



UPPSALA
UNIVERSITET

Statistical methods and machine learning in football

Predicting goals from shots and corners

Joakim Bergström, Sari Shaybany, Chun Wu

Project in Scientific Computing: Report

PROJECT

Abstract

In this study two kinds of models to evaluate the quality of play in football have been developed, looking at shots and corners, respectively, in order to give football clubs insights and possible ways to gain advantage over opposing teams. The first model aims at predicting whether a shot from a certain position of the pitch will result in a goal or not. At first, only the player in possession of the ball was considered in the model, after which the model was modified taking into account the surrounding players. The aim of the second model is to answer the question 'What makes a successful corner?'. The models were made in collaboration with the company Twelve and Hammarby IF. Using their existing expected goals model and data set, a logistic regression and a XGBoost classifier were trained, after which the models were evaluated on test sets. Compared to their counterpart using XGBoost, the shot model using logistic regression performed better in terms of accuracy (0.71) and area under the ROC-curve (AUC) (0.78). The larger AUC implies that the classifier is more capable of distinguishing between the classes (*goals* and *not goals/misses*). Taking into account the surrounding players of the field did not improve the performance of the shot models, which can be explained mainly by the fact that the data sets used were too small for training the models properly and the fact that the positions of all the players deviates from the real positions of the players due to the time delay of the observers manually keeping track of the positions of the players. In contrast to the shot models, the corner models using XGBoost performed slightly better than their counterpart using logistic regression. The AUC for the model predicting shots from corner was 0.63 using XGBoost. For the model predicting goals from corners, the AUC was 0.62 using XGBoost. The XGBoost decision tree helped to answer the question 'What makes a successful corner?'. Last but not least, a scout report describing the corners of each team was developed to give further insights and possible advantages against the opposing teams.

Contents

1	Introduction	3
2	Theory and Method	4
2.1	The event data	4
2.1.1	OPTA qualifiers	5
2.1.2	Twelve Analysis Tool	5
2.1.3	Pitch map	6
2.2	Data pre-processing	6
2.2.1	Shots	6
2.2.2	Shots, using tracking data	9
2.2.3	Corners	11
2.2.4	Data Set Balancing and the confusion matrix	13
2.3	Corner models	14
2.3.1	Shots from corners	14
2.3.2	Goals from shots	14
2.4	Training a classifier	14
2.5	Logistic regression	14
2.6	XGBoost	15
2.7	ROC-curve and AUC	16
2.8	Scout report	17
3	Results and discussion	17
3.1	Shots	17
3.2	Corner models	20
3.3	Scout report	20
4	Conclusions	30

1 Introduction

Machine Learning (ML) is the study of patterns and statistical methods which are used to perform tasks without predetermined outcomes. Using software and algorithms Machine Learning is usually considered as a sub-field of Artificial intelligence. As the popularity of artificial intelligence methods seem to be on a steady rise due to the fact that the methods are key to understanding what it means to predict future returns and their impacts, betting/lottery companies are considering these methods to a larger extent. This has created a whole field of business opportunities which are eyeopening to coaches and others whom are interested in the cross section between how sports analysis is becoming more and more data driven at its core. The coaches can use classification models as a basis to formulate successful strategies to ultimately win matches.[1]

Twelve company is an IT-consultancy which specialises in football analysis, and are now providing the service for many football clubs to model the quality of play in football. They have provided us with event data of the last three seasons in Allsvenskan, which is the top level of the Swedish football league system. In our case, Hammarby IF, also known by the nickname "Bajen" and notoriously known for their enthusiastic fans, is the main partner in the quest for a deeper understanding of football.

In modern football the importance of set pieces are becoming more and more understood, as a way to score. The amount of effort going into developing strategies for free kick situations, corners of various forms, etc., seems to be paying off. Therefore, most professional football teams are putting more and more training into their corner-tactics. The lesson from this is that football teams that are good in understanding set pieces will probably have advantage in today's modern football. In this project we studied both shots and corners as means of scoring goals.

The first aim of the project is to create models using logistic regression and Xgboost, respectively, evaluating the probability of a shot from a particular location on the football pitch becoming a goal and based on the prediction classifies the shot as either goal or miss. In order to classify a particular shot a classification model is built based on the training data set which was extracted from raw data of match events in Allsvenskan. The events describe the action and position of the player currently in possession of the ball, and hence the models did not, at first, account for the positions of the other 21 players. Later, tracking data of all the positions of players during the match in Allsvenskan were used to improve the performance of the models.

The second aim of the project is similar to the first one, but models are now created for corner situations to determine the combination of features more likely leading to goals via shots. A sub-goal is to generate actionable intelligence regarding the opponents tactics on corner kicks. This information could be used by the defending team to help stop the attacking team from scoring from a corner situation. This is also referred to as the *scout report*.

The models of the type studied in this project are sometimes referred to

as *Expected goals models*, or xG , namely they give the probability of a certain phenomenon in football (free kicks, penalties, corners, shots during open play, etc.) to result in a goal. The model can for example take the distance to the goal and the shot angle, and convert it to a number between 0 and 1, which can be interpreted as the probability of scoring. Having a probability of 1 indicates that scoring is most certainly the case, while a probability of 0 is a guaranteed failure of scoring.

The remainder of this report is organised as follows. In Section 2, the underlying theory and the methods used to train the models are explained, including an in dept description of the pre-processing actions. Section 3 provides the results which are discussed and compared to each other as well as other models out there. Finally, Section 4 concludes the report.

2 Theory and Method

Firstly, the structure and content of the event data is examined. Secondly, we provide a description of the data pre-processing. The resulting pre-processed data is then used to train the models using logistic regression and Xgboost, respectively, both of which are explained in dept. Other relevant theory to the project are explained as well.

2.1 The event data

As mentioned in the introduction, the training data used to learn the models were extracted from raw data of match events (passes, shots, etc.) in Allsvenskan, from the past three seasons (2017-2019). The event data consist of various information about the events. The files containing the data are the so called json-files. The format of these files is complex, consisting of groups of dictionaries inside lists and groups of lists are in turn enclosed by outer lists (see Figure 1). Each dictionary contains information about one certain event. The details of the events are referenced by numbers known here as OPTA qualifiers [7]. Furthermore, the qualifiers are divided into groups associated to specific "event types". The first column of a dictionary containing information about an event is an event ID, followed by the event type I.D., the outcome (true or false), the minutes and seconds played, the team ID, the player I.D., a time stamp and then a list of different qualifier I.D.s and their respective value grouped together in directories (see Figure 1). Finally, the x- and y-coordinates are specified and a mention of the event is given.

```
[{"eventId": 1375794403, "eventId": 1, "outcome": true, "min": 0, "sec": 3, "playerId": 192774, "timestamp": "2019-06-25T18:02:51.41", "qualifiers": [{"qualifierId": 17, "value": "17.6"}, {"qualifierId": 56, "value": "Back"}, {"qualifierId": 27, "value": "33.9"}, {"qualifierId": 140, "value": "33.9"}, {"qualifierId": 213, "value": "3.4"}, {"qualifierId": 43, "value": "43.3"}], "x": 49.9, "y": 50.1, "description": "Kick-off"}, {"eventId": 140277081, "eventId": 1, "outcome": false, "min": 0, "sec": 3, "teamId": 483, "playerId": 192774, "timestamp": "2019-06-25T18:02:53.951", "qualifiers": [{"qualifierId": 213, "value": "88.0"}, {"qualifierId": 140, "value": "82.5"}, {"qualifierId": 59, "value": "59.0"}, {"qualifierId": 1, "value": "Left"}, {"qualifierId": 56, "value": "Left"}, {"qualifierId": 1, "value": "65.4"}, {"qualifierId": 57, "value": "57.5"}, {"qualifierId": 140, "value": "21.6"}], "x": 17.8, "y": 17.8, "description": "Failed Pass"}]
```

Figure 1: Small piece of the event data in a json-file.

Each list of events in the event data is equivalent to linked and coordinated movements of players in what is called Possession Chains. A Possession Chain is a chain of events where the ball is moving between players of the same team. The chain is broken when the team either scores or loses the possession of the ball, i.e. the opposing team has been in contact with the ball twice, from which a new chain of events starts taking place. This creates series of patterns looked into in the creation of the corner models.

2.1.1 OPTA qualifiers

OPTA is one of the top companies handling sports data. The company provides the tracking data as well as the labels for the data used throughout this project. A possession chain consists of events, where each event is an action on the pitch. Examples of events are pass, shot, tackle and goal. To distinguish even further, the events themselves consists of qualifiers which further describes the event (e.g. "long ball", "free-kick", "Head pass", etc.). The most important events for this project are shots and passes (passes only of interest for the corner models). The relevant qualifiers are *Regular play* and *Big Chance* (for shot models), and *Corner taken*, *In-swinger*, *Out-swinger*, *Length*, *Angle*, *Pass End X* and *Pass End Y* (for corner models) [7].

These qualifiers are meaningful to investigate as input parameters to the models, except for *Regular play* and *Corner taken*, which do not serve as input parameters but rather are qualifiers describing the events from which the other qualifiers are to be extracted. For instance, *Big Chance* is a qualifier describing shots deemed by Opta analysts to be excellent opportunities to score (clear cut chances, e.g. one-on-one). This qualifier is expected to influence the prediction of the shot models. Other types of qualifiers, such as *Right footed*, may instead be redundant qualifiers having no or small impact on the predictions. The qualifiers used for the corner models are looked at in a similar manner.

