404 Final Paper

The Havertz Effect: How Havertz Shifts Arsenal's Match Outcomes

Matheus Grover
June 14, 2025

1: Introduction

Kai Havertz is a soccer player for Arsenal Football Club in the English Premier League(EPL). Since he signed for Arsenal in the Summer of 2023 he has been a very controversial figure. He was signed for around 65 Million euros compared to the average EPL's tranfer cost of 25 Million euros. He was also given the highest wages at the club. So, immediately there was a pressure for him to live up to his price tag.

As seen in Figure 1, he had a slow start to his first season, registering his first goal 10 games into the season and taking even longer to start the upward trend he's on today. Over the course of the season he went from a "flop" signing to one of our most valuable players. However with a major injury in the 2024/2025 season it left people with a lot of time to argue about whether the team is better with or without him.

In this paper I am going to analyze how Havertz shifts Arsenals match outcomes in a quest to prove he is important to our team.

1.1: Data

All the data was scraped from FBREF a publicly available soccer statistics page. I used the R (rvest) package to scrape match performance data for Havertz from the 2023–24 and 2024–25 Premier League seasons. These datasets included detailed individual statistics such as goals, assists, xG, and minutes played. I also scraped full-season squad statistics for Arsenal, along with fixture results and expected goals (xG) from the team schedule pages for the same seasons. I ended up combining the Havertz indivudal match stats with the fixture results to create a dataframe that had all 110 matches in the 2023-2025 seasons and Havertz Team Stats for each game. This allowed me to compare Arsenal's performance in matches where Havertz played versus those he missed, using both individual and team statistics.

1.2: Key Variables

Here are a few definitions of key variables used throughout the rest of this paper that would be helpful in understanding soccer concepts. `xG` or Expected Goals is one of the most important concepts to understand. Everytime a player takes a shot on goal it creates an expected goal percentage from 0.0 to 1.0, 0 meaning the shot going into the goal is impossible and 1 being a guaranteed goal. `xG` doesn't capture if the shot was converted into a goal, just the likelihood of scoring from the moment the shot is taken. It's a relatively new concept and is used to evaluate the quality of shots and assess team or player performance beyond just goals scored. Another important variable is `Points`. In soccer the winning team is given 3 points, a draw is 1 point, a loss is 0 points. Later, I use an indicator I created called `Win` for 1 if Arsenal won the game and 0 if they drew or lost.

2: Predictions

2.1: Exploratory Data Analysis

Figure 2 reports the basic distributions of the study variables. Team `xG` is right-skewed, with most matches clustered between 0.5 and 2. Havertz's personal `xG` is even more skewed, reflecting many appearances with few shots of high quality. The minute histogram confirms he usually plays at least one half, with a mode around 60–90 minutes.

Applying the $\log(1 + xG)$ transform to both `xG` measures reduces skew and produces roughly bell-shaped densities for team xG, however due to a large zero mass in Havertz xG it's not as effective but still helpful.

2.2 Predictions for Points

Before estimating any models I outlined five hypotheses for how Havertz might influence Arsenal's results:

1. **Havertz played:** matches in which he appears, yield more points.
2. **Havertz xG:** higher logged personal xG correlates with more points.
3. **Team xG:** higher logged team xG correlates with more points.
4. **Venue:** home fixtures deliver more points than away or neutral fixtures.
5. **Interaction:** Havertz's xG benefit diminishes when team xG is already high.

2.3 Linear Models

I estimated four linear specifications, each with **Points** as the dependent variable.

M0 (baseline)

$$\text{Points} = \beta_0 + \beta_1\,\texttt{HavertzPlayed} + \beta_2\,\texttt{xG} + \beta_3\,\texttt{Venue} + \beta_4\,\texttt{Season} + \varepsilon.$$

$\mathbf{M_{clean}}$ (minutes)

Same as M0, but `HavertzPlayed` is replaced by `Hav_Min`.

M1 (logged)

Same as M_{clean}, with team and player xG log-transformed:

$$\text{Points} = \beta_0 + \beta_1\,\texttt{Hav\_Min} + \beta_2\,\ln(1 + \texttt{xG}) + \beta_3\,\ln(1 + \texttt{Hav\_xG}) + \beta_4\,\texttt{Venue} + \beta_5\,\texttt{Season} + \varepsilon.$$

$\mathbf{M_{int}}$ (interaction)

Extends M1 with an interaction between the logged xG terms:

$$\text{Points} = \beta_0 + \beta_1\,\texttt{Hav\_Min} + \beta_2\,\ln(1 + \texttt{xG}) + \beta_3\,\ln(1 + \texttt{Hav\_xG}) + \beta_4\,\big(\ln(1 + \texttt{xG}) \times \ln(1 + \texttt{Hav\_xG})\big) + \beta_5\,\texttt{Venue} + \beta_6\,\texttt{Season} + \varepsilon.$$

2.4 Prediction Results

Across the four linear specifications, five headline findings emerge:

1. **Does simply playing Havertz improve results?** The binary `HavertzPlayed` coefficient is positive in the baseline model (0.44) but not significant ($p = 0.16$). When minutes replace the indicator or when log terms are added, the estimate shrinks toward zero ($p = 0.75$ and $p = 0.13$, respectively). In short, just putting Havertz on the pitch does not appear to be significant in changing points totals.

2. **Does Havertz's own shot quality matter?** His logged personal xG is positive in Model1 (0.62, $p = 0.32$). In the interaction specification the coefficient rises to 4.12 and the p-value falls to 0.058, which is statistically significant at the 10 percent level (though not at 5 percent). This provides moderate evidence that better shooting performances from Havertz convert into additional points.

3. **Does team attacking quality matter?** Logged team xG is associated with more points in every model. The effect strengthens as controls are added; in the interaction model, a one-unit increase in $\ln(1+\text{xG})$ is linked to 1.39 additional points ($p = 0.011$).

4. **Is there a home-field advantage?** Home indicators are positive in all models, but p-values sit between 0.16 and 0.26. The data therefore do not allow a firm conclusion about venue.

5. **How does Havertz's marginal impact vary with team xG?** The interaction between player and team xG is negative (–2.71, $p = 0.096$), significant at the 10 percent level. The sign implies that Havertz's individual xG is most valuable in low-xG matches and contributes slightly less once the team is already generating many chances.

3 Kernel Regression

3.1 Why Kernel Regression

The four OLS fits (Figure 3) enforce a single straight line between team xG and match points. That rigidity means they cannot detect plateaus, spikes, or thresholds in the points-scoring process. To explore possible non-linear structure, I turn to a kernel smoother, implemented in `C` for speed and called from `R`. Kernel regression adapts to local data density and can therefore reveal how the xG to points relationship changes across the range of observed xG.

3.2 Results

Figure 4 plots two kernel regression curves, one for matches where Havertz played and one for matches he missed.

- **With Havertz.** The gold curve rises consistently as team xG increases, suggesting that when Havertz plays, Arsenal steadily turn their chances into points.
- **Without Havertz.** The red curve initially follows the same upward path, peaks near 2.3 xG, and then drops sharply. This pattern implies that Arsenal were still creating high-quality chances without Havertz but struggled to convert those chances into goals and wins.
- **Interpretation.** Havertz appears to improve the efficiency with which team xG turns into match points, most notably in high-xG fixtures. When he is absent, the team can reach similar xG totals yet earns fewer points beyond the 2.3 xG mark.

4: 2D KDE of xG and Goals

Figures 5-7 overlays a two-dimensional kernel density estimate (2D KDE) on the scatter of team expected goals (`xG`) and goals scored (`GF`) for matches *with* and *without* Havertz.

- **Matches without Havertz.** The red 2D KDE clusters tightly below `xG` around 2.5 and `GF` greater than 3, suggesting both chance creation and goal output stay in a moderate range. High-goal outcomes are rare for any given `xG`.
- **Matches with Havertz.** The gold 2D KDE displays a denser core (`xG` around 1–1.5, `GF` around 1–2) and, importantly, a pronounced tail extending to `xG` greater than 3 and `GF` = 4–6. This indicates a higher scoring ceiling when Havertz is on the pitch.
- **Interpretation.** For comparable `xG` tallies, Arsenal convert chances into goals more efficiently with Havertz, and the team's upper bound on goals expands. Sample sizes differ, so the comparison is suggestive rather than definitive, but the 2D KDEs align with the kernel-regression evidence that Havertz boosts finishing efficiency and upside.

Figures 5-7 are provided in the appendix.

5: Logistic Regression

To translate match performance into a win–loss lens, I created a binary response variable called `Win` (1 = Arsenal win, 0 = draw or loss) and estimated the logit model

$$\Pr(\texttt{Win} = 1) = \text{logit}^{-1}\big(\beta_0 + \beta_1\,\texttt{xG} + \beta_2\,\texttt{HavertzPlayed}\big).$$

The coefficient on `HavertzPlayed` is $\hat{\beta}_2 = 0.8589$; exponentiating yields an odds ratio of $e^{0.8589} \approx 2.36$, meaning the odds of winning are about 2.4 times higher when Havertz appears. With $p = 0.07$ this effect is statistically significant at the 10 percent level, though not at 5 percent.

