NBA MATCH OUTCOME PREDICTION

Batuhan Koca, Alperen Bayrak

.

1- Introduction

(You can see the codes on the last pages.)

In the ever-evolving world of data-driven decision making, sports analytics has become a powerful tool not only for teams and coaches but also for fans, bettors and businesses. Among all major sports, the NBA stands out with its fast-paced structure and rich statistical ecosystem, which makes it a perfect candidate for predictive modeling.

The goal of this project is to build a machine learning model that predicts the outcome of an NBA game **before it starts**, based solely on **pre-game data**. This means the model only uses statistics and performance indicators that would have been realistically available before the matches played. Unlike models that explain the result after it happens (post-game analysis), our project aim to focus on **forecasting**, which can be used in legal betting, fantasy platforms, and sports analytics tools.

Legal betting and how odds are set

The legal sports betting industry is now worth over 200 billion dollars globally. Behind every moneyline, point spread or over under bet, there are algorithms and statistical models working in the background. These models estimate the likelihood of one team winning based on data that is known before the match.

This project follows a similar approach and recreates the base logic that helps set betting odds. It asks questions like:

- What are the chances that Boston beats Miami tonight based on their last five games
- Which features like rebounds, turnovers or shooting percentages have the most impact on match outcomes

With more development, a model like this could become the starting point for automatic odds creation. This kind of foundation is essential for sportsbooks, fantasy platforms and modern betting startups.

Smarter use for everyday fans

For both casual and experienced fans, this model can become a decision support tool. Instead of relying on emotions or biased opinions, a fan could use real statistical indicators to guide their thinking. This also makes watching games more analytical and fun.

Potential to become a real product

If improved and integrated in the right way, this project could grow into something beyond just an assignment. For example:

- A live prediction app or dashboard
- A tool that helps with fantasy sports decisions
- A data API that can be used by startups working in sports content, betting tools or interactive platforms

.

2- Data Collection

We sourced our data from https://www.basketball-reference.com/, a trusted archive of NBA statistics. The original data is player-level and includes basic and advanced box score tables for both home and away teams, detailing metrics such as field goal attempts, three-point percentages, rebounds, assists, turnovers, and more.

To make the data more usable and consistent for modeling, we use a **team-level version** of this dataset. This version was created by an open-source contributor who converted the player-level data into team-level data by computing:

- **Total values** for some metrics like rebounds and shot attempts
- **Averages** for some metrics like FG% or 3PT%
- **Maximums** for some metrics.

We uploaded this structured team-level dataset to GitHub and use it throughout the project to ensure faster data processing.

By modeling games at the team level, we can more effectively create patterns such as:

- Recent performance trends over the last 5 games
- Key metrics that correlate strongly with match outcomes

Game-level Dataset Structure

Each row in our dataset represents a single NBA match, with the following metrics:

Date, Home_Team, Away_Team, Home_PPG, Away_PPG, Home_Last5_FG%, Away_Last5_FG% Home_Win, and other metrics.

All other columns are features calculated before the game.

We created the metrics to describe both teams' performance in a way that reflects their current strength and momentum:

- Teams' Last 5 Games' PPG
- Teams' Last 5 Games' FG%

- Teams' Last 5 Games' Rebounds
- Teams' Last 5 Games' Turnovers, and so on

This gives the model a sense of temporary form situation, which is also very important.

.

3- Data Preparation, Data Visualization, Planning the Model

In this section, we prepared the dataset for modeling by identifying relevant features, visualizing their correlation with match outcomes, and splitting the data into training and test sets.

Correlation Analysis

To evaluate which variables are most associated with a team winning the game, we first filtered only the numeric columns in the dataset and calculated their correlation with the won variable. Then we selected only those columns that include the suffix _lag, as these represent team form based on the last five games as it is our main modeling input.

We visualized these correlations using a horizontal bar plot, where the fill color indicated the strength and direction of the correlation. This allowed us to quickly identify which features might be most useful for predicting match outcomes. While no single variable had a very strong correlation, some variables showed moderate predictive value in the ±0.08 to ±0.20 range.

