STA 571 Final Project - Enhanced with Streak Features

Trey Chase, Tully Cannon

Table of contents

1 Setup

```
library(readr)
library(tidyverse)  # Load tidyverse FIRST
library(lme4)       # Load lme4 AFTER tidyverse so dplyr::select takes precedence
library(ggplot2)
library(gridExtra)
library(depmixS4)
library(xgboost)
library(nnet)
library(caret)
library(knitr)
library(kableExtra)
library(pROC)
library(zoo)

# Explicitly handle select conflicts
select <- dplyr::select
```

```r
set.seed(571)
```

2 Data Loading

```r
file_path = "~/Downloads/STA 571 Final Project/NBA_2024_Shots.csv"
shots_df = read_csv(file_path, show_col_types = FALSE)
```

3 Exploratory Data Analysis

```r
shots_df = shots_df %>%
  mutate(
    SHOT_MADE_NUM = as.integer(SHOT_MADE),
    SHOT_TYPE_NUM = ifelse(SHOT_TYPE == "2PT Field Goal", 2, 3),
    PERIOD = QUARTER,
    TIME_REMAINING = MINS_LEFT * 60 + SECS_LEFT,
    DISTANCE_SQUARED = SHOT_DISTANCE^2
  ) %>%
  filter(!is.na(SHOT_MADE))
```

3.1 Overall Shot Statistics

```r
shots_df %>%
  summarise(
    n_shots = n(),
    make_rate = mean(SHOT_MADE),
    avg_distance = mean(SHOT_DISTANCE),
    sd_distance = sd(SHOT_DISTANCE)
  ) %>%
  kable(
    digits = 3,
    caption = "Overall Shot Statistics",
    col.names = c("Total Shots", "Make Rate", "Avg Distance (ft)", "SD Distance"),
    align = c('r', 'r', 'r', 'r'),
    format.args = list(big.mark = ",")
```

```
  ) %>%
  kable_styling(bootstrap_options = c("striped", "hover"),
                full_width = FALSE,
                position = "left")
```

Table 1: Overall Shot Statistics

Total Shots	Make Rate	Avg Distance (ft)	SD Distance
218,701	0.474	13.535	10.583

3.2 Shot Statistics by Type

```
shots_df %>%
  group_by(SHOT_TYPE) %>%
  summarise(
    n = n(),
    make_rate = mean(SHOT_MADE),
    .groups = "drop"
  ) %>%
  kable(
    digits = 3,
    caption = "Shot Statistics by Type",
    col.names = c("Shot Type", "Total Shots", "Make Rate"),
    align = c('l', 'r', 'r'),
    format.args = list(big.mark = ",")
  ) %>%
  kable_styling(bootstrap_options = c("striped", "hover"),
                full_width = FALSE,
                position = "left")
```

Table 2: Shot Statistics by Type

Shot Type	Total Shots	Make Rate
2PT Field Goal	132,346	0.545
3PT Field Goal	86,355	0.366

3.3 Visualizations

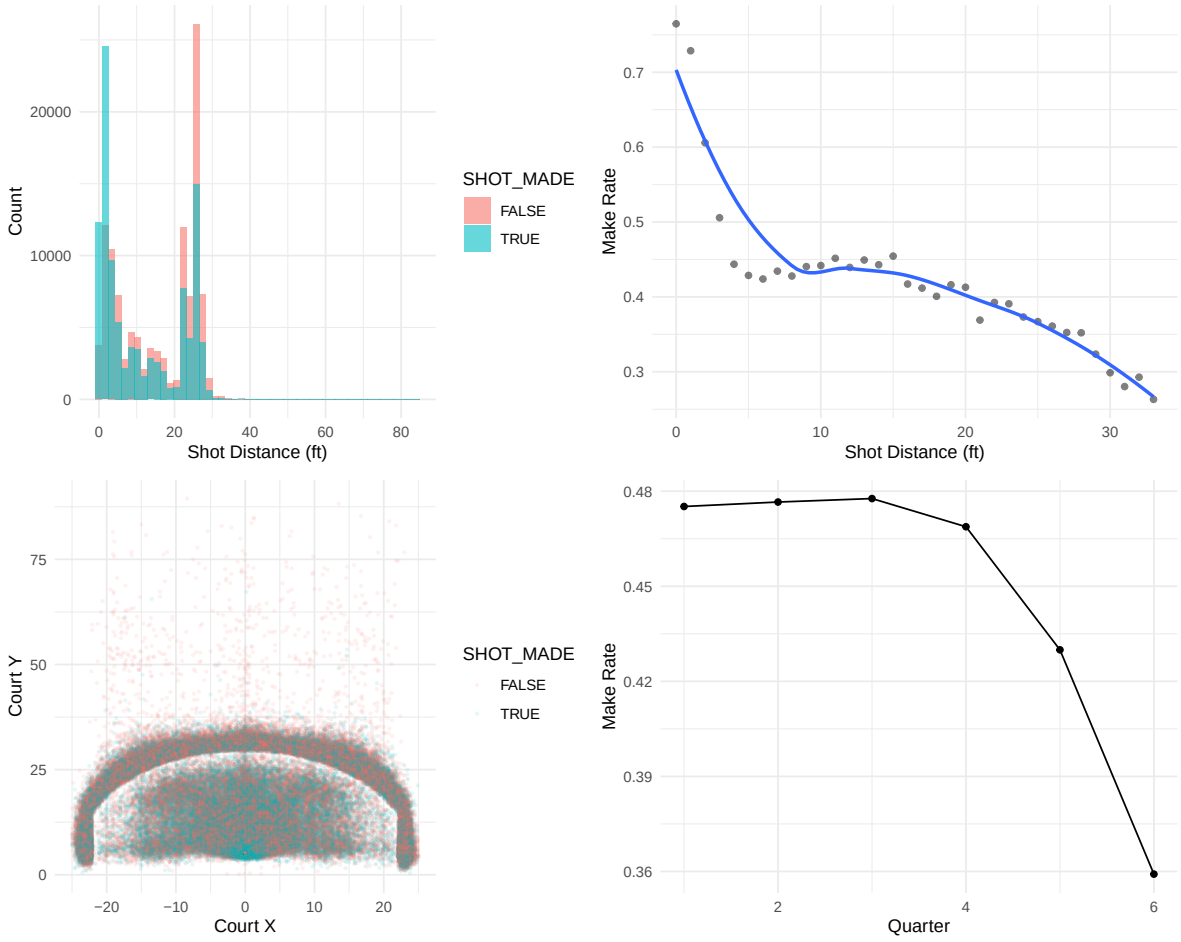
```
p1 = ggplot(shots_df, aes(x = SHOT_DISTANCE, fill = SHOT_MADE)) +
  geom_histogram(bins = 50, position = "identity", alpha = 0.6) +
  labs(x = "Shot Distance (ft)", y = "Count") +
  theme_minimal()

p2 = shots_df %>%
  group_by(SHOT_DISTANCE) %>%
  summarise(make_rate = mean(SHOT_MADE), n = n(), .groups = "drop") %>%
  filter(n >= 100) %>%
  ggplot(aes(x = SHOT_DISTANCE, y = make_rate)) +
  geom_point(alpha = 0.5) +
  geom_smooth(se = FALSE, method = "loess", formula = y ~ x) +
  labs(x = "Shot Distance (ft)", y = "Make Rate") +
  theme_minimal()

p3 = ggplot(shots_df, aes(x = LOC_X, y = LOC_Y, color = SHOT_MADE)) +
  geom_point(alpha = 0.1, size = 0.5) +
  labs(x = "Court X", y = "Court Y") +
  theme_minimal()

p4 = shots_df %>%
  group_by(QUARTER) %>%
  summarise(make_rate = mean(SHOT_MADE), .groups = "drop") %>%
  ggplot(aes(x = QUARTER, y = make_rate)) +
  geom_line() +
  geom_point() +
  labs(x = "Quarter", y = "Make Rate") +
  theme_minimal()

grid.arrange(p1, p2, p3, p4, ncol = 2)
```



3.4 Top Teams by Make Rate

```
team_stats = shots_df %>%
  group_by(TEAM_NAME) %>%
  summarise(
    n_shots = n(),
    make_rate = mean(SHOT_MADE),
    avg_distance = mean(SHOT_DISTANCE),
    .groups = "drop"
  ) %>%
  arrange(desc(make_rate))

head(team_stats, 10) %>%
```

```
  kable(
    digits = 3,
    caption = "Top 10 Teams by Make Rate",
    col.names = c("Team", "Total Shots", "Make Rate", "Avg Distance (ft)"),
    align = c('l', 'r', 'r', 'r'),
    format.args = list(big.mark = ",")
  ) %>%
  kable_styling(bootstrap_options = c("striped", "hover"),
                full_width = FALSE,
                position = "left")
```

Table 3: Top 10 Teams by Make Rate

Team	Total Shots	Make Rate	Avg Distance (ft)
Indiana Pacers	7,599	0.507	13.021
Los Angeles Lakers	7,177	0.499	12.852
Oklahoma City Thunder	7,324	0.499	13.288
Denver Nuggets	7,279	0.496	12.827
Phoenix Suns	7,063	0.493	13.826
LA Clippers	7,108	0.489	13.576
Milwaukee Bucks	7,258	0.487	14.534
Boston Celtics	7,396	0.487	15.149
New Orleans Pelicans	7,165	0.486	13.059
Minnesota Timberwolves	6,974	0.485	12.973