Using the fitted model I predicted a win probability for every match and smoothed those probabilities with a LOESS curve. Figure 8 compares the two smoothed curves.

- As team `xG` rises, win probability increases in both scenarios, confirming that creating quality chances drives results.
- Across the entire `xG` range the curve with Havertz lies roughly 10–15 percentage points above the curve without him. Thus, for any given chance Arsenal are more likely to win when Havertz is on the pitch.
- The sample of matches without Havertz is small, so the gap should be interpreted with caution, but it aligns with the earlier kernel and KDE evidence that Havertz improves chance conversion.

Figure 8 (predicted win probability vs. xG) is provided in the appendix.

6 Results

6.1 Average Points per Match

Arsenal collect an average of **2.13 points** when Havertz plays and **1.65 points** when he does not (Figure 9). The density plot (Figure 10) shows that matches with Havertz are skewed toward the three-point column, whereas matches without him have a much larger mass at zero or one point. Fewer low-point outcomes occur when he is on the pitch.

For context, Arsenal finished second this season with 74 points. Havertz missed 15 league games in which the team earned 25 points. If those matches had produced the squad's "with Havertz" average (15 × 2.13 32 points), the final tally would rise to 81 points still behind Liverpool's 84, but closer to the title pace.

6.2 Findings Across Models

Linear regressions. Across four different linear models:

- **HavertzPlayed by itself is not enough.** The simple "played or not" dummy is positive but never close to 5 percent significance ($p = 0.16$ in the baseline; $p \geq 0.13$ with added controls).
- **Personal xG helps but only marginally.** Havertz's own logged expected goals become meaningful only in the interaction model (coefficient = 4.12, $p = 0.058$), significant at the 10 percent level.
- **Team quality dominates.** Logged team `xG` is the strongest and most stable predictor: a one-unit increase adds about 1.4 points in the full interaction model ($p = 0.011$).
- **Venue remains inconclusive.** Home-field coefficients are positive but imprecise ($p \approx 0.16$–0.26), so we cannot confirm a consistent home advantage.
- **Diminishing returns.** The interaction between player and team `xG` is negative and marginally significant ($-2.71, p = 0.096$), implying Havertz's finishing is most valuable in lower-`xG` games and contributes slightly less when the team is already generating many chances. This was my favorite finding and it makes sense from a spectators perspective. It has to do with the playstyles of EPL teams, better teams play less defensively against Arsenal which makes Havertz less vital as Arsenal can find other outlets to score. When a team is more defensive and our creative outlets are blocked, he steps up to be the center of our attack up front.

Kernel regression. The flexible smoother reveals a clear divergence: with Havertz the `xG` to `Points` curve keeps climbing, whereas without him it flattens near 2.3 `xG`. This pattern suggests he helps convert high-quality chances into results.

Two-dimensional KDE. The joint density of `xG` and goals (Figures 5-7) shows a longer high-goal, high-`xG` tail when Havertz plays; the ceiling on goals is lower in matches he misses.

Logistic regression. The odds of winning are about **2.4 times** higher with Havertz ($p = 0.07$), significant at the 10 percent level. Across the entire `xG` range his presence adds roughly 10–15 percentage points to win probability (Figure 8).

Bottom line. Every method tells a consistent story: Kai Havertz is not merely present in winning line-ups, he is associated with turning expected goals into actual goals and goals into match points.

7. Conclusion

7.1: Limitations and Future Work

Data Limitation: The sample without Havertz is small. He missed over 3 months this season(24/25) and less last season(23/24).

xG limitation: Expected goals is a relatively new idea and calculated by FBREF. This might vary from what other statistics websites use to calculate.

Future Work: To attempt to solve these limitations I would like to focus more on minutes/what minutes he played. Havertzplayed being binary could have led to times where he only got a few minutes at the end of the game skew the results. I would also like to scrape other context variables: What position Havertz played in during the game, other players stats in similar positions, opponent team strength, other players who were purchased for around the same price tag.

Already, just looking at the effects he has on Arsenal games shows he impacts results and plays an important role in the team. But this additional work lets me quantify that impact and begin to evaluate whether his statistics justify his price tag and wages.

1 5. Appendix

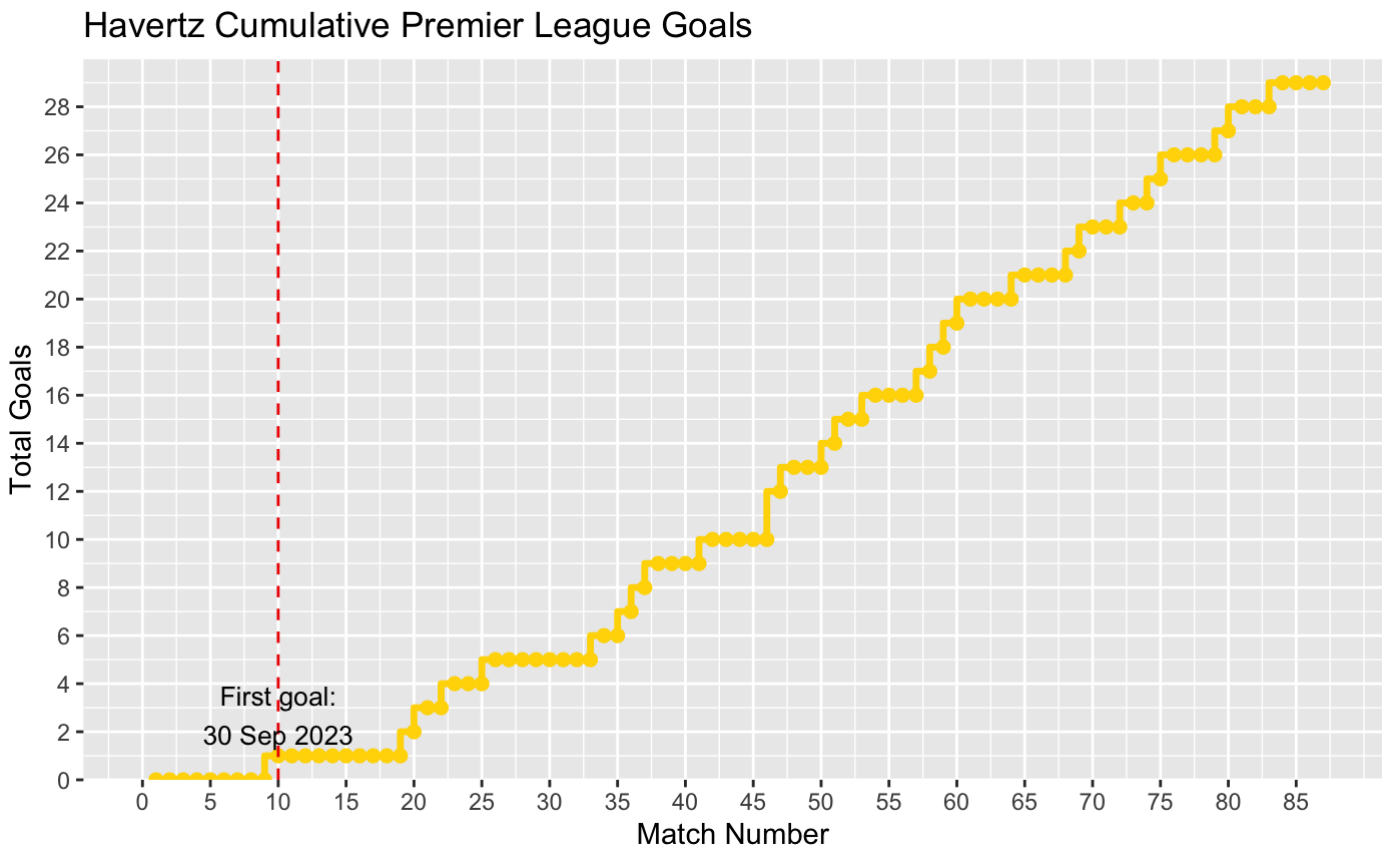

Figure 1: Havertz Total Goals over both Seasons(2023-2025)