2.1.2 Twelve Analysis Tool

The project was in collaboration with the company Twelve Football, which provided us with the event data. The company specialises in measured performance in football, which is possible thanks to their team of engineers and developers

whom all are working coordinated to fulfil the main goal of providing leveraged, scientifically based strategies and/or evaluations for their clients. Furthermore, Twelve has developed a website and a phone application containing their tools, which are, among other things, used to investigate a certain player. Figure 1 shows a visualisation of a football analysis tool developed by Twelve.[6]

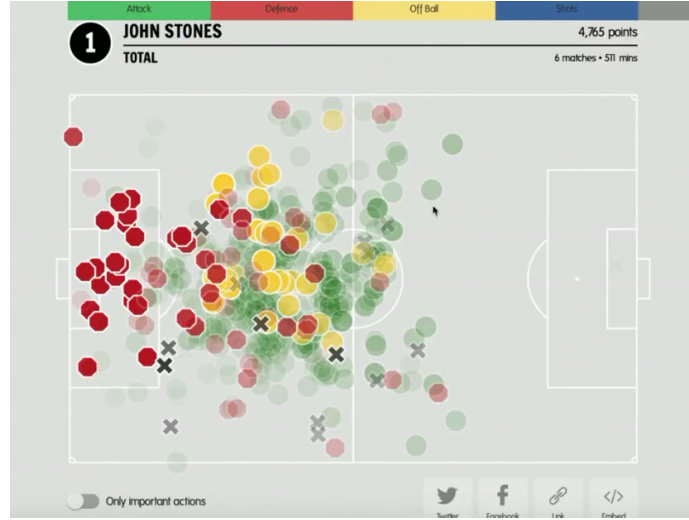


Figure 2: A demonstration of what the Twelve football analysis tool could look like. The points and their colours represent positions and significance.

2.1.3 Pitch map

The plots depicting a football pitch were done using the pitch map code made available by FCPython [2].

2.2 Data pre-processing

In order to train the models properly the raw data had to be pre-processed to produce the final training set, which forms a better set to train the models, excluding irrelevant or insignificant information of the events for the purpose of creating models predicting goals from shots/corners. This type of excess information would otherwise either not influence predictions of future shots or lead to bad future predictions of goals.

2.2.1 Shots

One of the main properties determining the likeliness of a shot being a goal is the location from which the shot was taken. The x- and y-coordinates of the shots are hence extracted from the events data. The shots of interest are only the ones occurring during open play as opposed to from a set play. This is because set plays, such as free kicks and penalties, are undesirable types of shots to include into the models, since these occur under different conditions. A penalty, for instance, is far more likely to result in a goal than an arbitrary shot from the same distance during regular play under pressure from attacking

opponents. In the implementation, the events initially looked at were the ones having event type I.D.s 13-16 (associated with shots), and from these events the x- and y-coordinates were extracted only for those including the qualifier *Regular play* in the list of qualifiers associated with the event.

To verify that only the coordinates associated with shots were extracted (i.e. excluding passes, free kicks, etc.), the extracted data were plotted for one match and one team (Figure 3).

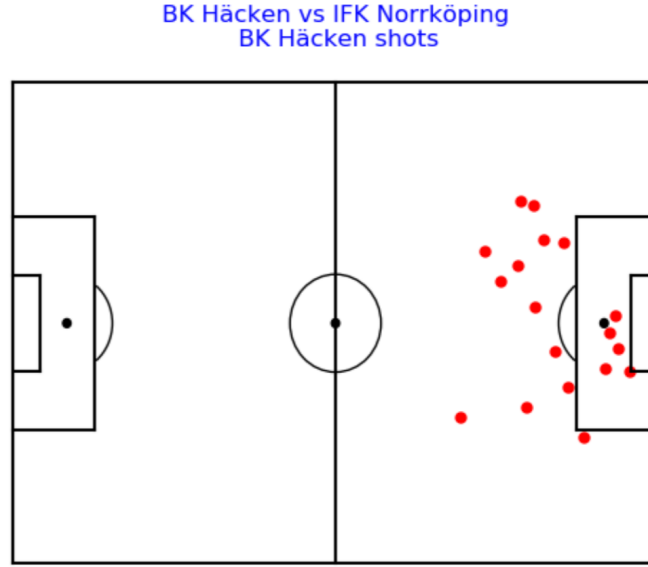


Figure 3: All shots of team BK Häcken during one match.

Now that the locations of the relevant shots have been collected and verified, the distance to the goal (origin at the midpoint of the goal) and the angle between the two goal posts (Figure 4) was calculated. The calculated angles and distances will then, among others, serve as input parameters to the model, i.e. part of the final training set. The distances were easily calculated using the formula for Euclidean distances, while the angle was calculated using trigonometric functions.

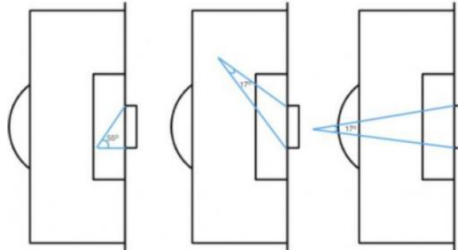


Figure 4: Illustration of the angle associated with probability of scoring. The larger the angle, the better chance of scoring.

When considering only the distance and the angle as factors of the probability of scoring, a fitted model of the probability distribution over the area close to the goal would somehow mimic the one shown in Figure 5, where the more red the colour, the higher the frequency of goals per shot from that particular position on the pitch.

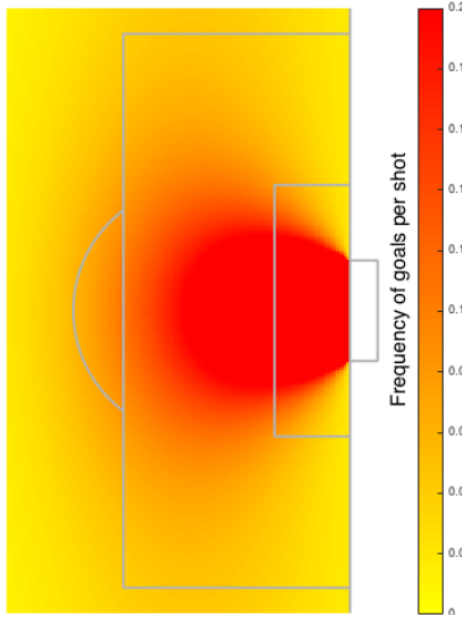


Figure 5: Probability of scoring goals from different positions around the box taking into account both the angle between the goal posts and distance to the centre of the goal.

The frequency of goals does not follow a perfect circle but rather a more squashed out circle. Though, if only the angle were considered, the red region would be circular, but since distance is taken into account the frequency of scoring from a bit wider out at larger distances is still relatively high [5].

Beside distance and angle, the final training set used as input data to the models contained the parameters *distance squared*, *angle times distance*, *angle squared*, as well as *big chance*. The later is, as mentioned, a shot qualifier and was extracted due to its remarkable impact in the prediction of goals, as opposed to other shot qualifiers which are not expected to influence the prediction to a similar extent. The total training set, i.e. the pre-processed data, can be seen in Figure 6.

```

,goal,bigchance,distance,angle,distance^2,angle*distance,angle*2
0,0,1,26.619046282690135,15.509847096577984,708.5736249999995,41
1,0,1,11.316316715256779,17.83550835804831,128.05902399999997,20
2,1,1,4.957902883276358,72.79581261158627,24.58080100000002,360.
3,0,1,13.26990414434106,23.034060916291356,176.09035600000004,30
4,0,1,20.363571027695507,14.49773614107088,414.675024999999983,29
5,1,1,12.417512834702446,17.283746324372014,154.19462499999997,2
6,0,0,27.063923514523903,14.502953630655695,732.4559559999999,39
7,0,0,79.24303634389587,5.193435774528187,6279.458809000002,411.
8,1,1,11.355572420622389,34.93061090321552,128.94902499999984,39
9,0,1,21.110913196733105,15.880285871379307,445.6706559999995,3

```

Figure 6: Pre-processed raw data, i.e. the training data.

2.2.2 Shots, using tracking data

The tracking data included the position of all players in a match, and the positions were recorded every 40ms. The position and time are, hence, the only information in this data. By using this data set we can find some useful information about the shot, e.g. the probability of scoring depending on whether the player shooting encounters defending opponents or not.

However, since the tracking data was collected by a different company than the one providing the event data, it was rather difficult to link the event data and tracking data together. The player ID in tracking data and event data were entirely different, and the only way to locate each shot in tracking data is through time. Thus, the time of the shots was used to link them and we found the result to be extremely noisy, i.e. the shot position from event data and position from tracking data are too distant from each other. The reason for this is simple; the sampling frequency in the tracking data not the same through the match, i.e. not always equal to 40ms, and since there are at least 135,000 frames in the tracking data, small time deviations may accumulate and cause great errors in positions of the players (due to the time delay) towards the end of the match. The task was therefore to figure out a way to remove this noise by first finding the player taking the shot, noting the time t in the event data and compare it to the time in the tracking data.