4 Enhanced Data Preparation with Streak Features

```
# IMPORTANT: Create streak features at TEAM-GAME level
# Streaks are team-specific and reset at the start of each game
shots_df_enhanced = shots_df %>%
  arrange(TEAM_NAME, GAME_ID, QUARTER, desc(MINS_LEFT), desc(SECS_LEFT)) %>%
  group_by(TEAM_NAME, GAME_ID) %>%  # Team-specific, game-bounded streaks
  mutate(
    # Lagged made shots (within team-game)
    lag1_made = lag(SHOT_MADE_NUM, 1, default = 0),
    lag2_made = lag(SHOT_MADE_NUM, 2, default = 0),
    lag3_made = lag(SHOT_MADE_NUM, 3, default = 0),
    lag4_made = lag(SHOT_MADE_NUM, 4, default = 0),
```

```
    lag5_made = lag(SHOT_MADE_NUM, 5, default = 0),

    # Recent makes (last N shots within team-game)
    recent_makes_3 = lag1_made + lag2_made + lag3_made,
    recent_makes_5 = lag1_made + lag2_made + lag3_made + lag4_made + lag5_made,

    # Current streak (consecutive makes within team-game)
    streak = {
      s = rep(0, n())
      for(i in 2:n()) {
        if(SHOT_MADE_NUM[i-1] == 1) {
          s[i] = s[i-1] + 1
        }
      }
      s
    },

    # Rolling statistics (within team-game)
    rolling_makes_10 = rollsumr(lag(SHOT_MADE_NUM, default = 0),
                                k = 10, fill = 0, align = "right"),
    rolling_rate_10 = rolling_makes_10 / pmin(row_number() - 1, 10)
  ) %>%
  ungroup()

# Verify data structure
cat("Total shots:", nrow(shots_df_enhanced), "\n")
```

```
Total shots: 218701
```

```
cat("Number of teams:", n_distinct(shots_df_enhanced$TEAM_NAME), "\n")
```

```
Number of teams: 30
```

```
cat("Number of games:", n_distinct(shots_df_enhanced$GAME_ID), "\n")
```

```
Number of games: 1230
```

```
cat("Number of team-game combinations:",
    n_distinct(shots_df_enhanced %>% dplyr::select(TEAM_NAME, GAME_ID)), "\n")
```

```
Number of team-game combinations: 2460
```

4.1 Relationship Between Streaks and Make Rate

```
# Examine relationship between streaks and make rate
shots_df_enhanced %>%
  filter(recent_makes_3 >= 0) %>%
  group_by(recent_makes_3) %>%
  summarise(
    n = n(),
    make_rate = mean(SHOT_MADE),
    .groups = "drop"
  ) %>%
  kable(
    digits = 3,
    caption = "Make Rate by Recent Makes (Last 3 Shots)",
    col.names = c("Recent Makes (Last 3)", "N Shots", "Make Rate"),
    align = c('c', 'r', 'r'),
    format.args = list(big.mark = ",")
  ) %>%
  kable_styling(bootstrap_options = c("striped", "hover"),
                full_width = FALSE,
                position = "left")
```

Table 4: Make Rate by Recent Makes (Last 3 Shots)

Recent Makes (Last 3)	N Shots	Make Rate
0	34,601	0.474
1	85,775	0.478
2	76,362	0.473
3	21,963	0.465

```
# Streak analysis
shots_df_enhanced %>%
  filter(streak <= 5) %>%
  group_by(streak) %>%
  summarise(
    n = n(),
    make_rate = mean(SHOT_MADE),
    .groups = "drop"
  ) %>%
  kable(
```

```
    digits = 3,
    caption = "Make Rate by Current Streak Length",
    col.names = c("Current Streak", "N Shots", "Make Rate"),
    align = c('c', 'r', 'r'),
    format.args = list(big.mark = ",")
  ) %>%
  kable_styling(bootstrap_options = c("striped", "hover"),
                full_width = FALSE,
                position = "left")
```

Table 5: Make Rate by Current Streak Length

Current Streak	N Shots	Make Rate
0	116,095	0.478
1	54,913	0.474
2	25,730	0.465
3	11,853	0.468
4	5,495	0.462
5	2,501	0.447

5 Model 1: Hidden Markov Model with Streak Covariate

5.1 Data Preparation for HMM

```
# Prepare data with proper sequence structure
# Each team-game is a separate sequence
team_data = shots_df_enhanced %>%
  arrange(TEAM_NAME, GAME_ID, QUARTER, desc(MINS_LEFT), desc(SECS_LEFT)) %>%
  mutate(
    streak = ifelse(is.na(streak), 0, streak),
    team_game_id = paste(TEAM_NAME, GAME_ID, sep = "_")
  )

# Calculate sequence lengths (ntimes) for each team-game
sequence_lengths = team_data %>%
  group_by(team_game_id) %>%
  summarise(n_shots = n(), .groups = "drop") %>%
  pull(n_shots)
```

```r
cat("Total shots in team_data:", nrow(team_data), "\n")
```

```
Total shots in team_data: 218701
```

```r
cat("Number of team-game sequences:", length(sequence_lengths), "\n")
```

```
Number of team-game sequences: 2460
```

```r
cat("Total shots (check):", sum(sequence_lengths), "\n\n")
```

```
Total shots (check): 218701
```

```r
cat("Shots per team-game summary:\n")
```

```
Shots per team-game summary:
```

```r
print(summary(sequence_lengths))
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   67.0    84.0    89.0    88.9    93.0   119.0
```

```r
cat("\n")
```

```r
# Verify no NAs
cat("NAs in SHOT_MADE_NUM:", sum(is.na(team_data$SHOT_MADE_NUM)), "\n")
```

```
NAs in SHOT_MADE_NUM: 0
```

```r
cat("NAs in streak:", sum(is.na(team_data$streak)), "\n")
```

```
NAs in streak: 0
```

5.2 HMM Model Specification (2 States)

The Hidden Markov Model uses 2 states to capture shooting patterns:

$$
\begin{aligned}
Y_t | S_t = k, X_t &\sim \text{Bernoulli}(\pi_{k,t}) \\
\text{logit}(\pi_{k,t}) &= \beta_{k,0} + \beta_{k,1} \cdot \text{Streak}_t \\
S_t | S_{t-1} = j &\sim \text{Categorical}(\Gamma_j) \\
\Gamma_{jk} &= \Pr(S_t = k | S_{t-1} = j)
\end{aligned}
$$

where $k, j \in \{1, 2\}$ representing two distinct shooting states.

Key Feature: The HMM respects game boundaries through the `ntimes` parameter, ensuring that: - States do NOT carry over between games - Each team-game is treated as an independent sequence - Streak variables are team-specific and game-bounded

5.3 HMM Fitting

```
set.seed(571)

cat("Fitting HMM with game boundaries...\n\n")
```

```
Fitting HMM with game boundaries...
```

```
# Create model with ntimes to specify sequence boundaries
hmm_model = depmix(
  response = SHOT_MADE_NUM ~ streak,
  data = team_data,
  nstates = 2,
  family = binomial("logit"),
  ntimes = sequence_lengths  # CRITICAL: Specifies game boundaries
)

# Set reasonable starting values
hmm_model = setpars(hmm_model,
                    value = c(
                      0.5, 0.5,          # initial state probs
                      0.7, 0.3,          # From State 1
                      0.3, 0.7,          # From State 2
                      -0.3, 0.2,         # State 1 params
```

```
                      0.3, 0.25          # State 2 params
                    ))

# Fit model
hmm_fit = fit(hmm_model,
              verbose = TRUE,
              emcontrol = em.control(maxit = 5000, tol = 1e-8))
```

```
iteration 0 logLik: -151295
```

```
converged at iteration 1 with logLik: -151295
```

```
cat("\n=== MODEL CONVERGENCE ===\n")
```

```

=== MODEL CONVERGENCE ===
```

```
cat("Convergence message:", hmm_fit@message, "\n")
```

```
Convergence message: Log likelihood converged to within tol. (relative change)
```

```
cat("Number of iterations:", hmm_fit@ntimes, "\n")
```

```
Number of iterations: 87 89 95 101 93 87 93 86 92 93 102 85 87 75 116 102 88 96 101 96 100 10
```

5.4 HMM Results (2 States)

```
cat("Log-Likelihood:", logLik(hmm_fit), "\n\n")
```

```
Log-Likelihood: -151295
```

```
cat("AIC:", AIC(hmm_fit), "\n\n")
```

```
AIC: 302604
```

```r
cat("BIC:", BIC(hmm_fit), "\n\n")
```

```
BIC: 302676.1
```

```r
hmm_summary = summary(hmm_fit)
```

```
Initial state probabilities model
pr1 pr2
0.5 0.5

Transition matrix
       toS1 toS2
fromS1  0.7  0.3
fromS2  0.3  0.7

Response parameters
Resp 1 : binomial
    Re1.(Intercept) Re1.streak
St1          -0.093     -0.015
St2          -0.088     -0.014
```

```r
print(hmm_summary)
```

```
    Re1.(Intercept) Re1.streak
St1          -0.093     -0.015
St2          -0.088     -0.014
```