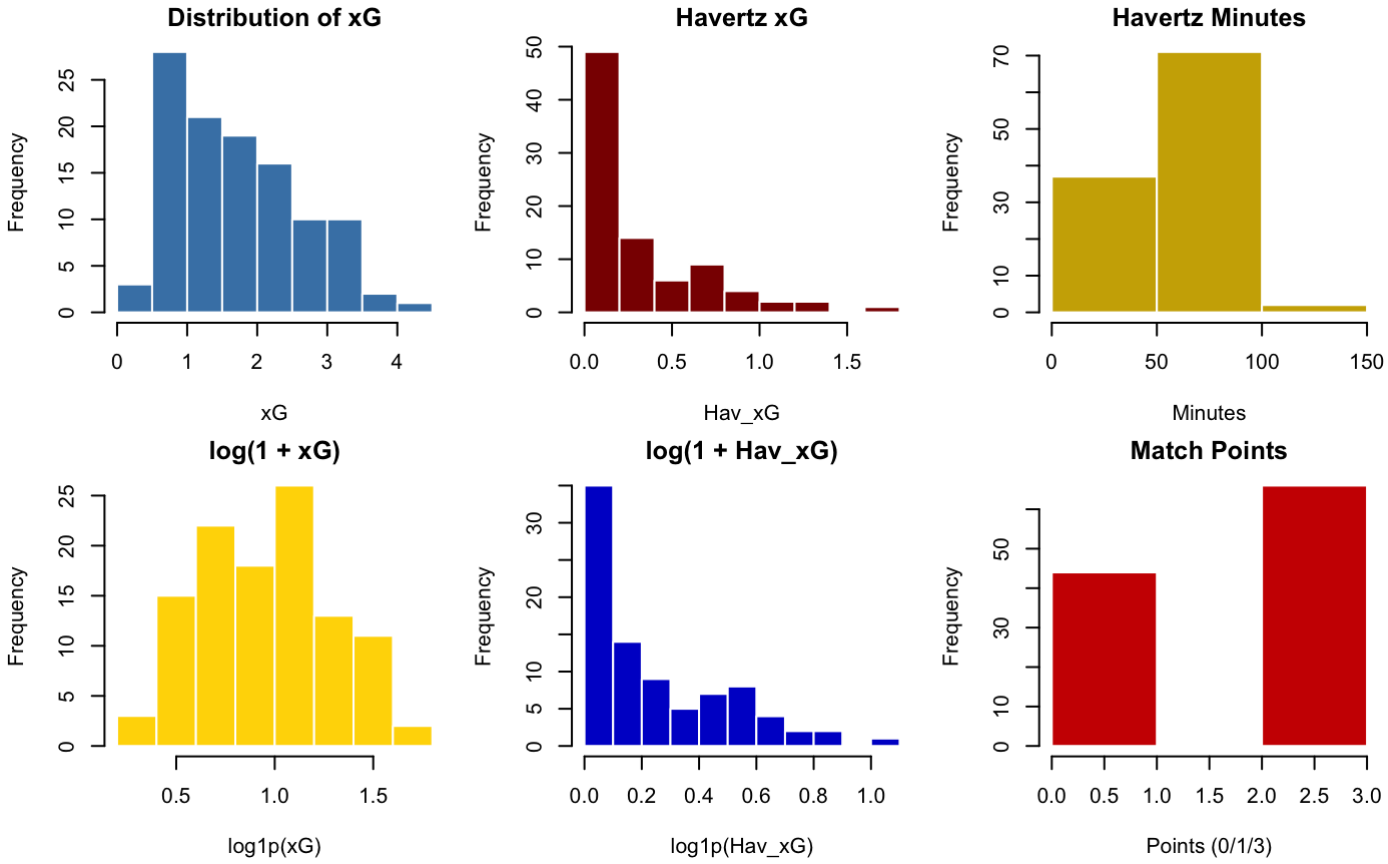

Figure 2: Exploratory Data Analysis)

Table 1: Linear Model Specifications

- **M0 (Baseline)**:

$$\text{Points} = \beta_0 + \beta_1 \cdot \text{HavertzPlayed} + \beta_2 \cdot \text{xG} + \beta_3 \cdot \text{Venue} + \beta_4 \cdot \text{Season} + \varepsilon$$

- **Model clean (Minutes)**:

$$\text{Points} = \beta_0 + \beta_1 \cdot \text{Hav\_Min} + \beta_2 \cdot \text{xG} + \beta_3 \cdot \text{Venue} + \beta_4 \cdot \text{Season} + \varepsilon$$

- **Model 1 (Logged)**:

$$\text{Points} = \beta_0 + \beta_1 \cdot \text{Hav\_Min} + \beta_2 \cdot \ln(1 + \text{xG}) + \beta_3 \cdot \ln(1 + \text{Hav\_xG}) + \beta_4 \cdot \text{Venue} + \varepsilon$$

- **Model Int (Interaction)**:

$$\text{Points} = \beta_0 + \beta_1 \cdot \text{Hav\_Min} + \beta_2 \cdot \ln(1 + \text{xG}) * \beta_3 \cdot \ln(1 + \text{Hav\_xG}) + \beta_4 \cdot \text{Venue} + \text{Season} + \varepsilon$$

