ISyE 6740 - Spring 2023
Final Project

Team Member Names:
Spencer Allgaier (GTID: 903844347)

Project Title:
A Better xG Model

GitHub Code:
https://github.com/SAllgaier96/Portfolio/blob/master/BetterXG/BetterXG.ipynb

Problem Statement:
Football (soccer) is notoriously difficult as a sport to predict a scoreline or even an individual's performance due to the low scoring nature of the game. In recent years, sports scientists at companies, both large and small, have created new data and metrics in order to address this issue and better define performance. One metric that has risen in popularity has been expected goals (xG) which aims to measure the quality of a shot based on available features, especially distance and angle of the shot taken relative to the goal. Simple public models do not take into account enough event information for optimal performance and are built with a limited development process. State of the art models and data may have a more robust process to improve results, but both are kept proprietary. The goal of this project is to create an xG model that performs as well as state of the art in addition to uncovering the data and process of creating such a model.

Data Source:
Sports data is plentifully open sourced and easily accessible. The data for this project is sourced from ShotData, a github csv file curated by the football analytics community. The community is contributed by organizations like Opta, FotMob, FBRef, Stats Perform, StatsBomb, Understat, Wyscout, as well as individual contributors. ShotData provides detailed information with 29 unique features on 10165 shot events from the top european football leagues. While there is a more robust outcome label with 6 different shot results (crossbar, save, block, etc.), the author will use the binomial isGoal label as the dependent variable to classify.

Feature Name	**Description**	**Data Type**	**Example**
match_minute	Minute of the game	integer	29
match_second	Second of the minute of the game	integer	54
position_x	Distance in meters from the target goal line	float	23.69

postion_y	Distance in meters relative to the middle of the field width, negative values are left of the goal (shooter's perspective)	float	4.99
play_type	What kind of play shot came from	string	Open Play
BodyPart	Body part used to execute the shot	string	Left
Number_Intervening_Opponents	How many opponents were in the shooter’s path to the goal	integer	4
Number_Intervening_Teammates	How many teammates were in the shooter’s path to the goal	integer	2
Interference_on_Shooter	Level of interference on the shooter	string	Medium
outcome	Variety of more descriptive outcomes	string	Missed
position_xM	Distance in meters from midfield, negative values are in defensive half	float	23.69
position_yM	Distance in meters from the middle of the field width	float	4.99
position_xM_r	Negative transformation of position_xM	float	29.31
position_yM_r	Negative transformation of position_yM	float	5.001769
position_xM_std	Standardized value of position_xM	float	82.31
position_yM_std	Standardized value of position_yM	float	39.001769
position_xM_std_r	Standardized value of position_xM_r	float	73.001769
position_yM_std_r	Standardized value of position_yM_r	float	135.31
BodyPartCode	Numeric value of body part used in shot (1 = Left, 2 = Right, 3 = Head)	integer	3
Interference_on_Shooter_Code	Numeric level of interference on shooter from 1-3	integer	3
isGoal	Shot scored (1) or not (0)	integer	0
distance_to_goalM	Distance in meters from center of the target goal	float	24.212265
distance_to_centerM	Distance in meters from center of the field	float	5.001769
angle	Angle of shot relative to the center of	float	11.922004

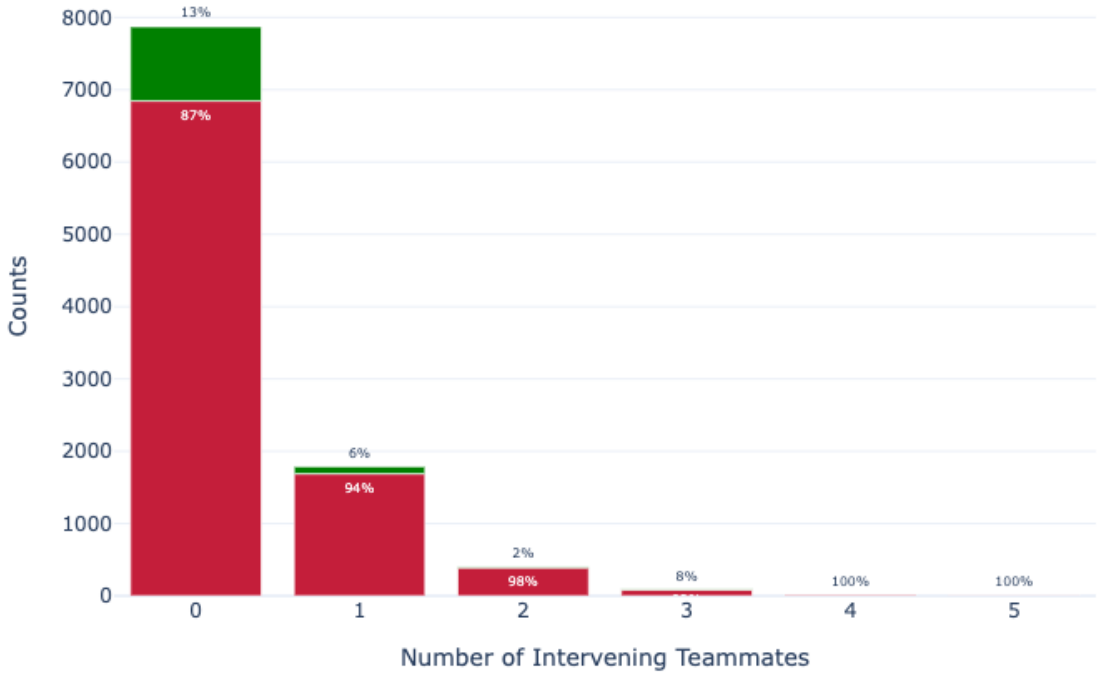
	target goal		
isFoot	Shot attempt with foot (1) or not (0)	integer	1
isHead	Shot attempt with head (1) or not (0)	integer	0
header_distance_to_goalM	Distance in meters from goal if header, otherwise (0)	float	0.0
High	Binomial dummy variable for Interference_on_Shooter High value	integer	0
Low	Binomial dummy variable for Interference_on_Shooter Low value	integer	0
Medium	Binomial dummy variable for Interference_on_Shooter Medium value	integer	1