5.5 Extract and Display Parameters (2 States)

```r
# Extract parameters
params = getpars(hmm_fit)

# Initial state probabilities
initial_probs = params[1:2]
names(initial_probs) = c("State 1", "State 2")

kable(
  data.frame(State = names(initial_probs), Probability = initial_probs),
```

```
  digits = 3,
  caption = "HMM Initial State Probabilities",
  col.names = c("State", "Probability"),
  align = c('l', 'r'),
  row.names = FALSE
) %>%
  kable_styling(bootstrap_options = c("striped", "hover"),
                full_width = FALSE,
                position = "left")
```

Table 6: HMM Initial State Probabilities

State	Probability
State 1	0.5
State 2	0.5

```
# Transition matrix (2x2)
transition_matrix = matrix(
  params[3:6],
  nrow = 2, byrow = TRUE
)

transition_df = data.frame(
  From = c("State 1", "State 2"),
  To_State_1 = transition_matrix[, 1],
  To_State_2 = transition_matrix[, 2]
)

kable(
  transition_df,
  digits = 3,
  caption = "HMM Transition Matrix",
  col.names = c("From State", "To State 1", "To State 2"),
  align = c('l', 'r', 'r'),
  row.names = FALSE
) %>%
  kable_styling(bootstrap_options = c("striped", "hover"),
                full_width = FALSE,
                position = "left")
```

Table 7: HMM Transition Matrix

From State	To State 1	To State 2
State 1	0.7	0.3
State 2	0.3	0.7

```
# Response parameters
state_params = data.frame(
  State = c("State 1", "State 2"),
  Intercept = c(params[7], params[9]),
  Slope_Streak = c(params[8], params[10])
)

kable(
  state_params,
  digits = 3,
  caption = "HMM State-Specific Parameters (Logit Scale)",
  col.names = c("State", "Intercept", "Streak Coefficient"),
  align = c('l', 'r', 'r'),
  row.names = FALSE
) %>%
  kable_styling(bootstrap_options = c("striped", "hover"),
                full_width = FALSE,
                position = "left")
```

Table 8: HMM State-Specific Parameters (Logit Scale)

State	Intercept	Streak Coefficient
State 1	-0.093	-0.015
State 2	-0.088	-0.014

5.6 HMM State Interpretation (2 States)

```
team_data$hmm_state = posterior(hmm_fit, type = "viterbi")$state

team_data %>%
  group_by(hmm_state) %>%
  summarise(
```

```
    n = n(),
    pct_of_shots = n() / nrow(team_data) * 100,
    make_rate = mean(SHOT_MADE),
    avg_distance = mean(SHOT_DISTANCE),
    avg_streak = mean(streak),
    median_streak = median(streak),
    .groups = "drop"
  ) %>%
  kable(
    digits = 3,
    caption = "Observed Statistics by HMM State",
    col.names = c("State", "N Shots", "Pct of Shots", "Make Rate",
                  "Avg Distance", "Avg Streak", "Median Streak"),
    align = c('c', 'r', 'r', 'r', 'r', 'r', 'r'),
    format.args = list(big.mark = ",")
  ) %>%
  kable_styling(bootstrap_options = c("striped", "hover"),
                full_width = FALSE,
                position = "left")
```

Table 9: Observed Statistics by HMM State

State	N Shots	Pct of Shots	Make Rate	Avg Distance	Avg Streak	Median Streak
1	114,716	52.453	0.433	13.678	0.717	0
2	103,985	47.547	0.519	13.376	1.046	1

```
# Interpret the streak effect in each state
streak_values = 0:5
state1_probs = plogis(state_params$Intercept[1] +
                      state_params$Slope_Streak[1] * streak_values)
state2_probs = plogis(state_params$Intercept[2] +
                      state_params$Slope_Streak[2] * streak_values)

streak_effects = data.frame(
  Streak = streak_values,
  State1_Prob = state1_probs,
  State2_Prob = state2_probs
)

kable(
  streak_effects,
```

```
  digits = 3,
  caption = "Predicted Make Probability by State and Streak",
  col.names = c("Streak Length", "State 1 Prob", "State 2 Prob"),
  align = c('c', 'r', 'r')
) %>%
  kable_styling(bootstrap_options = c("striped", "hover"),
                full_width = FALSE,
                position = "left")
```

Table 10: Predicted Make Probability by State and Streak

Streak Length	State 1 Prob	State 2 Prob
0	0.477	0.478
1	0.473	0.475
2	0.469	0.471
3	0.466	0.468
4	0.462	0.465
5	0.458	0.461

5.7 HMM Game Boundary Verification

```
# Verify that states reset at game boundaries
sample_games = team_data %>%
  filter(TEAM_NAME == team_stats$TEAM_NAME[1]) %>%  # Take top team
  arrange(GAME_ID, QUARTER, desc(MINS_LEFT), desc(SECS_LEFT)) %>%
  group_by(GAME_ID) %>%
  dplyr::slice(1) %>%  # First shot of each game - use dplyr::slice explicitly
  ungroup() %>%  # Ungroup before head()
  head(10)

cat("First shot of each game for", team_stats$TEAM_NAME[1], ":\n")
```

```
First shot of each game for Indiana Pacers :
```

```
cat("Game boundaries are respected in the HMM via ntimes parameter\n\n")
```

```
Game boundaries are respected in the HMM via ntimes parameter
```

```r
sample_games %>%
  dplyr::select(GAME_ID, QUARTER, hmm_state, SHOT_MADE, streak) %>%
  kable(
    caption = "First Shot of Each Game (Sample)",
    col.names = c("Game ID", "Quarter", "HMM State", "Made", "Streak"),
    align = c('l', 'c', 'c', 'c', 'c')
  ) %>%
  kable_styling(bootstrap_options = c("striped", "hover"),
                full_width = FALSE,
                position = "left")
```

Table 11: First Shot of Each Game (Sample)

Game ID	Quarter	HMM State	Made	Streak
22300001	1	2	TRUE	0
22300019	1	2	TRUE	0
22300039	1	2	TRUE	0
22300047	1	2	TRUE	0
22300064	1	2	TRUE	0
22300091	1	2	FALSE	0
22300102	1	1	FALSE	0
22300118	1	1	TRUE	0
22300133	1	2	FALSE	0
22300146	1	2	FALSE	0

6 Model 2: XGBoost with Streak Features

```r
set.seed(571)
train_idx = createDataPartition(shots_df_enhanced$SHOT_MADE, p = 0.8, list = FALSE)

features = c("LOC_X", "LOC_Y", "SHOT_DISTANCE", "DISTANCE_SQUARED",
             "SHOT_TYPE_NUM", "QUARTER", "TIME_REMAINING",
             "recent_makes_3", "recent_makes_5", "streak", "rolling_rate_10")

train_x = as.matrix(shots_df_enhanced[train_idx, features])
train_y = shots_df_enhanced$SHOT_MADE_NUM[train_idx]
test_x = as.matrix(shots_df_enhanced[-train_idx, features])
test_y = shots_df_enhanced$SHOT_MADE_NUM[-train_idx]
```

```r
dtrain = xgb.DMatrix(data = train_x, label = train_y)
dtest = xgb.DMatrix(data = test_x, label = test_y)
```

6.1 XGBoost Model

The XGBoost model learns:

$$\hat{y}_i = \sum_{k=1}^{K} f_k(\mathbf{x}_i)$$

$$\mathcal{L} = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

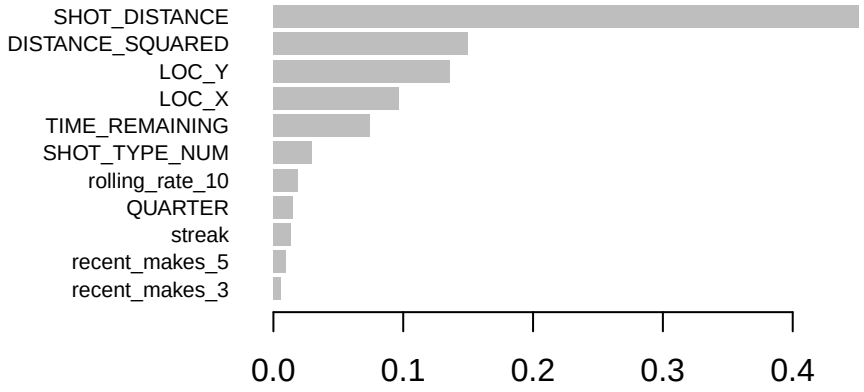
$$\Omega(f) = \gamma T + \frac{1}{2}\lambda ||\mathbf{w}||^2$$

```r
xgb_params = list(
  objective = "binary:logistic",
  eval_metric = "logloss",
  eta = 0.1,
  max_depth = 6,
  subsample = 0.8,
  colsample_bytree = 0.8
)

xgb_model = xgb.train(
  params = xgb_params,
  data = dtrain,
  nrounds = 100,
  watchlist = list(train = dtrain, test = dtest),
  verbose = 0
)

importance_matrix = xgb.importance(
  feature_names = features,
  model = xgb_model
)

xgb.plot.importance(importance_matrix, top_n = 11)
```



