College Basketball Post Season Prediction

Maggie Truong

Introduction

College Basketball Post Season

The college basketball post season, also known as March Madness, is a tournament in which Division 1 schools from the National Collegiate Athletic Association (NCAA) compete. The tournament starts with 68 teams, with the tournament rounds being; First 4, Round of 64, Round of 32, Sweet 16, Elite 8, Final 4, then the Championship.

Why this Model?

The goal of this project is to create a classification model that takes in the results of the Three-Point Shooting Percentage, Free Throw Rate, Two-Point Shooting Percentage, Turnover Percentage Allowed, Offensive Rebound Rate and Effective Field Goal Percentage Shot to try to predict if the team makes the post season and which ranking they get. These 6 will be the predictors while the POSTSEASON will be the response variable.

Predictors Definition

1. Three-Point Shooting Percentage (3P_O)
   - The percent of shooting Three Pointers
2. Free Throw Rate (FTR)
   - How often the team shoots Free Throws
3. Two-Point Shooting Percentage (2P_O)
   - The percent of shooting Two Pointers
4. Turnover Percentage Allowed (TOR)
   - The Turnover Rate
5. Offensive Rebound Rate (ORB)
   - The rate at which the Offense Rebounds
6. Effective Field Goal Percentage Shot (EFG_O)
   - The percent that the team makes field goal shots

Response Definitions

Post Season Ranking(POSTSEASON) The Output is what the team ranks in the Post Season

- Does Not Make Post Season
  - The Team did not make post season at all
- First Four (R68)
  - The team made it to the round of the First Four
- Round of 64 (R64)
  - The team made it to the round of 64
- Round of 32 (R32)
  - The team made it to the round of 32
- Sweet Sixteen (S16)

  - The team made to to the Sweet Sixteen round
- Elite Eight (E8)
  - The team made it to the Elite Eight round
- Final Four (F4)
  - The team made it to the Final Four
- Runner-Up(2ND)
  - The team was the runner up and got Second Place
- Champion
  - The team won the tournament and is the Champion

Origin of the Data

The dataset was found on Kaggle with the author being listed as Andrew Sundberg. The data gets updated every year and the version being used is from November 2024 as the 2025 post season is still not completed. The title on Kaggle is College Basketball Dataset and contains the data from the 2013-2025 season. cbb.csv is the specific data set used, which combines each individual dataset from each season. It has 24 variables with 3523 observations.

Exploratory Data Analysis (EDA)

Exploratory Data Analysis, also know as EDA, is the most important step to do before modeling the data. We will load our data, look at it, search for any issues, such as missing values, and then clean up the data. After this is complete and the data is in better shape, we will use visualization through plots and tables. Through the visualization we will able to gain insight into the data and this will help in our analysis. EDA is extremely important for machine learning as we are able to make sure the data is ideal for analysis, In addition, we can familiarize ourselves with the data and the relations between the variables.

We are using supervised learning algorithms for classification, including logistic regression, K-Nearest Neighbors (KNN), ...

Loading and Exploring Dataset

Here load the readr package from r to use the function read_csv() which will allow us to read in our dataset, which contains playoff college basketball data from 2013-2024

```
library(readr)
cbb <- read_csv("cbb.csv")
```

```
## Rows: 3523 Columns: 24
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (4): TEAM, CONF, POSTSEASON, SEED
## dbl (20): G, W, ADJOE, ADJDE, BARTHAG, EFG_O, EFG_D, TOR, TORD, ORB, DRB, FT...
##
## 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.
```

Now we will take a look at the first five rows with the head() function

```
kable(head(cbb))
```

TEAM	CONF	G	W	ADJOE	ADJDE	BARTHAG	EFG_O	EFG_D	TOR	TORD	ORB	DRB	FTR	FTRD	2P_O	2P_D	3P_O	3P_D	ADJ_T	WAB	POSTSEASON	SEED	YEAR
North Carolina	ACC	40	33	123.3	94.9	0.9531	52.6	48.1	15.4	18.2	40.7	30.0	32.3	30.4	53.9	44.6	32.7	36.2	71.7	8.6	2ND	1	2016