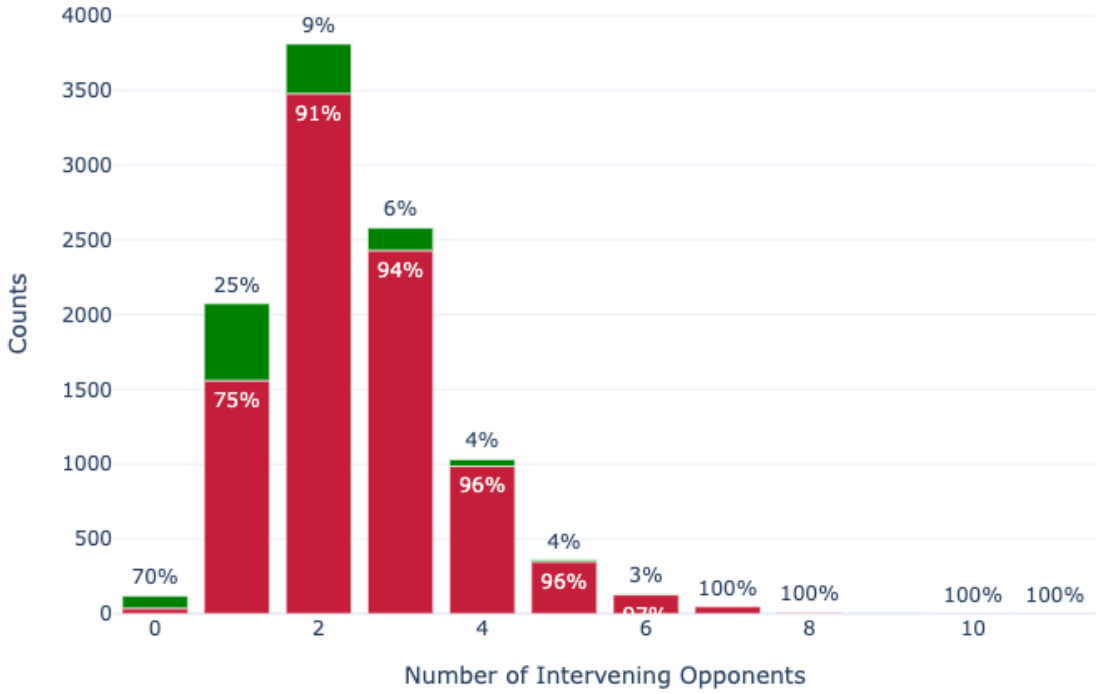
ShotData:
https://raw.githubusercontent.com/eddwebster/football_analytics/master/data/shots/engineered/complete_shots_engineered.csv

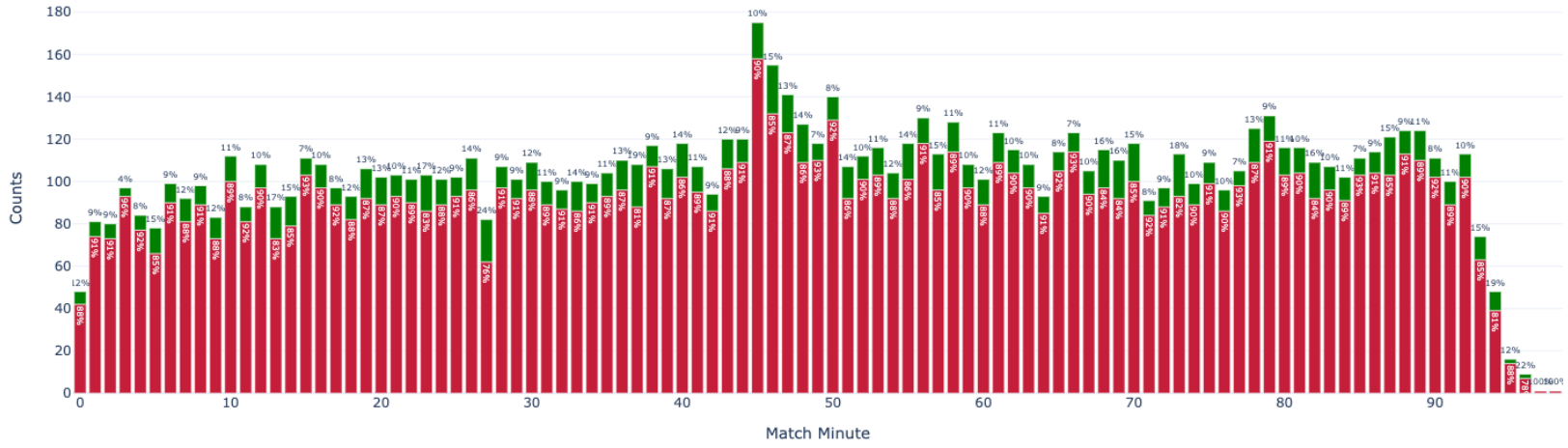
Description: https://github.com/eddwebster/football_analytics/blob/master/docs/shots/ShotData.txt