As already mentioned, the name of each player is not always the same in the two data set. A matching algorithm was therefore used to find the most matches having the most matching names. Then, using the time range $-4s + t$ to $4s + t$ we searched in the event data for the closest position of the player to the one in the tracking data, within the specified time range. This procedure avoided finding noise points outside this time frame. The result was only used in the case where the position difference is within 5 meters. One example is given in Figure 7,

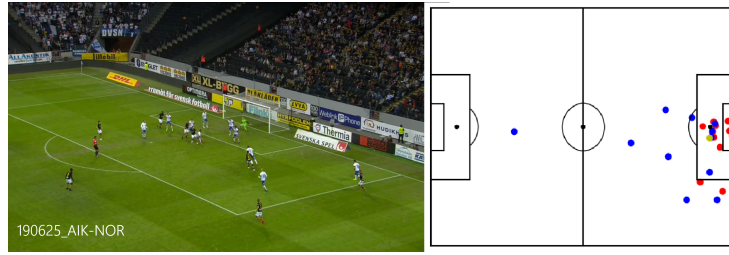


Figure 7: Position of each player(i).

and another one is shown in Figure 8.

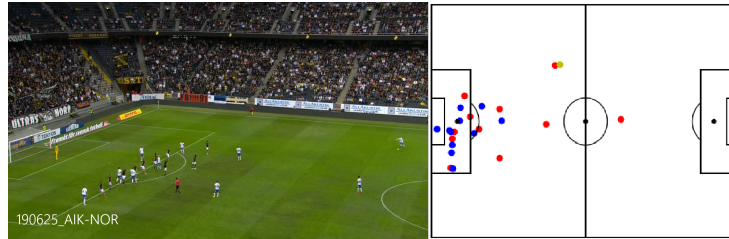


Figure 8: Position of each player(ii).

After finishing the searching process described above, a total of 3485 available results were found, from which data were extracted. From the data the number of opponents inside the shooting range, given by the angle between the two goal posts (Figure 9), were found and put into a vector. Also, the distance between the player taking the shot and the nearest opponent were found and put into a second vector. These two features may influence the performance of the model and were hence added to the training data of the shot model created without accounting for the tracking data (shown in Figure 6).

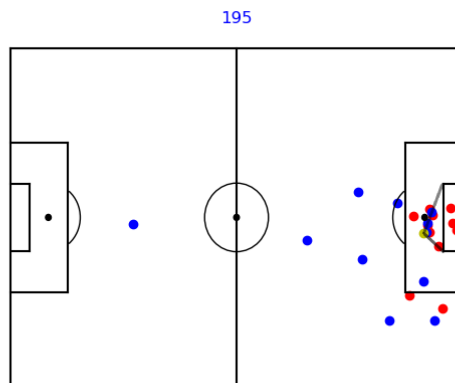


Figure 9: 7 opponents inside

2.2.3 Corners

One of the project goals was to utilise data analysis to investigate what makes a successful corner. In the data set, a corner is recorded as an event for the attacking team.

To avoid the accidental use of irrelevant events, only the possession chains containing a corner was extracted from the raw data. Furthermore, only events in that possession chain which took place after the corner kick were used in the analysis. The qualifiers of the corner kick event were saved, and if a shot or a goal was present in the following events, the corner data was labelled accordingly. In total, there were 7213 corner events, out of which 1810 corners lead to a shot and 151 corners lead to a goal.

One assumption made was that the corner qualities relevant for predicting shots and goals were *distance*, *angle*, *swing type* and *angle squared*.

An in-swinger is a corner taken such that the ball trajectory curls inwards towards the goal when entering the penalty box. On the other hand an out-swinger is a corner taken such that the ball trajectory curls away from the goal. A straight corner has no curl and a short corner is a pass which doesn't land in the penalty area.

The categorical values *in-swinger*, *out-swinger*, *straight* and *short* were transformed to one-hot-encoded values, see Table 1. This was done because the classification model needs the inputs to be numerical. The choice of one hot encoding over integer encoding ensures that the classifier does not rank the categorical values and assign higher importance to one of them because they happened to be encoded with an integer with higher value.

Table 1: One hot encoding of the swing type

Corner #	In-swinger	Out-swinger	Straight	Short
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1

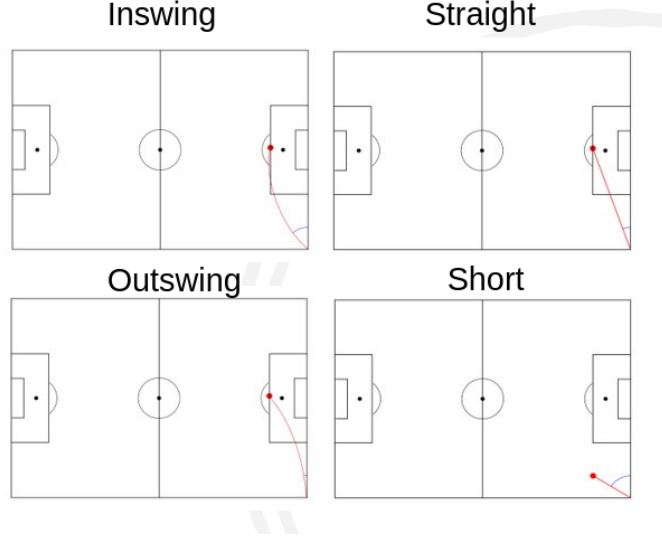


Figure 10: A graphical description of the different values of the corner qualifier *swing type*. The red line represents the ball trajectory. The blue curve indicates the angle of the trajectory. The red dot is the destination point of the ball.

The original x-coordinate, original y-coordinate and original angle (in degrees) were denoted as x , y and θ .

The second step was to project the coordinates from the range $[0 - 100]$ to be able to plot them on a pitch map. The pitch length was chosen to 130 m and the pitch width to 90 m. The coordinates were transformed as

$$x' = \frac{130x}{100}, \quad (1)$$

$$y' = \frac{90y}{100}. \quad (2)$$

As it turns out, the distance and angle qualifiers in the data set were not directly applicable to the real world. This led to the choice of using the x- and y-coordinates to compute both the distance and the angle.

The attacking direction in the data was supposed to always be from left to right, regardless of which team it was. A known error in the raw data was that sometimes the x-coordinates were flipped. One example was that a corner kick event had coordinates in the bottom left corner, but the corner destination coordinates were in the penalty area on the right hand side of the pitch. This could lead to massive computational errors for the distances and angles. To correct this, all event coordinates with an x-coordinate of less than half of the pitch length were mirrored across the middle line of the pitch. The distances d were computed as

$$d = \sqrt{(x - x_0)^2 + (y - y_0)^2}, \quad (3)$$

where x_0 and y_0 are the coordinates of the corner kick. The angles θ were computed as

$$\theta = 90^\circ - \arctan(y - y_0) / (x - x_0), \quad (4)$$

where x and y are the coordinates of the corner destination point. The shift of 90° was done because the angle was defined w.r.t. the goal line as in figure 10.

2.2.4 Data Set Balancing and the confusion matrix

From the training data it became obvious that we were dealing with an imbalanced data distribution. The number of instances where the goal event is labelled '1' is much smaller than those labelled '0' (most shots do not result in goals and even fewer corners lead to goals). This is problematic using the usual machine learning methods since these tend to be biased towards the majority class (ignoring the minority class) by assigning almost all new data point to the majority class in an attempt at increasing accuracy and will have major misclassification of the minority class (compared to the majority class). In order to use the machine learning methods for this project (logistic regression and XGBoost) an explicit algorithm handling the imbalanced class distribution would have to be applied to the models [3].

The algorithm used to create a balanced training set was the Synthetic Minority Over-sampling Technique (SMOTE), which over-samples the minority class instances and makes them equal in number to the majority class by replicating them. The virtual training records are generated by linear interpolation for the minority class and by randomly selecting the k-nearest neighbours (one or more) for each instance in the minority class. The new training data set was now fit for machine learning [3].

To obtain a more comprehensive view of the classifications, one can plot the confusion matrix for the training data. The confusion matrix displays the actual labels versus the predicted labels (Table 3). The recall, which is the fraction of the actual goals truly predicted as such [9], is expected to increase for the minority class ('1'), as the data becomes balanced. On the other hand, a decrease in accuracy (the fraction of predictions the model predicted correctly) is expected, due to the fact that the classifier no longer assigns most of the test data points to the dominating majority class.

Table 2: Confusion matrix consisting of the number of records being true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN).

		Predicted label	
		$\hat{y} = 0$	$\hat{y} = 1$
True label	$y = 0$	TN	FP
	$y = 1$	FN	TP

In mathematical terms, the recall is hence given by

$$\text{Recall} = \frac{TP}{TP + FN}. \quad (5)$$

2.3 Corner models

The corner models were made in two steps. The first model tries to predict if a given corner leads to a shot or not. The second model tries to predict if a corner that leads to a shot also leads to a goal. For the corner models, *LogisticRegressionCV* [4] with the 'lbfgs' solver, 'l2' penalty and 10-fold cross validation was used.

2.3.1 Shots from corners

The first model calculates the probability of a shot given the corner qualities, or in mathematical terms:

$$P(\text{shot} \mid \text{corner qualities}),$$

where shot is a binary variable (0/1).

