Predicting shot probabilities and analyizing match outcomes

STAT 380 Group 4

Maxwell Gerhart, Rohan Sampath, Jaikhush Thakkar, and Ryan Kaye

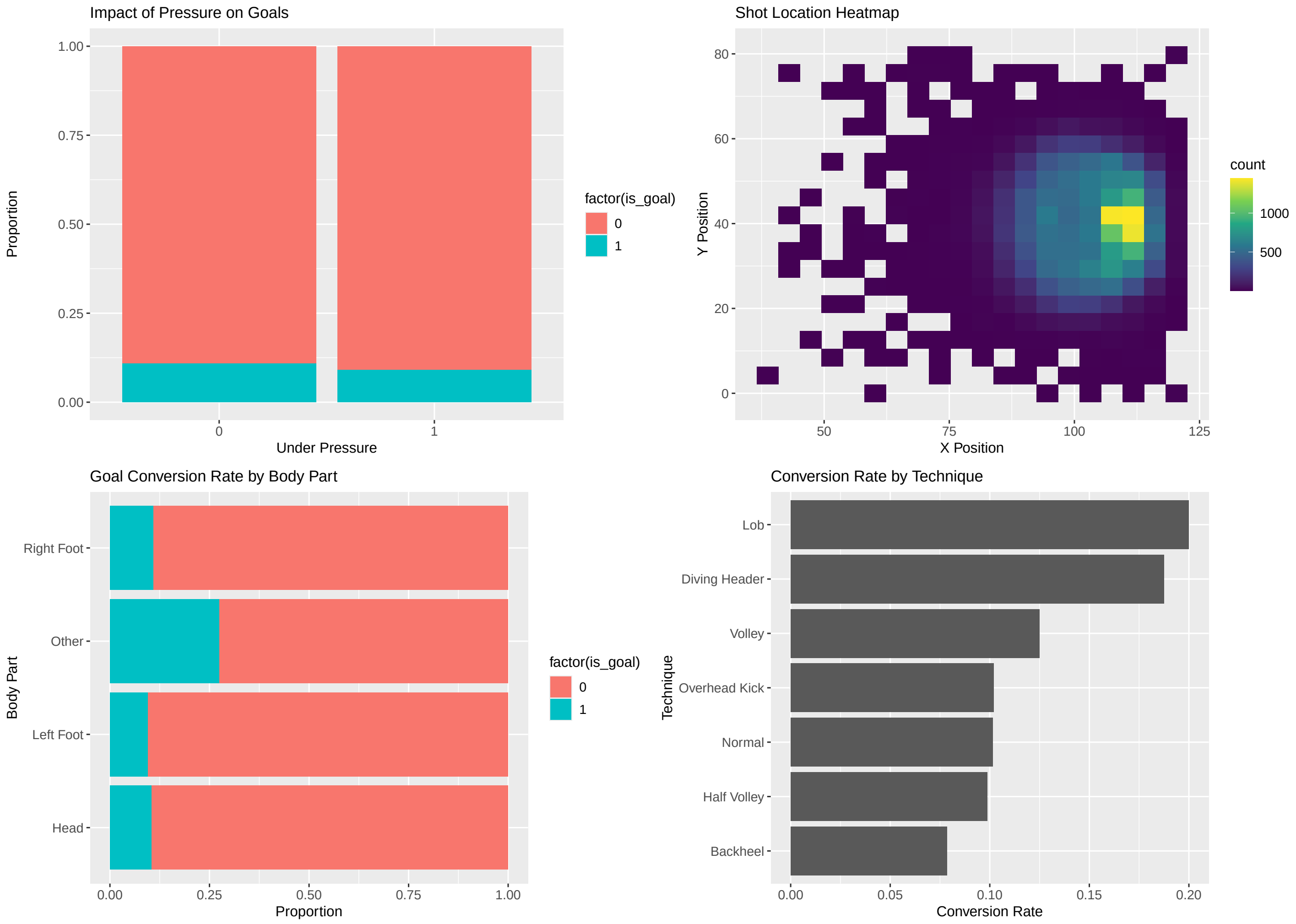
Our goal for the project was to create a model to calculate the probability of a shot resulting in a goal. In soccer this metric is commonly known as expected goals (xG for short). We plan on creating three models: a logistic regression model, Generalized Additive Model (GAM), and random forest model. Once we select the best model, we will put this model to use by calculating the expected points of an entire league season. Calculating expected points requires running simulations based on the shot xG values in order to calculate the proportion of wins, losses, and draws for each team.

$$\text{Expected Points} = 3 \times \text{Win Proportion} + 1 \times \text{Draw Proportion} + 0 \times \text{Loss Proportion}$$

This lets us determine what teams performed as expected and which did better or worse. The reason for the variance could be caused by many factors, but over time teams should regress towards their expected value.

The data is open to the public through the StatsBombR package in R. It includes detailed information about different competitions and the matches and events inside each competition. For our purposes we will filter down the event level data down to only shots. There are many columns within the event dataframe that don't apply to shots so we can remove those. In soccer there are five leagues that are consistently ranked as the top five: Premier League (England), La Liga (Spain), Bundesliga (Germany), Serie A (Italy), and Ligue 1 (France). We have chosen to train our model on the four other leagues outside the Premier League as it is reasonable to assume the player ability is similar between all five.

In terms of preprocessing the data was relatively clean, but there were a lot of columns unrelated to the event type shot so we removed those. Now we had a detailed, but simplified set of columns location, shot outcome, under pressure, body part, technique, and play pattern. From this we dummified shot the categorical variables and transformed location into distance and angle to the center of the goal.



The dataset contains 9908 shots where 988 resulted in goals meaning that approximately 10% of shots resulted in a goal. This means accuracy won't be the best for model assessment. The top left plot shows that the impact of pressure is surprisingly insignificant. The heat map on the top right shows the frequency of where shots were taken. From this we can see shots centrally and high into the attacking half which is what we would expect. The bottom left shows the conversion of different body parts they are all very even except for the other value which likely repersents very close shots from the torso bundled towards goal. The bottom right plot shows the conversion rate of different shot techniques surprisingly normal is towards the lower end. This is likely due to the fact that the more rare shot techniques are used in situations where a goal is more likely.

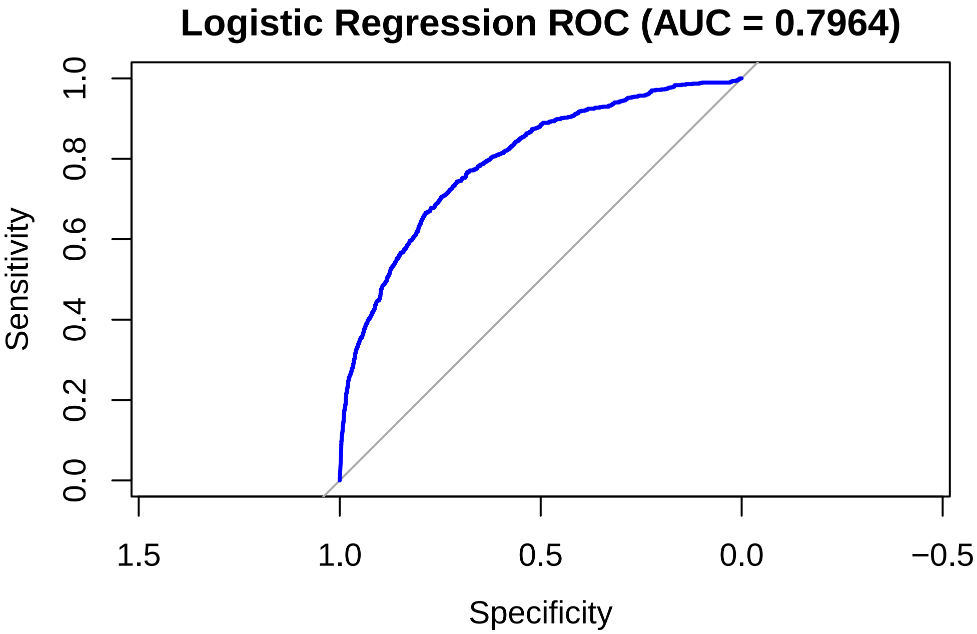
Logistic Regression

For the initial approach we began with a Logistic Regression model which is an extremely commonly used model for prediction tasks. We wanted to create a baseline model that would help us predict the expected goals scored in a total match by focusing on calculating the probability of a shot resulting in a goal. This is a great match because Logistic Regression works with binary factors, and in our case each shot would be either a Goal or Not a goal depending on the outcome of the shot.