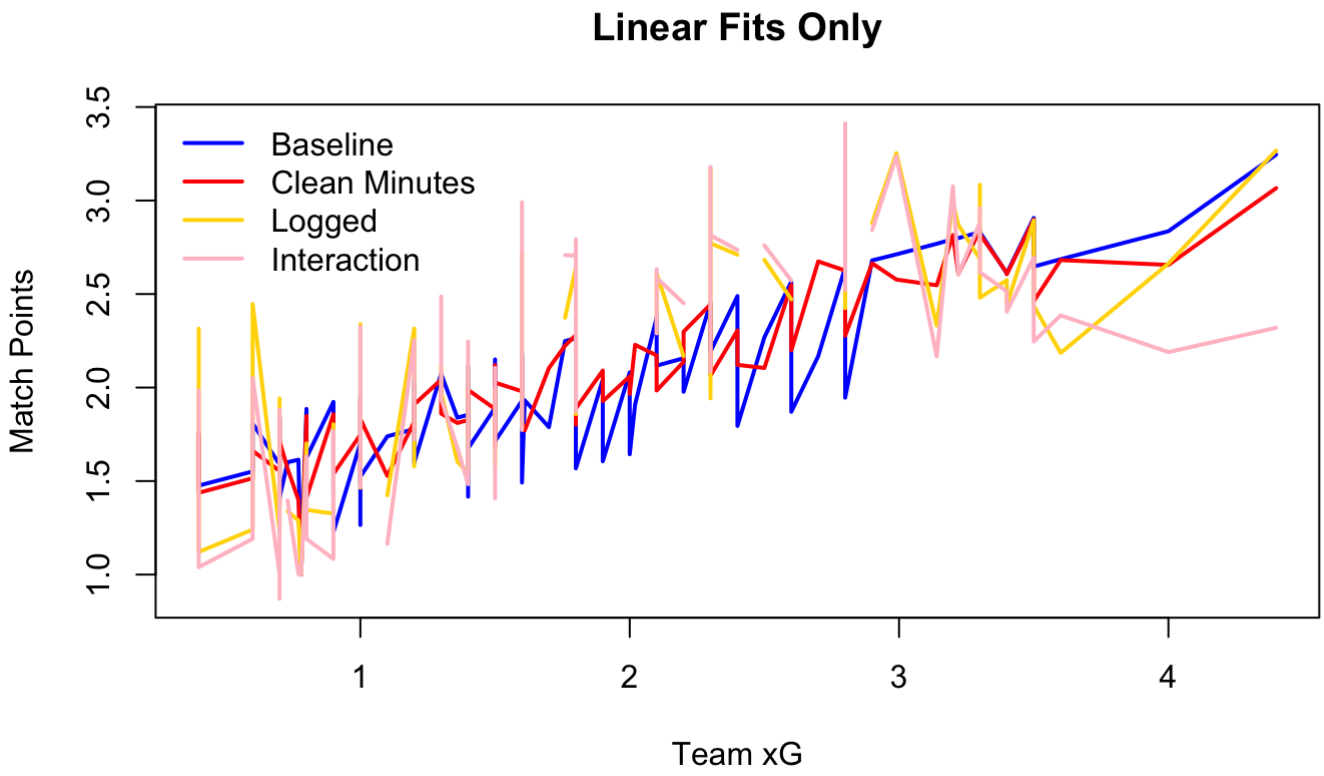

Figure 3: OLS Fits for Our Four Models

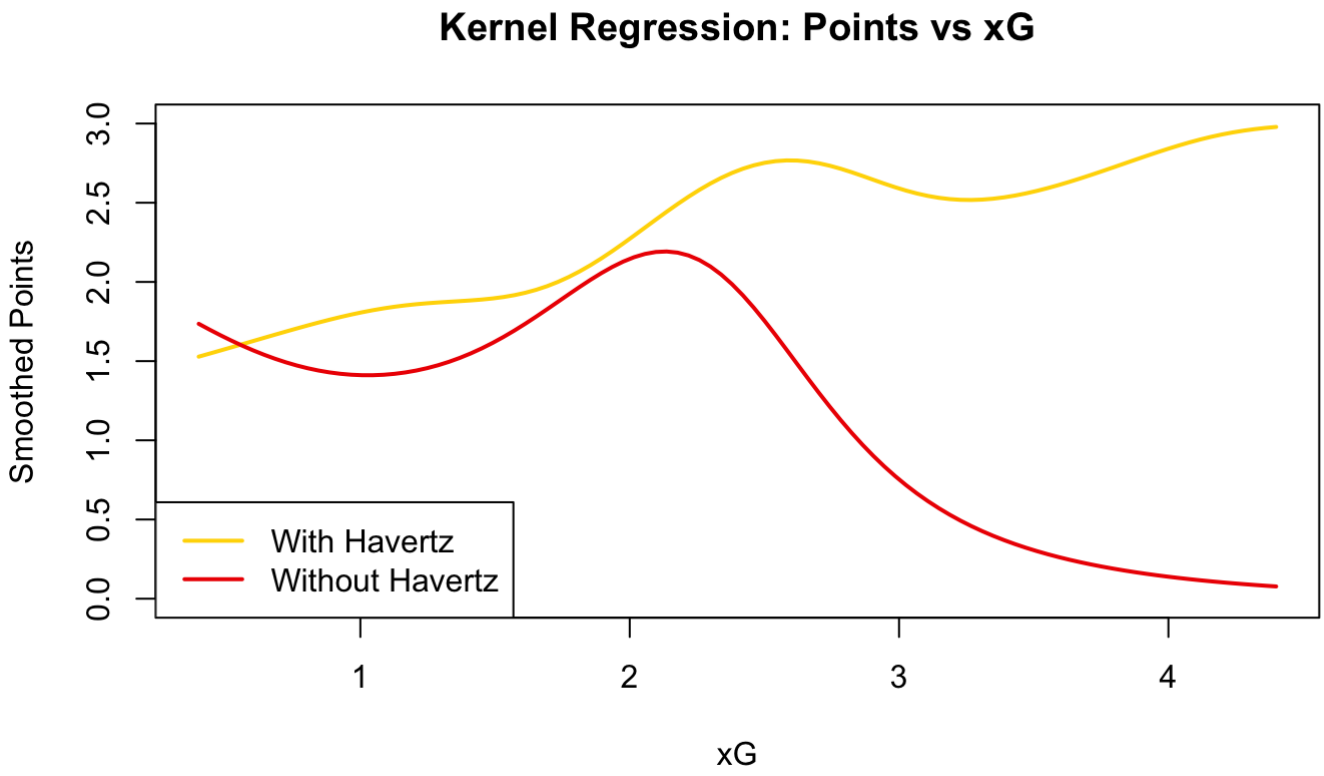

Figure 4: Kenrel Regression with C Smoother

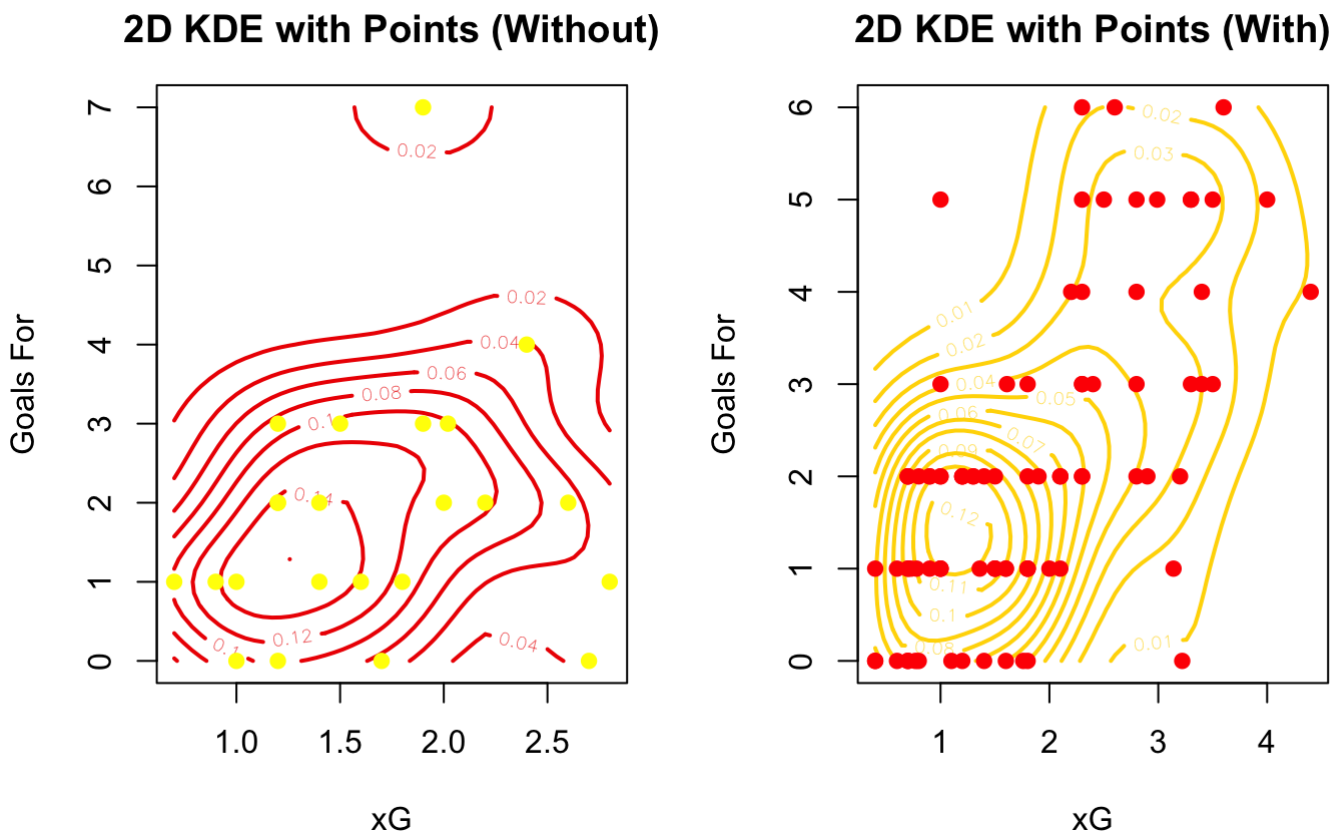

Figure 5: Side by Side 2D KDE With and Without Havertz

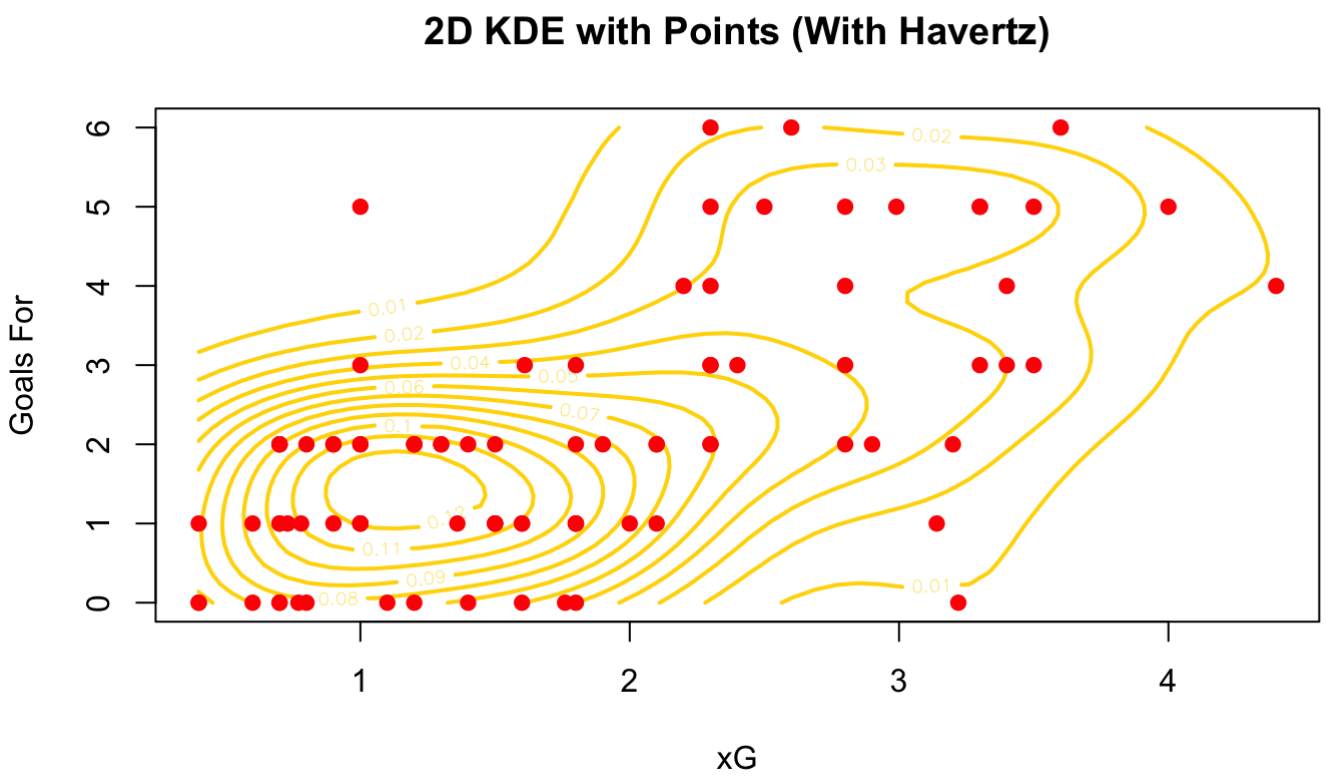

Figure 6: 2D Kernel Density Estimation With Havertz

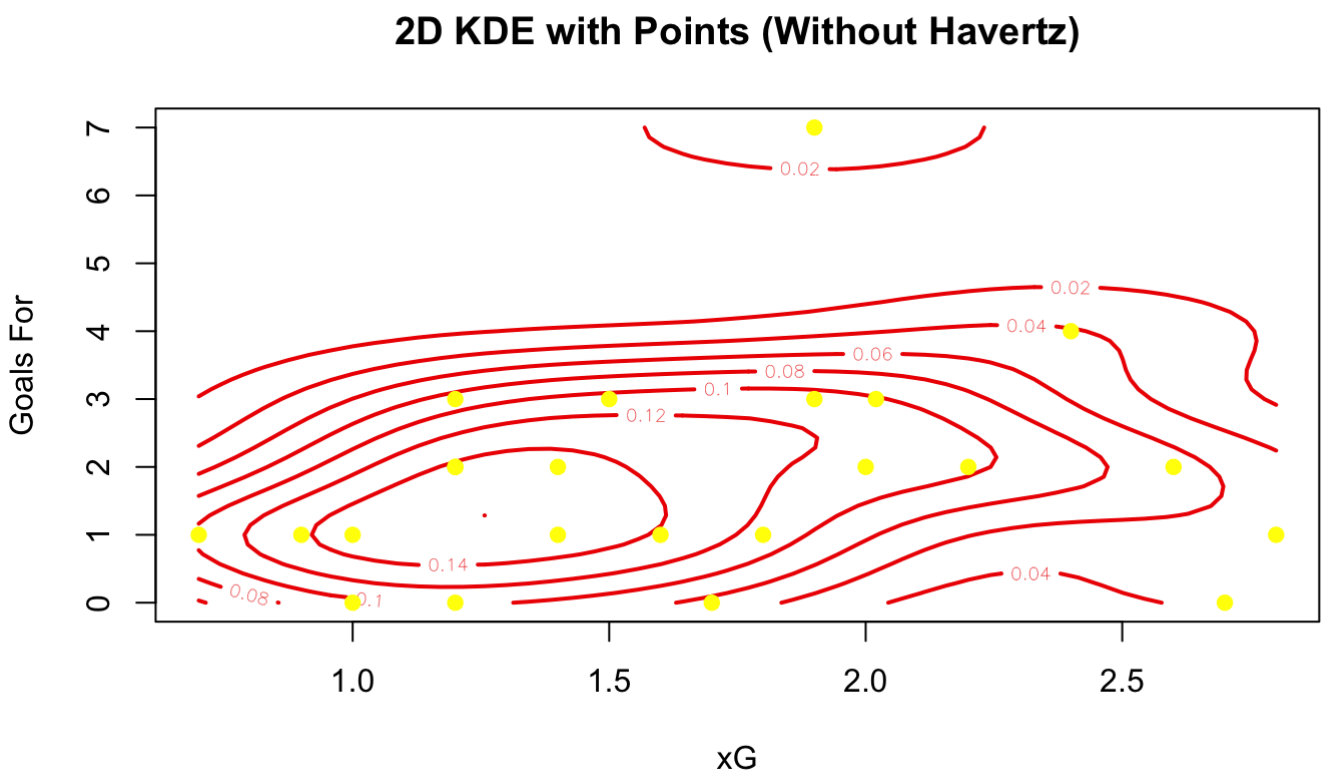

Figure 7: 2D Kernel Density Estimation Without Havertz

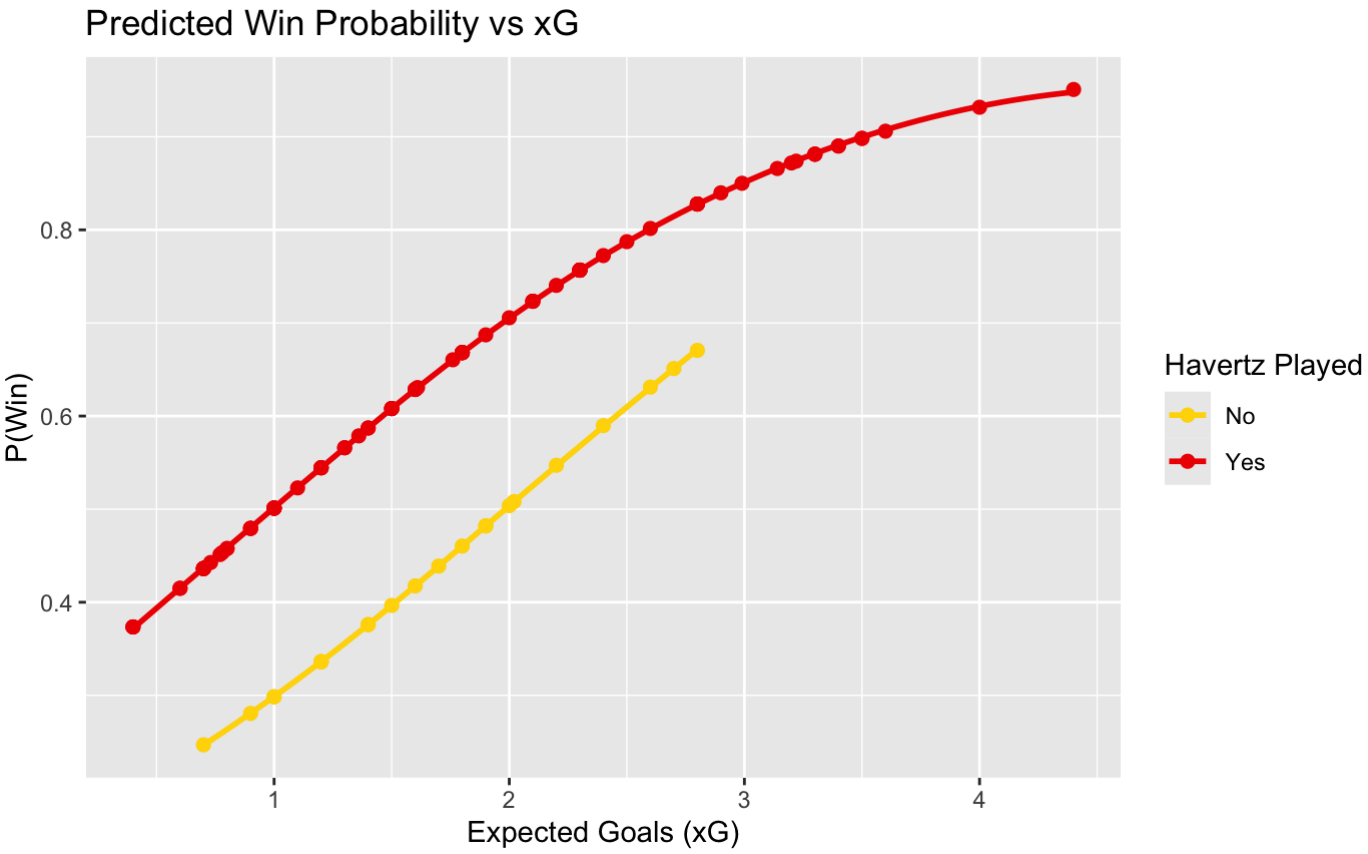

Figure 8: Logistic Regression of Predicted Win Probability