Selecting the Right Metrics

From the correlation analysis, we selected the _lag features with correlation ≥ 0.08. These are:

fg_max_opp_lag5: Maximum field goals made by any opponent player in their last 5 games

pts_max_opp_lag5: Highest point total by an opponent player across their last 5 matches

fga_max_opp_lag5: Maximum field goal attempts by an opponent player

ft%_max_lag5: Free throw percentage of the best-performing player on our team

ast%_max_lag5: Assist percentage of our top player

And some other metrics.

.

4- Modelling

Train–Test Split To evaluate model generalizability, we split the dataset into a training set (70%) and a test set (30%) using stratified sampling on the won variable to maintain class balance. This ensures that our classifier sees a fair distribution of both wins and losses in both subsets.

To remove redundant variables, we calculated pairwise correlations among all numeric features in the training set. Features with correlation above 0.70 were considered highly collinear, and we dropped them to avoid overfitting and model instability.

After this filtering step, we retained the final set of input variables (vars_clean) to be used in model training.

We then trained a logistic regression model using the cleaned training dataset (train_last). Logistic regression was selected as our baseline classifier due to its interpretability, efficiency, and robustness in binary classification tasks — especially when dealing with moderately sized feature sets and structured data like this.

After training, we applied the model to the test_last dataset to generate predicted probabilities (prob) for each match. These probabilities were converted into binary predictions using a 0.5 decision threshold.

Then, we applied the same steps to predict another team, Milwaukee Bucks.

We evaluated the models' performance using:

Accuracy: Measures the proportion of correctly classified games.

AUC (Area Under the ROC Curve): Captures the model's ability to rank positive cases higher than negative ones across all thresholds.

.

5- Result

After training our model, we tested it on the test dataset and got the following results:

Accuracies: 61.7%, 62.2%

AUCs (Areas Under Curve): 0.612, 0.619

This shows that our model is doing better than random guessing and has learned some useful patterns. It’s especially meaningful because we only used pre-game statistics, which would be available before the game actually started.

The model:

- Looked at each team’s last 5 games,
- Calculated average stats from those games,
- Used logistic regression to predict whether the team would win.

Even though the model is simple, it gives us a good starting point.

.

6- Conclusion

The main goal of this project was to predict the winner of an NBA game before it starts, based only on the recent form of the teams. And we were able to build a basic version of that idea.

What we learned:

- A team's performance in the last 5 games is a useful indicator of whether they will win the next one.
- Even a basic model like logistic regression can achieve over 60% accuracy.
- By using team-level data instead of player-level, we kept the model simple and easier to manage.

Potential Improvements:

- **Add more features (like home/away status, player injuries, travel fatigue, etc.)**,
- **Try more advanced models (like Random Forest or XGBoost)**,
- **Analyze data by season or playoff stage**.

But overall, this project proves that it's possible to make decent predictions using only pre-game data.

.

7- Business Use Case

With some developments, an improved version of our model can be used in several fields:

Sports Betting Platforms

Betting companies set odds based on predictions. A model like ours could:

- Help generate pre-game odds more accurately,
- Detect games where the odds may be off,
- Support betting tools and systems with statistical logic.

Fantasy Sports Apps

Fantasy NBA players need to make decisions before games start. This model could:

- Help users understand which teams are likely to win,
- Suggest which teams might perform better,
- Support users with data-driven decisions.

Sports Media & Content Creators

Media platforms or sports YouTubers could use this kind of prediction tool to:

- Add pre-game prediction visuals to content,
- Show recent form stats for both teams,
- Create interactive dashboards for fans.

NBA Coaches & Analysts

Even though real NBA teams have access to much deeper data, a simplified version like this could:

- Be used for high-level pre-game scouting,
- Quickly summarize opponent performance trends,
- Support junior analysts or interns with basic tools.

.