This is a model we have seen many times in class, and follows the basic commonly used pattern for logistic regression tasks. The incorporation of these variables works to help us to estimate and understand how each characteristic affects the probability of a shot resulting in a goal. For example the numerical variables such as the location variables work to tell us the coordinates of where on the field the shot is being taken from. Naturally, a shot attempt right in front of the goal is far more likely to result in a goal compared to a shot taken from the midfield line. Similarly the angle in which the shot was taken has a major influence on the prediction we decide on. When looking at the categorical predictors such as body parts or under pressure they can provide additional context for the model..

$$\text{logit}(P(\text{goal} = 1)) = \log\left(\frac{P(\text{goal} = 1)}{1 - P(\text{goal} = 1)}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots$$

This equation shows how logistic regression connects the predictors in our dataset to the probability of a goal. The model first forms a linear combination of all shot features, and then the logistic function converts that linear score into a probability between 0 and 1. This transformation is what allows the model to handle a binary outcome like scoring: no matter how large or small the linear predictor becomes, the logistic curve ensures the final output behaves like a valid probability. In our context, this means each variable contributes additively to the underlying scoring tendency, and the logistic function maps that tendency into an interpretable xG value for every shot.



The Logistic Regression model had an **AUC of 0.7964**, which is a strong score from the supervised learning method. The result is to be expected, as with using a linear model, we can capture broader relationships, but struggle to fully capture the more complex directional relationships.

Naturally, when looking at our results, the spatial predictors were not the most influential factors. In fact, the highest few predictors in terms of importance were shot techniques, such as diving header or lob. Although distance is not clearly associated, this correlation makes sense because a header would almost exclusively be tried when very close to the goal. Similarly, the purpose of a lob is a slow shot that's meant to go above the goalkeeper. A shot of this kind would only be effective if very close to the goal, so we know that a lob shot is very informative when trying to calculate xG.

Overall, the Logistic Regression model does a very good job of predicting and is useful for comparative tasks. However, the linear structure limits the ability to truly capture spatial complexity in the decision making process, limiting top end results.

GAM

To model the probability that a shot results in a goal, we used a **Generalized Additive Model (GAM)**. A GAM extends logistic regression by allowing key predictors to influence the response through smooth nonlinear functions rather than fixed linear slopes. This makes GAMs well suited for soccer analytics, where variables like shot distance and angle are known to have curved, diminishing effects on scoring probability.

Conceptual Description of the Method

Standard logistic regression models the log-odds of scoring as:

$logit(P(\text{goal})) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots$

This assumes that each one-unit increase in a predictor has the **same effect everywhere**, which is unrealistic in soccer:

- Going from **10 to 15 yards** drastically lowers scoring probability
- Going from **45 to 50 yards** changes almost nothing

A **GAM** relaxes this assumption by modeling distance and angle using smooth spline functions rather than linear terms.

Our model takes the following form:

A spline function can be expressed as:

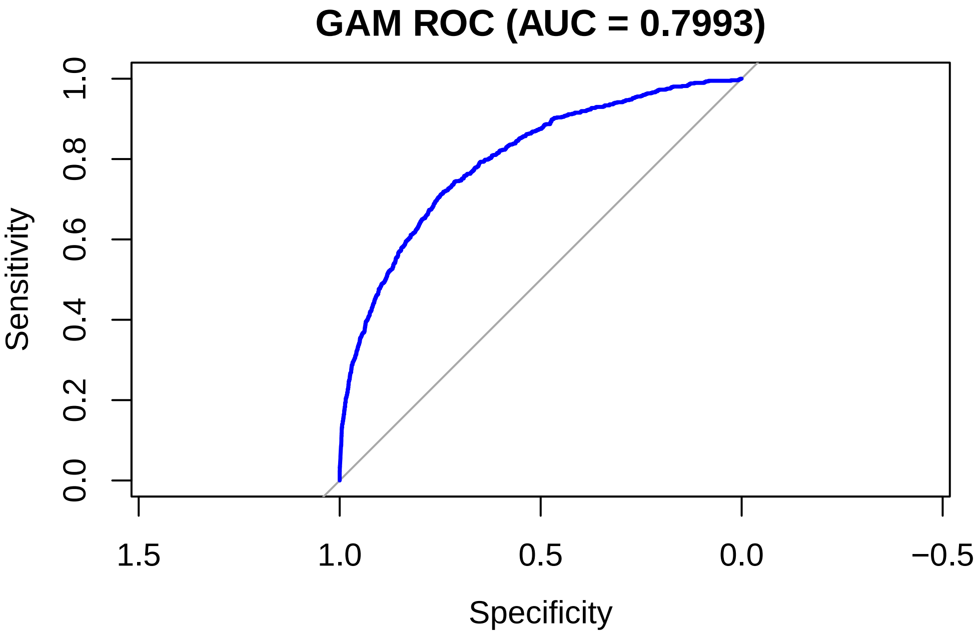
$$s(x) = \sum_{k=1}^{K} b_k(x)\theta_k$$

where:

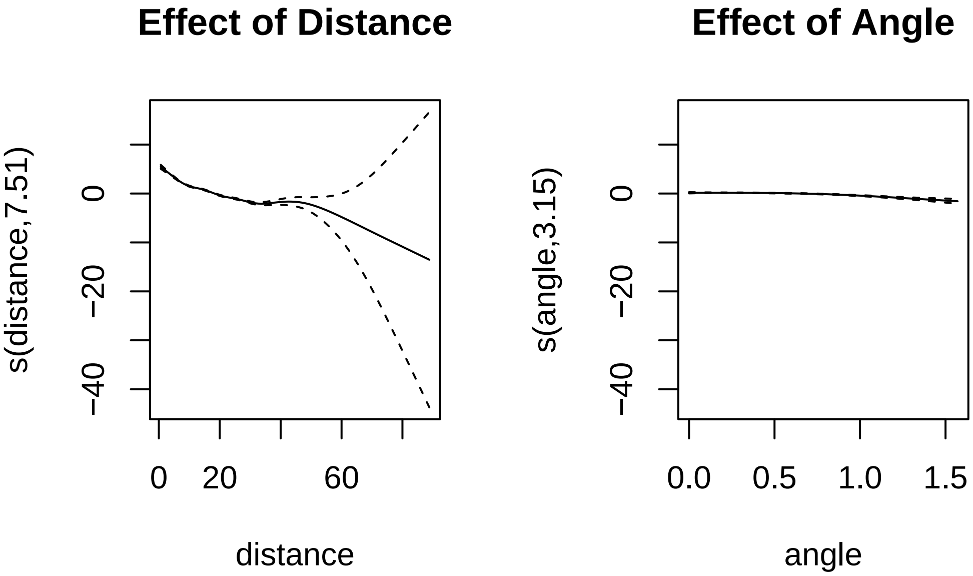
$b_k(x)$ are spline basis functions (piecewise cubic polynomials), θ_k are coefficients estimated during training, and smoothness is enforced by penalizing rapid curvature (wiggliness).