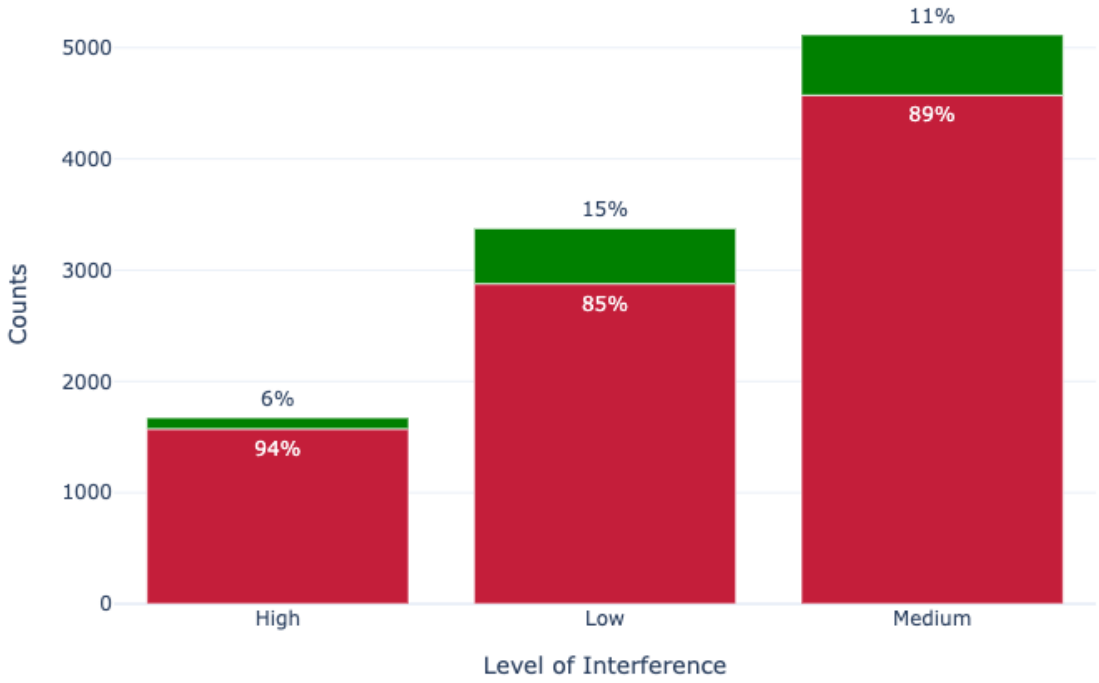
Methodology:

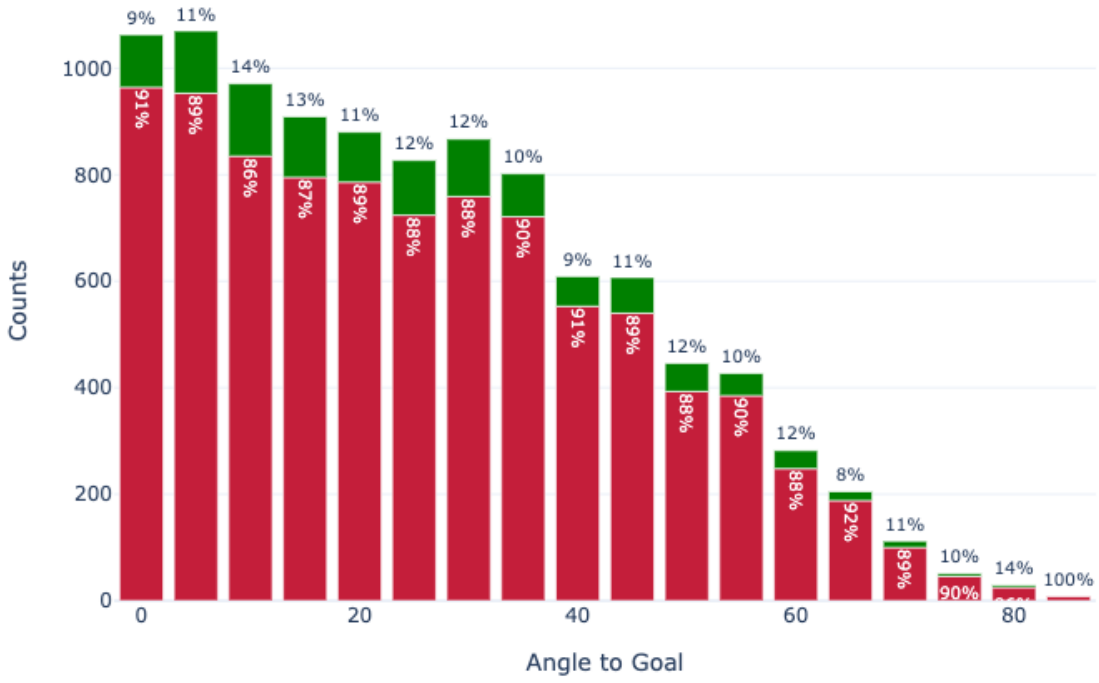
EDA
First, the author will conduct exploratory data analysis (EDA) of ShotData. Histograms of features with labels of conversion rates are shown at each category (or bucket for continuous variables) to reveal any significant variations. Each histogram bar will show the count of shots converted to goals in green stacked on shots missed in red. Features that show significant variations in conversion *rates* between categories are likely good predictors for models The features to be represented this way are Number_Intervening_Opponents, Number_Intervening_Teammates, match_minute, BodyPart, Interference_on_Shooter, angle, and distance_to_goalM.