2.3.2 Goals from shots

The second model calculates the probability of a goal given a shot and the corner qualities, or in mathematical terms:

$$P(\text{goal} \mid \text{shot} = 1, \text{corner qualities}),$$

where goal is a binary variable (0/1).

2.4 Training a classifier

To train a supervised machine learning classifier, one must have access to a large labelled data set. The goal is to tune the parameters of the model to minimise some chosen error metric. The data set is split into a training set and a test set, where the test set usually is much smaller. The model learns from the training set and is evaluated on the test set. A perfect performance on the training but poor performance on the test set is often caused by overfitting. To avoid overfitting, there are several regularisation techniques available which will improve the performance on the test set.

One example used was k-fold cross validation. It splits the training set into k subsets, and uses k-1 of those for training and one for validation. The training is then iterated k times to ensure each subset has been used as the validation set.

Another regularisation technique used was 'l2' penalty, which adds a penalty for large coefficients to the cost function being minimised.

2.5 Logistic regression

To solve binary classification problems, one of the classical machine learning methods out there is the Logistic regression, which models a dependant variable (goal/miss) in terms of independent variables corresponding to the features in the training data set. It is part of the supervised learning family, meaning that it learns from labelled training data. Hopefully, the model will be good

enough to generalise and correctly predict the true class of the new unseen data.

Logistic regression gives the probability of the unseen data belonging to class '0'/'1'. It uses the idea behind linear regression

$$p(X) = \beta_0 + \beta_1 X_1 + \dots + \beta_p X_p,$$

which inserted in the *logistic function*, modelling the probabilities describing the possible outcomes (goals/miss) [4], yields

$$p(X) = \frac{e^{\beta_0 + \beta_1 X_1 + \dots + \beta_p X_p}}{1 + e^{\beta_0 + \beta_1 X_1 + \dots + \beta_p X_p}},$$

to guarantee an output always lying between 0 and 1, for any given input matrix X . Here, X consists of p features, corresponding to the features of the training data.

In order to implement a logistic regression model in Python the library scikit-learn was imported, where the method *LogisticRegression()* can be applied to estimate the coefficients $\beta_0, \beta_1, \dots, \beta_p$ by finding the coefficient vector β and constant c that minimises the expression within brackets [4]:

$$\min_{\beta, c} \left(\frac{1}{2} \beta^T \beta + C \sum_{i=1}^n \log(\exp(-y_i(X_i^T \beta + c)) + 1) \right)$$

2.6 XGBoost

XGBoost is an algorithm for classification that is currently dominating the field of applied machine learning. It is an open-source software library providing gradient boosted framework for Python and is an implementation of gradient boosted decision trees, i.e. it produces a prediction model in the form of an ensemble of weak prediction models of decision trees [10]. Mathematically it is more complicated, but can be explained briefly: setting the mean square error (MSE) as the loss function and calculating the derivative of the loss function, one can prove that the residual r_{im} equals minus gradient. In other words, adding one more tree to fit the residual it will be going towards the minus gradient direction. The procedure of the algorithm (Fig.11) can be described as follow: calculate the residual r_{im} of the current model, add another tree into the model to fit the error and update the model. Iterate until the number of trees reach the limit (a hyperparameter).

for $D = \{(x_i, y_i)\}_{i=1}^n$ *and loss function* \mathcal{L}
Initialize $\mathcal{F}_0 = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^n \mathcal{L}(y_i, \gamma)$
For $m := 1 \leftarrow M$ *do*
 1. *Compute gradient w.r.t* $\mathcal{F}_{m-1}(x_i)$
 $r_{im} = - \left[\frac{\partial \mathcal{L}(y_i, \mathcal{F}_{m-1}(x_i))}{\partial \mathcal{F}_{m-1}(x_i)} \right], i = 1, \dots, n$
 2. *Fit C&RT decision tree* $h_m(x)$ *to* $\{(x_i, r_{im})\}_{i=1}^n$
 3. *Compute multiplier* γ_m
 $\gamma_m = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^n \mathcal{L}(y_i, \mathcal{F}_{m-1}(x_i) + \gamma \cdot h_m(x_i))$
 4. *Update the model*
 $\mathcal{F}_m = \mathcal{F}_{m-1} + \gamma_m \cdot h_m$
end
return \mathcal{F}_M

Figure 11: Algorithmic procedure of XGBoost.

This greedy algorithm is used to search for a best splitting point. Thus the searching procedure is expected to be time-consuming and reach local optimal point instead of global optimal point.

2.7 ROC-curve and AUC

One way to evaluate the performance of the model is by the ROC-curve, or more accurately, the area under the ROC-curve (AUC). The ROC-curve plots the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings, where the TRP is equivalent to the recall in Eq. (4) and the FPR is given by

$$\text{FPR} = \frac{FP}{FP + TN}. \quad (6)$$

Equation (2) is equivalent to the fraction of the actual misses/non-goals truly predicted as such [9]. An example of a ROC curve is shown in Figure 12 (retrieved from the Results section).

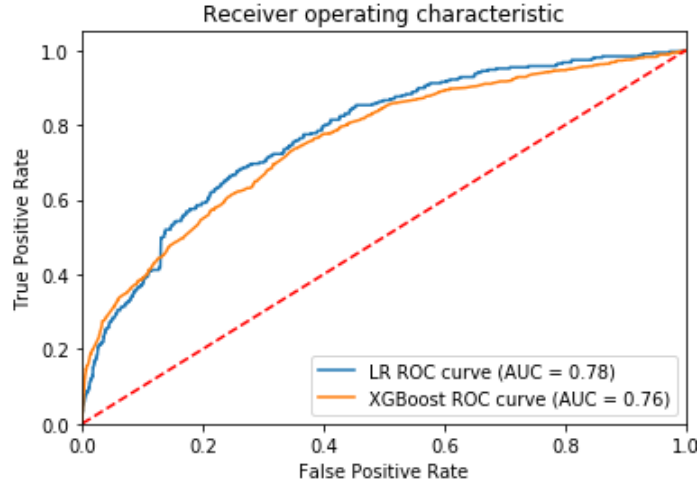


Figure 12: A ROC curve plotting the true positive rate versus the false positive rate at various threshold settings.

The AUC is equal to the probability that the classifier ranks a randomly chosen goal instance higher than a randomly chosen non-goal instance and for that reason frequently used for model comparison. An AUC of 0.5, yielded from the ROC curve represented by a diagonal dashed line in Figure 11, is equivalent to random guessing (uninformative classifier), while the AUC for an ideal classifier is equal to one. Most classifiers have AUCs laying somewhere in between, as will be seen in the Results.[8]

2.8 Scout report

To assemble the scout report, the corner events needed to be associated with the corresponding team. The aim was to be able to provide a simple yet non-trivial overview of the teams in Allsvenskan and their corner kicks. To compensate for the fact that some teams are relegated from and promoted to Allsvenskan after each season, some of the team statistics were normalised with respect to the actual number of games present in the data.

3 Results and discussion

In this sections some results from the implementation of the trained models can be seen, both in terms of accuracy and in the form of ROC curves with corresponding AUCs. The results are discussed further and comparisons are made between the shot models with and without tracking data, and the corner models. A comparison between the performance of logistic regression and XGBoost was done as well.

3.1 Shots

After implementing the SMOTE algorithm described in section 2.2.4 to the models the increase in recall became obvious (Table 3). For instance, for the

model using logistic regression the value of the recall ten-folded, i.e. going from mere 0.07 before the balancing of the training data to about 0.70 after SMOTE was applied. On the other hand, the accuracy has, as expected, decreased to reasonable values; 0.71 for logistic regression and 0.70 for XGBoost.

Table 3: Obtained results for recall and accuracy, before and after oversampling for recall and accuracy, respectively, when using logistic regression and XGBoost, respectively.

	Instances of '1'/'0'	Recall for '1'	Accuracy
After (before) oversampling using Logistic regression	14363 (1592) /14363	0.70 (0.07)	0.71 (0.89)
After (before) oversampling using XGBoost	8422 (963) /8422	0.67 (0.14)	0.70 (0.91)

The results from Table 4 suggests that logistic regression is slightly superior to XGBoost in terms of accuracy, i.e. out of the total number of predictions of the test data logistic regressions will have a larger percentage of truly predicted records. Furthermore, the AUC is seen in Figure 13 to be larger for logistic regression by 0.02, having an AUC of 0.78 compared to 0.76 for XGBoost.

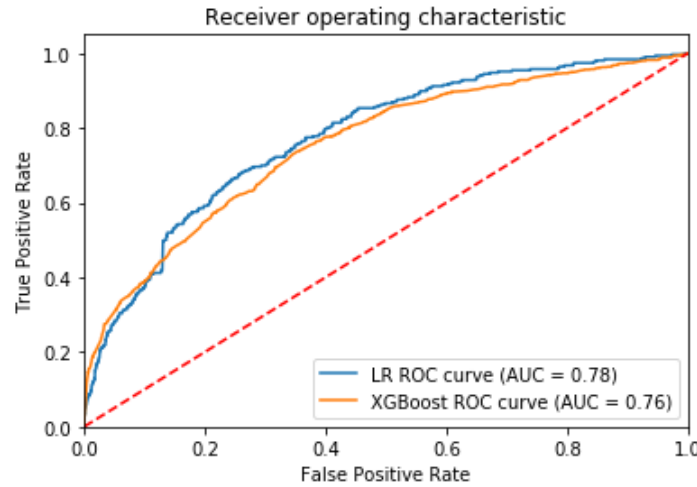


Figure 13: The ROC curves for the shot models (without using tracking data) using logistic regression and XGBoost, respectively, where the true positive rate has been plotted against the false positive rate at various threshold settings. The AUC for logistic regression is 0.78 and for XGBoost it is 0.76.

