss scores paint a different picture with random forest and ada-boost scoring the worst. The brier scores are strong for all models except adaptive boost. Finally the sum of xG scores show that most of the models are performing better than one would expect based on the other metrics. There were a total of 549 goals scored in the validation set, meaning that asides from ada-boost which performs very poorly on this metric, the furthest model was only 17 away from the validation set, and the others were within 5 goals of the target. This metric can be somewhat misleading at this level however, as it is possible for the models to overfit on the training and validation sets as they were repeatedly trained and tuned. The quality of performance on this metric, even from algorithms with poor scores otherwise like logistic regression does demonstrate the importance of quality and quantity of data for this project. For much of the semester when I was working without the number of defenders, shot type and under pressure features, models trained on just distance from goal, height and angle to goal were capable of reproducing similar results on other metrics across the board, but were never universally close to the target number of goals.

With solid results from the raw data, I retrained the algorithms using over-sampled data in an attempt to improve performance. After training with different ratios of oversampling, the best ratio was 0.3, which is the lowest accepted by the SMOTE algorithm. Which only increased the minority class from 2183 samples in the training set to 2461.

The results of the training are shown below.

Algorithm	Precision	Recall	AUC	MCC	Log Loss	Brier Score	Sum of xG
LogisticRegression	0.58	0.15	0.75	0.22	0.45	0.14	594.54
GradientBoostingClassifier	0.59	0.25	0.76	0.29	0.44	0.14	596.88
AdaBoostClassifier	0.51	0.28	0.74	0.27	0.66	0.23	1234.34
SVC	0.59	0.12	0.69	0.19	0.49	0.15	600.77
RandomForestClassifier	0.63	0.39	0.78	0.39	0.60	0.13	595.80

Figure 2: Model Results with Over-sampled Data

The improvements in model performance were very minimal when using SMOTE. When a higher ratio of over-sampling was used, there was a trade off in precision and recall scores while auc remained largely the same. This follows intuition as the over-sampling in training data results in a higher classification rate of goals which meant a higher true positive rate and an increase in recall score. Conversely this also meant a major increase in the false positive rate, which lead to a decrease in precision score. As mentioned this trade off meant that auc scores where largely unaffected by the ratio of over-sampling or the inclusion of SMOTE at all. The Matthew's correlation coefficient scores increased across the board with a higher over-sampling ratio as more balanced data meant more balance between the four confusion matrix metrics. Log loss and Brier score also increased across the board with higher levels of over-sampling which is a negative indicator of model performance. Most importantly, over-sampling meant that the sum of xG increased drastically making all of the models much more inaccurate according to this metric.

As a result of the negligible increase in model performance according to the typical evaluation metrics and the drastic negative impact on the sum of xG, it was determined that minority sampling was not beneficial to model performance, and would not be used with the final selected model. This is somewhat counter-intuitive, as a machine learning model with a noticeable class imbalance should benefit from some form of over or under-sampling. This is an area for future improvement of this model and this project, as it is likely due to my implementation of the over-sampling that it had a negative impact.

Across all metrics, with and without over-sampling the gradient boosting and random forest algorithms consistently outperformed the others. Based on results from prior works this is an expected result, as an extreme gradient boosting algorithm was the best performer in Anzer and Bauer's model [4] and the random forest algorithm was the best performer in Cavus and Biecek's work [16]. This follows intuition as well, as ensemble models should perform better than solitary classifiers when on even footing. The poor performance of ada-boost, particularly in the sum of xG metric was surprising and again is likely down to my implementation and potentially the decision tree base model.

As mentioned previously, the best performing algorithms were then further evaluated on the testing set, with a focus on the sum of xG metric, as this is the statistic that would suffer the most from over-fitting, as well as being the most important to the real world application of the model. The total number of goals scored in the testing set was 723. The random forest algorithm returned a relatively strong score of 714.5 while the gradient boosting algorithm return a weaker score of 675.8. Interestingly, when evaluated on the testing set, both algorithms performed better when trained using over-sampled data, once again with a ratio of 0.3. The random forest algorithm with over-sampling returned 731.5 goals while the gradient boosting algorithm improved massively with a result of 730.4. These results show that while not competitive with industry leaders in professional soccer, my expected goals model is accurate enough to be a valuable statistic in evaluating individual and team performance in professional rocket league.