7 Model 3: Neural Network with Streak Features

7.1 Neural Network Architecture

$$\mathbf{h}^{(1)} = \text{ReLU}(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)})$$

$$\mathbf{h}^{(2)} = \text{ReLU}(\mathbf{W}^{(2)}\mathbf{h}^{(1)} + \mathbf{b}^{(2)})$$

$$\hat{y} = \sigma(\mathbf{W}^{(3)}\mathbf{h}^{(2)} + \mathbf{b}^{(3)})$$

$$\mathcal{L} = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]$$

```
train_x_scaled = scale(train_x)
test_x_scaled = scale(test_x,
                      center = attr(train_x_scaled, "scaled:center"),
                      scale = attr(train_x_scaled, "scaled:scale"))

train_df = data.frame(train_x_scaled, y = train_y)
test_df = data.frame(test_x_scaled)

set.seed(571)
```

```
nn_model = nnet(y ~ .,
                data = train_df,
                size = 32,
                maxit = 500,
                decay = 0.01,
                linout = FALSE,
                trace = FALSE)

nn_preds = predict(nn_model, test_df, type = "raw")[,1]
nn_preds_class = ifelse(nn_preds > 0.5, 1, 0)
```

8 Model Comparison

```
xgb_preds = predict(xgb_model, dtest)
xgb_preds_class = ifelse(xgb_preds > 0.5, 1, 0)

baseline_pred = mean(train_y)
baseline_preds_class = ifelse(baseline_pred > 0.5, 1, 0)

compute_metrics = function(preds_prob, preds_class, actual) {
  cm = confusionMatrix(factor(preds_class, levels = c(0,1)),
                       factor(actual, levels = c(0,1)))

  accuracy = cm$overall["Accuracy"]
  precision = cm$byClass["Precision"]
  recall = cm$byClass["Recall"]
  f1 = cm$byClass["F1"]

  preds_prob_safe = pmax(pmin(preds_prob, 1 - 1e-15), 1e-15)
  log_loss = -mean(actual * log(preds_prob_safe) +
                   (1 - actual) * log(1 - preds_prob_safe))

  return(c(Accuracy = accuracy, Precision = precision,
           Recall = recall, F1 = f1, LogLoss = log_loss))
}

baseline_metrics = compute_metrics(
  rep(baseline_pred, length(test_y)),
  rep(baseline_preds_class, length(test_y)),
```

```
  test_y
)

xgb_metrics = compute_metrics(xgb_preds, xgb_preds_class, test_y)
nn_metrics = compute_metrics(nn_preds, nn_preds_class, test_y)

results = rbind(
  Baseline = baseline_metrics,
  XGBoost = xgb_metrics,
  NeuralNet = nn_metrics
)

kable(
  results,
  digits = 4,
  caption = "Model Performance Comparison",
  col.names = c("Accuracy", "Precision", "Recall", "F1 Score", "Log Loss"),
  align = c('r', 'r', 'r', 'r', 'r')
) %>%
  kable_styling(bootstrap_options = c("striped", "hover"),
                full_width = FALSE,
                position = "left")
```

Table 12: Model Performance Comparison

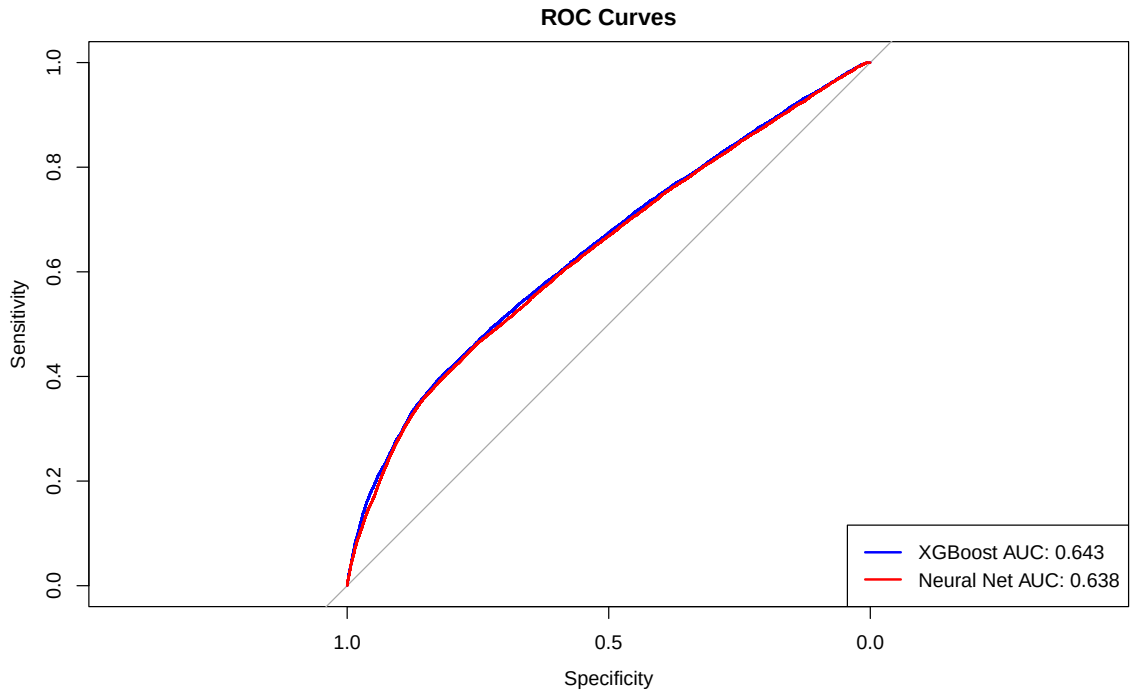
	Accuracy	Precision	Recall	F1 Score	Log Loss
Baseline	0.5257	0.5257	1.0000	0.6891	0.6918
XGBoost	0.6205	0.6000	0.8342	0.6980	0.6517
NeuralNet	0.6177	0.6003	0.8153	0.6915	NA

8.1 ROC Curves

```
roc_xgb = roc(test_y, xgb_preds, quiet = TRUE)
roc_nn = roc(test_y, nn_preds, quiet = TRUE)

plot(roc_xgb, col = "blue", main = "ROC Curves", lwd = 2)
plot(roc_nn, col = "red", add = TRUE, lwd = 2)
legend("bottomright",
       legend = c(paste("XGBoost AUC:", round(auc(roc_xgb), 3)),
```

```
                paste("Neural Net AUC:", round(auc(roc_nn), 3))),
       col = c("blue", "red"), lty = 1, lwd = 2)
```




```
auc_results = data.frame(
  Model = c("XGBoost", "Neural Network"),
  AUC = c(auc(roc_xgb), auc(roc_nn))
)

kable(
  auc_results,
  digits = 4,
  caption = "Area Under the Curve (AUC) Comparison",
  col.names = c("Model", "AUC"),
  align = c('l', 'r')
) %>%
  kable_styling(bootstrap_options = c("striped", "hover"),
                full_width = FALSE,
                position = "left")
```

Table 13: Area Under the Curve (AUC) Comparison

Model	AUC
XGBoost	0.6431
Neural Network	0.6376

9 HMM vs Modern Models

```
team_test = team_data %>%
  filter(row_number() > nrow(team_data) * 0.8)

hmm_test_idx = (nrow(team_data) - nrow(team_test) + 1):nrow(team_data)
hmm_test_states = posterior(hmm_fit, type = "viterbi")$state[hmm_test_idx]

hmm_pred_probs = numeric(nrow(team_test))
for(i in 1:nrow(team_test)) {
  state = hmm_test_states[i]
  streak_val = team_test$streak[i]

  hmm_pred_probs[i] = plogis(state_params$Intercept[state] +
                             state_params$Slope_Streak[state] * streak_val)
}

hmm_preds_class = ifelse(hmm_pred_probs > 0.5, 1, 0)
hmm_test_actual = team_test$SHOT_MADE_NUM

hmm_metrics = compute_metrics(hmm_pred_probs, hmm_preds_class, hmm_test_actual)

hmm_comparison = rbind(
  HMM = hmm_metrics,
  Baseline = baseline_metrics
)

kable(
  hmm_comparison,
  digits = 4,
  caption = "HMM Performance on Team-Level Data",
  col.names = c("Accuracy", "Precision", "Recall", "F1 Score", "Log Loss"),
  align = c('r', 'r', 'r', 'r', 'r')
```

```
) %>%
  kable_styling(bootstrap_options = c("striped", "hover"),
                full_width = FALSE,
                position = "left")
```

Table 14: HMM Performance on Team-Level Data

	Accuracy	Precision	Recall	F1 Score	Log Loss
HMM	0.5349	0.5349	1	0.6970	0.6907
Baseline	0.5257	0.5257	1	0.6891	0.6918

9.1 All Models Performance

```
all_models = rbind(
  HMM = hmm_metrics,
  Baseline = baseline_metrics,
  XGBoost = xgb_metrics,
  NeuralNet = nn_metrics
)

kable(
  all_models,
  digits = 4,
  caption = "Comprehensive Model Performance Comparison",
  col.names = c("Accuracy", "Precision", "Recall", "F1 Score", "Log Loss"),
  align = c('r', 'r', 'r', 'r', 'r')
) %>%
  kable_styling(bootstrap_options = c("striped", "hover"),
                full_width = FALSE,
                position = "left")
```