The AUCs retrieved from Figure 13 suggests that logistic regression is more capable of distinguishing between the classes. Overall, logistic regression seems to perform better than XGBoost for this model of predicting goals from shots excluding the tracking data due to the higher AUC value.

Compared to other models its performance in terms of AUC is among the worst (Figure 14).

Model	RMSE	McFadden	AUC
DeadSpin	0.301	0	0.5
Conversion Ratio	0.291	0.04	0.558
Standard	0.273	0.17	0.787
Stacked Equations	0.273	?	0.787
Big Chance Only	0.271	0.18	0.725
Standard + Big Chance	0.264	0.22	0.807
Martin Eastwood SVM	0.269	?	?
Perfect	0	1	1

Figure 14: AUC of all models

Unfortunately, due to the relatively small data set, the tracking data could not improve the value of the AUC (Fig15).

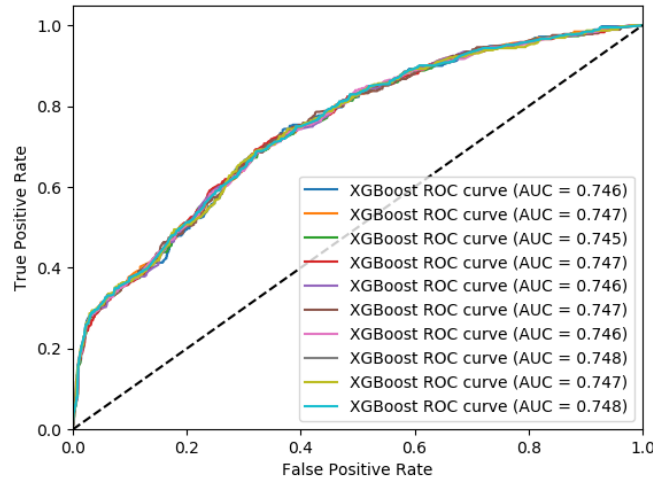


Figure 15: Several ROC curves corresponding to different simulations for the shot models using tracking data for XGBoost, where the true positive rate has been plotted against the false positive rate at various threshold settings. The average AUC is about 0.75.

The result from 15 is not surprising due to the fact that the event data only contained 540 matches for all three seasons, and the tracking data was for 174 matches only. Such small data size may not be sufficient for training a model with good generalisation ability. Furthermore, since the position of shots in event data were obtained by observers tapping on an iPad, the error is, to say

the least, considerable. The positions of the players in the tracking data is expected to deviate from the real positions of the players when the shot is taken according to the event data. For instance, a player may pass a position twice or even more during the time error between the two data sets, which makes it difficult to identify the true positions of the players. Even worse is the fact that the time error adds up during the matches, making it larger and larger for every shot instance (e.g. the total time error of the match AIK vs NORR is 45 seconds).

3.2 Corner models

The ROC-curve for the corner model predicting shots can be seen in Figure 17. In Figure 16a, one sees that the accuracy before using SMOTE is higher than it is afterwards for the Logistic Regression model and XGBoost model. This is due to the fact that the models classifies all test data as *miss*. The oversampling reduces the accuracy but increases the recall for *shots*. The performance of both LR and XGBoost are very similar.

The ROC-curve for the model predicting goals can be seen in Figure 18. In figure 16b, one sees that the increase in recall for *goal* is paid for by a decrease in accuracy. An important observation to make is that for the goal predictions, XGBoost has a higher accuracy than LR.

The XGBoost decision tree gives an insight of what makes a successful corner. The average number of shots per corner for the whole data set is 0.251. The average number of goals per corner and shot for the whole data set is 0.0834. In Figure 19 and Figure 20, decision trees for the two models are shown. Leaves number 5, 6 and 8 in Figure 19 gives the conditions under which the shots per corner ratio is higher than the average ratio. In Figure 20, leaves number 1, 3 and 6 gives the conditions under which the 'goals per corner and shot' ratio is almost twice the average ratio.

3.3 Scout report

Figures 21-25 show different statistics from the event data. The teams have four to six corners per game, but only about 20-30% of them leads to a shot. The best teams at scoring goals from corners are *Djurgårdens IF* and *IFK Norrköping*. Their goals per corner number is about 50% higher than the average of all the teams. A surprising insight regarding *Helsingborgs IF*: they have the highest goals per shot number, but the worst shots per corner number and corners per game number.

Figures 21-24 show the corner statistics for all teams who played in Allsvenskan during at least one of the seasons 2017, 2018 and 2019. Breaking it down to look at individual teams, the visualisation can look like Figure 25. The pie charts at Figure 25a shows the same data as figures 22-24, but in a different way. Figure 25b shows all of Hammarby IF's corner kicks, grouped by swing type. Out-swingers and in-swingers are the most common types of corner kicks.

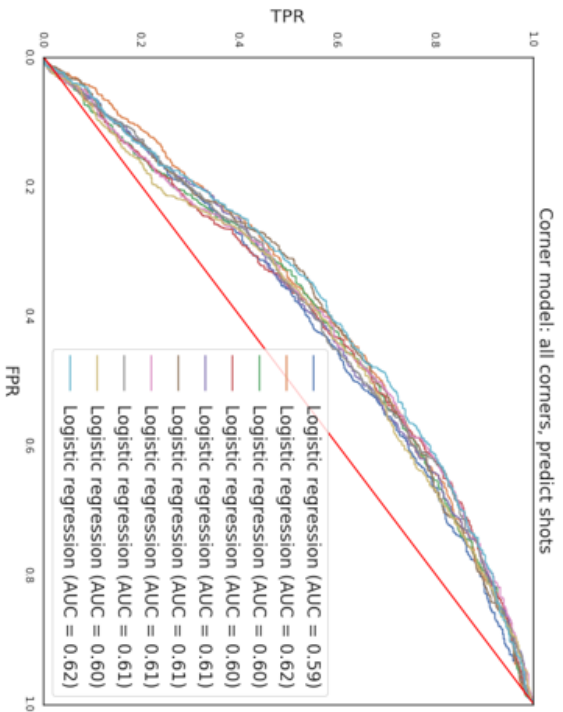
	Recall for '1'	Accuracy	Number of '1'/'0'
After (before) oversampling for Logistic regression	0.56 (0.00)	0.61 (0.75)	3784 (1265) / 3784
After (before) oversampling for XGBoost	0.51 (0.01)	0.62 (0.76)	3788 (1261) / 3788

(a) Table of the recall, accuracy and number of elements in each class for the shot prediction model. The first row shows the result using Logistic Regression. The second row shows the result using XGBoost.

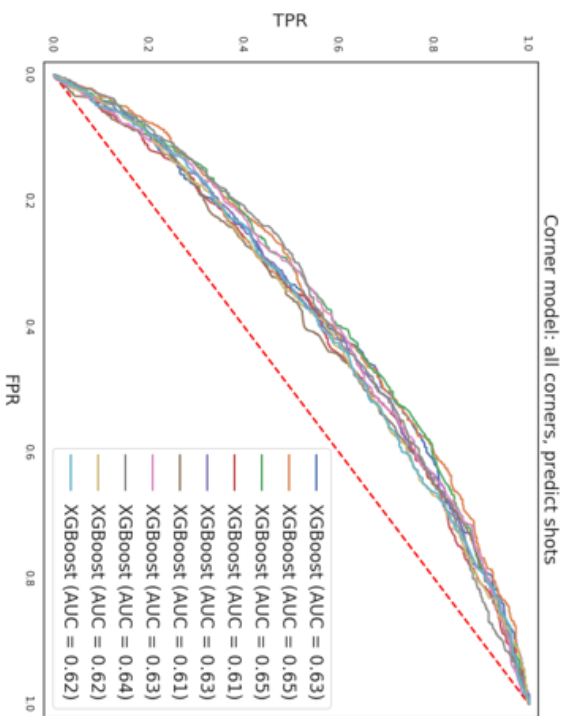
	Recall for '1'	Accuracy	Number of '1'/'0'
After (before) oversampling for Logistic regression	0.64 (0.00)	0.62 (0.91)	1160 (107) / 1160
After (before) oversampling for XGBoost	0.35 (0.00)	0.72 (0.89)	1176 (91) / 1176

(b) Table of the recall, accuracy and number of elements in each class for the goal prediction model. The first row shows the result using Logistic Regression. The second row shows the result using XGBoost.

Figure 16: Tables describing the two corner models: the shot prediction model and the goal prediction model. After oversampling, the recall for '1' is increased and the accuracy is decreased for both LR and XGBoost.