With the two algorithms selected for their performance, create graphs showing the Shapley values of feature importance for the models. Shapley values are metric adapted from cooperative game theory to show the magnitude of importance each feature played in the model's performance [5]. The graph below shows the impact on the model output based on feature value for the gradient boosting algorithm. As expected distance from goal and height are the most important features, with higher values correlating with a lower expected goals value, This follows the intuition which expected goals models are built off of, that one has a higher chance of scoring the closer they are to the goal.

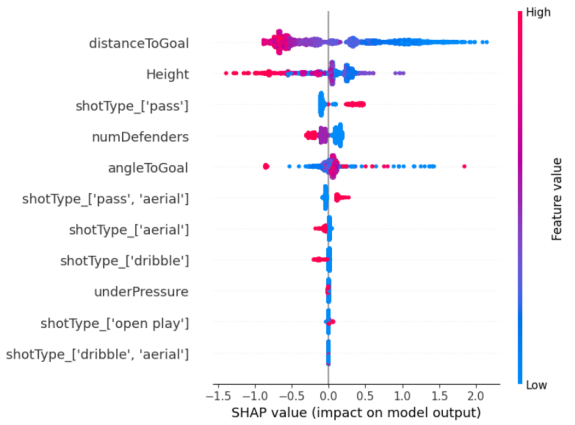

Figure 3: Gradient Boost Shapley Values

The graph below shows the magnitude of impact that each feature has on the model, once again for the gradient boosting algorithm. We can see that distance to goal is by far the most important feature of the model. It is somewhat surprising to see that angle to goal is rated lower than the shot type "pass" and number of defenders, as this is widely considered secondary only to distance from goal in most models for soccer. Fans of professional rocket league will

appreciate the importance of shooting after receiving a pass, as these are difficult plays to execute in high pressure scenarios, but when done properly can generate shot power that is not possible in most other scenarios.

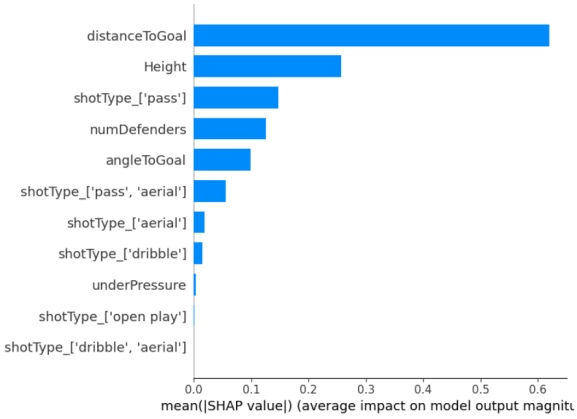

Figure 4: Gradient Boost Feature Magnitude

The final graph shows the same magnitude of feature importance for the random forest algorithm, this time denoting the difference for each class, goal and no goal. As this is only a binary class separation, there is no difference in magnitude of feature between classes.

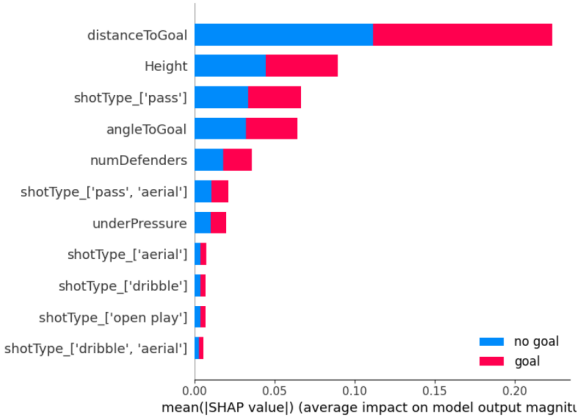

Figure 5: Random Forest Feature Magnitude

7 Ethical Considerations

The ethical considerations for this project can be split into two main categories. The first being the potential ethical issues that go along with any use of machine learning for analysis and the second being problems that could arise with publishing the model and making it available for any player's usage in game.

7.1 General Concerns of Machine Learning

The explosion of the use of artificial intelligence and machine learning in the last decade has been followed by an equally large influx of ethical issues to do with the biases that are internalized by computer algorithms. These biases reproduce and sometimes expand on real world discrimination and stereotypes. Some famous examples of such recreation of bias are reports in 2015 that the top results of a google search for "CEO" returned pictures of exclusively white men. Also in 2015 Flickr's image recognition tool would tag black people in pictures as apes [25]. These biases are still present today in many forms of artificial intelligence that are used by millions of people. While much attention has been given to these issues, as Safiay Noble discusses in her book Algorithms of Oppression, when companies like Google are questioned on the bias in their algorithms it is often claimed that they do not have control over what the AI learns and outputs [18].