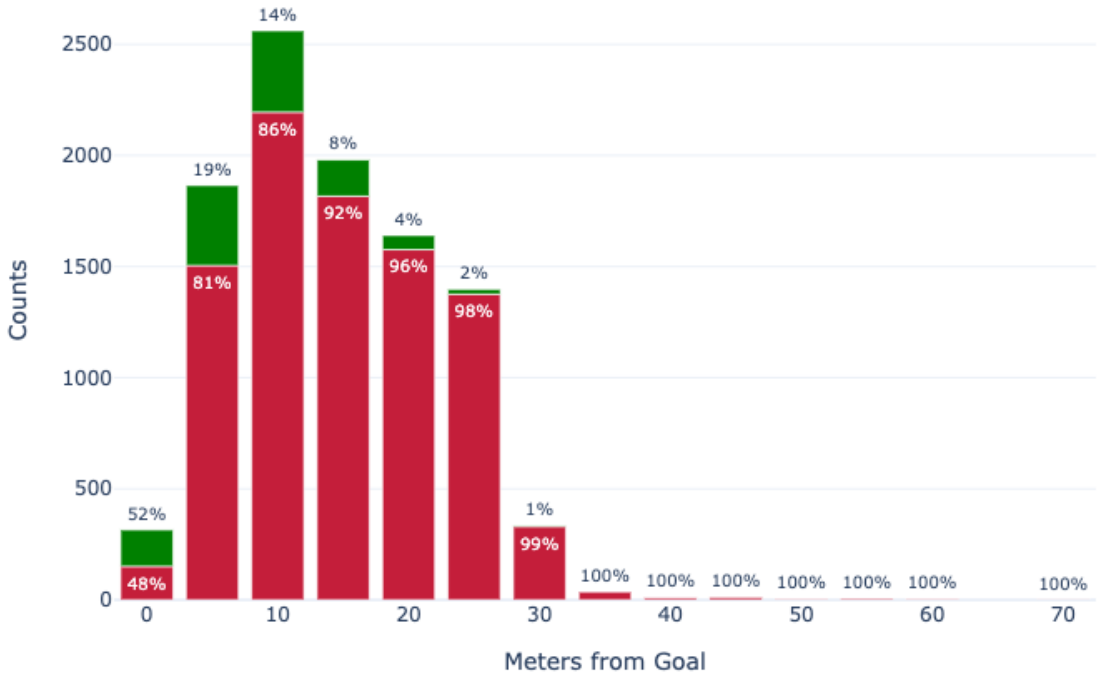

8000
7000
6000
5000
4000
3000
2000
1000
0
Counts
13%
87%
6%
94%
2%
98%
8%
100%
100%
0
1
2
3
4
5
Number of Intervening Teammates

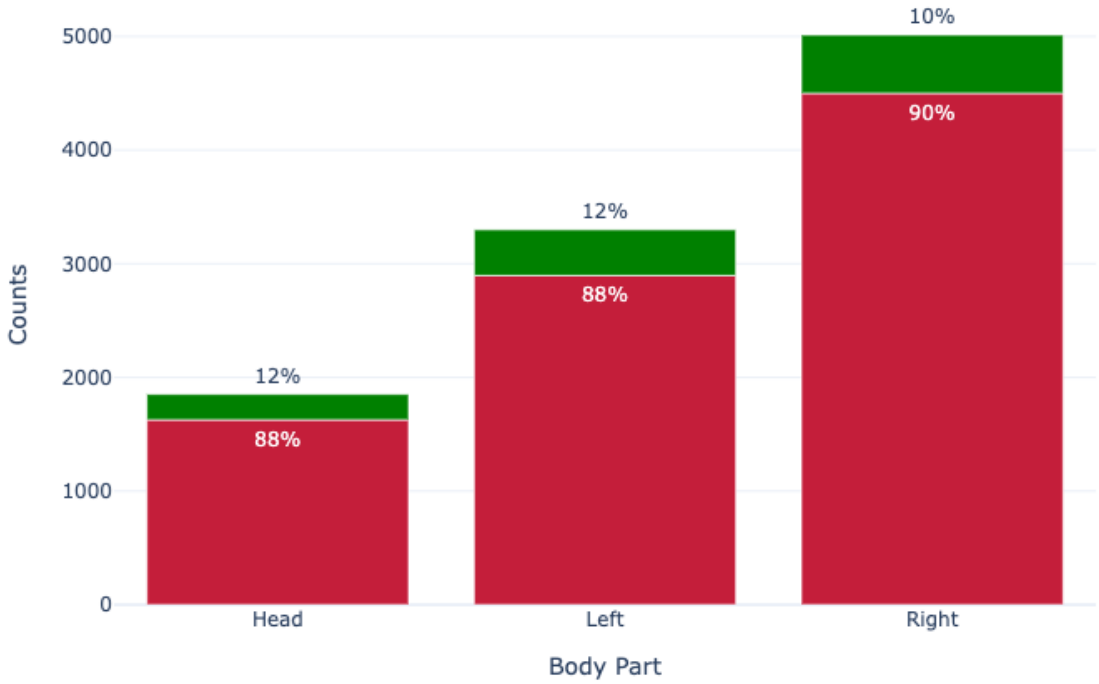

4000
3500
3000
2500
2000
1500
1000
500
0
Counts
70%
25%
75%
9%
91%
6%
94%
4%
96%
4%
96%
3%
100%
100%
100%
100%
0
2
4
6
8
10
Number of Intervening Opponents


Counts
Match Minute
0
20
40
60
80
100
120
140
160
180
0
10
20
30
40
50
60
70
80
90