8- Codes

```
library(dplyr)
library(tidyr)
library(purrr)
library(caret)
library(MASS)
library(car)
library(ggplot2)
library(corrplot)
library(pROC)
library(zoo)


,Attaching package: ‘dplyr’
,
,
,The following objects are masked from ‘package:stats’:
,
,    filter, lag
,
,
,The following objects are masked from ‘package:base’:
,
,    intersect, setdiff, setequal, union
,
,
,Loading required package: ggplot2
```

```
,    cov, smooth, var
,
,
,
,Attaching package: ‘zoo’
,
,
,The following objects are masked from ‘package:base’:
,
,    as.Date, as.Date.numeric
,
,
```

```r
data_initial <-
readr::read_csv("https://raw.githubusercontent.com/alpaalp/csv_proje/
refs/heads/main/nba_games%20(1).csv")
```

```
New names:
,• `` -> `...1`
,• `mp` -> `mp...2`
,• `mp` -> `mp...3`
,• `mp_max` -> `mp_max...37`
,• `mp_max` -> `mp_max...38`
,• `mp_opp` -> `mp_opp...76`
,• `mp_opp` -> `mp_opp...77`
,• `mp_max_opp` -> `mp_max_opp...111`
,• `mp_max_opp` -> `mp_max_opp...112`
,Rows: 17772 Columns: 151
,── Column specification
────────────────────────────────────────────────────────
,Delimiter: ","
,chr    (2): team, team_opp
,dbl  (141): ...1, mp...2, mp...3, fg, fga, fg%, 3p, 3pa, 3p%, ft,
fta, ft%, ...
,lgl    (7): +/-, mp_max...37, mp_max...38, +/-_opp, mp_max_opp...111,
mp_max...
,date   (1): date
,
,ℹ Use `spec()` to retrieve the full column specification for this
data.
,ℹ Specify the column types or set `show_col_types = FALSE` to quiet
this message.
```

```r
head(data_initial)
```

```
  ...1 mp...2 mp...3 fg fga fg%   3p 3pa 3p%   ft … tov%_max_opp usg
%_max_opp
1 0    240    240    39  81 0.481  6 20  0.300 14 … 22.8         29.0

2 1    240    240    36 100 0.360  7 31  0.226 16 … 50.0         32.6
```

```
3 2     240     240     37   85 0.435   8 19   0.421 17 ⋯ 20.0          30.9
4 3     240     240     41   89 0.461   8 21   0.381 17 ⋯ 28.6          30.9
5 4     240     240     27   86 0.314   6 26   0.231 15 ⋯ 16.8          30.9
6 5     240     240     34   99 0.343 11 39   0.282 13 ⋯ 33.9          27.2
```

```
  ortg_max_opp drtg_max_opp team_opp total_opp home_opp season date
won
1 178          111          DAL        95      1        2016   2015-
12-09  TRUE
2 152          111          ATL        98      0        2016   2015-
12-09 FALSE
3 148          116          SAS       107      1        2018   2017-
10-18 FALSE
4 138          118          MIN        99      0        2018   2017-
10-18  TRUE
5 157           90          MEM        92      1        2021   2021-
04-30 FALSE
6 106          106          ORL        75      0        2021   2021-
04-30  TRUE
```

```
str(data_initial[1:5])
```

```
tibble [17,772 × 5] (S3: tbl_df/tbl/data.frame)
, $ ...1  : num [1:17772] 0 1 2 3 4 5 6 7 8 9 ...
, $ mp...2: num [1:17772] 240 240 240 240 240 240 240 240 240 240 ...
, $ mp...3: num [1:17772] 240 240 240 240 240 240 240 240 240 240 ...
, $ fg    : num [1:17772] 39 36 37 41 27 34 35 42 44 39 ...
, $ fga   : num [1:17772] 81 100 85 89 86 99 83 86 87 77 ...
```

```
data_initial <- data_initial %>% arrange(date) # arranged by date from
earliest to latest

data_initial %>% distinct(team) %>% head() # to see distinct teams
```

```
  team
1 DET
2 ATL
3 NOP
4 GSW
5 CLE
6 CHI
```

```
team_distinct <- data_initial %>% distinct(team) %>% unlist %>%
unname

team_distinct
```

```
 [1] "DET" "ATL" "NOP" "GSW" "CLE" "CHI" "BRK" "IND" "TOR" "MEM" "PHI"
"BOS"
[13] "DEN" "HOU" "SAS" "OKC" "POR" "LAC" "SAC" "MIN" "LAL" "DAL" "PHO"
"WAS"
[25] "ORL" "UTA" "CHO" "MIA" "NYK" "MIL"
```

```
df_vector <- vector("list", length(team_distinct))
names(df_vector) <- team_distinct     # created a list to later on use
in a loop

for (t in team_distinct) {
  obj_name <- paste0("df_", t)
  assign(
    obj_name,
    dplyr::filter(data_initial, team == t),
    envir = .GlobalEnv
  )
}   # every team is now stored in ^df_{teamcode}^ dataframes

team_vec <-
c("DET","ATL","NOP","GSW","CLE","CHI","BRK","IND","TOR","MEM",

"PHI","BOS","DEN","HOU","SAS","OKC","POR","LAC","SAC","MIN",

"LAL","DAL","PHO","WAS","ORL","UTA","CHO","MIA","NYK","MIL")