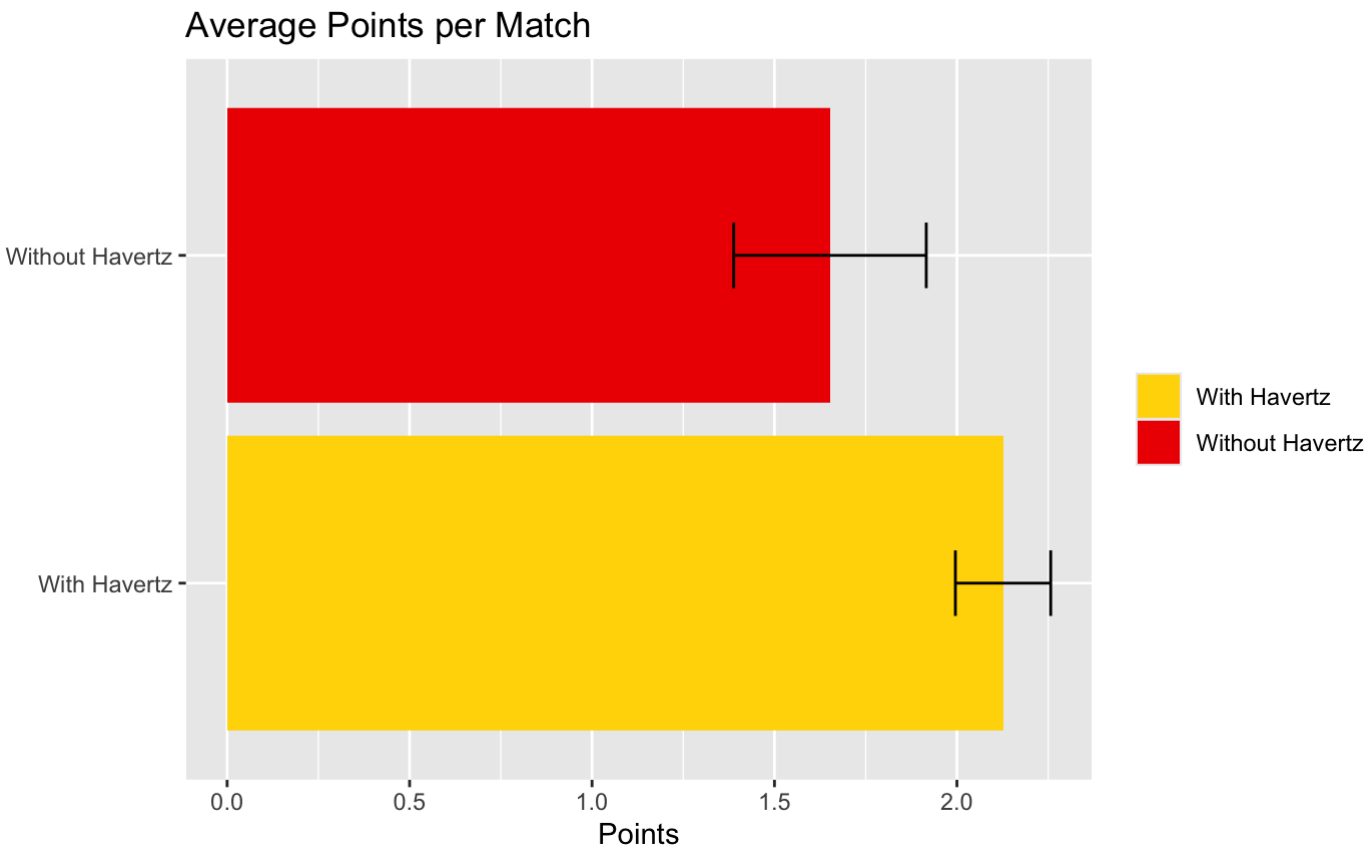

Figure 9: Average Points per Match Bar Plot

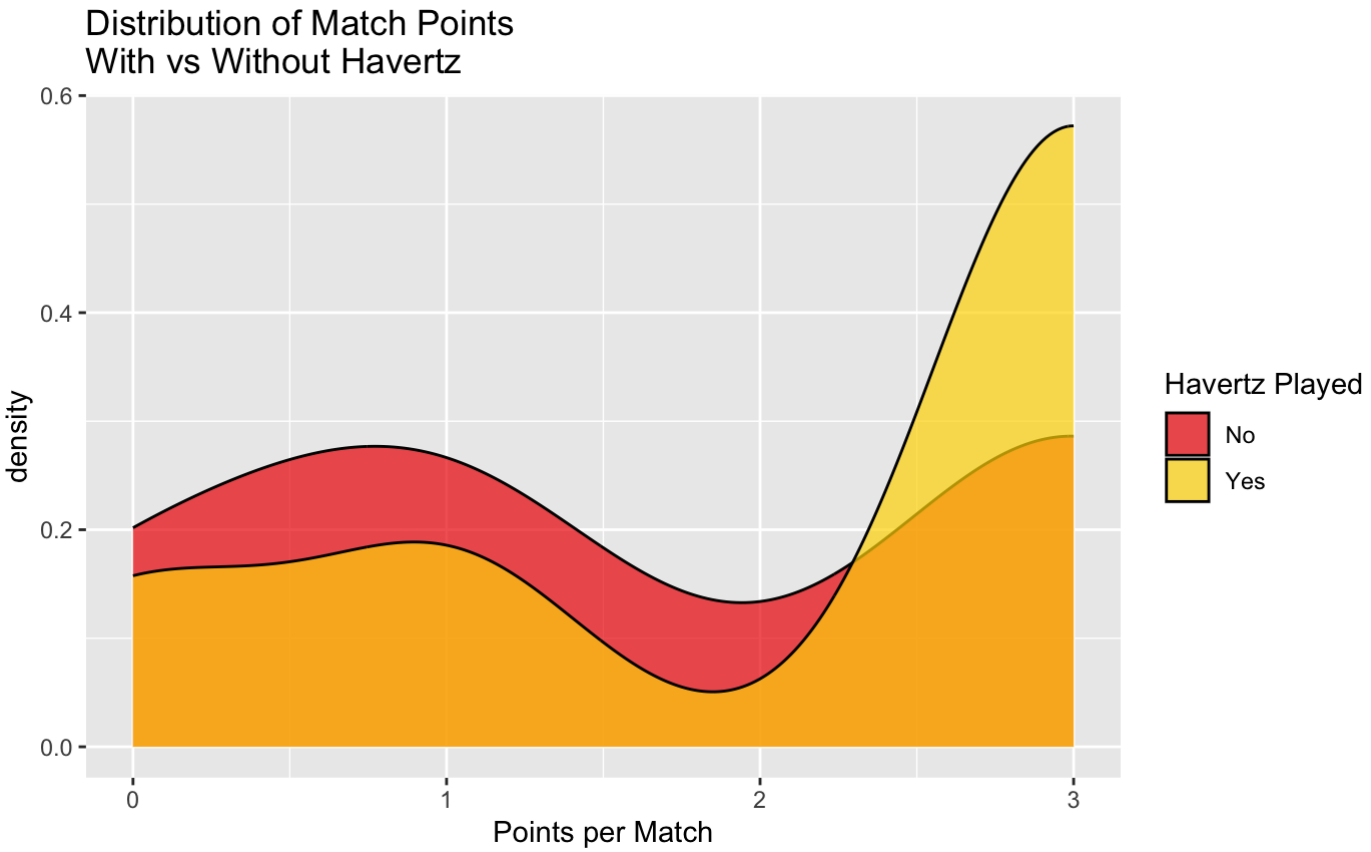

Figure 10: Distribution of Points With and Without Havertz

6: Random Forest

For lack of space I included this in the Appendix, Which Havertz/possession/venue metrics matter most for predicting Win (1 or 0): Random Forest: 110 Matches and 500 trees, OOB error is 31 percent meaning our forest correctly predicts a win around 69 percent of the time. Variable importance scores using RF: The further right the variable is the more important it is to predicting correctly: Havertz Goals increase win likelihood, Touches and Minutes mean more wins, Possession is very weak which is surprising.

Random forest Results. The Random Forest places `HavertzPlayed` among the top three predictors of points once complex interactions are allowed. Most interesting find was Possession scored very low for accuracy when I ran a Variable Importance Score which is surprising because I've always assumed that us having more of the ball would lead to more points but it actually clouds our accuracy.