This structure allows the GAM to learn realistic curves for distance and angle without requiring manual transformations. Categorical Predictors include variables such as under_pressure, shot_body_part, shot_technique, and play_pattern. These enter the model as parametric (linear) terms represented by dummy variables. These terms adjust the baseline scoring probability depending on shot context.

We selected GAM for four main reasons. First, it captures nonlinear shot difficulty because distance and angle do not affect scoring probability linearly, and GAM automatically learns the correct curved relationships. Second, it produces interpretable visual outputs, since smooth functions produce partial-effect plots that directly show how distance and angle affect scoring probability. Third, it is widely used in real soccer analytics because GAM-based models are standard in academic and professional xG research. Fourth, it satisfies project requirements. GAM qualifies as a method not covered in class but remains transparent and statistically grounded.



The GAM was trained on shots from four major European leagues (2015–16 season) and evaluated on a held-out test set. The model achieved **AUC = 0.7993**. This indicates strong discriminative performance. AUC values near 0.80 are typical for event-only expected goals models and show that our predictors, especially distance, hold meaningful predictive value.



The Distance Smooth is the most influential variable in the model. There is a sharp drop in scoring probability within 0 to 20 yards and a gradual decay beyond 25 to 40 yards. There is near-zero probability beyond 40 yards, with wide confidence intervals due to fewer observations. This matches established soccer analytics research. The Angle Smooth shows very little variation in scoring probability across its range. The effect remains zero with a slight downward dip that suggests that angle does not affect scoring as strongly as distance. This makes sense as more centralized shots tend to have higher scoring probability and narrower angles reduce scoring likelihood.

The linear terms provide contextual adjustments. Headers typically lower xG. Shots under pressure decrease scoring likelihood. Shot technique influences difficulty such as volleys compared to placed shots. Play pattern matters such as fast breaks versus settled play. These effects shift the baseline probability but do not require nonlinear modeling.

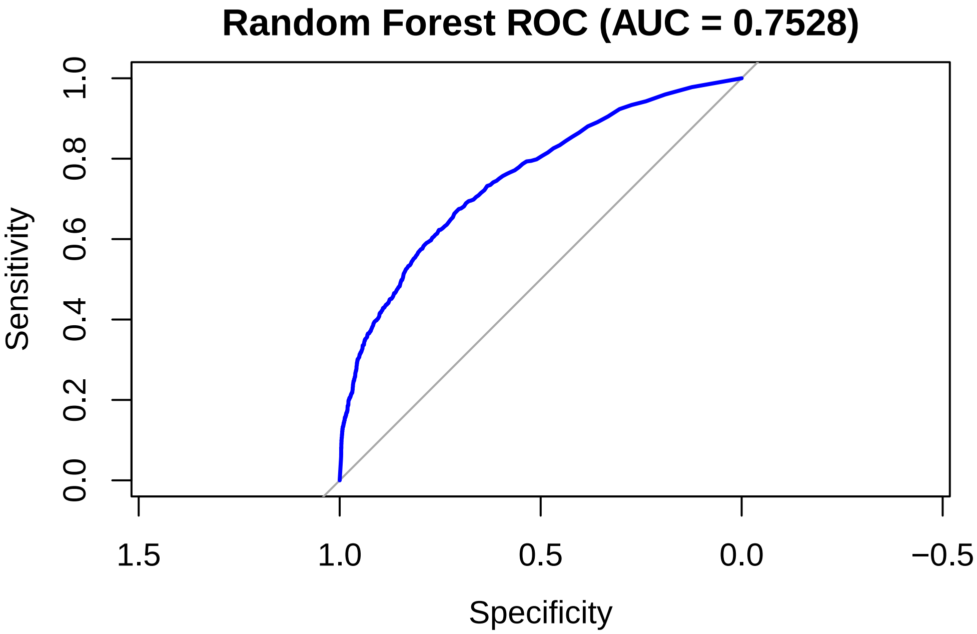
The GAM effectively models nonlinear spatial effects using cubic and TPR splines, balances interpretability with flexibility, produces realistic xG curves consistent with soccer knowledge, achieves strong predictive accuracy with an AUC around 0.80, and provides a strong baseline for comparison with more advanced models. This makes GAM a strong and well-justified modeling choice for expected goals prediction.

Random Forest

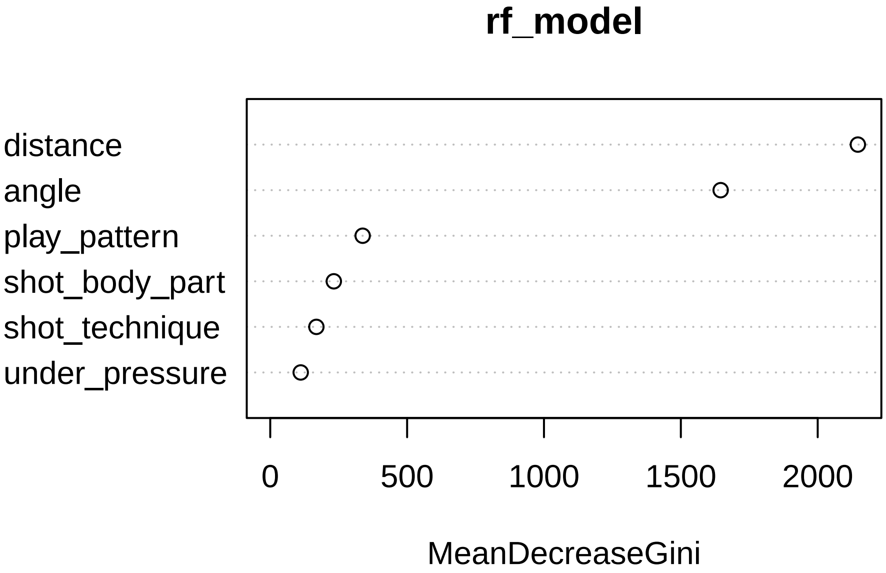
For the last model here we wanted to employ a Random Forest classifier to help with our prediction goals. We maintain our goals with the task of this ensemble model with the use of many decision trees, and finding the average of our results. This is a good fit for this problem because we have many predictors and variables, this way we prevent overfitting issues and improve accuracy compared to a single decision tree. Similarly to GAM this works to help capture nonlinear patterns and interactions.

A Random Forest works by building many different decision trees and then combining their predictions to produce a more stable and accurate result. Each tree is trained on a slightly different sample of the data and considers only a random subset of predictors at each split, which helps the model reduce overfitting and capture complex relationships that a single tree would miss. By averaging the predictions from all trees in the forest, the model is able to generalize better and handle nonlinear patterns in the data, making it well suited for a task like estimating xG where many variables interact in non-linear ways.

We still employ the same predictors as aforementioned, maintaining consistency and reproducibility in our approach. For random forest we also made sure to convert to factors to appropriately run the model. That way we have a predictor for not a goal and goal scored, helping to effectively create our xG. We used 500 trees in our model, and kept the mtry value to 3.