TEAM	CONF	G	W	ADJOE	ADJDE	BARTHAG	EFG_O	EFG_D	TOR	TORD	ORB	DRB	FTR	FTRD	2P_O	2P_D	3P_O	3P_D	ADJ_T	WAB	POSTSEASON	SEED	YEAR
Wisconsin	B10	40	36	129.1	93.6	0.9758	54.8	47.7	12.4	15.8	32.1	23.7	36.2	22.4	54.8	44.7	36.5	37.5	59.3	11.3	2ND	1	2015
Michigan	B10	40	33	114.4	90.4	0.9375	53.9	47.7	14.0	19.5	25.5	24.9	30.7	30.0	54.7	46.8	35.2	33.2	65.9	6.9	2ND	3	2018
Texas Tech	B12	38	31	115.2	85.2	0.9696	53.5	43.0	17.7	22.8	27.4	28.7	32.9	36.6	52.8	41.9	36.5	29.7	67.5	7.0	2ND	3	2019
Gonzaga	WCC	39	37	117.8	86.3	0.9728	56.6	41.1	16.2	17.1	30.0	26.2	39.0	26.9	56.3	40.0	38.2	29.0	71.5	7.7	2ND	1	2017
Kentucky	SEC	40	29	117.2	96.2	0.9062	49.9	46.0	18.1	16.1	42.0	29.7	51.8	36.8	50.0	44.9	33.2	32.2	65.9	3.9	2ND	8	2014

From here we can clean the data by making NA values into actual values. The NA values only come from teams that did not make the post season and thus have no post season seed or post season status.

Data Cleaning

```
cbb$POSTSEASON[is.na(cbb$POSTSEASON)] <- "No Postseason"
cbb$SEED[is.na(cbb$SEED)] <- "No Postseason"
cbb <- cbb %>% mutate(POSTSEASON = na_if(POSTSEASON, "N/A"))
cbb <- cbb %>%
  mutate(POSTSEASON = replace_na(POSTSEASON, "No Postseason"))
```

Now, we will break the data set down to only include our predictors and response variable and rename our variables so they are easier to work with

```
cbb <- cbb %>%
  rename(Three_Pointer = '3P_O',Free_Throw = FTR,
         Two_Pointer = '2P_O', Turnover = TOR,
         Rebound = ORB, Field_Goal = 'EFG_O')
cbb <- cbb %>% dplyr::select(Free_Throw, Two_Pointer, Three_Pointer, Turnover, Rebound, Field_Goal, POS
```

Now that the data is cleaned up and our NA values have already been reworked, we can look at all the data we are working with.

```
cat("Here are the amounts (rows, columns) currently in our dataset: (",
    nrow(cbb), ",", ncol(cbb), ")\n")
```

```
## Here are the amounts (rows, columns) currently in our dataset: ( 3523 , 7 )
```

Because all of our values had data, and the only NAs were changed to Did Not Make Postseason, we still have the same number of observations, however our column amount has changed since we now only have predictors and response.

Visual EDA

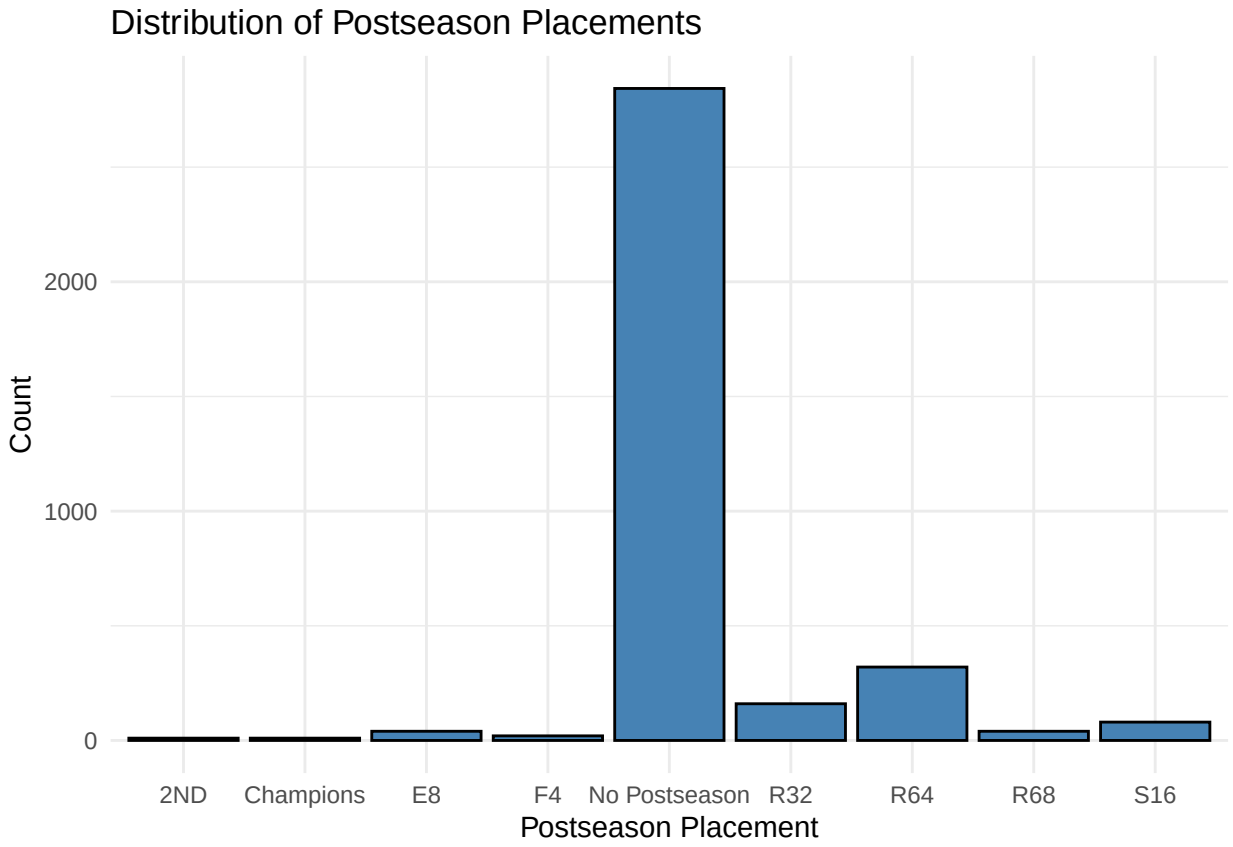
For this section we will now being using Visual EDA. First we will have a bar graph to see how many teams have made each round in the post season. Next we will have Box plots to show relationship between Three Pointers, Free Throws and Post Season Position and finally we will have a correlation Heat Map to see the relationship between all of our predictors.