lagify_df <- function(df) {
  df %>%
    arrange(date) %>%
    mutate(across(
      where(is.numeric),
      ~ rollapplyr(lag(.x), 5, mean, fill = NA, align = "right"),
      .names = "{.col}_lag5"
    ))
}

for (t in team_vec) {
  nm <- paste0("df_", t)
  assign(nm, lagify_df(get(nm)), envir = .GlobalEnv)
}
# Respective to their match date, added variebles' average value of
itselfs, input is: last 5 games before that game, output is average of
all 5 game

df_bos <- get("df_BOS")
lag_cols <- grep("_lag5$", names(df_bos), value = TRUE)

df_bos_full <- df_bos %>%
  mutate(
    opp_stats = pmap(
```

```r
      list(opp   = team_opp,
           gdate = date),
      function(opp, gdate) {
        opp_df <- get(paste0("df_", opp))
        opp_row <- opp_df %>% filter(date == gdate)
        if (nrow(opp_row) == 0) {
          as_tibble(setNames(as.list(rep(NA, length(lag_cols))),
                             paste0(lag_cols, "_opp")))
        } else {
          opp_row %>%
            dplyr::select(all_of(lag_cols)) %>%
            rename_with(~ paste0(.x, "_opp"))
        }
      }
    )
  ) %>%
  unnest(opp_stats)
## this part was the hardest, since every match, in this case BOS team
is only team in the "team" column, has many different opponents,
stored in the team_opp column.
## this function tooks the team_opp gets the respective teams df in
paste0(df_, opp) part, and calculates its lag5 average, exact match
played on that exact date also stored in every df.
## 2015-11-10 match played with MIL and BOS, is both on df_MIL and
df_BOS dataframe. difference is df_MIL takes team==MIL which includes
every MIL matches no matter its home or away team.Which i can
calculate opp_team_lag5 in df_BOS

var_cols <- c(
  "fg%_lag5",   "3p%_lag5",  "ft%_lag5",
  "orb_lag5",   "drb_lag5",  "ts%_lag5",
  "fg%_max_lag5","3p%_max_lag5","ft%_max_lag5",
  "ts%_max_lag5","orb%_max_lag5","drb%_max_lag5",
  "fg%_opp_lag5",
  "3p%_opp_lag5",
  "ft%_opp_lag5",
  "orb_opp_lag5",
  "drb_opp_lag5",
  "ts%_opp_lag5",
  "fg%_max_opp_lag5",
  "3p%_max_opp_lag5",
  "ft%_max_opp_lag5",
  "ts%_max_opp_lag5",
  "orb%_max_opp_lag5",
  "drb%_max_opp_lag5",
  "fta_max_opp_lag5", "won", "team_opp"
)

df_boston_latest <- df_bos %>% dplyr::select(all_of(var_cols))  # i
choose all of the lag5 columns that i created
```

```r
df_boston_latest %>% head
```

```
  fg%_lag5 3p%_lag5 ft%_lag5 orb_lag5 drb_lag5 ts%_lag5 fg%_max_lag5
1     NA       NA       NA      NA       NA          NA     NA
2     NA       NA       NA      NA       NA          NA     NA
3     NA       NA       NA      NA       NA          NA     NA
4     NA       NA       NA      NA       NA          NA     NA
5     NA       NA       NA      NA       NA          NA     NA
6 0.4136   0.3158   0.826    12.6     32.6      0.5152   0.9
  3p%_max_lag5 ft%_max_lag5 ts%_max_lag5 ⋯ ts%_opp_lag5 fg
%_max_opp_lag5
1     NA       NA                NA       ⋯     NA             NA

2     NA       NA                NA       ⋯     NA             NA

3     NA       NA                NA       ⋯     NA             NA

4     NA       NA                NA       ⋯     NA             NA

5     NA       NA                NA       ⋯     NA             NA

6 0.6642        1           0.9838       ⋯ 0.5152       0.7284

  3p%_max_opp_lag5 ft%_max_opp_lag5 ts%_max_opp_lag5 orb%_max_opp_lag5
1   NA               NA                    NA               NA
2   NA               NA                    NA               NA
3   NA               NA                    NA               NA
4   NA               NA                    NA               NA
5   NA               NA                    NA               NA
6 0.83                1               0.9048            22.84
  drb%_max_opp_lag5 fta_max_opp_lag5 won   team_opp
1    NA                NA                TRUE PHI
2    NA                NA               FALSE TOR
3    NA                NA               FALSE SAS
4    NA                NA               FALSE IND
5    NA                NA                TRUE WAS
6 42.28               8.6                TRUE MIL
```