The Random Forest Model produced an **AUC of 0.7528**. This is a relatively strong predictor, showing the model does well when trying to predict whether or not a goal is scored. Although it is not on the level we achieved with GAM, we can still see that the model can meaningfully differentiate between a good and a poor shot.



Furthermore, when taking a deeper dive into our visualization, which we created above, we can analyze the importance of each predictor based on MeanDecreaseGini. As you would naturally assume, the most important factor was the distance, which is how far away the shot is taken from the location of the goal. Then, not too far behind, was the angle at which the shot was taken towards the goal. This makes sense because a shot taken heavily from one side would allow the goalkeeper to position themselves in a manner where they are essentially covering all the area the ball could travel to based on the angle.

Overall, the use of the Random Forest was successful and allowed us to compute xG values in an accurate and meaningful way. We were able to compute values for each shot that align with historical data, and have a quality accuracy measurement in AUC. Although it was not the best performance, it is reliable and a good tool to have when comparing between models.

Model Selection

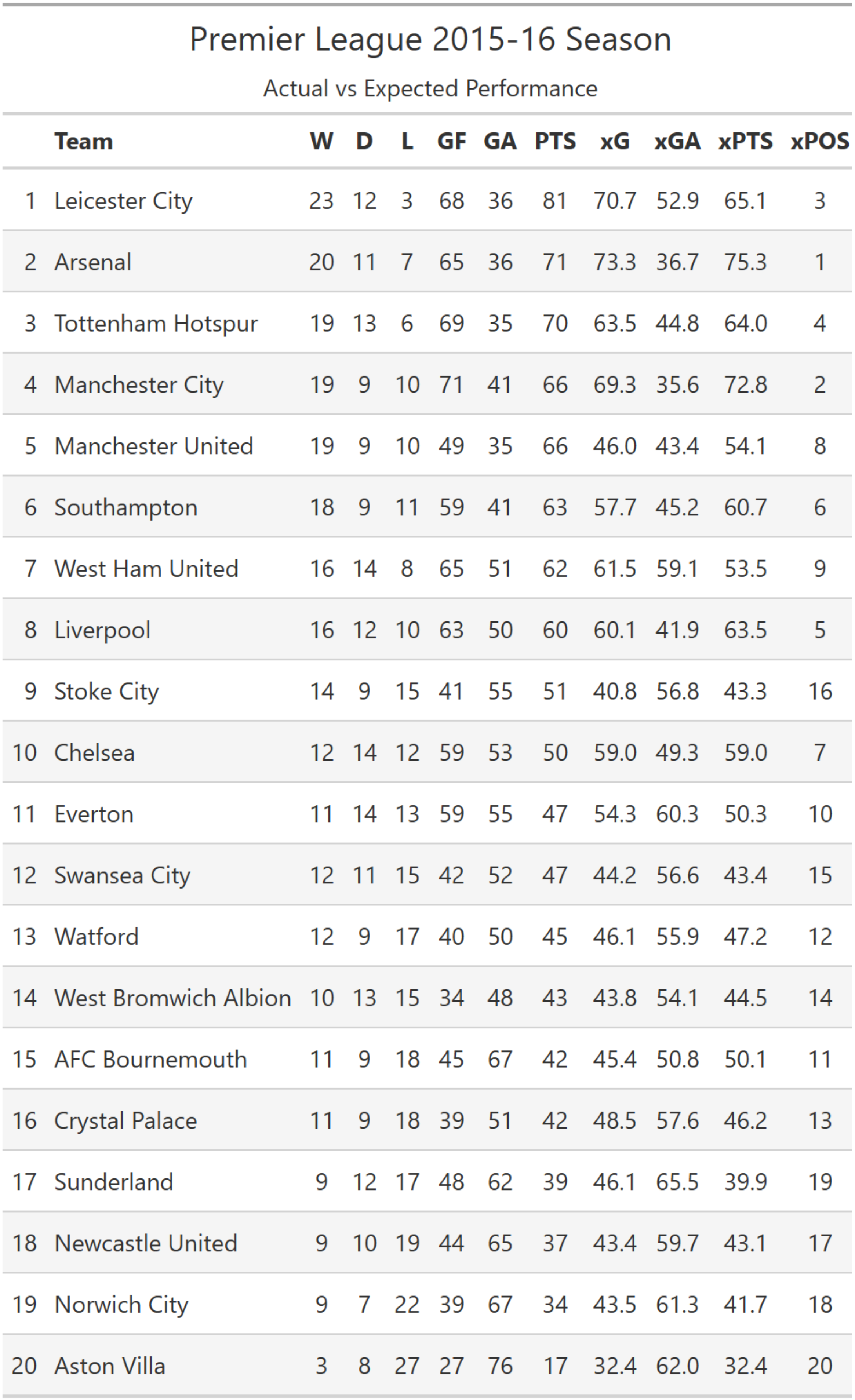
Overall, we had three separate models that all worked well. However, one model stood out among the rest. For our Model Selection, we chose the GAM model due to its improved performance and superior interpretability. The strongest metric we used across all three models was AUC. Our Logistic Regression performed well with an AUC of 0.7964. Similarly, the Random Forest model outputted an AUC of 0.7528. However, our GAM model stood far above the rest, with an AUC score of 0.7993. Another key reason we chose the GAM model was its ability to capture nonlinear and spatial patterns that the other models did not capture as well. Soccer, and more specifically shooting, does not follow a linear relationship with xG distance, so the appropriate model should account for this. Furthermore, the partial effects plots provide a great explanation of the model's behavior, making the GAM both interpretable and accurate. For those reasons, we found the GAM model to be the best choice for testing.

League Table

After fitting our xG model, we applied it to all Premier League matches in the 2015–16 season to estimate each team's expected points (xPTS). Expected points reflect the average number of points a team should have earned given the quality of shots they created and conceded, assuming a match is replayed many times using our xG-driven simulation procedure.

For each match, we simulated 10,000 outcomes using the predicted xG for every shot, calculated team-level win/draw/loss probabilities, and translated those into expected points using the standard scoring rule:

Aggregating across all matches yields a full model-based Premier League table, which we compared to the actual results.



Premier League 2015-16 Season

Actual vs Expected Performance

	Team	W	D	L	GF	GA	PTS	xG	xGA	xPTS	xPOS
1	Leicester City	23	12	3	68	36	81	70.7	52.9	65.1	3
2	Arsenal	20	11	7	65	36	71	73.3	36.7	75.3	1
3	Tottenham Hotspur	19	13	6	69	35	70	63.5	44.8	64.0	4
4	Manchester City	19	9	10	71	41	66	69.3	35.6	72.8	2
5	Manchester United	19	9	10	49	35	66	46.0	43.4	54.1	8
6	Southampton	18	9	11	59	41	63	57.7	45.2	60.7	6
7	West Ham United	16	14	8	65	51	62	61.5	59.1	53.5	9
8	Liverpool	16	12	10	63	50	60	60.1	41.9	63.5	5
9	Stoke City	14	9	15	41	55	51	40.8	56.8	43.3	16
10	Chelsea	12	14	12	59	53	50	59.0	49.3	59.0	7
11	Everton	11	14	13	59	55	47	54.3	60.3	50.3	10
12	Swansea City	12	11	15	42	52	47	44.2	56.6	43.4	15
13	Watford	12	9	17	40	50	45	46.1	55.9	47.2	12
14	West Bromwich Albion	10	13	15	34	48	43	43.8	54.1	44.5	14
15	AFC Bournemouth	11	9	18	45	67	42	45.4	50.8	50.1	11
16	Crystal Palace	11	9	18	39	51	42	48.5	57.6	46.2	13
17	Sunderland	9	12	17	48	62	39	46.1	65.5	39.9	19
18	Newcastle United	9	10	19	44	65	37	43.4	59.7	43.1	17
19	Norwich City	9	7	22	39	67	34	43.5	61.3	41.7	18
20	Aston Villa	3	8	27	27	76	17	32.4	62.0	32.4	20

Key Insights

Teams that overperformed their xPTS achieved more points than our GAM-based model predicted from the underlying shot quality. Elite finishing, excellent goalkeeping, tactical match states, or short-term volatility could all contribute to this disparity.