Postseason Bar Graph

Below is a bar graph that displays the number of teams within each postseason placement. Clearly, most teams did not make the postseason and very few groups did make it to the R32 and R64 placements.

```
ggplot(cbb, aes(x = factor(POSTSEASON))) +
  geom_bar(fill = "steelblue", color = "black") +
```

```
  labs(title = "Distribution of Postseason Placements",
       x = "Postseason Placement",
       y = "Count") +
  theme_minimal()
```

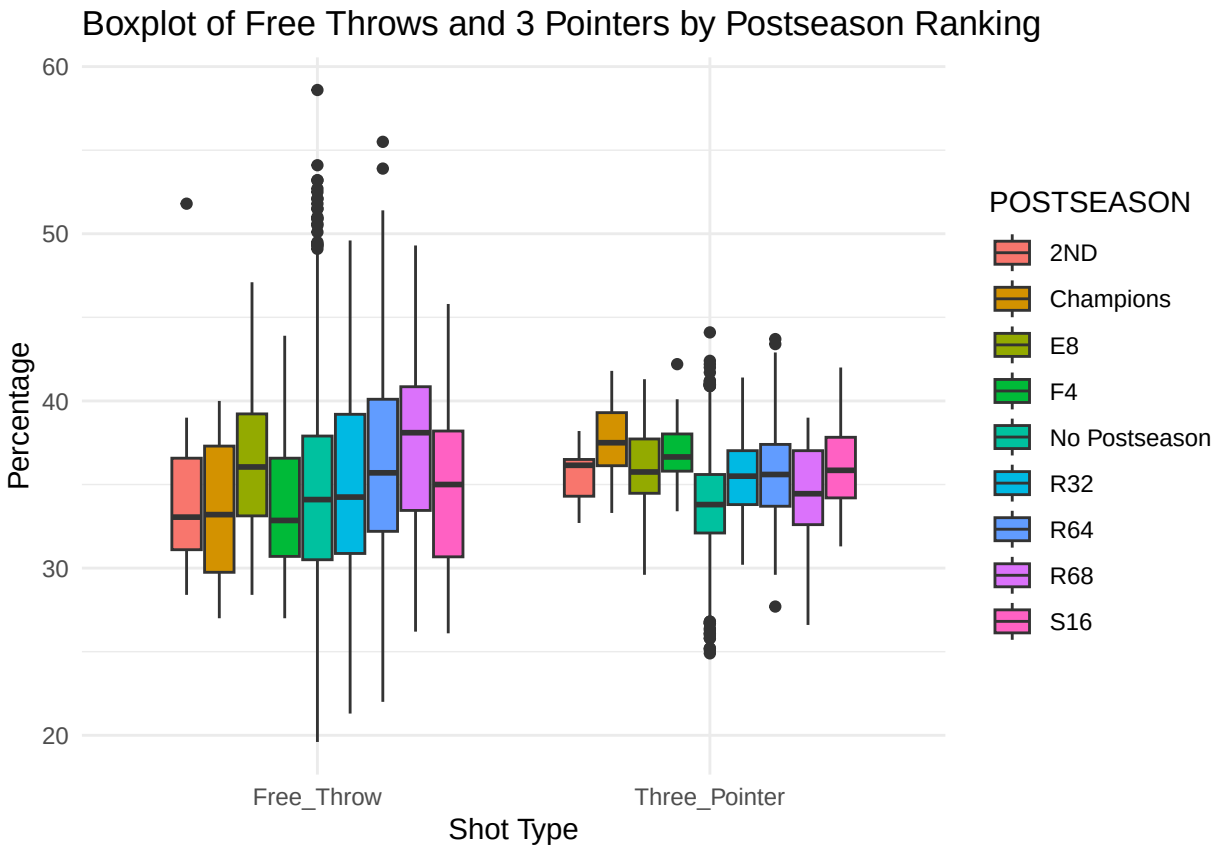


Shooting Boxplot by Postseason Ranking

The graph below illustrates a boxplot comparing free throw and three pointer percentages across different postseason rankings in college basketball. The postseason rankings are displayed in different colors to make it easy to understand.

```
cbb$POSTSEASON <- as.factor(cbb$POSTSEASON)
cbb_long <- cbb %>%
  dplyr::select(POSTSEASON, Free_Throw, Three_Pointer) %>%
  pivot_longer(cols = c(Free_Throw, Three_Pointer),
               names_to = "Metric", values_to = "Value")

ggplot(cbb_long, aes(x = Metric, y = Value, fill = POSTSEASON)) +
  geom_boxplot() +
  labs(x = "Shot Type",
       y = "Percentage",
       title = "Boxplot of Free Throws and 3 Pointers by Postseason Ranking",
       fill = "POSTSEASON") +
  theme_minimal()
```



Key Takeaways - **Free Throw vs. Three Pointer Distribution** - The distribution for free throws has a wider spread compared to the three-pointer distribution. - **Outliers in Postseason Rankings** - Most of the outliers (black dots) occur within the ranking of **no postseason**. - **Variation in Free Throw Performance** - Teams with postseason rankings from **no postseason to S16** tend to have more variation in performance compared to other rankings.

Correlation Matrix of Basketball metrics

Below is a correlation (heatmap) matrix that displays values from -1 to 1, meaning it is negatively correlated or positively correlated, repsectively. This data visualization helps us get a better understanding between the relationships between the variables in our dataset.