7.2 Data Collection & Project Specifics

The data analysis and use of machine learning in my project will not involve any player demographics that could reproduce structural discrimination. However it is important to recognize this aspect of machine learning and the potential issues that go along with it. The biases output by an algorithm are based on the data set it is trained on. To create the expected goals model I will be using data from replay files uploaded to ballchasing.com. Ballchasing is a popular website in the rocket league community where players upload their own gameplay for public viewing and the statistical analysis that ballchasing offers its users [6]. As a result there are no potential issues with consent for players use of data, as it is all taken from a public platform where individuals are choosing to make their data available. It does however raise a potential for data bias in that professional players who upload most or all of their replays will be over represented in the data set. The aim of an expected goals model is to calculate the quality of a chance based on the average shooting ability of all players so this bias could impede the validity of the model. The best solution to this issue will be to use only replay files from official professional tournaments, which I will do if there is enough data from these matches available.

In addition to minimizing data collection bias, when working with machine learning it is important to maintain algorithm transparency to mitigate biases that may occur in training the algorithm [12]. The expected goals model will be a deterministic algorithm, there are few controlled inputs and one output every time, which limits the opportunities for bias to appear in data training.

7.3 Public Use of Expected Goals

There are a couple of aspects of accessibility of the expected goals metric that should be considered. While the statistic can be explained in simple terms and certainly can be understood by anyone, a detailed understanding of the model requires a certain level of education or experience with mathematical concepts. Many fans of e-sports, and a large part of the player base are young and may not have reached that level of education yet, and of course many older users will have not gone to college or have little to no memory of their stats class if they took one. While none of these factors exclude a person from enjoying the analysis of expected goals, it is important to remember that the model should be publicized in a user friendly way, and be welcoming to people at all levels of math experience.

The second thing to consider is that the model is based on and made for professional gameplay which means that it is not accessible to 99% of the player base to evaluate their own gameplay[15]. There are some good reasons for this, mainly being data availability and practical uses of the model are focused on the analysis of pro play. In addition lower skill games have a much larger factor of randomness as players are not able to control the ball as consistently. For this scope of this project this randomness will make it much more difficult to train the model accurately. In addition to a focus on the pro level, this model will be made for the 3v3 game mode as that is what the vast majority of tournaments are held in. However for the general player base the 2v2 and 1v1 modes are equally popular, which once again makes the model less accessible to most players own game play.

It can be argued that expected goals being accessible for the average player's games is not important. If we compare to real world sports, such in depth statistics are only used for top level professionals but are still used by many fans to supplement their understanding of the sports they love watching, and often in hopes of improving their betting odds. That being said these statistics are not reserved for the top players/leagues because they are the only ones who can benefit from them, but because of the major financial barriers to collecting necessary data for real world sports [14]. Rocket league is an exception to this as all the data is coded in the game and I think its important to take advantage of this to make expected goals accessible to non-professional players/athletes. While it is out of the scope of this project, the model certainly could be trained using data sets from all levels of players (would be organized by the different classes of the ranking system) as well as the different game modes.

7.4 xG in the Professional Game

From the perspective of professional rocket league players, it is important to remember this project will be creating a statistic the evaluate the efficiency and ability of individual players, not just teams as a whole. While the motivation behind creating such in depth descriptive statistics is to improve analysis and thus improve coaching and player performance, they can also be used by teams and fans to mistreat individual players [19]. Professional esports players are subject to a constant stream of praise or criticism online just like or more so than athletes in real world sports. A player under-performing their expected goals value means they are shooting at a below league average quality and may be subject to valid criticism from their organization. However it also provides another way for fans to attack players online and potentially cause harm.

A Replication Instructions

Below is a list of all the packages and plugins one would need to replicate this project

- carball: python tool used to decompile and analyse rocket replays into json files. Can be installed using the command 'pip install carball' or following the instructions on the carball github [20]
- pandas dataframe to handle data to train model
- sklearn library for everything machine learning
- imblearn library for SMOTE implementation
- numpy for various calculations
- scipy for various calculations, particular hyperparamter training
- shap for feature importance analyis. Can be installed with 'pip install shap'
- json package to handle json files
- csv package to create csv files with data
- os package for sorting files
- math package for calculations when deriving data

B Code Architecture Overview

This was not a very code heavy project in terms of pure code volume, as my code can be divided into two main parts. The python script to convert rocket league replay files into json and analyzed json files and the create the data set that my models were trained on. I used pycharm to create this, but one could use any IDE or text editor of their choosing. In the script, each variable has its own method and they are all put together to get the data for a single shot in the getstats() method. The functions to convert replays to json files and to write the csv files should also be compartmentalized to make the main method clean and simple.