According to our model, teams that performed poorly in comparison to xPTS probably produced enough quality chances but either failed to convert them or gave up goals more frequently than anticipated.

Leicester City, the 2015–16 champions, exhibited a pronounced overperformance in comparison to their projected expectations, which is indicative of their clinical finishing and effective defensive structure despite their mediocre quality of players,which is what makes their title win one of the greatest sporting achievements ever.

Teams below their xPTS, on the other hand, might have had trouble converting opportunities, keeping their defensive shape, or just had negative variance.

We may determine whether there were deviations in offense, defense, or both by looking at our summary table, which shows each team's actual points, expected points (xPTS), expected goals for (xG), and expected goals against (xGA).

Interpretation

Our GAM's xG and xGA measures the quality of opportunities generated and lost by simulating the nonlinear impacts of shot distance, angle, and geographical position.By simulating matches based only on modeled chance quality rather than finishing results, xPTS converts these anticipated goals into hypothetical league outcomes.

GAMs are commonly utilized in contemporary soccer analytics because they can adapt to nonlinear patterns,such as sharply increasing risk in center zones or diminishing benefits from long-range shots. Our findings are in good agreement with well-established patterns from models like Understat's xG/xPTS estimates.This suggests that our GAM provides a solid basis for analyzing team performance throughout the season and captures significant structure in shot quality.

Shot Map

We created comprehensive shot maps for Leicester City and Liverpool for the 2015–16 campaign to go along with our findings. Every league shot that is tried is displayed in each plot, and the GAM-predicted xG for each shot is reflected by the size and color of the circles.

Compared to conventional linear or logistic models, the generated xG values account for positional context more flexibly since GAMs may simulate smooth, nonlinear spatial interactions. Thus, the shot maps aid in confirming if the GAM appropriately rates the threat of various shooting scenarios.

We employ a continuous xG color gradient (green to yellow to orange to red) and a uniform half-pitch template. The GAM predicts higher-quality chances when the circles are larger and warmer in color.We can examine the spatial distribution of opportunities, the locations where teams created their best chances, and the accuracy with which the GAM distinguishes between high-probability and low-probability shoots using these shot maps.

Liverpool attempted 635 shots, one of the highest volumes in the league, with a dense cluster inside and around the penalty area. Their GAM-based xG total (60.08) closely matches their actual output of 62 goals, indicating they finished chances at roughly the expected rate.

The GAM assigns high xG to close-range and central shots, which appear as larger red and yellow circles on the plot. Liverpool's map shows sustained pressure, frequent box entries, and continuous shot generation,all consistent with their German philosophy and attacking identity during the 2015–16 season, which stems from their iconic mid-season appointment of now club legend, manager Jurgen Klopp.

Leicester City attempted 525 shots, fewer than Liverpool, but produced a significantly higher GAM-based xG (70.66) and scored 67 goals.

Their high-value chances are slightly deeper but very central, a hallmark of their elite counterattacking during the title season. The GAM identifies these breakaway or semi-transition shots as high danger because their combination of angle, proximity, and open play conditions strongly increases scoring probability.

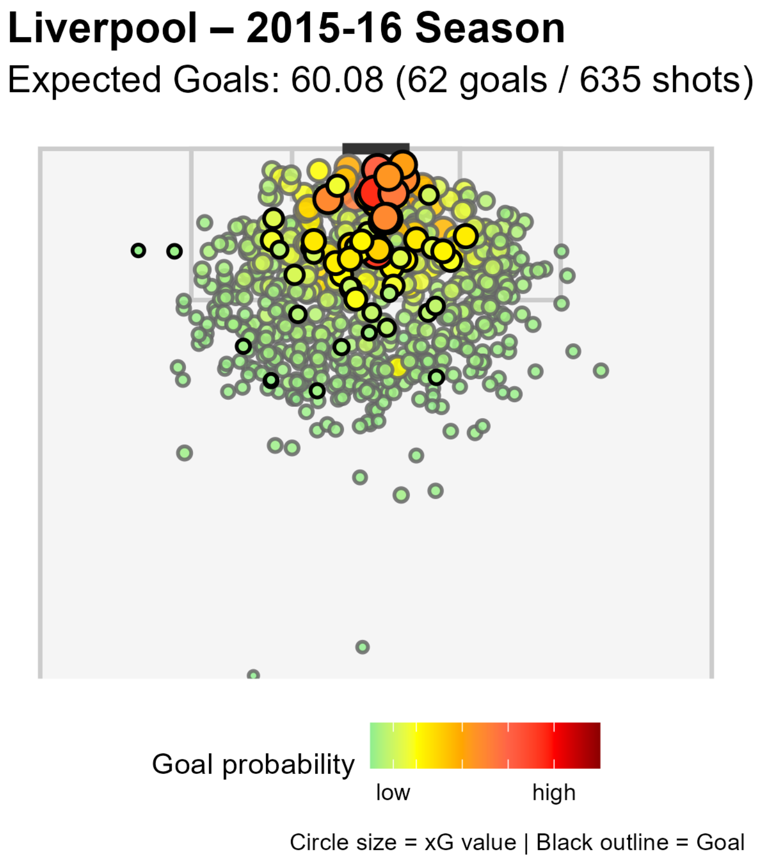
Their shot map contains fewer circles but a higher concentration of medium-to-high xG chances, demonstrating their efficiency and tactical discipline. This shows the elite planning,strategy and execution of the roster and management and is and will always be a prime example of maximum utilization of limited resources.

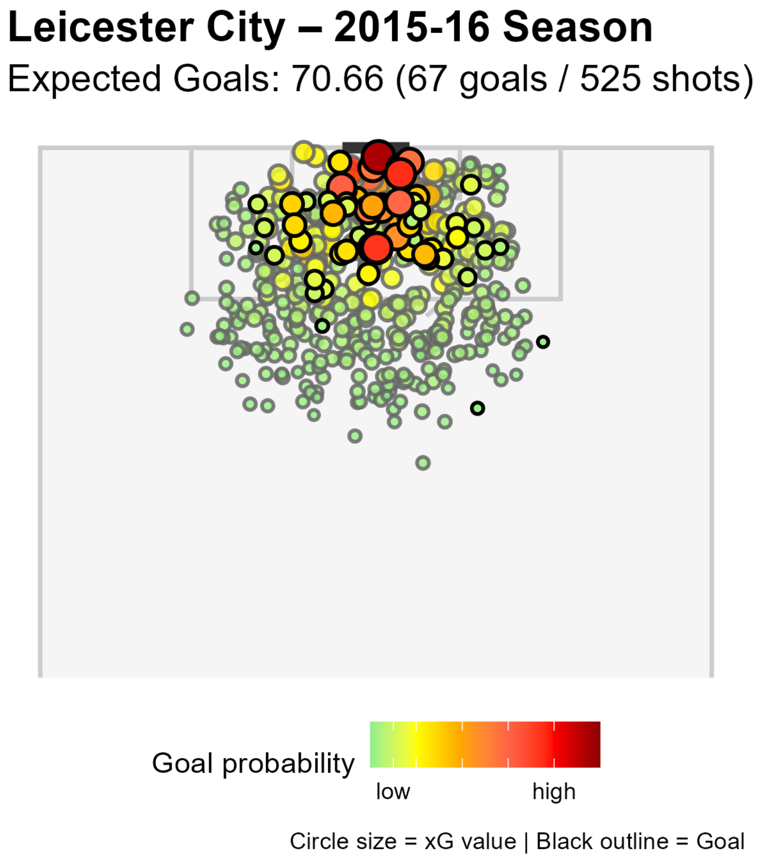
Two crucial aspects are supported by these numerical and visual patterns:

Close-range or high-angle shots are assigned significantly larger xG by the GAM, which accurately reflects nonlinear spatial effects. This behavior is consistent with contemporary professional models.