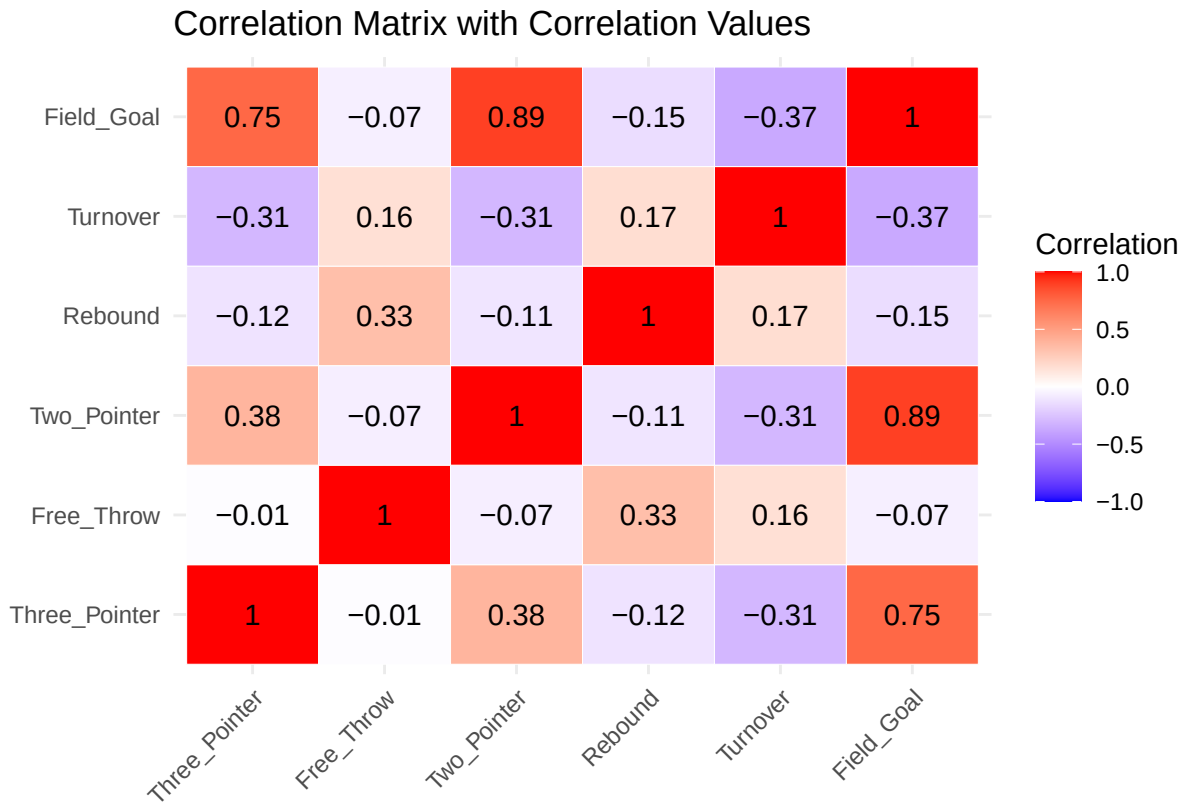
```
cor_data <- cbb %>%
  dplyr::select(Three_Pointer, Free_Throw, Two_Pointer, Rebound, Turnover, Field_Goal)

cor_matrix <- cor(cor_data, use = "complete.obs")

cor_df <- as.data.frame(as.table(cor_matrix))

ggplot(cor_df, aes(Var1, Var2, fill = Freq)) +
  geom_tile(color = "white") +
  geom_text(aes(label = round(Freq, 2)), color = "black", size = 4) +
  scale_fill_gradient2(low = "blue", mid = "white", high = "red",
                       midpoint = 0, limit = c(-1, 1), space = "Lab",
                       name = "Correlation") +
  labs(title = "Correlation Matrix with Correlation Values",
       x = "", y = "") +
  theme_minimal() +
```

```
theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



###

Key Takeaways Here are the correlations categorized by their strength:

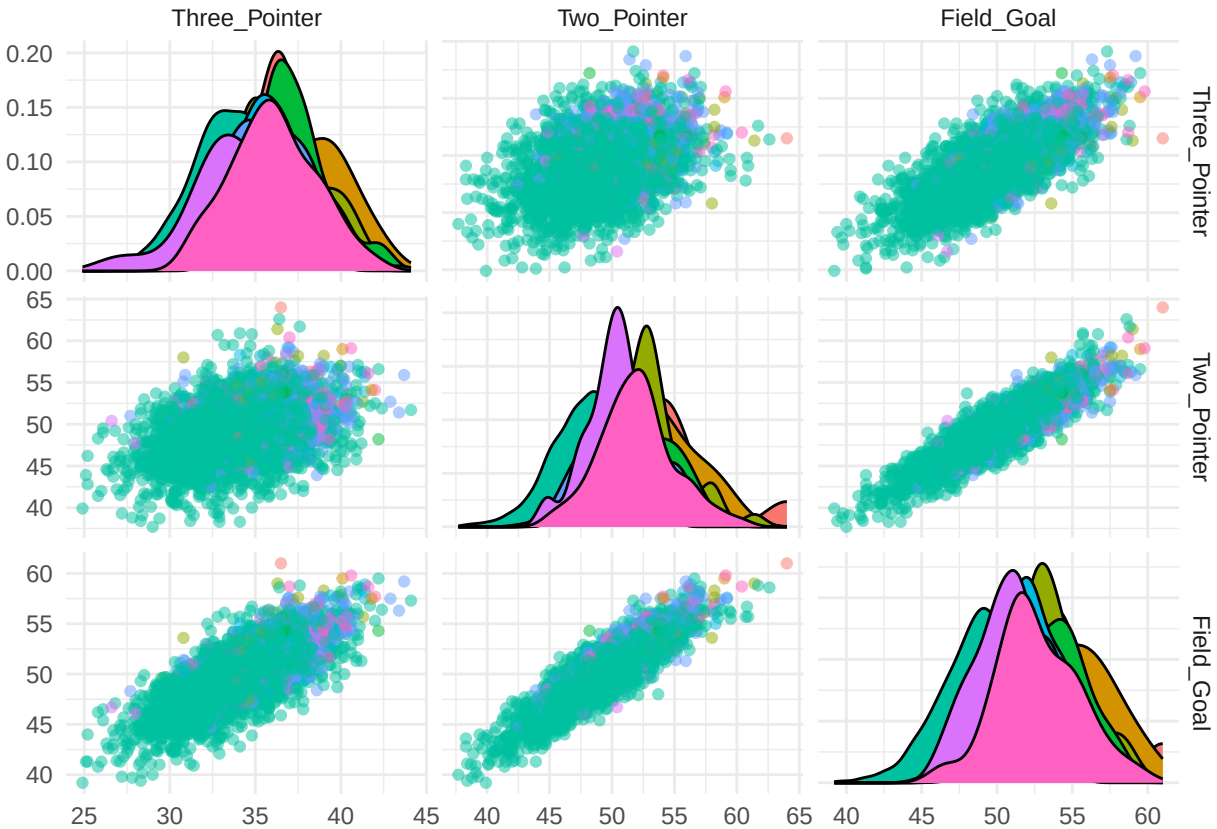
- **Negative Correlations (purple)**
  - **Weak Correlations**
    - Three_Pointer : Rebound (-0.12)
    - Three_Pointer : Turnover (-0.31)
    - Free_Throw : Two_Pointer (-0.07)
    - Free_Throw : Field_Goal (-0.07)
    - Two_Pointer : Rebound (-0.11)
    - Two_Pointer : Turnover (-0.31)
    - Rebound : Field_Goal (-0.15)
    - Turnover : Field_Goal (-0.37)
- **No Correlation (white)**
  - Three_Pointer : Free_Throw (-0.01)
- **Positive Correlations (red)**
  - **Weak Correlations**
    - Three_Pointer : Two_Pointer (0.38)
    - Free_Throw : Rebound (0.33)
    - Free_Throw : Turnover (0.16)
    - Rebound : Turnover (0.17)
  - **Strong Correlations**
    - Three_Pointer : Field_Goal (0.75)
    - Two_Pointer : Field_Goal (0.89)

Combine Metrics Pair

Below you will see a visual that contains six different scatter plots based on three basketball performance metrics: Three-Pointer Percentage, Two-Pointer Percentage, and Field Goal Percentage, with Postseason Ranking represented by color. These three metrics are key indicators of a team's shooting efficiency. The diagonal plots show the distribution of each metric, separated by postseason ranking. The scatter plots let you observe relationships between any two of the three shooting metrics, which helps make it easy to identify potential trends in team performance.