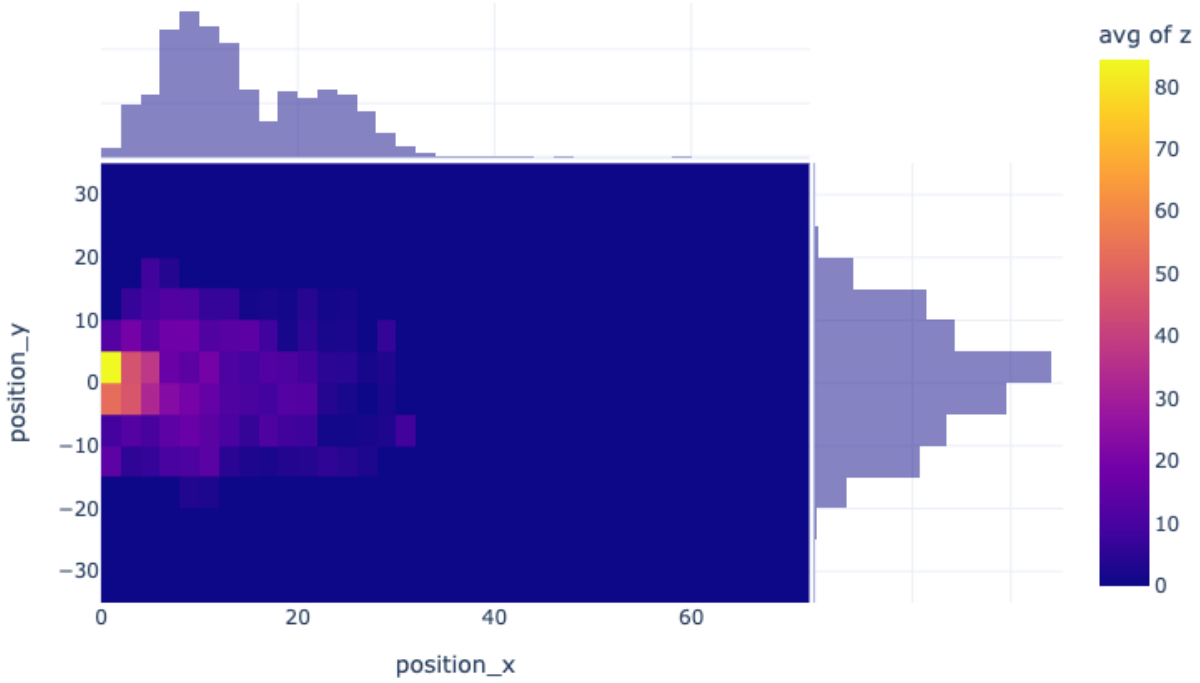

11%
89%
15%
85%
6%
94%
5000
4000
3000
2000
1000
0
Counts
High
Low
Medium
Level of Interference


9%
11%
14%
13%
11%
12%
12%
10%
9%
11%
12%
10%
12%
8%
11%
10%
14%
100%
1000
800
600
400
200
0
Counts
0
20
40
60
80
Angle to Goal


Counts
2500
2000
1500
1000
500
0
52%
48%
19%
81%
14%
86%
8%
92%
4%
96%
2%
98%
1%
99%
100%
100%
100%
100%
100%
100%
100%
0
10
20
30
40
50
60
70
Meters from Goal


Counts
5000
4000
3000
2000
1000
0
12%
88%
12%
88%
10%
90%
Head
Left
Right
Body Part

For the next visualization, location coordinates (position_x and position_y) are shown on a heatmap, colored by a continuous value, conversion rate of isGoal for a represented coordinate value. Using Plotly's marginal histogram feature on this visualization will also show the relationship of number of shot events and both positional coordinates. This shows positions on the field, relative to the center of the goal line, that correlate to goal conversions for our model and just how much position matters.



Feature Selection and Engineering

From our EDA, the most influential features that we should be aware of are position_x, position_y, angle, and Number_Intervening_Opponents. In order to decide on which features to use during the model training, tuning, and testing process, find the feature importances. Fit a simple random forest model with everything except duplicate features (BodyPart, Interference_on_Shooter), match_second, and outcome since we don't need those. This will give a more precise value on which features we can remove and which are more important.

Feature	**Importance**
distance_to_goalM	0.105371
match_minute	0.099539
angle	0.080740
Number_Intervening_Opponents	0.060489

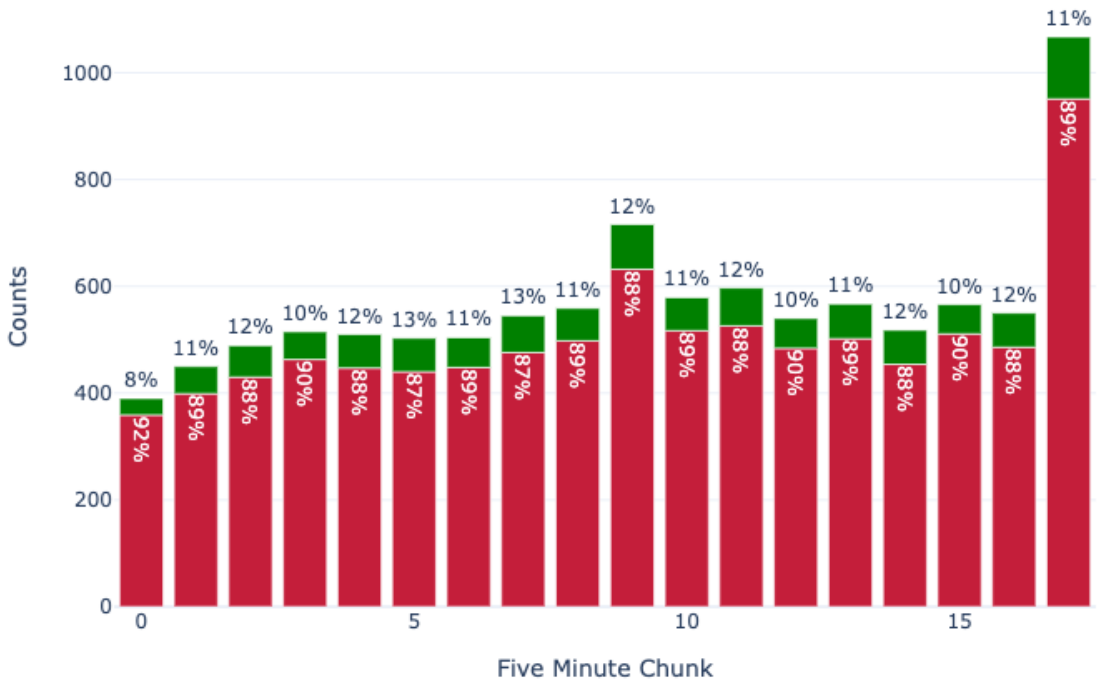
distance_to_centerM	0.055883
position_xM_r	0.049911
position_xM_std	0.049681
position_x	0.049109
position_xM_std_r	0.047831
position_xM	0.047534
postiion_ym_std	0.047440
position_yM_std_r	0.047269
position_yM_r	0.047050
position_yM	0.046918
position_y	0.046509
header_distance_to_goalM	0.023322
BodyPartCode	0.021033
Interference_Intervening_Teammates	0.018591
Number_Intervening_Teammates	0.013372
Low	0.012458
Medium	0.010442
High	0.009557
isFoot	0.005175
isHead	0.004775