The maps inherently reveal team tactics; Liverpool,with their intense volume and ongoing pressure and selective yet excellent central opportunities for Leicester

Overall, the shot maps show significant variations in chance creation between these two clubs during the 2015–16 Premier League season and confirm that the GAM acts in accordance with soccer domain knowledge.





Conclusion

From this project, several key takeaways emerge. First, xG and xPTS offer insightful information about underlying performance , showing where teams performed better or worse than expected and emphasizing tactical variations in the creation of opportunities. Second, shot maps provide an easy-to-understand visual confirmation of these trends by illustrating the manifestation of team identity on the field. When combined, these metrics significantly outperform simple goal totals or match results in capturing the quantity and caliber of opportunities.

The sport is moving toward more accurate, personalized, and data-driven evaluations as clubs continue to use advanced analytics. However, this shift also has downsides. Relying too heavily on predictive models can reduce creativity, make tactics too rigid, and push teams to copy the same “optimal” strategies. This can make soccer more efficient, but sometimes less expressive and less enjoyable to watch.

Future work could explore these concerns by studying how analytics changes playing styles over time, whether teams become more similar tactically, and how data use affects creativity, risk-taking, and overall entertainment value. Understanding this balance will be important as soccer continues to evolve with more data and technology.

Author Contribution

Maxwell Gerhart, Rohan Sampath, Jaikhush Thakkar, and Ryan Kaye discussed and planned the research project. Rohan Sampath and Ryan Kaye designed the models and performed model assessment. Maxwell Gerhart and Jaikhush Thakkar compiled the league table and analyzed the results as well as the shot map visualizations. All authors provided critical feedback and helped shape the report.

Code Appendix

```
library(StatsBombR)
library(tidyverse)
library(mgcv)
library(pROC)

comps <- FreeCompetitions()

prem1516 <- comps %>%
  filter(competition_id == "2", season_id == "27")

laliga1516 <- comps %>%
  filter(competition_id == "11", season_id == "27")

bundes1516 <- comps %>%
  filter(competition_id == "9", season_id == "27")

ligue1516 <- comps %>%
  filter(competition_id == "7", season_id == "27")

seriea1516 <- comps %>%
  filter(competition_id == "12", season_id == "27")


# fetch matches
prem1516_matches <- FreeMatches(prem1516)
laliga1516_matches <- FreeMatches(laliga1516)
bundes1516_matches <- FreeMatches(bundes1516)
ligue1516_matches <- FreeMatches(ligue1516)
seriea1516_matches <- FreeMatches(seriea1516)

# combine all matches
matches_1516 <- bind_rows(
  laliga1516_matches,
  bundes1516_matches,
  ligue1516_matches,
  seriea1516_matches,
)

# get all events from matches data frame
events <- free_allevents(MatchesDF = matches_1516, Parallel = TRUE)

# filter shots
shots <- events %>%
  filter(type.name == "Shot") %>%
  mutate(
    x = map_dbl(location, 1),
    y = map_dbl(location, 2)
  ) %>%
  select(
    is_goal = shot.outcome.name,
    x,
```

```r
    y,
    under_pressure,
    shot_body_part = shot.body_part.name,
    shot_technique = shot.technique.name,
    play_pattern = play_pattern.name
  ) %>%
  mutate(
    is_goal = ifelse(is_goal == "Goal", 1, 0),
    under_pressure = ifelse(is.na(under_pressure), 0, 1)
  )

goal_x <- 120
goal_y <- 40

shots <- shots %>%
  mutate(
    distance = sqrt((goal_x - x)^2 + (goal_y - y)^2),
    angle = abs(atan((y - goal_y) / (goal_x - x)))
  )
```

```r
# Train/Test Split
set.seed(123)
idx <- sample(1:nrow(shots), 0.8 * nrow(shots))

train <- shots[idx, ]
test  <- shots[-idx, ]

log_model <- glm(
  is_goal ~
    distance +
    angle +
    under_pressure +
    shot_body_part +
    shot_technique +
    play_pattern,
  data = train,
  family = binomial(link = "logit")
)

library(mgcv)

gam_model <- gam(
  is_goal ~
    s(distance) +
    s(angle) +
    under_pressure +
    shot_body_part +
    shot_technique +
    play_pattern,
  data = train,
  family = binomial
)
```

```r
library(randomForest)

# Random Forest requires a factor target
train$is_goal_factor <- factor(train$is_goal, levels = c(0, 1))

# Fiting the Random Forest model
set.seed(123)
rf_model <- randomForest(
  is_goal_factor ~ distance + angle + under_pressure +
    shot_body_part + shot_technique + play_pattern,
  data = train,
  ntree = 500,
  mtry = 3
)
```

```r
library(pROC)

#Prediction + AUC
test$log_xg <- predict(log_model, newdata = test, type = "response")

auc_value <- roc(test$is_goal, test$log_xg)$auc
```

```
Setting levels: control = 0, case = 1

Setting direction: controls < cases
```

```r
auc_value
```

```
Area under the curve: 0.7964
```

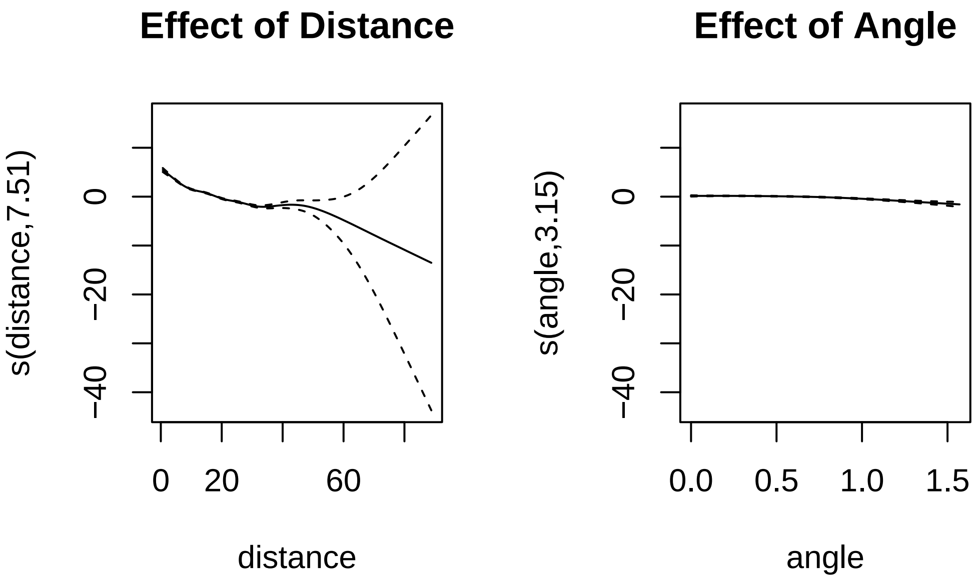
```r
#Prediction + AUC
test$gam_xg <- predict(gam_model, newdata = test, type = "response")

auc_value <- roc(test$is_goal, test$gam_xg)$auc
```

```
Setting levels: control = 0, case = 1
Setting direction: controls < cases
```

```r
auc_value
```

```
Area under the curve: 0.7993
```

```r
#GAM Plots
par(mfrow = c(1, 2))
plot(gam_model, select = 1, main = "Effect of Distance")
plot(gam_model, select = 2, main = "Effect of Angle")
```