```
cbb_selected <- cbb %>%
  dplyr::select(`Three_Pointer`, `Two_Pointer`, `Field_Goal`, `POSTSEASON`) %>%
  mutate(POSTSEASON = as.factor(POSTSEASON))

ggpairs(
  cbb_selected,
  columns = 1:3,
  mapping = aes(color = POSTSEASON),
  diag = list(continuous = wrap("densityDiag")),
  lower = list(continuous = wrap("points", alpha = 0.5)),
  upper = list(continuous = wrap("points", alpha = 0.5))
) +
  theme_minimal()
```



Key Insights Let's take a look at two of the plots and derive key insights on the relationship: - **Two_Pointer vs. Field_Goal (Bottom Right Scatter Plot)** - Shows a strong positive correlation. - Teams with high accuracy in two-pointers also tend to have strong overall shooting efficiency.

- **Three_Pointer Percentage (Top Left Density Curve)**
  - The distribution varies depending on the postseason ranking.
  - Most follow a somewhat normal distribution, while others appear slightly skewed.

Model Creation

Spliting the data: Training and Testing

In this section, we will split the data into 70% for training and 30% for testing. This ensures that we have a sufficient amount of data to train our models while reserving a separate portion to evaluate their performance later. The 30% test data will remain untouched until the final model evaluation to fairly assess its performance on unseen data.

We will set a random seed (set.seed(123)) to ensure that the results of the split remain consistent every time the code is rerun. This helps maintain reproducibility when training the model. The data is randomly sampled to create the training and testing sets.

Next, we will separate the predictor variables (X) and the target variable (y). The predictor variables include Three_Pointer, Free_Throw, Two_Pointer, Turnover, Rebound, and Field_Goal, while the target variable is POSTSEASON, which represents the postseason ranking.