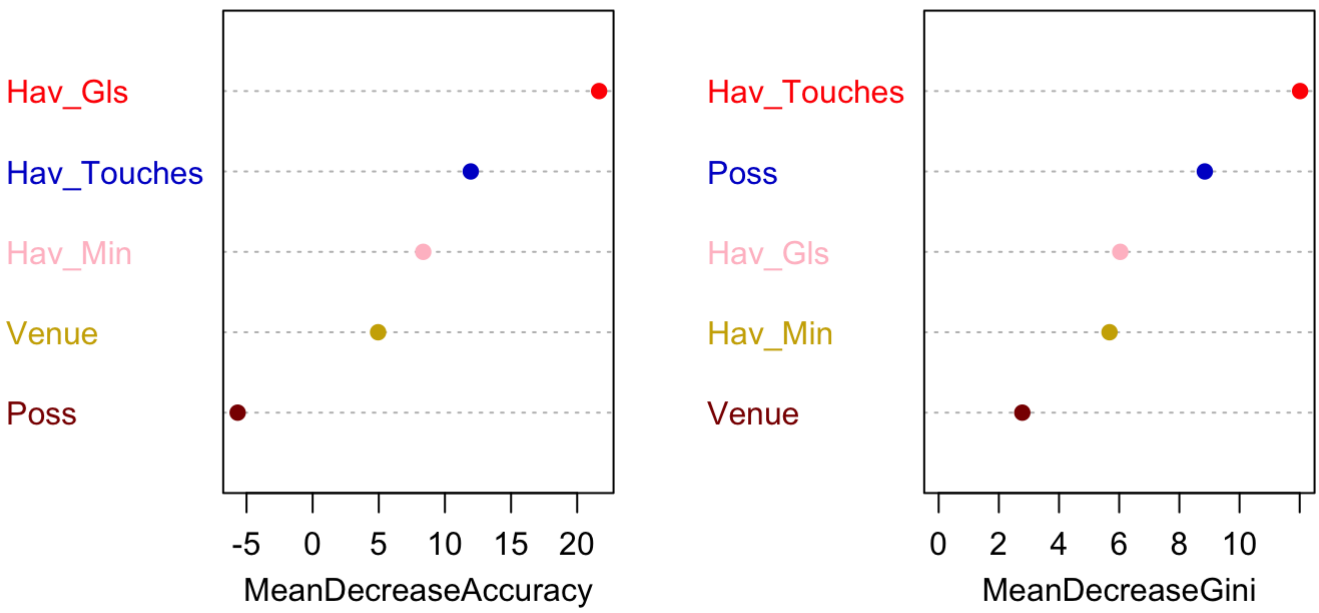

Figure 11: Variable Importance Scores for Random Forest

Appendix B: R Markdown Code

````
---
title: "404 Final Clean"
author: "Matheus Grover"
date: "2025-05-28"
output: pdf_document
---


Load the Data in:
```{R}
setwd("~/Desktop/404 Final")
library(readr)
library(dplyr)
library(lubridate)
library(tidyr)

#6 CSV's
all_players_23_24<-read_csv("arsenal_player_stats_2023_24.csv")
all_players_24_25<-read_csv("arsenal_player_stats_2024_25.csv")
schedule_23_24<-read_csv("arsenal_schedule_2023_24.csv")
schedule_24_25<-read_csv("arsenal_schedule_2024_25.csv")
havertz_23_24<-read_csv("havertz_2324_full_columns.csv")
havertz_24_25<-read_csv("havertz_2425_full_columns.csv")
```

Mutate and Clean:
```{R}
library(readr)
library(dplyr)
#create a master CSV
sched_all <- bind_rows(
  read_csv("arsenal_schedule_2023_24.csv") %>% mutate(Season = "2023-24"),
  read_csv("arsenal_schedule_2024_25.csv") %>% mutate(Season = "2024-25")
) %>%

  rename(HavertzPlayed = `Havertz Played`) %>%
  mutate(
    HavertzPlayed = case_when(
      HavertzPlayed %in% c(1, "1", TRUE, "TRUE", "Yes") ~ 1L,
      TRUE                                              ~ 0L
    ),
    # pull out W/D/L and score points
    ResultLetter = substr(trimws(Result), 1, 1),
    Points       = recode(ResultLetter, W = 3L, D = 1L, L = 0L),
    # turn Venue and Comp into factors now
    Venue        = factor(Venue),
    Comp         = factor(Comp)
  )
```
More Mutating
```{R}
library(readr)
library(dplyr)

havertz_stats <- bind_rows(
  read_csv("havertz_2324_full_columns.csv"),
  read_csv("havertz_2425_full_columns.csv")
)

havertz_stats_clean <- havertz_stats %>%
  rename_with(
    .cols = -Date,
    .fn   = ~paste0("Hav_", .x)
  )

intersect(names(sched_all), names(havertz_stats_clean))
sched_all <- sched_all %>%
  left_join(havertz_stats_clean, by = "Date")
write_csv(sched_all, "sched_all_with_havertz.csv")
names(sched_all)[names(sched_all) == "Hav_Tocuhes"] <- "Hav_Touches"
```
Graph of Total Goals:
````

```{R}
first_idx   <- which(hav_all$CumGoals > 0)[1]
first_date  <- hav_all$Date_dt[first_idx]
max_m       <- max(hav_all$MatchNo)
max_g       <- max(hav_all$CumGoals)

ggplot(hav_all, aes(MatchNo, CumGoals)) +
  geom_step(direction = "vh", size = 1.2, color = "gold") +
  geom_point(       size = 2,     color = "gold") +
  geom_vline(xintercept = first_idx, linetype = "dashed", color = "red2") +
  annotate("text",
           x     = first_idx,
           y     = 1.5,
           label = paste0("First goal:\n", format(first_date, "%d %b %Y")),
           vjust = 0, size = 3.5) +
  scale_x_continuous(breaks = seq(0, max_m, by = 5),
                     limits = c(0, max_m)) +
  scale_y_continuous(breaks = seq(0, max_g, by = 2),
                     expand = expansion(add = c(0, 1))) +
  labs(title = "Havertz Cumulative Premier League Goals",
       x     = "Match Number",
       y     = "Total Goals")
```
```
Histogram Distributions:
```{R}

vars_raw  <- c("xG", "Hav_xG", "Hav_Min", "Points")
vars_log  <- c("log_xG", "log_Hav_xG")
vars_sqrt <- c("sqrt_Hav_Min")

sched_all <- sched_all %>%
  mutate(
    log_xG        = log1p(xG),
    log_Hav_xG    = log1p(Hav_xG),
    sqrt_Hav_Min  = sqrt(Hav_Min)
  )

all_vars <- c(vars_raw, vars_log, vars_sqrt)


fill_cols  <- c("steelblue", "darkred", "gold3", "red3", "gold", "blue3", "gold")
names(fill_cols) <- all_vars




old_par <- par(no.readonly = TRUE)
par(mfrow = c(2, 3),
    mar = c(4, 4, 2, 1))
plot_hist <- function(v, main_lab, x_lab) {
  hist(
    sched_all[[v]],
    breaks = "FD",
    freq   = TRUE,
    main   = main_lab,
    xlab   = x_lab,
    col    = fill_cols[v],
    border = "white"
  )
}


plot_hist("xG",        "Distribution of xG",          "xG")
plot_hist("Hav_xG",    "Havertz xG",                  "Hav_xG")
plot_hist("Hav_Min",   "Havertz Minutes",             "Minutes")

plot_hist("log_xG",        "log(1 + xG)",            "log1p(xG)")
plot_hist("log_Hav_xG",    "log(1 + Hav_xG)",        "log1p(Hav_xG)")
plot_hist("Points",        "Match Points",           "Points (0/1/3)")

par(old_par)
```
```

Regression:
```{R}
library(dplyr)

sched_all <- sched_all %>%
  mutate(
    lnxG     = log1p(xG),
    lnHav_xG = log1p(Hav_xG),
    Hav_Min  = replace_na(Hav_Min, 0)
  )
```

```{R}
m0 <- lm(
  Points ~ HavertzPlayed + xG + Venue + Season,
  data = sched_all
)
summary(m0)
```

```{R}
m_clean <- lm(
  Points ~ Hav_Min + xG + Venue + Season,
  data = sched_all
)
summary(m_clean)

car::vif(m_clean)
AIC(m0, m_clean)
anova(m0, m_clean)
```

```{R}
m1 <- lm(Points ~ Hav_Min + lnxG + lnHav_xG + Venue, data = sched_all)
summary(m1)
#anova(m_clean, m1)

```
```{R}
sched_all <- sched_all %>%
  mutate(lnxG      = log1p(xG),
         lnHav_xG = log1p(Hav_xG))

m_int <- lm(
  Points ~ Hav_Min + lnxG * lnHav_xG + Venue + Season,
  data = sched_all
)
summary(m_int)

```
Helps copy and paste summary to the slide
```{R}
library(broom)
library(knitr)

tidy_tbl <- tidy(m1, conf.int = TRUE)
kable(tidy_tbl, digits = 3)
```
Linear Regression Graphs:
```{R}

ord        <- order(x)
pb         <- predict(m0,       newdata = sched_all)[ord]
pc         <- predict(m_clean,  newdata = sched_all)[ord]
pl         <- predict(m1,       newdata = sched_all)[ord]
pi         <- predict(m_int,    newdata = sched_all)[ord]


xrange <- range(x,              na.rm = TRUE)
yrange <- range(c(pb, pc, pl, pi), na.rm = TRUE)


plot(1, type = "n",
```

```
     xlim = xrange,
     ylim = yrange,
     xlab = "Team xG",
     ylab = "Match Points",
     main = "Linear Fits Only")

lines(x[ord], pb, col = "blue",      lwd = 2)
lines(x[ord], pc, col = "red",       lwd = 2)
lines(x[ord], pl, col = "gold",      lwd = 2)
lines(x[ord], pi, col = "pink", lwd = 2)

legend("topleft",
       legend = c("Baseline", "Clean Minutes", "Logged", "Interaction"),
       col    = c("blue","red","gold","pink"),
       lwd    = 2, bty = "n")
```

```

Kernel Regression + C
```{R}
dyn.load("HW4Q1C.so")