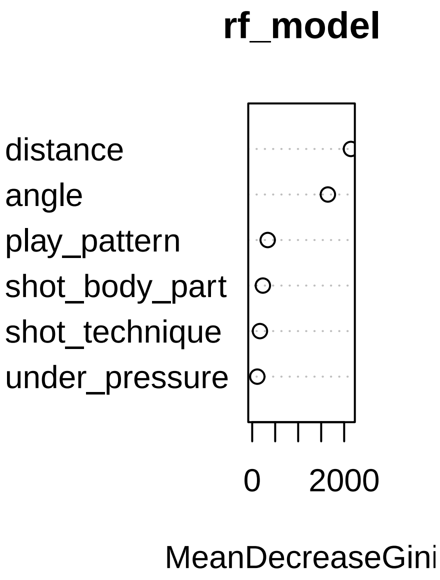
```
#Prediction + AUC
probs <- predict(rf_model, test, type = "prob")
rf_pred <- probs[,2]


# Compute AUC to analyze
rf_auc <- auc(roc(test$is_goal, rf_pred))
```

```
Setting levels: control = 0, case = 1
Setting direction: controls < cases
```

```
rf_auc
```

```
Area under the curve: 0.7528
```

```
# Plot variable importance
varImpPlot(rf_model)
```



```r
library(ggplot2)
library(dplyr)
library(gt)
library(StatsBombR)

prem_events <- free_allevents(
  MatchesDF = prem1516_matches,
  Parallel  = TRUE
) %>%
  allclean()

# filter Shots
prem_shots <- prem_events %>%
  filter(type.name == "Shot") %>%
  mutate(
    x = map_dbl(location, 1),
    y = map_dbl(location, 2)
  ) %>%
  select(
    match_id,
    team.name = team.name,
    is_goal = shot.outcome.name,
    x,
    y,
    under_pressure,
    shot_body_part = shot.body_part.name,
    shot_technique = shot.technique.name,
    play_pattern = play_pattern.name
  ) %>%
  mutate(
    is_goal = ifelse(is_goal == "Goal", 1, 0),
    under_pressure = ifelse(is.na(under_pressure), 0, 1)
  )
```

```r
# creating distance and angle
goal_x <- 120
goal_y <- 40

prem_shots <- prem_shots %>%
  mutate(
    distance = sqrt((goal_x - x)^2 + (goal_y - y)^2),
    angle = abs(atan((y - goal_y) / (goal_x - x)))
  )

prem_shots <- prem_shots %>%
  mutate(
    xg = predict(gam_model, newdata = ., type = "response")
  ) %>%
  filter(!is.na(xg)) %>%
  select(match_id, team.name, x, y, xg, is_goal)


simulate_match_from_xg <- function(match_shots, n_sims = 10000) {
  xg   <- match_shots$xg
  team <- match_shots$team.name
  teams <- unique(team)

  if (length(xg) == 0L || length(teams) != 2L) {
    return(tibble(
      team.name   = teams,
      win_prob    = 0,
      draw_prob   = 1,
      loss_prob   = 0,
      exp_points  = 1
    ))
  }

  idx1 <- which(team == teams[1])
  idx2 <- which(team == teams[2])

  n_shots <- length(xg)

  # prob matrix
  prob_mat <- matrix(rep(xg, each = n_sims),
                     nrow = n_sims, ncol = n_shots, byrow = FALSE)

  # simulate goals per shot
  goal_mat <- matrix(
    rbinom(n_sims * n_shots, size = 1, prob = as.vector(prob_mat)),
    nrow = n_sims, ncol = n_shots
  )

  # goals per team per simulation
  goals_team1 <- rowSums(goal_mat[, idx1, drop = FALSE])
  goals_team2 <- rowSums(goal_mat[, idx2, drop = FALSE])
  team1_wins <- goals_team1 > goals_team2
  team2_wins <- goals_team2 > goals_team1
  draws      <- goals_team1 == goals_team2

  team1_win_prob <- mean(team1_wins)
  team2_win_prob <- mean(team2_wins)
  draw_prob      <- mean(draws)

  team1_exp_pts <- 3 * team1_win_prob + 1 * draw_prob
  team2_exp_pts <- 3 * team2_win_prob + 1 * draw_prob

  tibble(
    team.name  = teams,
    win_prob   = c(team1_win_prob, team2_win_prob),
    draw_prob  = draw_prob,
    loss_prob  = c(mean(!team1_wins & !draws),
                   mean(!team2_wins & !draws)),
    exp_points = c(team1_exp_pts, team2_exp_pts)
  )
}

# expected points sim
set.seed(380)

match_expected_points <- prem_shots %>%
  group_by(match_id) %>%
  nest(data = -match_id) %>%
  mutate(
    sim_results = map(data, ~ simulate_match_from_xg(.x, n_sims = 10000))
  ) %>%
  select(match_id, sim_results) %>%
  unnest(sim_results)

# season totals for expected points
season_expected_points <- match_expected_points %>%
  group_by(team.name) %>%
  summarise(
    exp_points_season = sum(exp_points),
    .groups = "drop"
  )

# calculate xG and xGA totals
xg_summary <- prem_shots %>%
  group_by(team.name) %>%
  summarise(
    xG = sum(xg),
    .groups = "drop"
  )

# xGA (xG conceded)
xga_summary <- prem_shots %>%
  left_join(
    prem1516_matches %>%
      select(match_id, home_team.home_team_name, away_team.away_team_name),
    by = "match_id"
  ) %>%
  mutate(
    opponent = if_else(team.name == home_team.home_team_name,
                       away_team.away_team_name,
                       home_team.home_team_name)
  ) %>%
  group_by(opponent) %>%
  summarise(
    xGA = sum(xg),
    .groups = "drop"
  ) %>%
  rename(team.name = opponent)

# get actual results
actual_results <- prem1516_matches %>%
  transmute(
    team.name = home_team.home_team_name,
    W = if_else(home_score > away_score, 1, 0),
    D = if_else(home_score == away_score, 1, 0),
    L = if_else(home_score < away_score, 1, 0),
    GF = home_score,
    GA = away_score,
    points = case_when(
      home_score > away_score ~ 3,
      home_score == away_score ~ 1,
      TRUE ~ 0
    )
  ) %>%
  bind_rows(
    prem1516_matches %>%
      transmute(
        team.name = away_team.away_team_name,
        W = if_else(away_score > home_score, 1, 0),
        D = if_else(away_score == home_score, 1, 0),
        L = if_else(away_score < home_score, 1, 0),
        GF = away_score,
        GA = home_score,
        points = case_when(
          away_score > home_score ~ 3,
          away_score == home_score ~ 1,
          TRUE ~ 0
        )
      )
  ) %>%
  group_by(team.name) %>%
  summarise(
    W = sum(W),
    D = sum(D),
    L = sum(L),
    GF = sum(GF),
    GA = sum(GA),
    PTS = sum(points),
    .groups = "drop"
  )

# create final table
final_table <- actual_results %>%
  left_join(xg_summary, by = "team.name") %>%
  left_join(xga_summary, by = "team.name") %>%
  left_join(season_expected_points, by = "team.name") %>%
  rename(xPTS = exp_points_season) %>%
  arrange(desc(PTS)) %>%
  select(team.name, W, D, L, GF, GA, PTS, xG, xGA, xPTS)

# format table
gt_tbl <- final_table %>%
  mutate(
    Rank = row_number(),
    xPOS = rank(-xPTS, ties.method = "min"),
    .before = team.name
  ) %>%
  gt() %>%
  tab_header(
    title = "Premier League 2015-16 Season",
    subtitle = "Actual vs Expected Performance"
  ) %>%