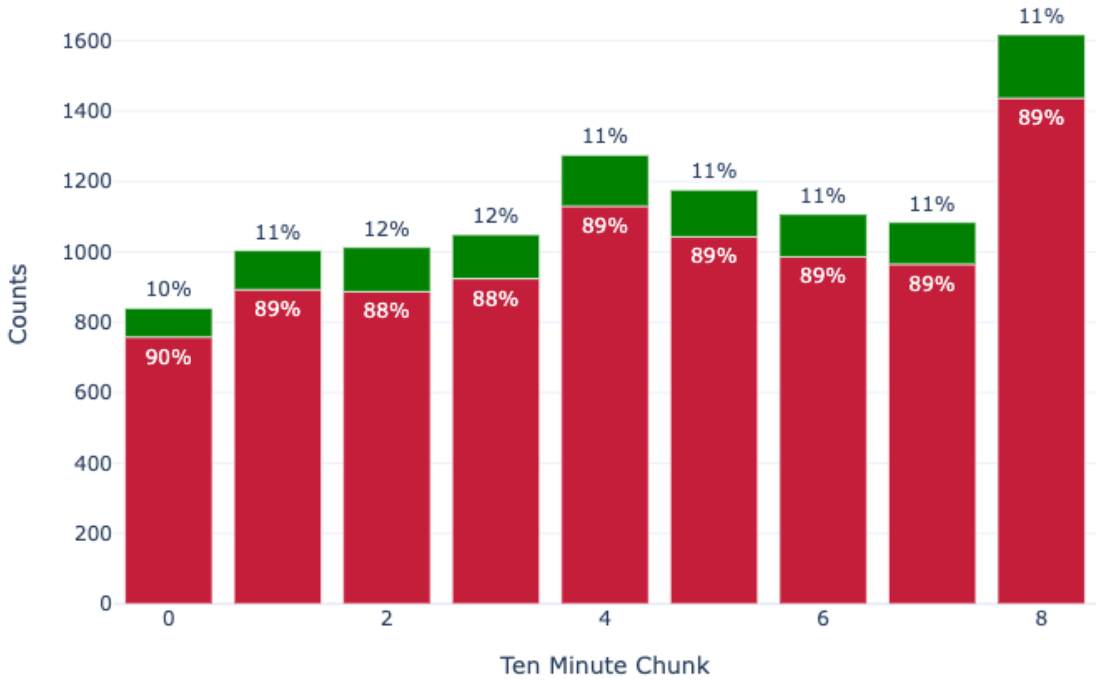
It's interesting to see that the most influential feature for converting a shot to a goal is distance_to_goalM. This makes intuitive sense, but putting a numeric value to compare against other features is pretty cool. Further, the specific foot/head dummy variables are less telling than the code counterpart, so we'll use the code features and drop to dummy variables. The sum of the dummy variables for interference level is higher than the coded feature, conversely, so we'll use the dummies. Position coordinates are very important, especially the x coordinate since it relates more closely to the distance from the goal, but we have 10 features that can be reduced to two. Since position_x and position_y are higher up the list in feature_importances_, we'll use these moving forward and drop the others.

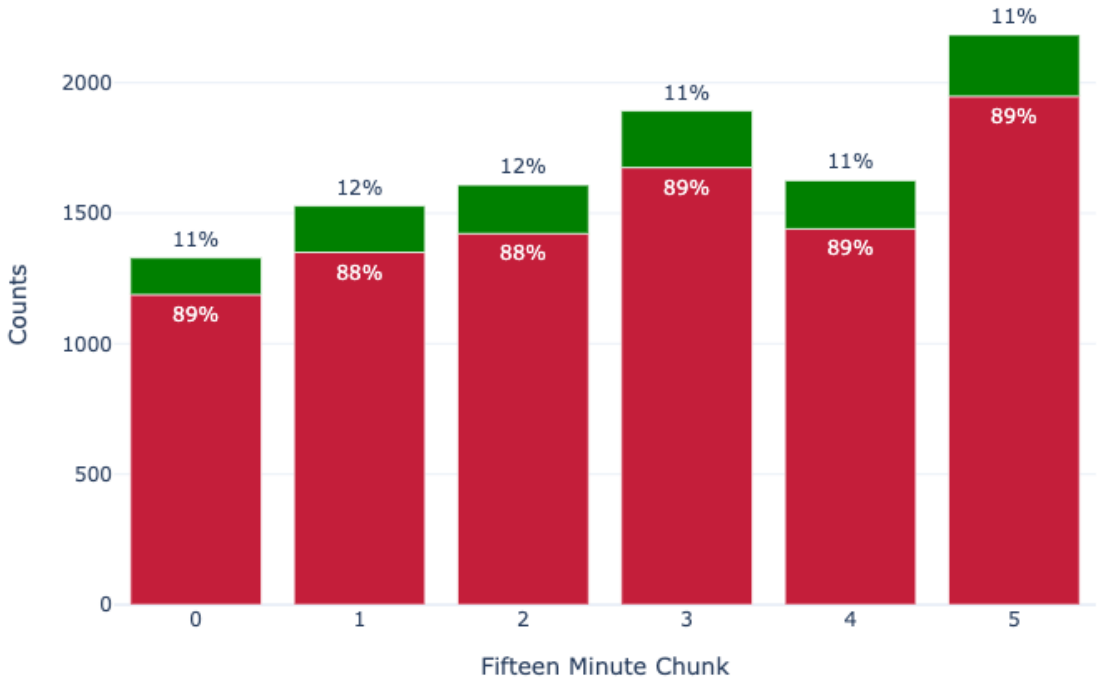
Somewhat unexpected is the impact the match_minute shows in this test. It seemed like the rates were randomly distributed, but this shows that there is significant differences minute to minute. This would make sense since the start of a game might yield lower accuracy, or the end of a game with high stakes could sharpen a player's skills to score in the final minutes. Still, continuing on a minute by minute basis wouldn't be prudent, so let's create a new set of variables for 5/10/15/45 minutes chunks in order to give more insight into strategy. Additionally, we'll engineer a total number of intervening players as the sum of teammates and opponents to see if that influences the result more than the two separated.