# x   <- data$Tax
# y   <- data$Consumption
# n   <- length(x)
xG_k<-sched_all$xG

with<- sched_all %>% filter(HavertzPlayed == 1)
without<- sched_all %>% filter(HavertzPlayed == 0)

x_with    <- with$xG
y_with    <- with$Points

x_without <- without$xG
y_without <- without$Points


# #SR
# b   <- bw.nrd(x)
b<- bw.nrd(xG_k)

# m   <- 80
m <- 100

# g2  <- seq(min(x), max(x), length.out = m)
g2  <- seq(min(xG_k), max(xG_k), length.out = m)

xG_kernel <- function(x, y, b, g2, m) {
  .C("HW4Q1C",
     as.double(x),
     as.double(y),
     as.integer(length(x)),
     as.double(b),
     as.double(g2),
     as.integer(m),
     res2 = double(m))$res2
}
# out <- .C("HW4Q1C",
#           as.double(x), as.double(y), as.integer(n),
#           as.double(b),
#           as.double(g2), as.integer(m),
#           res2 = double(m))
# fhat <- out$res2
fhat_with<- xG_kernel(x_with, y_with, b, g2, m)
fhat_without<-xG_kernel(x_without, y_without, b, g2, m)
#
# # Plot
# colors <- heat.colors(length(y))[rank(y)]
# plot(x, y,
#      #ylim = c()
#      main = "Kernel Regression Estimates",
#      xlab="Tax",
#      ylab="Consumption",
#      pch=21,
```
```

```
#       bg = colors,
#       col="black"
#       )
#
# lines(g2, fhat, col="skyblue1", lwd=2)

#Plot With
colors <- heat.colors(length(y_with))[rank(y_with)]
plot(x_with, y_with,
     #ylim = c()
     main = "Kernel Regression Estimates With",
     xlab="XG",
     ylab="Points",
     pch=21,
     bg = colors,
     col="black"
     )

lines(g2, fhat_with, col="skyblue1", lwd=2)

#Plot Without
colors <- heat.colors(length(y_without))[rank(y_without)]
plot(x_without, y_without,
     #ylim = c()
     main = "Kernel Regression Estimates Without",
     xlab="XG",
     ylab="Points",
     pch=21,
     bg = colors,
     col="black"
     )

lines(g2, fhat_without, col="orchid", lwd=2)
```
```
Combine the Kernel Regressions into one
```{R}
plot(NULL, xlim = range(g2), ylim = c(0, 3),
     main = "Kernel Regression: Points vs xG",
     xlab = "xG", ylab = "Smoothed Points")

lines(g2, fhat_with, col = "gold", lwd = 2)
lines(g2, fhat_without, col = "red2", lwd = 2)
legend("bottomleft", legend = c("With Havertz", "Without Havertz"),
       col = c("gold", "red2"), lwd = 2)

```

2D KDE
```{R}
library(MASS)

# Estimate 2D KDEs
kde_with <- with(with, kde2d(xG, GF, n = 50))
kde_without <- with(without, kde2d(xG, GF, n = 50))

par(mfrow = c(1, 2), mar = c(4, 4, 3, 2))
contour(kde_without,
        xlab = "xG", ylab = "Goals For",
        main = "2D KDE with Points (Without)",
        lwd = 2, col = "red2")

# Overlay actual match data
points(without$xG, without$GF, pch = 19, col = rgb(1, 1, 0, 1))

contour(kde_with,
        xlab = "xG", ylab = "Goals For",
        main = "2D KDE with Points (With)",
        lwd = 2, col = "gold")

points(with$xG, with$GF, pch = 19, col = rgb(1, 0, 0, 1))

```
```
By themselves:
```{R}


```

```
contour(kde_without,
        xlab = "xG", ylab = "Goals For",
        main = "2D KDE with Points (Without Havertz)",
        lwd = 2, col = "red2")

# Overlay actual match data
points(without$xG, without$GF, pch = 19, col = rgb(1, 1, 0, 1))

contour(kde_with,
        xlab = "xG", ylab = "Goals For",
        main = "2D KDE with Points (With Havertz)",
        lwd = 2, col = "gold")

points(with$xG, with$GF, pch = 19, col = rgb(1, 0, 0, 1))
```

```
Logit
```{R}
sched_all <- sched_all %>%
  mutate(Win = if_else(Points == 3, 1L, 0L))

logit_W <- glm(Win ~ xG + HavertzPlayed, data = sched_all, family = binomial())

summary(logit_W)


library(ggplot2)

sched_all <- sched_all %>%
  mutate(PredictedProb = predict(logit_W, type = "response"))

ggplot(sched_all, aes(x = xG, y = PredictedProb, color = factor(HavertzPlayed))) +
  geom_point(size =2, alpha=1) +
  geom_smooth(method = "loess", se = FALSE) +
  scale_color_manual(
    name   = "Havertz Played",
    values = c(
      "0" = "gold",
      "1" = "red2"
    ),
    labels = c("No", "Yes")
  ) +
  labs(
    title = "Predicted Win Probability vs xG",
    x     = "Expected Goals (xG)",
    y     = "P(Win)"
  )


```

```
Random Forest
```{R}
cols <- c("Win",
          "Hav_Gls",
          "Hav_Touches",
          "Hav_Min",
          "Poss",
          "Venue")


rf_custom_df <- sched_all[complete.cases(sched_all[, cols]), cols]

set.seed(123)
rf_custom <- randomForest(
  factor(Win) ~ .,
  data       = rf_custom_df,
  ntree      = 500,
  importance = TRUE
)
```

```
print(rf_custom)
importance(rf_custom)

my_cols <- c("darkred", "gold3", "pink", "blue3", "red")

varImpPlot(
  rf_custom,
  n.var = 5,
  main  = "Top Predictors",
  col   = my_cols,
  pch   = 19
)
```

```
```

```
Ending
Points Per Match:
```{R}
library(dplyr)
library(ggplot2)


summary_df <- sched_all %>%
  group_by(HavertzPlayed) %>%
  summarise(
    n         = n(),
    AvgPoints = mean(Points),
    SE        = sd(Points) / sqrt(n),
    .groups   = "drop"
  ) %>%
  mutate(
    Status = if_else(HavertzPlayed == 1, "With Havertz", "Without Havertz")
  )

ggplot(summary_df, aes(x = AvgPoints, y = Status, fill = Status)) +
  geom_col() +
  geom_errorbarh(aes(xmin = AvgPoints - SE, xmax = AvgPoints + SE),
                 height = 0.2) +
  scale_fill_manual(values = c(
    "With Havertz"    = "gold",
    "Without Havertz" = "red2"
  )) +
  labs(
    title = "Average Points per Match",
    x     = "Points",
    y     = NULL,
    fill  = NULL
  )
```
```{R}
ggplot(sched_all, aes(x = Points, fill = factor(HavertzPlayed))) +
  geom_density(alpha = 0.7) +
  scale_fill_manual(
    name   = "Havertz Played",
    values = c("0" = "red2", "1" = "gold"),
    labels = c("No", "Yes")
  ) +
  labs(
    x     = "Points per Match",
    title = "Distribution of Match Points\nWith vs Without Havertz"
  )
```
```

Finale

````
```{R}
df_2425 <- sched_all %>% filter(Season == "2024-25",
                                Comp   == "Premier League")
with25    <- df_2425 %>% filter(HavertzPlayed == 1)
without25 <- df_2425 %>% filter(HavertzPlayed == 0)

avg_with <- mean(with25$Points)
#avg_without <- mean(without25$Points)

missed <- nrow(without25)

#Our actual points this season
actual_total <- sum(df_2425$Points)

#portion without havets
actual_points_missed <- sum(without25$Points)

#What we should've got
expected_points_missed <- avg_with * missed

#where we would be
adjusted_total <- actual_total - actual_points_missed + expected_points_missed

print(missed)
print(actual_total)
print(actual_points_missed)
print(adjusted_total)

```
````

Listing 1: Full R Markdown file: $528_404_Final.Rmd$

Appendix C: C Code for Kernel Regression

```
#include <R.h>
#include <Rmath.h>


void HW4Q1C(double *x, double *y, int *n, double *b,
                    double *g2, int *m, double *res2)
{
    int i, j;
    double a1, a2, c, d;

    for(i = 0; i < *m; i++){
        a1 = 0.0;
        a2 = 0.0;
        for(j = 0; j < *n; j++){
            c = (g2[i] - x[j]) / *b;

            d = dnorm(c, 0.0, 1.0, 0) / *b;

            a1 += y[j] * d;
            a2 +=     d;
        }
        res2[i] = (a2 > 0.0 ? a1 / a2 : 0.0);
    }
}
```

Listing 2: C code file: `HW4Q1C.c`