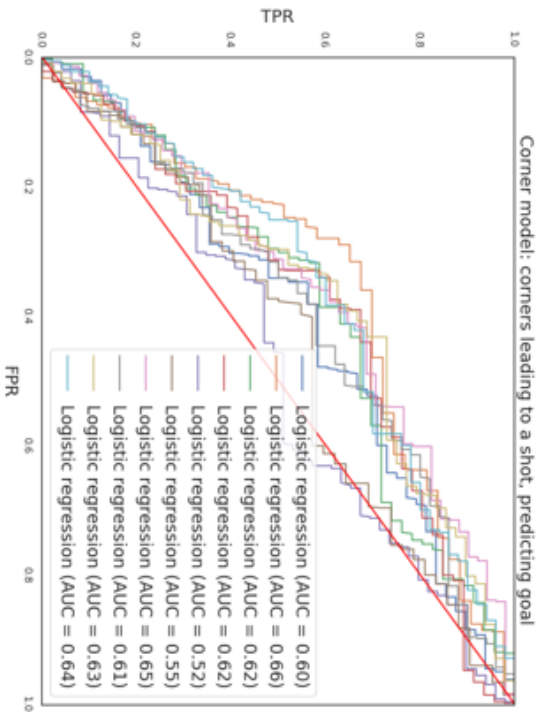


(a) The ROC curve for the corner shot model using logistic regression. The average AUC is about 0.61.

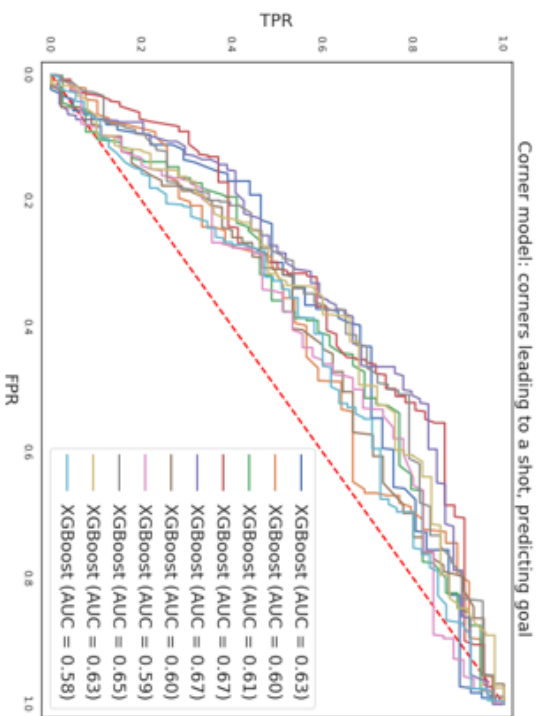


(b) The ROC curve for the corner shot model using XGBoost. The average AUC is about 0.63.

Figure 17: ROC curves for the corner shot model.



(a) The ROC curve for the corner goal model using logistic regression. The average AUC is about 0.61



(b) The ROC curve for the corner goal model using XGBoost. The average AUC is about 0.62.

Figure 18: ROC curves for the corner goal model.

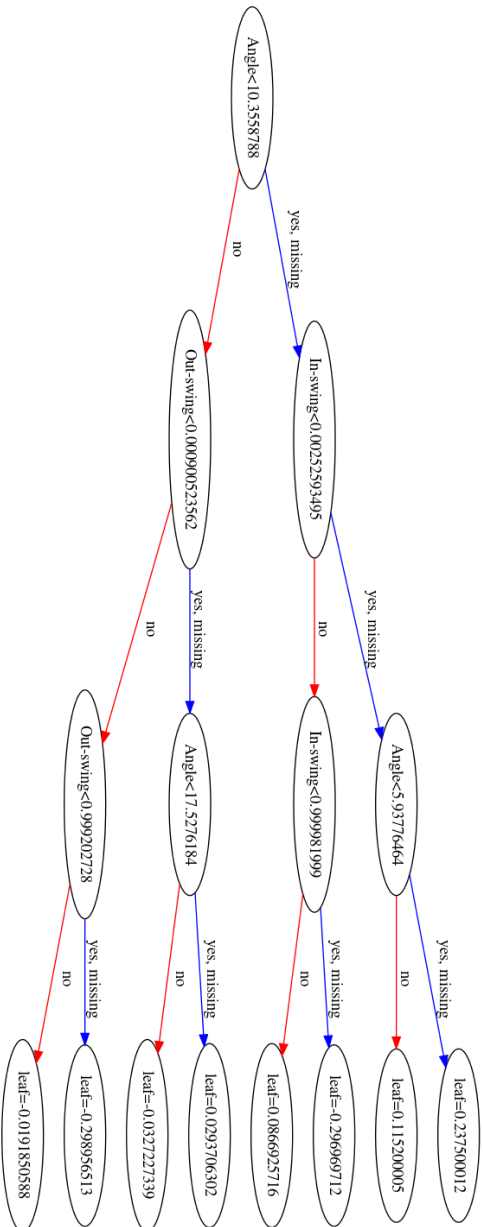


Figure 19: Decision tree for the corner model predicting shots. Leaves number 5, 6 and 8 represent subsets of the data set where the shot per corner is higher than the average of the whole data set.

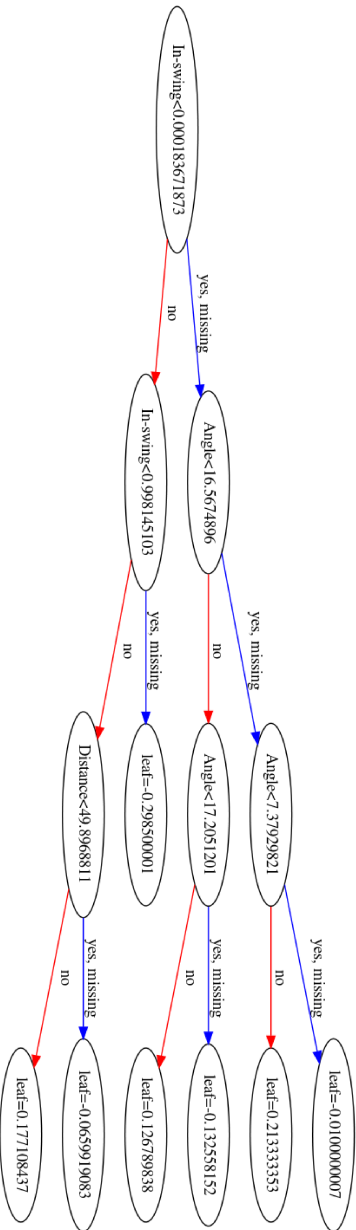


Figure 20: Decision tree for the corner model predicting goals. Leaves number 1,3, and 6 represent subsets of the data set where the goals per corner is higher than the average of the whole data set.