  cols_label(
    Rank = "",
    team.name = "Team",
    W = "W",
    D = "D",
    L = "L",
    GF = "GF",
    GA = "GA",
    PTS = "PTS",
    xG = "xG",
    xGA = "xGA",
    xPTS = "xPTS",
    xPOS = "xPOS"
  ) %>%
  fmt_number(
    columns = c(xG, xGA, xPTS),
    decimals = 1
  ) %>%
  tab_style(
    style = cell_fill(color = "#f5f5f5"),
    locations = cells_body(
      rows = Rank %% 2 == 0
    )
  ) %>%
  tab_style(
    style = cell_text(weight = "bold"),
    locations = cells_column_labels()
  ) %>%
  tab_options(
    table.font.size = 14,
    heading.title.font.size = 20,
    heading.subtitle.font.size = 14,
    column_labels.font.weight = "bold"
  ) %>%
  cols_align(
    align = "center",
    columns = c(W, D, L, GF, GA, PTS, xG, xGA, xPTS, xPOS)
  ) %>%
  cols_align(
    align = "left",
    columns = team.name
  ) %>%
  cols_move(
    columns = xPOS,
    after = xPTS
  )

gt::gtsave(gt_tbl, "prem_table.png")
```

```
# filter each team
liverpool_shots <- prem_shots %>%
  filter(team.name == "Liverpool")

leicester_shots <- prem_shots %>%
  filter(team.name == "Leicester City")

# create xG color palette
xg_colors <- c("#90EE90", "#FFFF00", "#FFA500", "#FF6347", "#FF0000", "#8B0000")

# liverpool
p_liv <- ggplot(liverpool_shots, aes(x = y, y = x)) +

  annotate("rect", xmin = 0, xmax = 80, ymin = 0, ymax = 120,
           fill = "#F5F5F5", color = "#CCCCCC", linewidth = 0.8) +

  annotate("rect", xmin = 18, xmax = 62, ymin = 102, ymax = 120,
           fill = NA, color = "#CCCCCC", linewidth = 0.8) +

  annotate("rect", xmin = 30, xmax = 50, ymin = 114, ymax = 120,
           fill = NA, color = "#CCCCCC", linewidth = 0.8) +

  annotate("segment", x = 36, xend = 44, y = 120, yend = 120,
           color = "#333333", linewidth = 2) +

  annotate("path",
           x = 40 + 10 * sin(seq(0.64, 2.5, length.out = 50)),
           y = 108 + 10 * cos(seq(0.64, 2.5, length.out = 50)),
           color = "#CCCCCC", linewidth = 0.8) +

  # misses
  geom_point(data = filter(liverpool_shots, is_goal == 0),
             aes(size = xg, fill = xg),
             color = "#666666",
             shape = 21,
             stroke = 1,
             alpha = 0.85) +

  # goals
  geom_point(data = filter(liverpool_shots, is_goal == 1),
             aes(size = xg, fill = xg),
             color = "#000000",
             shape = 21,
             stroke = 1,
             alpha = 1) +

  scale_size_continuous(range = c(1, 5), guide = "none") +
  scale_fill_gradientn(colors = xg_colors,
                       limits = c(0, 1),
                       breaks = c(0.1, 0.2, 0.4, 0.6, 0.8),
                       labels = c("low", "", "", "", "high"),
                       name = "Goal probability") +

  coord_fixed(xlim = c(0, 80), ylim = c(60, 121)) +

  theme_minimal() +
  theme(
    panel.grid = element_blank(),
    axis.text = element_blank(),
    axis.title = element_blank(),
    legend.position = "bottom",
    plot.title = element_text(face = "bold", size = 16),
    plot.subtitle = element_text(size = 14)
  ) +

  labs(
    title = "Liverpool - 2015-16 Season",
    subtitle = paste0("Expected Goals: ", round(sum(liverpool_shots$xg), 2),
                      " (", sum(liverpool_shots$is_goal),
                      " goals / ", nrow(liverpool_shots), " shots)"),
    caption = "Circle size = xG value | Black outline = Goal"
  )

ggsave("liverpool_shotmap.png", p_liv, width = 4, height = 6, dpi = 300)

# leicester

p_lei <- ggplot(leicester_shots, aes(x = y, y = x)) +

  annotate("rect", xmin = 0, xmax = 80, ymin = 0, ymax = 120,
           fill = "#F5F5F5", color = "#CCCCCC", linewidth = 0.8) +

  annotate("rect", xmin = 18, xmax = 62, ymin = 102, ymax = 120,
           fill = NA, color = "#CCCCCC", linewidth = 0.8) +

  annotate("rect", xmin = 30, xmax = 50, ymin = 114, ymax = 120,
           fill = NA, color = "#CCCCCC", linewidth = 0.8) +

  annotate("segment", x = 36, xend = 44, y = 120, yend = 120,
           color = "#333333", linewidth = 2) +

  annotate("path",
           x = 40 + 10 * sin(seq(0.64, 2.5, length.out = 50)),
           y = 108 + 10 * cos(seq(0.64, 2.5, length.out = 50)),
           color = "#CCCCCC", linewidth = 0.8) +

  geom_point(data = filter(leicester_shots, is_goal == 0),
             aes(size = xg, fill = xg),
             color = "#666666",
             shape = 21,
             stroke = 1,
             alpha = 0.85) +

  geom_point(data = filter(leicester_shots, is_goal == 1),
             aes(size = xg, fill = xg),
             color = "#000000",
             shape = 21,
             stroke = 1,
             alpha = 1) +

  scale_size_continuous(range = c(1, 5), guide = "none") +
  scale_fill_gradientn(colors = xg_colors,
                       limits = c(0, 1),
                       breaks = c(0.1, 0.2, 0.4, 0.6, 0.8),
                       labels = c("low", "", "", "", "high"),
                       name = "Goal probability") +

  coord_fixed(xlim = c(0, 80), ylim = c(60, 121)) +

  theme_minimal() +
  theme(
    panel.grid = element_blank(),
    axis.text = element_blank(),
    axis.title = element_blank(),
    legend.position = "bottom",
    plot.title = element_text(face = "bold", size = 16),
    plot.subtitle = element_text(size = 14)
  ) +

  labs(
    title = "Leicester City - 2015-16 Season",
    subtitle = paste0("Expected Goals: ", round(sum(leicester_shots$xg), 2),
                      " (", sum(leicester_shots$is_goal),
                      " goals / ", nrow(leicester_shots), " shots)"),
    caption = "Circle size = xG value | Black outline = Goal"
  )

ggsave("leicester_shotmap.png", p_lei, width = 4, height = 6, dpi = 300)

knitr::include_graphics("liverpool_shotmap.png")
knitr::include_graphics("leicester_shotmap.png")
```