To do the machine learning side of the project, I would recommend using anaconda, google colab or jupyter notebook which is what I used because they work well with the sklearn library and are well set up for machine learning tasks in general. The jupyter notebook is organized very simply, with the only main reused method being the modelEvaluation() method. It is possible to have methods to create machine learning algorithms quickly, but for ease of customization I chose not to do this.

References

[1] (online alias), twobackfromtheend. "Expected Goals: A new stat in rocket league". In: 2019.

[2] Alexander Fairchild, Konstantinos Pelechrinis and Kokkodis, Marios. "Journal of Sports Analytics 4". In: 2018, pp. 165–174.

[3] Anderson, Chris and Sally, David. "The Numbers Game: Why Everything You Know About Soccer is Wrong". In: Penguin Books, 2013. Chap. 1.

[4] Anzer, Gabriel and Bauer, Pascal. "Frontiers in sports and active living". In: 2021, pp. 1–15.

[5] Caelen, Olivier. "What is the Shapley value ?" In: *Medium* (2022).

[6] Can't, Fly. "Ballchasing FAQs". In: *Ballchasing.com* (2023).

[7] Davide Chicco, Guiseppe Jurman. "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation". In: *BMC Genomics*. 2020.

[8] Dixon, Michael. "How is big data analytics changing sports?" In: *Selertiy* (2021).

[9] ECMWF. "Ranked Probability Score". In: *Statistical Concepts - Probablistic Data* (2022).

[10] Egels, H.P.H. "Eindhoven University of Technology". In: 2016, pp. 1–63.

[11] Geron, Aurelion. "Support Vector Machines". In: *Hands-on Machine Learning*. 2019.

[12] Goldenfien, Jake. "Algorithmic Transparency and Decision-Making Accountability: Thoughts for Buying Machine Learning Algorithms". In: *University of Melbourne* (2019).

[13] Hewitt, James and Karakus, Oktay. "A Machine Learning Approach for Player and Position Adjusted Expected Goals in Football (Soccer)". In: *Cardiff University, School of Computer Science and Informatics* (2023).

[14] Hytner, Dave. "Arsenal's 'secret' signing: club buys £2m revolutionary data company". In: *The Guardian* (2014).

[15] Manfred, Tony. "Here Are The Odds That Your Kid Becomes A Professional Athlete (Hint: They're Small)". In: *Business Insider* (2012).

[16] Mustafa Cavus, Przemyslaw Biecek. "Explainable Expected Goals Model for Performance Analysis in Football Analytics". In: *Warsaw University of Technology*. 2022.

[17] Nitesh Chawla Kevin Bowyer, Lawrence Hall and Kegelmeyer, Philip. "SMOTE: Synthetic Minority Over-sampling Technique". In: *Journal of Artificial Intelligence Research 16*. 2002, pp. 321–357.

[18] Noble, Safiya Umoja. "Algorithms of Oppression. How Seach Engines Reinforce Racism". In: *Algorithms of Oppression* (2018).

[19] Ryan DeSalvio Ishaiq Hussain, Samuel Drolet. "Milestone 1 - NHL Expected Goals Model". In: *Milestone 1* (2021).

[20] SatieRl (alias). *Carball plugin for decompiling and analysing replays*. 2019. URL: `https://github.com/SaltieRL/carball`.

[21] Setia, Megha. "Log Loss vs. Mean Squared Error: Choosing the Right Metric for Classification". In: *Analytics Vidhya* (2023).

[22] Statsbomb. "What is expected goals?" In: *statsbomb* (2022).

[23] Syracuse. "With Viewership and Revenue Booming, Esports Set to Compete with Traditional Sports". In: *Syracuse University* (2019).

[24] Virginia, University of. "A Brief on Brier Scores". In: *Research Data Services & Social, Natural, and Engineering Sciences* (2023).

[25] Yapo, Adrienne and Weiss, Joseph. "Ethical Implications of Bias in Machine Learning". In: *Hawaii International Conference on System Sciences* (2018).

[1] `https://www.overleaf.com/read/fwswznqzdgsy`