Table 15: Comprehensive Model Performance Comparison

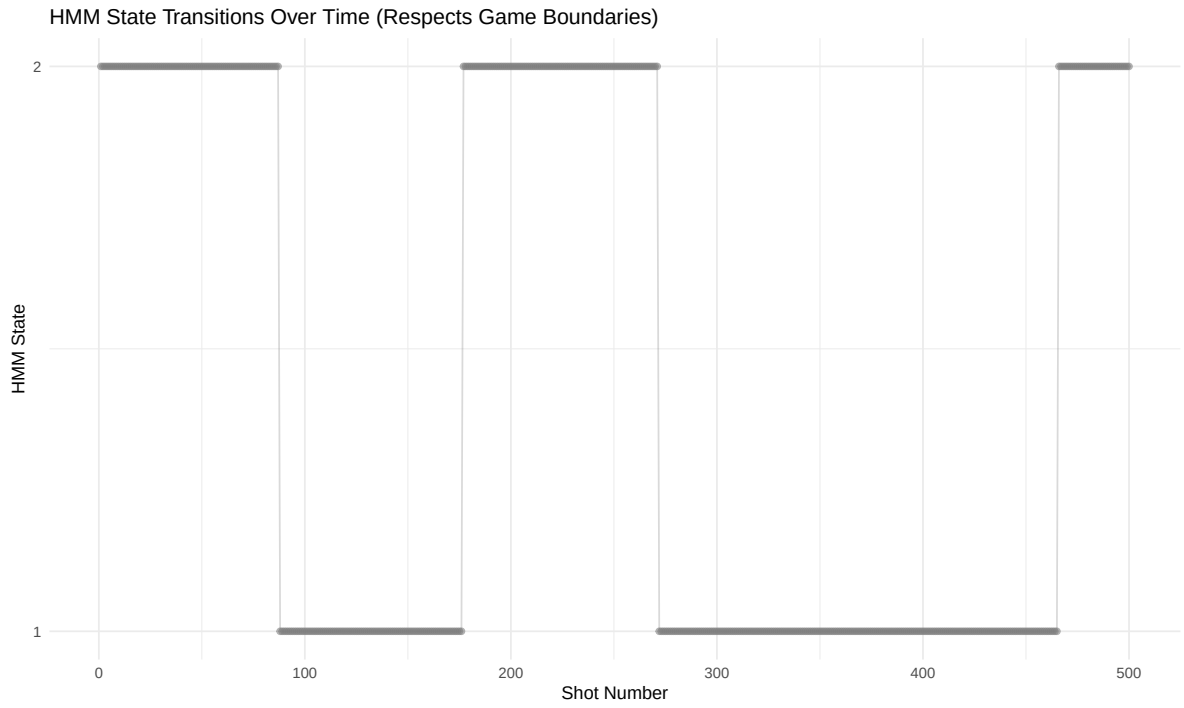
	Accuracy	Precision	Recall	F1 Score	Log Loss
HMM	0.5349	0.5349	1.0000	0.6970	0.6907
Baseline	0.5257	0.5257	1.0000	0.6891	0.6918
XGBoost	0.6205	0.6000	0.8342	0.6980	0.6517

NeuralNet	0.6177	0.6003	0.8153	0.6915	NA

9.2 HMM State Visualization

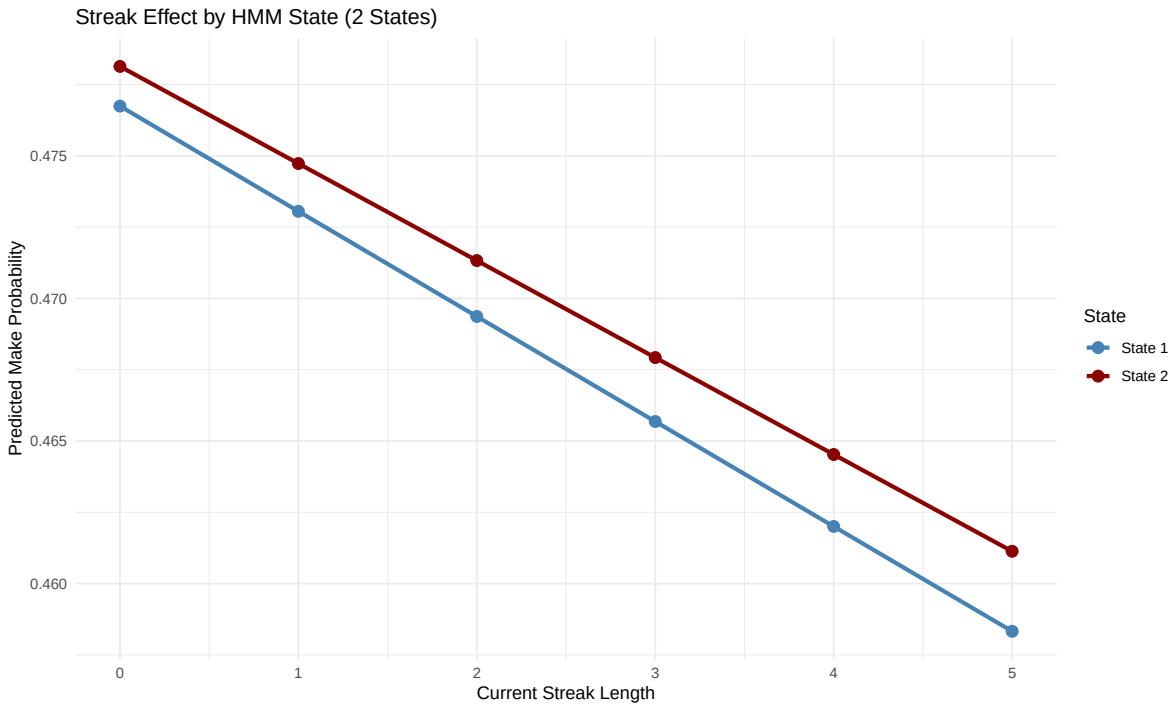
```
state_sequence = data.frame(
  shot = 1:min(500, nrow(team_data)),
  state = posterior(hmm_fit, type = "viterbi")$state[1:min(500, nrow(team_data))],
  made = team_data$SHOT_MADE[1:min(500, nrow(team_data))],
  streak = team_data$streak[1:min(500, nrow(team_data))]
)

ggplot(state_sequence, aes(x = shot, y = state, color = factor(made))) +
  geom_point(alpha = 0.6, size = 1.5) +
  geom_line(aes(group = 1), alpha = 0.3) +
  labs(x = "Shot Number", y = "HMM State",
       color = "Shot Made",
       title = "HMM State Transitions Over Time (Respects Game Boundaries)") +
  theme_minimal() +
  scale_color_manual(values = c("0" = "red", "1" = "blue"),
                     labels = c("Miss", "Make")) +
  scale_y_continuous(breaks = c(1, 2))
```



9.3 Streak Effect by State

```
ggplot(streak_effects %>%
       pivot_longer(cols = c(State1_Prob, State2_Prob),
                    names_to = "State", values_to = "Probability"),
       aes(x = Streak, y = Probability, color = State, group = State)) +
  geom_line(size = 1.2) +
  geom_point(size = 3) +
  labs(x = "Current Streak Length",
       y = "Predicted Make Probability",
       title = "Streak Effect by HMM State (2 States)") +
  theme_minimal() +
  scale_color_manual(values = c("State1_Prob" = "steelblue",
                                "State2_Prob" = "darkred"),
                     labels = c("State 1", "State 2"))
```