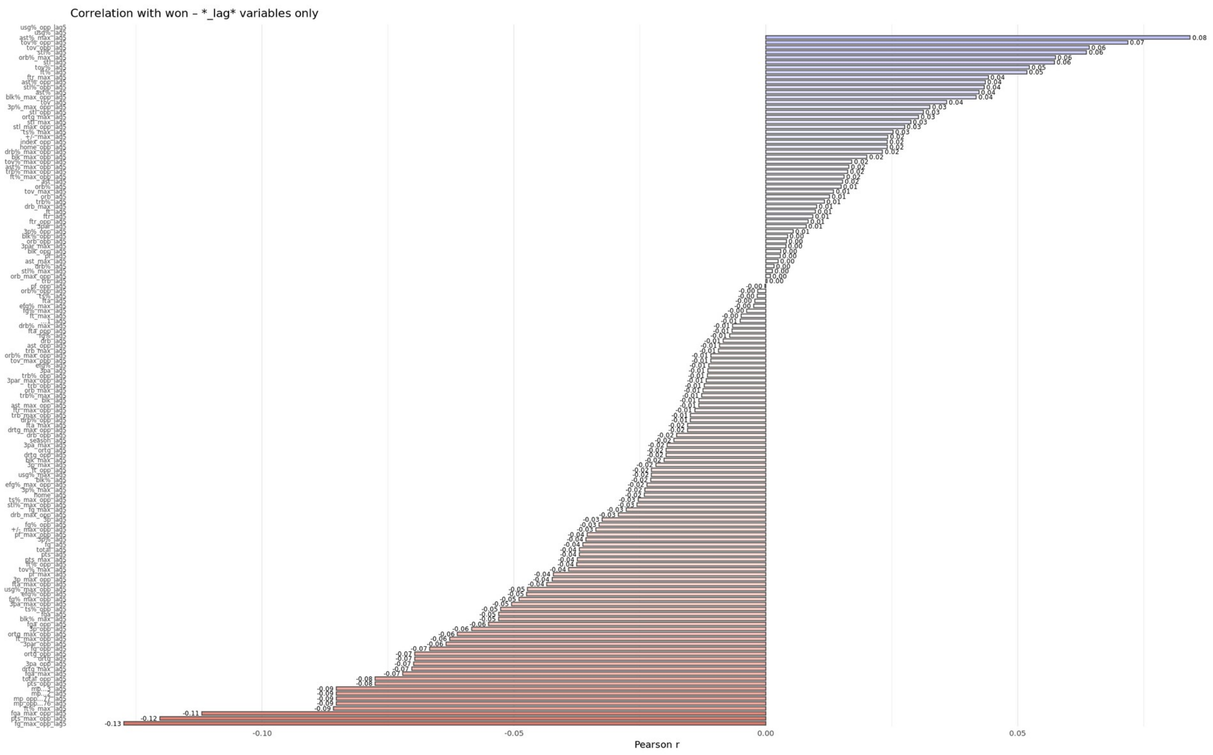
```r
df_boston_latest <- df_boston_latest %>%
  mutate(won = as.numeric(won)) %>%
  dplyr::select(where(is.numeric))   # taking only numeric columns

df_boston_latest[1:10,]
```

```
   fg%_lag5 3p%_lag5 ft%_lag5 orb_lag5 drb_lag5 ts%_lag5 fg%_max_lag5
1      NA       NA       NA      NA       NA          NA      NA
2      NA       NA       NA      NA       NA          NA      NA
3      NA       NA       NA      NA       NA          NA      NA
4      NA       NA       NA      NA       NA          NA      NA
5      NA       NA       NA      NA       NA          NA      NA
6   0.4136   0.3158   0.8260   12.6     32.6      0.5152   0.90
```

```
7  0.4102   0.2958   0.8018   13.4     32.6     0.5046   0.95
8  0.4182   0.2754   0.7896   13.6     32.0     0.5046   0.91
9  0.4284   0.3006   0.8158   13.8     32.8     0.5188   0.91
10 0.4392   0.3048   0.7822   13.2     33.0     0.5232   0.91
   3p%_max_lag5 ft%_max_lag5 ts%_max_lag5 … drb_opp_lag5 ts%_opp_lag5
1      NA       NA                NA      …   NA              NA
2      NA       NA                NA      …   NA              NA
3      NA       NA                NA      …   NA              NA
4      NA       NA                NA      …   NA              NA
5      NA       NA                NA      …   NA              NA
6  0.6642        1            0.9838      … 36.8          0.5152
7  0.7642        1            1.0026      … 36.8          0.5130
8  0.6308        1            0.9684      … 34.2          0.5072
9  0.7308        1            0.9734      … 32.0          0.5158
10 0.8166        1            0.9734      … 30.4          0.5140
   fg%_max_opp_lag5 3p%_max_opp_lag5 ft%_max_opp_lag5 ts%_max_opp_lag5
1      NA              NA               NA                      NA
2      NA              NA               NA                      NA
3      NA              NA               NA                      NA
4      NA              NA               NA                      NA
5      NA              NA               NA                      NA
6  0.7284            0.83                1                  0.9048
7  0.7368            0.95                1                  0.8988
8  0.7234            0.85                1                  0.8858
9  0.7234            0.77                1                  0.8026
10 0.6934            0.67                1                  0.8118
   orb%_max_opp_lag5 drb%_max_opp_lag5 fta_max_opp_lag5 won
1     NA                NA                NA               1
2     NA                NA                NA               0
3     NA                NA                NA               0
4     NA                NA                NA               0