```
predictors <- c("Three_Pointer", "Free_Throw", "Two_Pointer", "Turnover", "Rebound", "Field_Goal")
outcome <- "POSTSEASON"

set.seed(123)
train_indices <- sample(1:nrow(cbb), size = 0.7 * nrow(cbb))

cbb_train <- cbb[train_indices, ]
cbb_test  <- cbb[-train_indices, ]

X_train <- cbb_train[, predictors]
y_train <- cbb_train[, outcome, drop = FALSE]


X_test  <- cbb_test[, predictors]
y_test  <- cbb_test[, outcome, drop = FALSE]
```

Preprocessing

We will now create a recipe for preprocessing our data, which we will apply to any model we choose to fit. This ensures consistency in how the data is transformed before training. We start by specifying POSTSEASON as the outcome variable and applying centering and scaling transformations to all predictors. Then, we prep the recipe using the training data to estimate the necessary transformations, and finally, we bake the transformations into a new dataset. This transformed dataset will be used to train and test our models, ensuring consistency in feature scaling.

```
cbb_recipe <- recipe(POSTSEASON ~ ., data = cbb) %>%
  step_center(all_predictors()) %>%
  step_scale(all_predictors())

cbb_recipe_prepped <- prep(cbb_recipe, training = cbb)
cbb_transformed <- bake(cbb_recipe_prepped, new_data = cbb)
```

Model Pipelines

Within this section, we will focus on creating different machine learning models, including Logistic Regression, K-Nearest Neighbors (KNN), Random Forest, and Support Vector Machine (SVM). These models will be used to classify teams based on their postseason ranking (POSTSEASON) using six key basketball performance

metrics. These pipelines will allow us to train, evaluate, and compare models to determine the most effective approach for predicting postseason rankings based on team performance metrics.

```
logistic_model <- multinom_reg(mode = "classification", penalty = 0.5, mixture = 1) %>%
  set_engine("glmnet")
logistic_workflow <- workflow() %>%
  add_recipe(cbb_recipe) %>%
  add_model(logistic_model)

knn_model <- nearest_neighbor(mode = "classification") %>%
  set_engine("kknn")
knn_workflow <- workflow() %>%
  add_recipe(cbb_recipe) %>%
  add_model(knn_model)

rf_model <- rand_forest(mode = "classification", trees = 1000) %>%
  set_engine("ranger")
rf_workflow <- workflow() %>%
  add_recipe(cbb_recipe) %>%
  add_model(rf_model)

svm_model <- svm_rbf(mode = "classification") %>%
  set_engine("kernlab")
svm_workflow <- workflow() %>%
  add_recipe(cbb_recipe) %>%
  add_model(svm_model)
```

Performance Metrics of Models

This section evaluates the four untuned models (Logistic Regression, K-Nearest Neighbors, Random Forest, and Support Vector Machine) based on their Area Under the Curve (AUC) and Error Rate (1 - Accuracy). These metrics help us determine the best-performing model before tuning.

```
evaluate_performance <- function(model_workflow, X_data, y_data, model_name) {

  fitted_model <- model_workflow %>% fit(data = bind_cols(X_data, y_data))

  predictions <- predict(fitted_model, X_data) %>%
    bind_cols(predict(fitted_model, X_data, type = "prob")) %>%
    bind_cols(y_data)

  class_levels <- levels(y_data$POSTSEASON)
  pred_cols <- paste0(".pred_", class_levels)

  predictions <- predictions %>%
  mutate(POSTSEASON = as.factor(POSTSEASON))

  auc <- roc_auc(predictions, truth = POSTSEASON, all_of(pred_cols))
  acc <- accuracy(predictions, truth = POSTSEASON, estimate = .pred_class)

  error_rate <- 1 - acc$.estimate

  cat(paste0(model_name, " AUC: ", round(auc$.estimate, 4), "\n"))
  cat(paste0(model_name, " Error Rate: ", round(error_rate, 4), "\n\n"))
```

```
  return(list(AUC = auc$.estimate, Error_Rate = error_rate, Fitted_Model = fitted_model, Predictions = p
}

log_untuned_metrics <- evaluate_performance(logistic_workflow, X_train, y_train, "Logistic Regression")
```

```
## Logistic Regression AUC: 0.5
## Logistic Regression Error Rate: 0.1979
```

```
knn_untuned_metrics <- evaluate_performance(knn_workflow, X_train, y_train, "K-Nearest Neighbors")
```

```
##
## Attaching package: 'kknn'
```

```
## The following object is masked from 'package:caret':
##
##     contr.dummy
```

```
## K-Nearest Neighbors AUC: 0.9985
## K-Nearest Neighbors Error Rate: 0.0689
```

```
rf_untuned_metrics  <- evaluate_performance(rf_workflow, X_train, y_train, "Random Forest")
```

```
## Random Forest AUC: 0.9992
## Random Forest Error Rate: 0.086
```

```
svm_untuned_metrics <- evaluate_performance(svm_workflow, X_train, y_train, "Support Vector Machine")
```

```
## Support Vector Machine AUC: 0.6434
## Support Vector Machine Error Rate: 0.1971
```

Looking at the output, you can see that KNN is the best model with a low error rate of 0.0689 and almost 1 AUC. On the other hand, the worst model appears to be the logistic regression with the highest error rate (just slightly higher than SVM) of 0.1979.

Best Model

We concluded that KNN is the best model for our dataset.