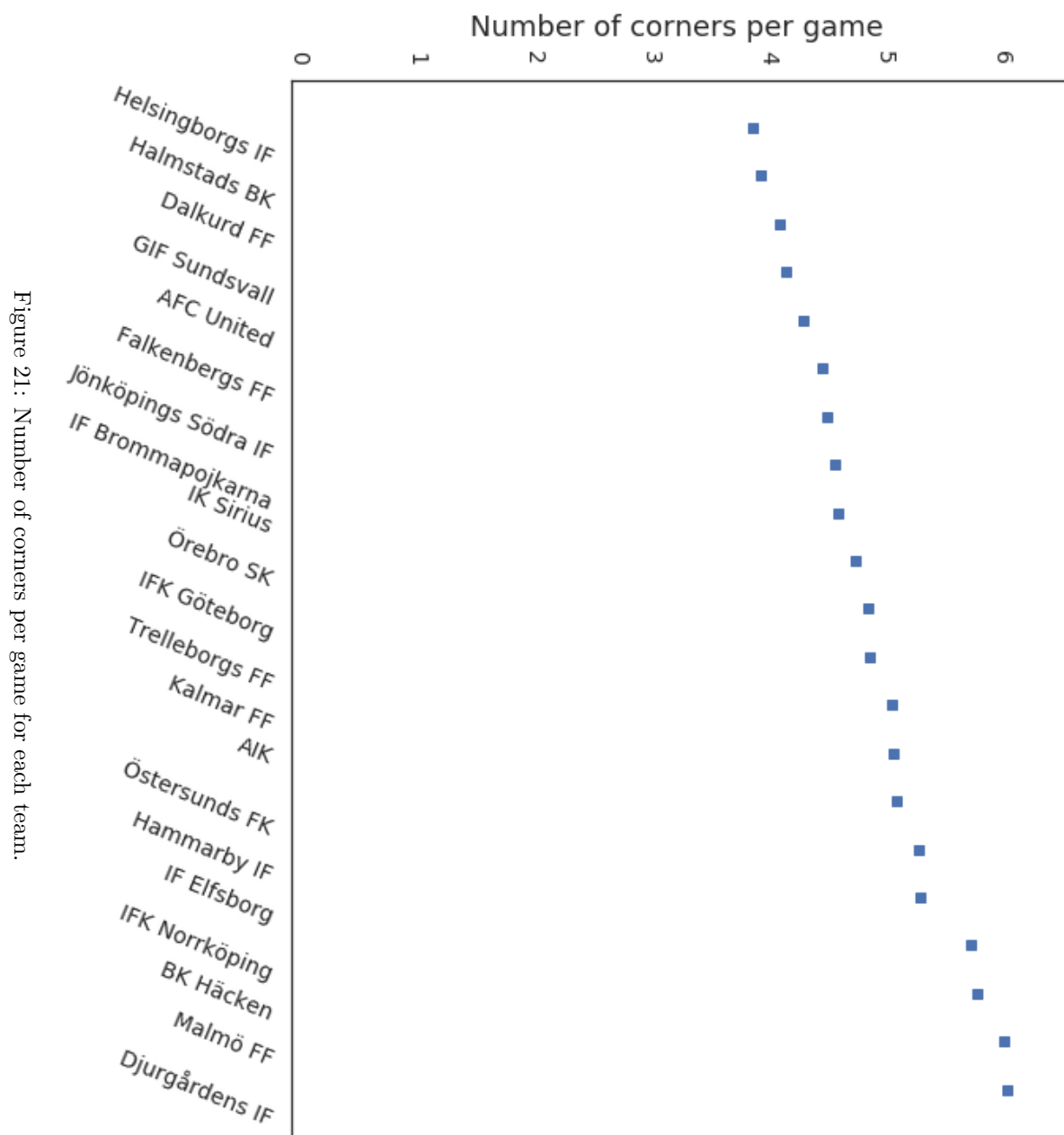
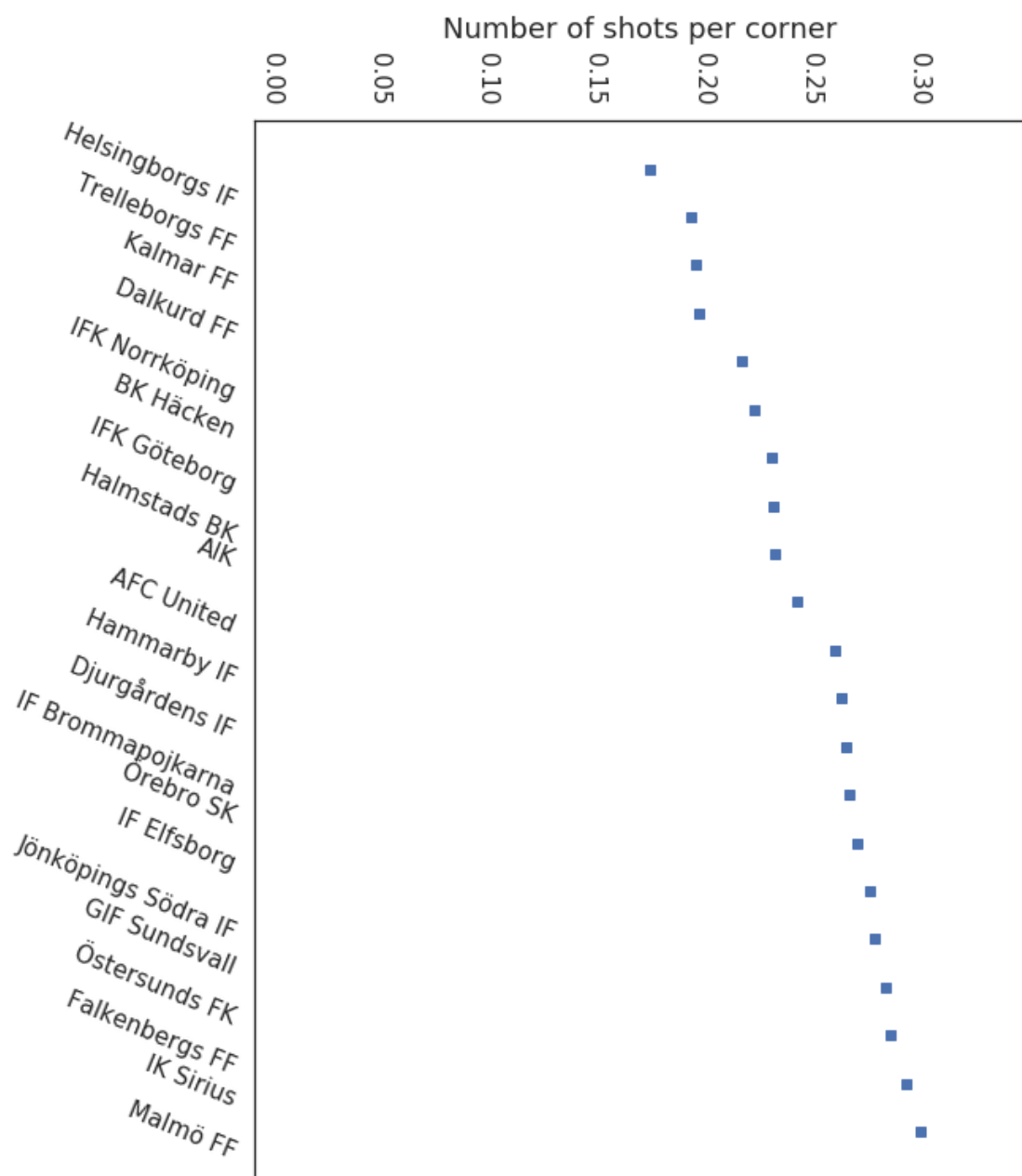


Figure 21: Number of corners per game for each team.

Figure 22: Number of shots per corner for each team.



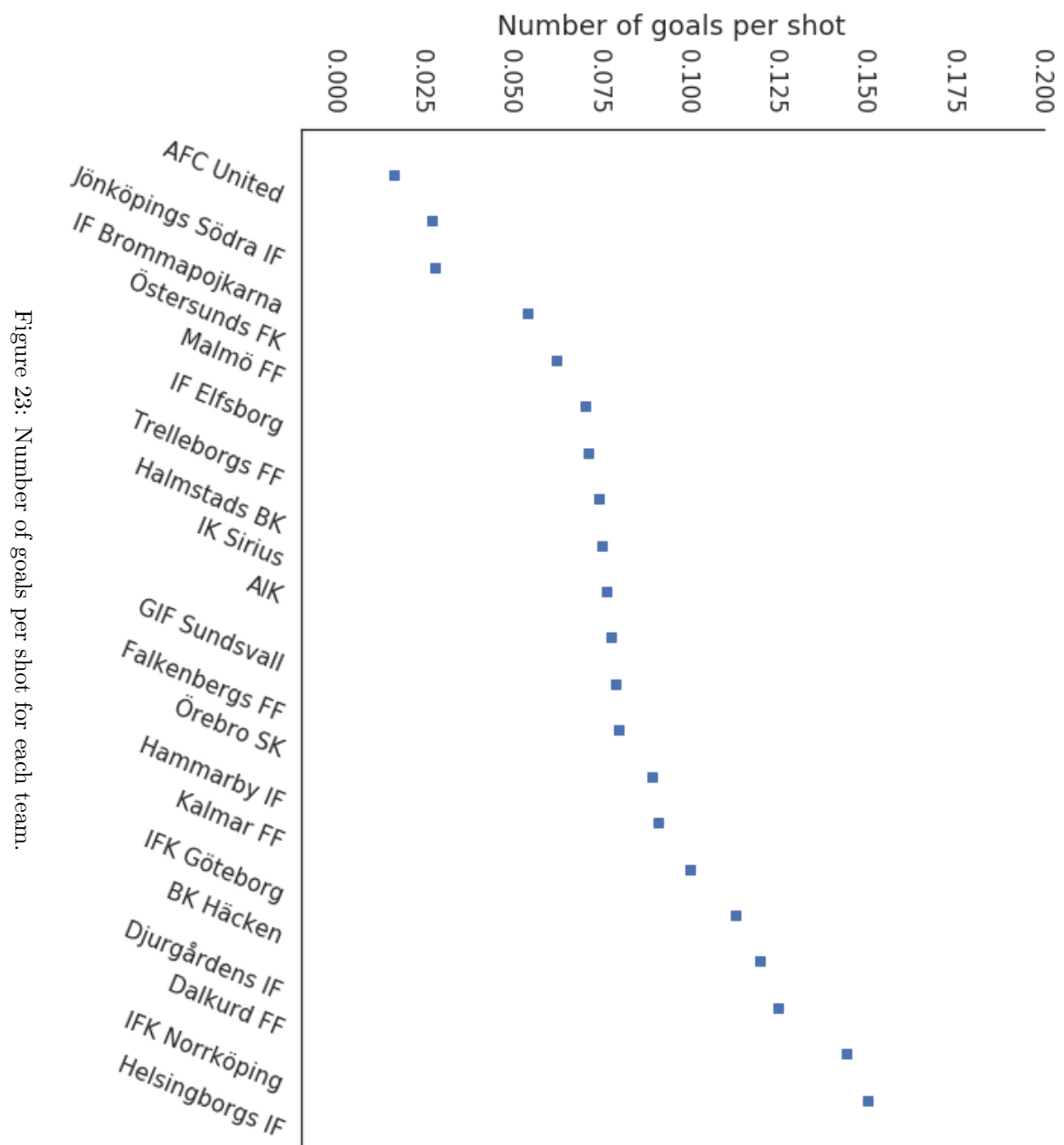


Figure 23: Number of goals per shot for each team.