5     NA                NA                NA               1
6  22.84             42.28               8.6               1
7  20.04             42.38               8.2               0
8  20.18             28.38               7.0               1
9  16.34             27.90               6.4               1
10 15.32             28.66               8.2               1

bos_num <- df_bos %>%
  mutate(won = as.numeric(won)) %>%
  dplyr::select(where(is.numeric))

corr_vec <- sapply(
  bos_num %>% dplyr::select(-won),
  function(x) cor(x, bos_num$won, use = "pairwise.complete.obs")
)
corr_df <- tibble(
  variable = names(corr_vec),
  corr     = as.numeric(corr_vec)
) %>%
```

```r
  filter(grepl("_lag", variable)) %>%  ## only selecting _lag columns
  arrange(desc(abs(corr)))

options(repr.plot.width = 20, repr.plot.height = 12)

plot_corr <- ggplot(corr_df, aes(x = reorder(variable, corr), y =
corr, fill = corr)) +
  geom_col(width = 0.7, colour = "grey30") +
  coord_flip() +
  geom_text(aes(label = sprintf("%.2f", corr),
                hjust = ifelse(corr > 0, -0.15, 1.15)),
            size = 2.5) +
  scale_fill_gradient2(
    low   = "firebrick",
    mid   = "white",
    high = "royalblue",
    midpoint = 0,
    limits   = c(-0.20, 0.20),
    oob      = scales::squish
  ) +
  labs(title = "Correlation with won – *_lag* variables only",
       x = NULL, y = "Pearson r") +
  theme_minimal(base_size = 11) +
  theme(
    legend.position    = "none",
    panel.grid.major.y = element_blank(),
    axis.text.y        = element_text(size = 7)
  )
plot_corr
```

```
Warning message in cor(x, bos_num$won, use = "pairwise.complete.obs"):
,“the standard deviation is zero”
,Warning message in cor(x, bos_num$won, use =
"pairwise.complete.obs"):
,“the standard deviation is zero”
,Warning message in cor(x, bos_num$won, use =
"pairwise.complete.obs"):
,“the standard deviation is zero”
,Warning message in cor(x, bos_num$won, use =
"pairwise.complete.obs"):
,“the standard deviation is zero”
,Warning message:
,“Removed 2 rows containing missing values or values outside the scale
range
,(`geom_col()`).”
,Warning message:
,“Removed 2 rows containing missing values or values outside the scale
range
,(`geom_text()`).”
```



```
corr_df %>% filter(abs(corr) >= 0.08) %>% dplyr::select(variable) %>%
unlist %>% unname

[1] "fg_max_opp_lag5"  "pts_max_opp_lag5" "fga_max_opp_lag5" "ft
%_max_lag5"
[5] "mp...2_lag5"      "mp...3_lag5"      "mp_opp...76_lag5"
"mp_opp...77_lag5"
[9] "ast%_max_lag5"

vars_keep <- c(
  "fg_max_opp_lag5","pts_max_opp_lag5","fga_max_opp_lag5",
  "ft%_max_lag5","mp...2_lag5","mp...3_lag5",
  "mp_opp...76_lag5","mp_opp...77_lag5","ast%_max_lag5"
)

data_mod <- df_bos %>%
  mutate(won = factor(ifelse(won, 1, 0))) %>%