```
# Load necessary libraries
library(tidymodels)
library(yardstick)

# Set up a tuning grid for the KNN model
knn_grid <- grid_regular(
  neighbors(range = c(1, 30)),  # Exploring values for k between 1 and 30
  levels = 30  # You can adjust the number of levels depending on the precision you want
)

# Set up a resampling strategy (e.g., 10-fold cross-validation)
cv_folds <- vfold_cv(cbb_train, v = 10)

# Create the workflow for the KNN model with the preprocessing recipe
knn_model <- nearest_neighbor(mode = "classification", neighbors = tune()) %>%
  set_engine("kknn")

knn_workflow <- workflow() %>%
  add_recipe(cbb_recipe) %>%
```

```
  add_model(knn_model)

# Correctly tune the KNN model by explicitly referencing the correct metric functions
tuned_knn <- knn_workflow %>%
  tune_grid(
    resamples = cv_folds,
    grid = knn_grid,
    metrics = metric_set(yardstick::roc_auc, yardstick::accuracy)
  )


# View the best tuning results
tuned_knn_results <- tuned_knn %>%
  collect_metrics() %>%
  arrange(desc(mean))

# Show the best k value
best_k <- tuned_knn_results %>%
  filter(.metric == "roc_auc") %>%
  slice_max(order_by = mean) %>%
  pull(neighbors)

cat("Best k value:", best_k, "\n")
```

```
## Best k value: 29
```

```
# Refit the model with the best k value
best_knn_model <- nearest_neighbor(mode = "classification", neighbors = best_k) %>%
  set_engine("kknn")

# Create a new workflow with the best k value
best_knn_workflow <- workflow() %>%
  add_recipe(cbb_recipe) %>%
  add_model(best_knn_model)

# Fit the tuned KNN model on the training data
tuned_knn_model <- best_knn_workflow %>%
  fit(data = cbb_train)

# Evaluate the tuned KNN model
tuned_knn_metrics <- evaluate_performance(best_knn_workflow, X_train, y_train, "Tuned K-Nearest Neighbo
```

```
## Tuned K-Nearest Neighbors AUC: 0.9106
## Tuned K-Nearest Neighbors Error Rate: 0.176
```

Conclusion

In this project, we were able to show how machine learning can be used to predict models for sports. We chose to do the prediction of the Post Season status of college basketball teams based on their offensive prowess. To begin the project, we started by introducing what college basketball was and what the playoffs were. Next we went on to the most important part before modeling, and that is the Exploratory Data Analysis. This section allowed us to clean up the data, and then analyze it through various visual means. After this was completed, we went to splitting our data into training and testing data, which we decided to do a 70/30 split. Then we created our pipelines, called workflows in R, which were the Logistic Regression,

K-Nearest Neighbors, Random Forest and Simple Vector Machines. The untuned models had been fit based on which were the best parameters that gave us the highest accuracy, the lowest error rate (1 - accuracy).

Reflecting on our project, we decided to look closely on the offensive capabilities of a team as the more points a team scored could possibly lead to their victory which would help the model predict the post season easier. All models used had good potential on their predictions, however it came down to two models that could be the best. The Random Forest model had a slightly better area under the curve,with 0.9992, which was only .07 more than K Nearest Numbers with an AUC of 0.9985. The difference was that K Nearest Numbers only had an error rate of 0.0689, which was less than the Random Forest which had an error rate of 0.0835. Overall, K Nearest Numbers appears to be the better model, but only just slightly better than the Random Forest model. The worst model is undoubtedly the Logistic Regression model. With the model having an AUC of only 0.5 and an error rate of 0.1979, Logistic Regression solidifies its status as the worst model in predicting the Post Season status for college basketball. If we were to go back and revamp the project, we would go more into depth about why each model incorrectly predicted the post season and see which teams they incorrectly predicted in making the post season and in their placement. If we were to add more predictors we would look at a teams Power Rating and their Adjusted Offensive Efficiency, as these might be able to help the model more accurately predict the post season status.