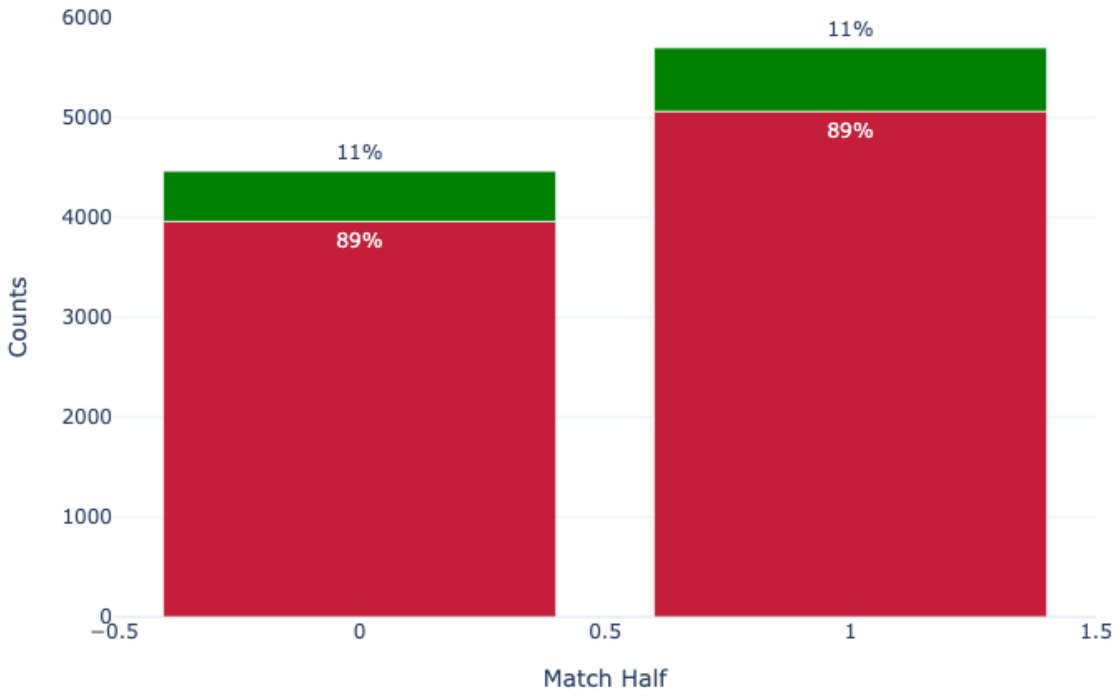
Feature	**Importance**
distance_to_goalM	0.095619
angle	0.068963
match_minute	0.063744
distance_to_centerM	0.051198
position_xM	0.044890
position_xM_std	0.044833
position_yM_std	0.044782
position_yM_std_r	0.043896
position_x	0.043607
position_y	0.042929
Number_Intervening_Total	0.042876
position_yM	0.042789
position_xM_std_r	0.042694
five_minute_chunk	0.042321
position_xM_r	0.042231
position_yM_r	0.041195
Number_Intervening_Opponents	0.038631
ten_minute_chunk	0.030083
header_distance_to_goalM	0.022543
fifteen_minute_chunk	0.022524

Interference_on_Shooter_Code	0.018210
BodyPartCode	0.016557
Low	0.011439
Number_Intervening_Teammates	0.009082
Medium	0.008521
High	0.007518
match_half	0.006851
isFoot	0.005194
isHead	0.004280




Counts
1600
1400
1200
1000
800
600
400
200
0
10%
11%
12%
12%
11%
11%
11%
11%
90%
89%
88%
88%
89%
89%
89%
89%
0
2
4
6
8
Ten Minute Chunk


Counts
2000
1500
1000
500
0
11%
12%
12%
11%
11%
11%
89%
88%
88%
89%
89%
89%
0
1
2
3
4
5
Fifteen Minute Chunk



According to this output, the addition of total intervening players does improve our reading of goal conversion. It also shows that as the game time is represented more precisely, the better the predictor becomes. match_half is statistically irrelevant, five_minute_chunk is almost significant, match_minute is one of the best predictors of the whole set. Looking at the visuals for these time divisors, it becomes clear that the reason is likely due to random sampling chance rather than an explanation of shot conversion. Since coaches usually make substitutions during the 60th minute, there should be a noticeable difference in that time window that simply doesn't appear. It's a case of being given a narrower window so the result seems easier to predict because there are more chances for spikes due to randomization to occur.

Using this information, the features that should be used during model training, tuning, and testing process should be distance_to_goalM, header_distance_to_goalM, match_minute, angle, Number_Intervening_Opponents, Number_Intervening_Total, position_x, position_y, Interference_on_Shooter_Code, BodyPartCode, High, Low, Medium.

Performance Metrics

The baseline random forest model before hyperparameter tuning is performing at around 88.7% accuracy with an ROC_AUC value of 0.594. However, given the fact that about 11.2% of shots from our data convert to goals, a poor model classifying anything as no goal would yield 91% accuracy. To account for the skew in classification from our dataset, incorporate two new metrics to better evaluate during development. Precision Recall AUC measures the area under the curve of recall (x) vs precision (y) where recall = (true positives) / ((true positives) + (false negatives)) and precision = (true positives) / ((true

positives) + (false positives)). Cohen's Kappa similarly used the confusion matrix values to show (probability of agreement) - (probability of random agreement) / (1 - probability of random agreement).