  dplyr::select(won, all_of(vars_keep))

set.seed(11)
idx          <- createDataPartition(data_mod$won, p = 0.7, list =
FALSE)
train_data   <- data_mod[idx, ]
test_data    <- data_mod[-idx, ]

glm_fit <- glm(won ~ ., data = train_data, family = binomial)

prob <- predict(glm_fit, newdata = test_data, type = "response")
```

```r
pred <- factor(ifelse(prob > 0.5, "1", "0"), levels = c("0","1"))

cm   <- confusionMatrix(pred, test_data$won, positive = "1")
roc_ <- roc(as.numeric(test_data$won), prob)
auc_ <- auc(roc_)

print(cm$table)
cat("\nAccuracy :", round(cm$overall["Accuracy"], 3),
    "\nAUC      :", round(auc_, 3), "\n")
```

```
Warning message in predict.lm(object, newdata, se.fit, scale = 1, type
= if (type == :
,“prediction from a rank-deficient fit may be misleading”
,Setting levels: control = 1, case = 2
,
,Setting direction: controls < cases
,
```

```
          Reference
,Prediction   0   1
,         0  11  10
,         1  65 107
,
,Accuracy : 0.611
,AUC      : 0.583
```

```r
# To avoid overfitting, we cut off variables which correlate more than
0.7.

X <- train_data %>% dplyr::select(-won)
high_corr <- findCorrelation(cor(X, use = "pairwise.complete.obs"),
cutoff = 0.70)

vars_clean <- colnames(X)[-high_corr]
vars_clean
```

```
[1] "pts_max_opp_lag5" "ft%_max_lag5"     "mp_opp...77_lag5" "ast
%_max_lag5"
```

```r
train_last <- train_data %>% dplyr::select(won, all_of(vars_clean))
test_last  <- test_data %>% dplyr::select(won, all_of(vars_clean))


glm_fit <- glm(won ~ ., data = train_last, family = binomial)


prob <- predict(glm_fit, newdata = test_last, type = "response")
pred <- factor(ifelse(prob > 0.5, "1", "0"), levels = c("0", "1"))

cm   <- confusionMatrix(pred, test_last$won, positive = "1")
roc_ <- roc(as.numeric(test_last$won), prob)
```

```r
auc_ <- auc(roc_)

print(cm$table)
cat("\nAccuracy :", round(cm$overall["Accuracy"], 3),
    "\nAUC      :", round(auc_, 3), "\n")
```

```
Setting levels: control = 1, case = 2
,
,Setting direction: controls < cases
,
```

```
          Reference
,Prediction   0   1
,         0   7   5
,         1  69 112
,
,Accuracy : 0.617
,AUC      : 0.612
```

```r
# Now, predicting the mathes of Milwaukee Bucks, with the same steps
as we did for Boston Celtics.

df_mil <- get("df_MIL")
lag_cols_mil <- grep("_lag5$", names(df_mil), value = TRUE)

df_mil_full <- df_mil %>%
  mutate(
    opp_stats = pmap(
      list(opp = team_opp, gdate = date),
      function(opp, gdate) {
        opp_df <- get(paste0("df_", opp))
        opp_row <- opp_df %>% filter(date == gdate)
        if (nrow(opp_row) == 0) {
          as_tibble(setNames(as.list(rep(NA, length(lag_cols_mil))),
paste0(lag_cols_mil, "_opp")))