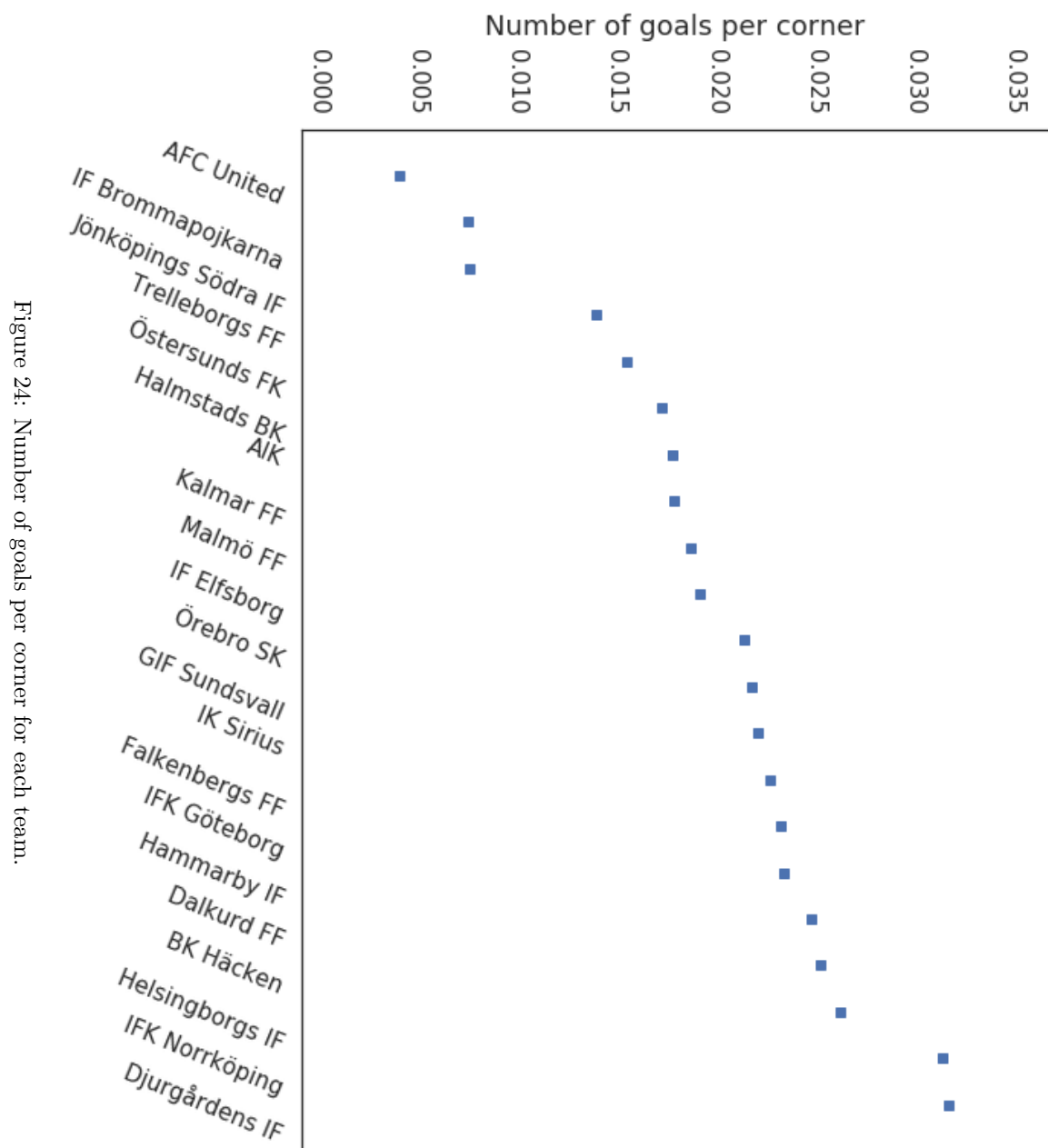
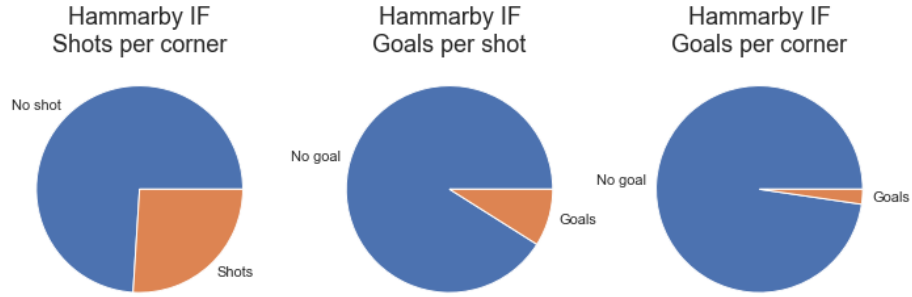
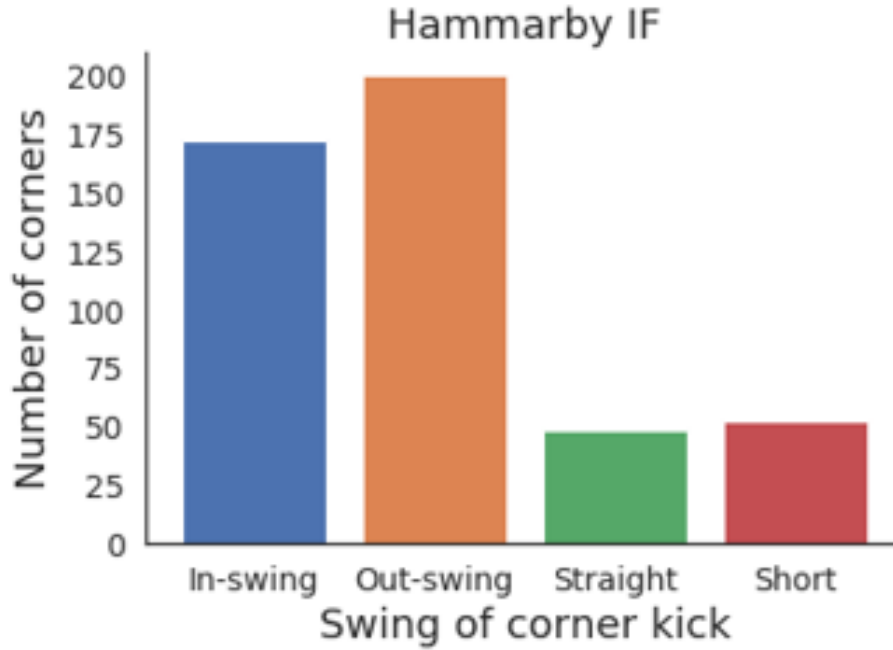


Figure 24: Number of goals per corner for each team.



(a) Pie chart showing the fractions of shots per corner, goals per shot and goals per corner for Hammarby IF. The actual numbers are 473 corners, 123 shots and 11 goals.



(b) Number of corners for Hammarby IF, grouped by swing type.

Figure 25: More detailed corner statistics about Hammarby IF.

4 Conclusions

Out of the classifiers used for the shot models, Logistic Regression seemed to be best suited to distinguish between goals and misses due to a larger AUC. Using tracking data did unfortunately not improve the performance of the shot models, which can be explained through various factors, where the main ones being insufficient size of the data sets for training the models with good enough generalisation ability and the fact that the positions in the tracking data deviates from the real positions of the players when the shot is taken because of the time delay of the observers manually keeping track of the positions of the players. To improve the shot model more features given by the OPTA qualifiers

can be taken into consideration under the premise that a larger, good quality tracking data set is used.

The corner models tell a different story. For both predicting shot and predicting goals, XGBoost performed a little better than Logistic Regression. The decision trees from XGBoost made it possible to give an answer to the question ‘What makes a successful corner?’ asked in the introduction. As a potential aid to the coaches of Allsvenskan, the scout report has some interesting data about the other teams. It is easy to see how well each team performs at taking corners, and also compare the teams with each other. With all the data already available, the possibility to add more custom visualisations is enormous.

References

- [1] R. P. Bunker and F Thabtah. “A machine learning framework for sport result prediction”. In: *Applied Computing and Informatics* 15.1 (2017), pp. 27–33. DOI: <https://doi.org/10.1016/j.aci.2017.09.005>.
- [2] FCPython. *Drawing a Pitchmap â Adding Lines Circles in Matplotlib*. URL: <https://fcpython.com/visualisation/drawing-pitchmap-adding-lines-circles-matplotlib>. (accessed: 21.01.2020).
- [3] GeeksforGeeks. *ML — Handling Imbalanced Data with SMOTE and Near Miss Algorithm in Python*. URL: <https://www.geeksforgeeks.org/ml-handling-imbalanced-data-with-smote-and-near-miss-algorithm-in-python/>. (accessed: 21.01.2020).
- [4] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.
- [5] D. Sumpter. *The Geometry of Shooting*. URL: <https://medium.com/@Soccermatics/the-geometry-of-shooting-ae7a67fdf760>. (accessed: 07.01.2020).
- [6] Twelve. *How to use Twelve’s algorithm to analyse players*. URL: <http://blog.twelve.football/about-us/>. (accessed: 20.01.2020).
- [7] S. Varrall. *OPTA Qualifiers*. URL: <https://github.com/fluidpixel/RePlayed/wiki/OPTA-Qualifiers>. (accessed: 07.01.2020).
- [8] Wikipedia. *Receiver operating characteristic*. URL: https://en.wikipedia.org/wiki/Receiver_operating_characteristic. (accessed: 21.01.2020).
- [9] Wikipedia. *Sensitivity and specificity*. URL: https://en.wikipedia.org/wiki/Sensitivity_and_specificity. (accessed: 21.01.2020).
- [10] Wikipedia. *XGBoost*. URL: <https://en.wikipedia.org/wiki/XGBoost>. (accessed: 23.01.2020).