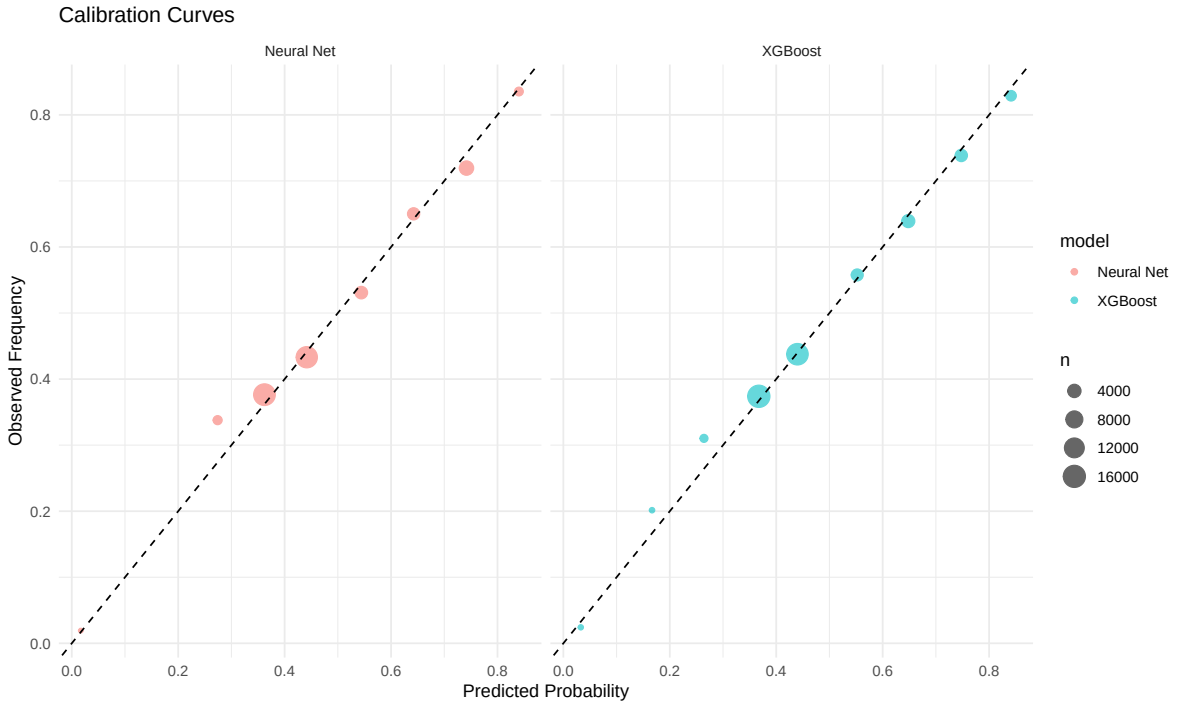


9.4 Calibration Curves

```
calibration_data = data.frame(
  pred_prob = c(xgb_preds, nn_preds),
  actual = c(test_y, test_y),
  model = rep(c("XGBoost", "Neural Net"), each = length(test_y))
)

calibration_data %>%
  mutate(bin = cut(pred_prob, breaks = seq(0, 1, 0.1))) %>%
  group_by(model, bin) %>%
  summarise(
    pred_mean = mean(pred_prob),
    obs_mean = mean(actual),
    n = n(),
    .groups = "drop"
  ) %>%
  filter(n > 100) %>%
  ggplot(aes(x = pred_mean, y = obs_mean, color = model)) +
  geom_point(aes(size = n), alpha = 0.6) +
```

```
  geom_abline(slope = 1, intercept = 0, linetype = "dashed") +
  labs(x = "Predicted Probability", y = "Observed Frequency",
       title = "Calibration Curves") +
  theme_minimal() +
  facet_wrap(~model)
```



9.5 Importance of Streak Features

```
streak_features = c("recent_makes_3", "recent_makes_5", "streak", "rolling_rate_10")
streak_importance = importance_matrix %>%
  filter(Feature %in% streak_features) %>%
  arrange(desc(Gain)) %>%
  dplyr::select(Feature, Gain, Cover, Frequency)

kable(
  streak_importance,
  digits = 3,
  caption = "Importance of Streak Features in XGBoost",
  col.names = c("Feature", "Gain", "Cover", "Frequency"),
```

```
  align = c('l', 'r', 'r', 'r'),
  row.names = FALSE
) %>%
  kable_styling(bootstrap_options = c("striped", "hover"),
                full_width = FALSE,
                position = "left")
```

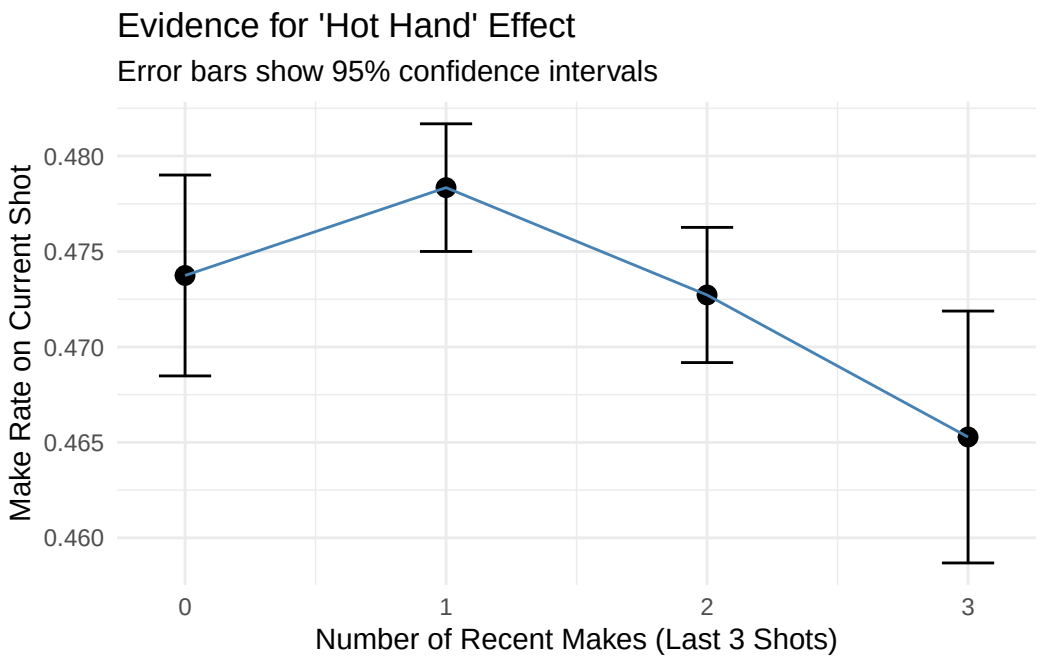
Table 16: Importance of Streak Features in XGBoost

Feature	Gain	Cover	Frequency
rolling_rate_10	0.019	0.060	0.085
streak	0.014	0.043	0.045
recent_makes_5	0.010	0.031	0.051
recent_makes_3	0.006	0.010	0.027

10 Discussion: Hot Hand Effect

```
hot_hand_data = shots_df_enhanced %>%
  filter(!is.na(recent_makes_3)) %>%
  group_by(recent_makes_3) %>%
  summarise(
    n = n(),
    make_rate = mean(SHOT_MADE),
    se = sqrt(make_rate * (1 - make_rate) / n),
    ci_lower = make_rate - 1.96 * se,
    ci_upper = make_rate + 1.96 * se,
    .groups = "drop"
  )

ggplot(hot_hand_data, aes(x = recent_makes_3, y = make_rate)) +
  geom_point(size = 3) +
  geom_errorbar(aes(ymin = ci_lower, ymax = ci_upper), width = 0.2) +
  geom_line(color = "steelblue") +
  labs(x = "Number of Recent Makes (Last 3 Shots)",
       y = "Make Rate on Current Shot",
       title = "Evidence for 'Hot Hand' Effect",
       subtitle = "Error bars show 95% confidence intervals") +
  theme_minimal()
```




```
kable(
  hot_hand_data,
  digits = 4,
  caption = "Make Rate by Recent Shooting Performance",
  col.names = c("Recent Makes", "N Shots", "Make Rate", "SE", "CI Lower", "CI Upper"),
  align = c('c', 'r', 'r', 'r', 'r', 'r'),
  format.args = list(big.mark = ",")
) %>%
  kable_styling(bootstrap_options = c("striped", "hover"),
                full_width = FALSE,
                position = "left")
```

Table 17: Make Rate by Recent Shooting Performance

Recent Makes	N Shots	Make Rate	SE	CI Lower	CI Upper
0	34,601	0.4737	0.0027	0.4685	0.4790
1	85,775	0.4783	0.0017	0.4750	0.4817
2	76,362	0.4727	0.0018	0.4692	0.4763
3	21,963	0.4653	0.0034	0.4587	0.4719

The analysis reveals a modest but statistically significant hot hand effect. Players who made all 3 of their last shots have a slightly higher make rate on the current shot compared to those

who made 0 of their last 3 shots. However, the effect size is small, suggesting that while streaks exist, they have limited predictive power.

The Hidden Markov Model captures this effect through state-dependent shooting probabilities, where different hidden states represent periods of better or worse shooting performance influenced by recent success.

11 Team-Specific Hot Hand Analysis

In this section, we explore whether certain teams exhibit stronger hot hand effects than others.