        } else {
          opp_row %>%
            dplyr::select(all_of(lag_cols_mil)) %>%
            rename_with(~ paste0(.x, "_opp"))
        }
      }
    )
  ) %>%
  unnest(opp_stats)


df_mil_latest <- df_mil_full %>% dplyr::select(all_of(var_cols))
df_mil_latest <- df_mil_latest %>%
  mutate(won = as.numeric(won)) %>%
  dplyr::select(where(is.numeric))
```

```
mil_num <- df_mil %>%
  mutate(won = as.numeric(won)) %>%
  dplyr::select(where(is.numeric))

corr_vec_mil <- sapply(
  mil_num %>% dplyr::select(-won),
  function(x) cor(x, mil_num$won, use = "pairwise.complete.obs")
)

corr_df_mil <- tibble(
  variable = names(corr_vec_mil),
  corr     = as.numeric(corr_vec_mil)
) %>%
  filter(grepl("_lag", variable)) %>%
  arrange(desc(abs(corr)))


vars_keep_mil <- c(
  "fg_max_opp_lag5","pts_max_opp_lag5","fga_max_opp_lag5",
  "ft%_max_lag5","mp...2_lag5","mp...3_lag5",
  "mp_opp...76_lag5","mp_opp...77_lag5","ast%_max_lag5"
)

data_mod_mil <- df_mil %>%
  mutate(won = factor(ifelse(won, 1, 0))) %>%
  dplyr::select(won, all_of(vars_keep_mil))


set.seed(11)
idx_mil        <- createDataPartition(data_mod_mil$won, p = 0.7, list
= FALSE)
train_data_mil <- data_mod_mil[idx_mil, ]
test_data_mil  <- data_mod_mil[-idx_mil, ]


X_mil          <- train_data_mil %>% dplyr::select(-won)
high_corr_mil <- findCorrelation(cor(X_mil, use =
"pairwise.complete.obs"), cutoff = 0.70)
vars_clean_mil <- colnames(X_mil)[-high_corr_mil]


train_last_mil <- train_data_mil %>% dplyr::select(won,
all_of(vars_clean_mil))
test_last_mil  <- test_data_mil %>% dplyr::select(won,
all_of(vars_clean_mil))


glm_fit_mil <- glm(won ~ ., data = train_last_mil, family = binomial)

prob_mil <- predict(glm_fit_mil, newdata = test_last_mil, type =
"response")
pred_mil <- factor(ifelse(prob_mil > 0.5, "1", "0"), levels = c("0",
"1"))

cm_mil   <- confusionMatrix(pred_mil, test_last_mil$won, positive =
"1")
roc_mil  <- roc(as.numeric(test_last_mil$won), prob_mil)
auc_mil  <- auc(roc_mil)

print(cm_mil$table)
cat("\nAccuracy :", round(cm_mil$overall["Accuracy"], 3),
    "\nAUC      :", round(auc_mil, 3), "\n")

Warning message in cor(x, mil_num$won, use = "pairwise.complete.obs"):
,“the standard deviation is zero”
,Warning message in cor(x, mil_num$won, use =
"pairwise.complete.obs"):
,“the standard deviation is zero”
,Warning message in cor(x, mil_num$won, use =
"pairwise.complete.obs"):
,“the standard deviation is zero”
,Warning message in cor(x, mil_num$won, use =
"pairwise.complete.obs"):
,“the standard deviation is zero”
,Setting levels: control = 1, case = 2
,
,Setting direction: controls < cases
,

          Reference
,Prediction   0   1
,         0  11   8
,         1  62 104
,
,Accuracy : 0.622
,AUC      : 0.619
```