The baseline random forest model doesn't perform very well adjusting for the predictor skewness, especially in predicting goals (52 out of 244). The PR-AUC value is at 0.233 and Cohen Kappa is at 0.278.

Model Selection and Hyperparameter Tuning

Compare a series of models from the scikit-learn python package to find the best results for an xG predictor based on the features from the EDA/feature engineering. Use a train/test split of 80/20 with random split assignments. Logistic Regression, Naive Bayes, Decision Trees, and ADA Boost have limited parameters and do not require further adjustment. Hyperparameter tuning is implemented using GridSearchCV with a five-fold replication system for each parameter combination set to account for random variation. It is important for model performance metrics to take into account that the proportion of shots that convert to goals is low, so creating a poor predictor that guesses 100% of shots will miss would be as accurate as the conversion rate of the dataset by traditional measures (i.e. if 90% of shots convert to goals, this model will be 90% accurate). To solve this issue, GridSearchCV is able to change the scoring metric to our proposed PR-AUC metric. After each combination of a specified set of parameter options are repeated five times, the setup with the best score will be assigned to the model for evaluation. The results are included in the evaluation and final results section.

Reviewing the results, the best classifier models are Random Forest, and Naive Bayes. Two very different methods which likely yield different predictions. To combine the best of the models, an ensemble model can be built. Ensemble models generate predictions from multiple models and each gets a vote in the final prediction. Voting can be hard or soft meaning a “hard” binomial prediction is generated from each model and the majority gets the ensemble prediction output or a “soft” probability prediction is added from each model and the highest value gets the ensemble prediction output. Use both to compare performance results, but in the final xG prediction column will be generated using the soft voting method since hard voting doesn’t allow for a probabilistic prediction output.

Evaluation and Final Results:

Model	**Customized Parameters**	**Accuracy**	**ROC-AUC**	**PR-AUC**	**Cohen Kappa**
Baseline Random Forest	None	0.89179	0.59873	0.23297	0.27821
Logistic Regression	None	0.88834	0.56856	0.19888	0.20573
Naive Bayes	None	0.82292	0.70826	0.24849	0.33193
Decision Tree	None	0.82784	0.59956	0.17078	0.19634
ADA Boost	learning_rate: 0.1	0.87998	0.50000	0.12002	0.00000

KNN	n_neighbors: 12	0.88244	0.53148	0.15250	0.10163
SVM	Gamma: 0.01 C: 3.0	0.88047	0.54806	0.16311	0.14520
Gradient Boost	learning_rate: 0.1 loss: deviance n_estimators: 60	0.88883	0.57759	0.20750	0.22771
Random Forest	n_estimators: 70 min_samples_split: 150 class_weight: {0: 1, 1:3}	0.87850	0.66728	0.26494	0.36776
Neural Network	activation: logistic alpha: 0.1	0.88933	0.57610	0.20784	0.22509
Ensemble of Naive Bayes, Random Forest	Inherited voting = hard	0.88785	0.55756	0.18909	0.17820
Ensemble of Naive Bayes, Random Forest	Inherited voting = soft	0.87654	0.67678	0.27037	0.37691

Clearly, some of the skew-weighted metrics differ from traditional methods, so it was good that PR-AUC and Cohen Kappa were included in the evaluations. The baseline random forest model actually turned out to perform quite well, but Naive Bayes and Random Forest (optimized) models all stood out in the evaluation. Random Forest even jumped by a significant 0.09 points after tuning. The ensemble method shows a slight bump in performance metrics compared to Random Forest (optimized). This could also be caused by random chance with the specific test/train split, but predictions by combining models probably aren't improving.

Overall, test the performance of our produced xG value using the ensemble model. The top 5 highest xG values are all correctly predicting goals being scored with 71.3% being the highest xG that turned out to be a miss and 76.0% was the absolute highest. The top 625 lowest xG values are also all correctly predicted with 1.7% being the lowest xG that turned out to be a goal and 0.6% was the absolute lowest. The sum of the predictions over the entire dataset is 1133.03 compared to 1142 true goals scored.

If there's one addition this study adds to the football world, it's that it's easy to predict missed shots, but predicting goals is hard. There is a wide disparity between the upper and lower xG estimates which bodes well for the usefulness of the model. Most upper and lower end estimates were also accurate. Over time this xG model will closely resemble real life results. There are better shots and strategies that can be implemented in gameplay, but the football gods are a stubborn bunch who have the final say before even the best chances turn into a goal.

Future Improvements and Limitations:

ShotData had some limitations that did not prevent an effective model to be built, but probably diminishes the quality. One limitation was that it only had open play types available, but it would have been interesting to compare and include free kicks, plays off of corners or indirect free kicks, and penalties. There could also be more features to include player info and dates for a time dependent method to account for the shooter's finishing quality or a goalkeeper's blocking quality during a recent time period. League information would also be influential since they vary wildly in quality between nations, tiers, and competition types. Other features that would likely improve a model are goalkeeper's position relative to the shooter and the goal, if the shot was deflected, the scoreline at the time of the shot, whether or not in stoppage time or extra time, and the shooter's (and goalkeeper's) movement direction and speed.

Although not necessary, it would improve the understanding of classification predictions done by each model if a dimension reduction technique like PCA, TSNE, or UMAP was applied to the dataframes during the model preparation process. This would also aid in viewing decision boundaries and incorrect predictions could go into the ensemble model selection so that if one model performed poorly, but correctly captured some goals that were incorrectly classified by better models, it could be included in the ensemble.

Understat provides a live API that could be used to train a similar model and generate live xG values during games to compare against their industry generated value. The features are slightly different, but similar enough for future implementation.

Understat: https://understat.readthedocs.io/en/latest/classes/understat.html (get_player_shots)