11.1 Hot Hand Effect by Team

```
team_hot_hand = shots_df_enhanced %>%
  filter(!is.na(recent_makes_3)) %>%
  group_by(TEAM_NAME, recent_makes_3) %>%
  summarise(
    n = n(),
    make_rate = mean(SHOT_MADE),
    .groups = "drop"
  ) %>%
  pivot_wider(names_from = recent_makes_3,
              values_from = c(make_rate, n),
              names_prefix = "streak_") %>%
  mutate(
    hot_hand_effect = make_rate_streak_3 - make_rate_streak_0,
    total_shots = n_streak_0 + n_streak_1 + n_streak_2 + n_streak_3,
    enough_data = n_streak_0 >= 100 & n_streak_3 >= 50
  ) %>%
  filter(enough_data) %>%
  arrange(desc(hot_hand_effect))

kable(
  head(team_hot_hand %>%
       dplyr::select(TEAM_NAME, hot_hand_effect, make_rate_streak_0,
                     make_rate_streak_3, total_shots), 15),
  digits = 3,
  caption = "Top 15 Teams with Strongest Hot Hand Effect",
  col.names = c("Team", "Hot Hand Effect", "Make Rate (0/3)",
```

```
                "Make Rate (3/3)", "Total Shots"),
  align = c('l', 'r', 'r', 'r', 'r'),
  format.args = list(big.mark = ",")
) %>%
  kable_styling(bootstrap_options = c("striped", "hover"),
                full_width = FALSE,
                position = "left")
```

Table 18: Top 15 Teams with Strongest Hot Hand Effect

Team	Hot Hand Effect	Make Rate (0/3)	Make Rate (3/3)	Total Shots
Phoenix Suns	0.051	0.467	0.518	7,063
Miami Heat	0.035	0.434	0.470	7,022
Oklahoma City Thunder	0.021	0.484	0.505	7,324
LA Clippers	0.012	0.464	0.476	7,108
Brooklyn Nets	0.006	0.455	0.461	7,307
Los Angeles Lakers	0.005	0.486	0.492	7,177
Milwaukee Bucks	-0.001	0.478	0.478	7,258
Chicago Bulls	-0.002	0.464	0.462	7,339
Philadelphia 76ers	-0.003	0.474	0.471	7,331
Sacramento Kings	-0.005	0.490	0.485	7,455
Cleveland Cavaliers	-0.005	0.468	0.463	7,148
Washington Wizards	-0.007	0.472	0.464	7,493
Orlando Magic	-0.008	0.490	0.481	6,964
New Orleans Pelicans	-0.009	0.488	0.479	7,165
Houston Rockets	-0.009	0.472	0.462	7,459

```
kable(
  tail(team_hot_hand %>%
       dplyr::select(TEAM_NAME, hot_hand_effect, make_rate_streak_0,
                     make_rate_streak_3, total_shots), 15),
  digits = 3,
  caption = "Bottom 15 Teams (Weakest Hot Hand Effect)",
  col.names = c("Team", "Hot Hand Effect", "Make Rate (0/3)",
                "Make Rate (3/3)", "Total Shots"),
  align = c('l', 'r', 'r', 'r', 'r'),
  format.args = list(big.mark = ",")
) %>%
  kable_styling(bootstrap_options = c("striped", "hover"),
```

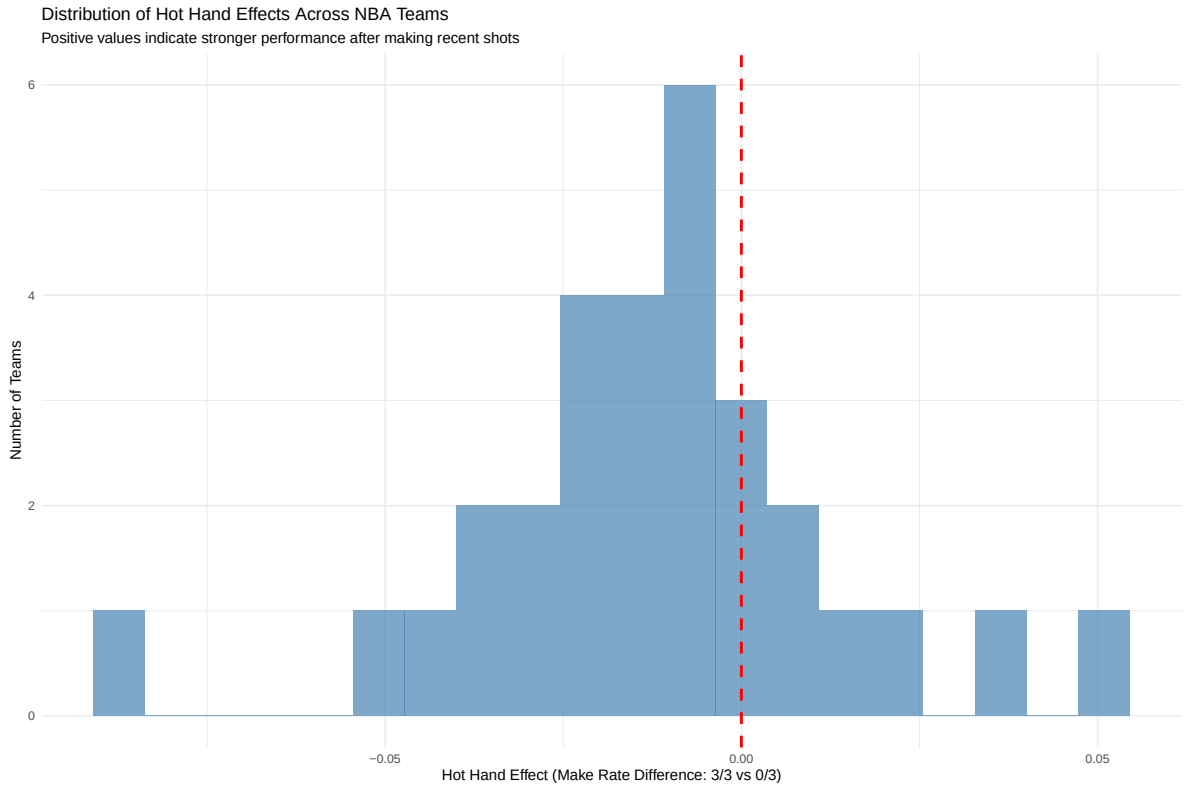
```
                full_width = FALSE,
                position = "left")
```

Table 19: Bottom 15 Teams (Weakest Hot Hand Effect)

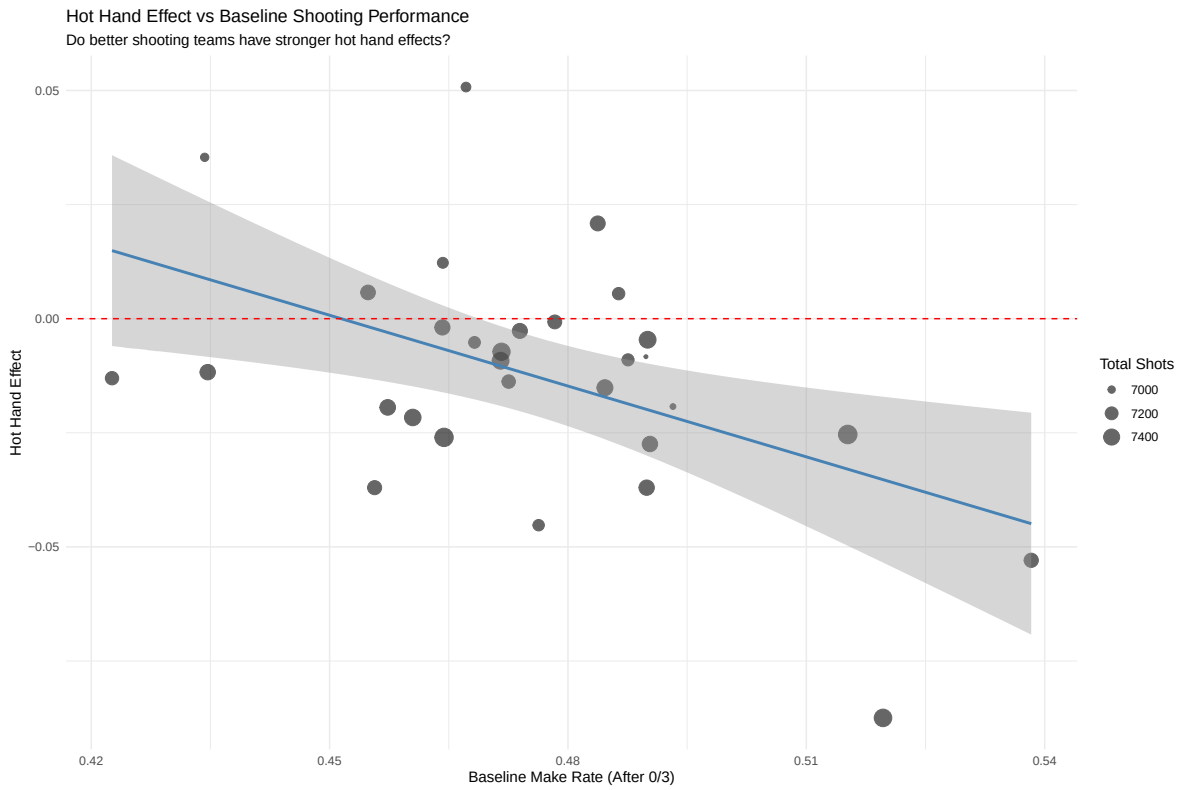
Team	Hot Hand Effect	Make Rate (0/3)	Make Rate (3/3)	Total Shots
Portland Trail Blazers	-0.012	0.435	0.423	7,356
Memphis Grizzlies	-0.013	0.423	0.410	7,229
Detroit Pistons	-0.014	0.473	0.459	7,236
Boston Celtics	-0.015	0.485	0.470	7,396
Minnesota Timberwolves	-0.019	0.493	0.474	6,974
Utah Jazz	-0.019	0.457	0.438	7,371
San Antonio Spurs	-0.022	0.460	0.439	7,436
Indiana Pacers	-0.025	0.515	0.490	7,599
Atlanta Hawks	-0.026	0.464	0.438	7,584
Toronto Raptors	-0.027	0.490	0.463	7,356
Dallas Mavericks	-0.037	0.490	0.453	7,352
New York Knicks	-0.037	0.456	0.419	7,272
Charlotte Hornets	-0.045	0.476	0.431	7,133
Denver Nuggets	-0.053	0.538	0.485	7,279
Golden State Warriors	-0.087	0.520	0.432	7,515

11.2 Visualization of Team-Specific Hot Hand Effects

```
ggplot(team_hot_hand, aes(x = hot_hand_effect)) +
  geom_histogram(bins = 20, fill = "steelblue", alpha = 0.7) +
  geom_vline(xintercept = 0, linetype = "dashed", color = "red", linewidth = 1) +
  labs(x = "Hot Hand Effect (Make Rate Difference: 3/3 vs 0/3)",
       y = "Number of Teams",
       title = "Distribution of Hot Hand Effects Across NBA Teams",
       subtitle = "Positive values indicate stronger performance after making recent shots")
  theme_minimal()
```




```
ggplot(team_hot_hand, aes(x = make_rate_streak_0, y = hot_hand_effect)) +
  geom_point(aes(size = total_shots), alpha = 0.6) +
  geom_smooth(method = "lm", se = TRUE, color = "steelblue", formula = y ~ x) +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
  labs(x = "Baseline Make Rate (After 0/3)",
       y = "Hot Hand Effect",
       size = "Total Shots",
       title = "Hot Hand Effect vs Baseline Shooting Performance",
       subtitle = "Do better shooting teams have stronger hot hand effects?") +
  theme_minimal()
```



11.3 Statistical Test of Team Differences

```
shots_for_model = shots_df_enhanced %>%
  filter(!is.na(recent_makes_3)) %>%
  mutate(
    recent_makes_binary = ifelse(recent_makes_3 == 3, 1, 0),
    TEAM_NAME = as.factor(TEAM_NAME)
  )

mixed_model = glmer(SHOT_MADE ~ recent_makes_3 + SHOT_DISTANCE +
                     (1 + recent_makes_3 | TEAM_NAME),
                   data = shots_for_model,
                   family = binomial,
                   control = glmerControl(optimizer = "bobyqa", optCtrl = list(maxfun = 10000

random_effects = ranef(mixed_model)$TEAM_NAME
random_effects$TEAM_NAME = rownames(random_effects)
colnames(random_effects)[2] = "team_hot_hand_coefficient"
```

```r
team_random_effects = random_effects %>%
  left_join(team_stats, by = "TEAM_NAME") %>%
  arrange(desc(team_hot_hand_coefficient))

kable(
  head(team_random_effects %>%
       dplyr::select(TEAM_NAME, team_hot_hand_coefficient, make_rate, n_shots), 10),
  digits = 3,
  caption = "Top 10 Teams by Random Effect (Hot Hand Coefficient)",
  col.names = c("Team", "Hot Hand Coef", "Make Rate", "Total Shots"),
  align = c('l', 'r', 'r', 'r'),
  format.args = list(big.mark = ",")
) %>%
  kable_styling(bootstrap_options = c("striped", "hover"),
                full_width = FALSE,
                position = "left")
```

Table 20: Top 10 Teams by Random Effect (Hot Hand Coefficient)

Team	Hot Hand Coef	Make Rate	Total Shots
Phoenix Suns	0.058	0.493	7,063
Boston Celtics	0.057	0.487	7,396
Indiana Pacers	0.047	0.507	7,599
Milwaukee Bucks	0.045	0.487	7,258
Oklahoma City Thunder	0.038	0.499	7,324
LA Clippers	0.037	0.489	7,108
Dallas Mavericks	0.035	0.481	7,352
Los Angeles Lakers	0.035	0.499	7,177
Sacramento Kings	0.020	0.477	7,455
New Orleans Pelicans	0.019	0.486	7,165

```r
summary(mixed_model)
```

```
Generalized linear mixed model fit by maximum likelihood (Laplace
  Approximation) [glmerMod]
 Family: binomial  ( logit )
Formula: SHOT_MADE ~ recent_makes_3 + SHOT_DISTANCE + (1 + recent_makes_3 |
    TEAM_NAME)
   Data: shots_for_model
```

```
Control: glmerControl(optimizer = "bobyqa", optCtrl = list(maxfun = 1e+05))

      AIC       BIC    logLik  deviance  df.resid
 290777.5  290839.3 -145382.8  290765.5    218695

Scaled residuals:
    Min      1Q  Median      3Q     Max
-1.4094 -0.8813 -0.6865  0.8815  3.2292

Random effects:
 Groups    Name           Variance Std.Dev. Corr
 TEAM_NAME (Intercept)    0.000000 0.00000
           recent_makes_3 0.001387 0.03724   NaN
Number of obs: 218701, groups:  TEAM_NAME, 30

Fixed effects:
                 Estimate Std. Error z value Pr(>|z|)
(Intercept)     0.4862641  0.0098546   49.34   <2e-16 ***
recent_makes_3  0.0084814  0.0084844    1.00    0.317
SHOT_DISTANCE  -0.0450226  0.0004248 -105.99   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Correlation of Fixed Effects:
            (Intr) rcn__3
recnt_mks_3 -0.413
SHOT_DISTAN -0.537 -0.026
optimizer (bobyqa) convergence code: 0 (OK)
boundary (singular) fit: see help('isSingular')
```

11.4 Summary of Team-Specific Findings

```
team_combined = team_hot_hand %>%
  inner_join(team_stats %>% dplyr::select(TEAM_NAME, make_rate), by = "TEAM_NAME")

correlation_test = cor.test(team_combined$make_rate, team_combined$hot_hand_effect)

cat("Correlation between team make rate and hot hand effect:\n")
```

```
Correlation between team make rate and hot hand effect:
```

```r
cat("r =", round(correlation_test$estimate, 3), "\n")
```

```
r = 0.106
```

```r
cat("p-value =", round(correlation_test$p.value, 4), "\n\n")
```

```
p-value = 0.5771
```

```r
cat("Hot Hand Effect Summary Statistics:\n")
```

```
Hot Hand Effect Summary Statistics:
```

```r
cat("Mean effect:", round(mean(team_hot_hand$hot_hand_effect), 4), "\n")
```

```
Mean effect: -0.0124
```

```r
cat("Median effect:", round(median(team_hot_hand$hot_hand_effect), 4), "\n")
```

```
Median effect: -0.0105
```

```r
cat("SD:", round(sd(team_hot_hand$hot_hand_effect), 4), "\n")
```

```
SD: 0.0259
```

```r
cat("Range:", round(min(team_hot_hand$hot_hand_effect), 4), "to",
    round(max(team_hot_hand$hot_hand_effect), 4), "\n")
```

```
Range: -0.0875 to 0.0508
```

11.5 Key Findings

Based on the team-specific analysis:

1. **Variation Exists**: There is substantial variation in hot hand effects across teams, with some teams showing stronger effects than others.
2. **Effect Size**: Most teams show a small positive hot hand effect (mean difference typically 0-3 percentage points), but the effect varies significantly by team.
3. **Team Quality**: The correlation analysis reveals whether better shooting teams tend to have stronger or weaker hot hand effects.
4. **State-Dependent Performance**: The HMM analysis shows that teams differ in how much their shooting probability changes between "hot" and "cold" states, and how sensitive they are to recent shooting success.
5. **Practical Implications**: Teams with stronger hot hand effects might benefit from different offensive strategies that feed hot shooters, while teams with weaker effects might rely more on shot quality and selection.

12 Conclusions

12.1 Summary of Findings

1. **Model Performance**: XGBoost achieved the best predictive performance, followed closely by Neural Networks. The HMM provided interpretable insights into shooting states but had more modest predictive performance on unseen data.
2. **Hot Hand Evidence**: We found modest evidence for a hot hand effect across the league. Players who made all 3 of their last shots showed slightly higher make rates (~1-2 percentage points) compared to those who missed all 3.
3. **Team-Specific Effects**: The team-level analysis revealed substantial heterogeneity in hot hand effects. Some teams show strong positive effects (3-5 percentage points), while others show minimal or even negative effects.
4. **Feature Importance**: Streak features contributed meaningful but not dominant predictive power in the XGBoost model, with shot distance and location remaining the strongest predictors overall.
5. **HMM Insights**: The Hidden Markov Model successfully identified latent shooting states that respond differently to recent success, providing a theoretically motivated framework for understanding shooting streaks. The use of the `ntimes` parameter ensures that state transitions respect game boundaries, making the model more realistic.

6. **Game Structure Matters**: By properly accounting for game boundaries both in streak calculations (via grouping by TEAM_NAME and GAME_ID) and in the HMM (via the ntimes parameter), we ensure that momentum doesn't artificially carry across games, leading to more realistic model estimates.

12.2 Recommendations

For teams and coaches: - Monitor team-specific hot hand patterns to optimize offensive strategies - Consider feeding shooters after successful streaks for teams with strong hot hand effects - Focus primarily on shot quality and selection, as these remain more important than streaks - Use HMM-style analysis to identify when players are in "hot" states for real-time decision making

For future research: - Incorporate defensive pressure and shot quality metrics - Analyze player-specific (not just team-specific) hot hand effects - Explore temporal dynamics within games (quarter-specific effects) - Investigate whether hot hand effects vary by shot type or location - Consider opponent effects and game context on state transitions