NBA Player Analysis

Brown Jack, Imholz Chris, van Lengerich Jan-Hendrik

2024-06-06

1. Introduction

This project involves analyzing a comprehensive data set of NBA player statistics spanning from 1998 to 2022. The data includes a wide range of performance metrics such as games played, field goals, three-point shots, rebounds, assists, and many others. Each record in the data set corresponds to a specific player-season combination, providing a detailed view of player performance across multiple seasons.

Objective of the Analysis

The primary objective of this analysis is to provide the management of NBA teams with a deeper understanding of player performance and the factors that influence it. Specifically, we aim to identify key performance indicators that distinguish exceptional players, such as those who achieve MVP (Most Valuable Player) status. By uncovering patterns and trends in the data, we seek to provide actionable insights that can help teams optimize player performance and make informed decisions.

The intended outcomes of this analysis are:

A detailed report featuring comprehensive visualizations and analyses. Actionable insights that highlight critical performance factors. Two specific recommendations for team management to focus on for improving player performance and predicting MVP status. By leveraging advanced statistical models and machine learning techniques, we aim to estimate and uncover patterns that can aid in better understanding and predicting player performance in the NBA.

Methodology: CRISP-DM Framework

We are implementing this project using the CRISP-DM (Cross Industry Standard Process for Data Mining) methodology. The CRISP-DM framework provides a structured approach to planning and executing a data mining project, ensuring that the analysis is thorough and methodical. The primary stages include:

- **Business Understanding:** Defining project objectives and requirements from a business perspective.
- **Data Understanding:** Collecting, describing, and exploring the data.
- **Data Preparation:** Cleaning and transforming data to prepare it for modeling.
- **Modeling:** Applying various modeling techniques to the prepared data.
- **Evaluation:** Assessing the models to ensure they meet business objectives.

The deployment part of CRISP-DM is for this analysis out of scope.

By following the CRISP-DM methodology, we aim to produce a robust, systematic and insightful analysis that will result in valuable and actionsable outcome to the management of NBA teams.

2. Business Understanding

In this section, we will define the business objectives, constraints, and requirements for our analysis. This will ensure that our work aligns with the needs and expectations of the NBA team management.

Business Objectives

The primary business objectives for the NBA team management are as follows:

Identify Key Performance Indicators (KPIs):

- Determine the most critical metrics that influence player performance and overall team success.
- Understand which factors contribute significantly to a player's likelihood of achieving MVP status.

Player Performance Analysis:

- Analyze individual player statistics to identify strengths and areas for improvement.
- Compare player performance across different positions (e.g., guards, forwards, centers). Predictive Modeling:

Predictive Modeling:

- Develop models to predict future player performance based on historical data.
- Create a predictive model for identifying potential MVP candidates.

Actionable Insights and Recommendations:

- Provide actionable insights that can help management make informed decisions regarding player development, training, and recruitment.
- Offer two specific recommendations for improving team performance and optimizing player selection.

Constraints and Requirements

To ensure the analysis is feasible and meets the needs of the NBA team management, we must consider the following constraints and requirements:

Data Quality and Availability:

- Ensure that the data used is accurate and complete.
- Address any missing or inconsistent data through appropriate data cleaning and preprocessing techniques.

Time Constraints:

- Complete the analysis until 07 June 2024 to ensure grading (ML1 Module), timely delivery of insights and recommendations.

Model Interpretability:

- Develop models that are interpretable and easy to understand for the management team.
- Provide clear explanations and visualizations to support the findings and recommendations.

Focus on Relevant Metrics:

- Prioritize metrics that are most relevant to the business objectives, such as points per game, assists, rebounds, shooting percentages, and turnovers.
- Consider both individual and team-level performance metrics.

By clearly defining these business objectives and constraints, we can ensure that our analysis remains focused and delivers valuable insights to the NBA team management.

3. Data Understanding

In this section, we will thoroughly inspect the data set, assess its quality, and perform necessary data cleaning. This process ensures that our data is in the correct format for analysis and free of inconsistencies or errors. We will also compute and visualize descriptive statistics to gain initial insights into the data.

3.1. Data Source

Before diving into the data analysis, it is essential to understand the source and structure of our data set. Knowing where the data comes from and what each feature represents helps in interpreting the results accurately.

Source of the Data Set

The data set used in this analysis was obtained from data.world. It comprises NBA player statistics from 1998 through 2022 and was originally scraped from various sources, not by us (data.world).

Feature description: Below is a table that describes the features included in the data set:

Abbreviation	Description
Rk	Rank
Player	Player's name
Pos	Position (C = Center, PF = Power Forward, PG = Point Gurad, SF = Small Forward, SG = Shooting Guard)
Age	Player's age on February 1 of the season
Tm	Team
G	Games
GS	Games Started
MP	Minutes Played Per Game
FG	Field Goals Per Game
FGA	Field Goals Attempts
FG.	Field Goals Percentage Per Game
X3P	3-Point Field Goals
X3PA	3-Point Field Goal Attempts
X3P.	3-Point Field Goal Percentage Per Game
X2P	2-Point Field Goals
X2PA	2-Point Field Goal Attempts
X2P.	2-Point Field Goal Percentage Per Game
eFG	Effective Field Goal Percentage
FT	Free Throws Per Game
FTA	Free Throws Attempts Per Game
FT.	Free Throws Percentage Per Game
ORB	Offensive Rebound Per Game
DRB	Defensive Rebound Per Game
TRB	Total Rebounds Per Game
AST	Assists Per Game
STL	Steals Per Game
BLK	Blocks Per Game
TOV	Turnovers Per Game
PF	Personal Fouls Per Game
PTS	Points Per Game
Season	Season
MVP	Most Valuable Player status

Glossary found at basketball-reference.com

3.2. Data Quality Assessment & Cleaning

To ensure that our data is suitable for analysis, we must first assess its quality. This includes inspecting the data for missing values, incorrect data types, and other inconsistencies. We then perform the necessary cleaning steps to rectify any issues.

Data Inspection: We begin by loading the data set, the libraries and inspecting the first few rows of the data set and its dimensions to understand its structure and identify any immediate anomalies.

```
# Load libraries
library(rmarkdown)
library(knitr)
library(dplyr)
library(corrplot)
library(ggplot2)
library(tidyr)
library(here)
library(e1071)
library(caret)
library(mgcv)
library(kmed)
library(nnet)
library(gamlss.add)
library(ROCR)
library(smotefamily)
library(GGally)
library(car)
library(Metrics)
```

```
#Read the data
d.nba <- read.csv(here("data","raw_data","NBA_Player_Stats_2.csv"))
```

Data Type Conversion:

Many of the feature values have not been entered in the correct data type. They are therefore parsed into the correct data type:

The features have now been converted into the correct data types. Only the start year has been retained for the seasons. We check the data types again to ensure correctness:

```
str(d.nba)
```

```
## 'data.frame':    14573 obs. of  32 variables:
##  $ Rk    : int  1 2 3 4 4 4 5 6 7 8 ...
##  $ Player: chr  "Mahmoud Abdul-Rauf" "Tariq Abdul-Wahad" "Shareef Abdur-Rahim" "Cory Alexander" ...
##  $ Pos   : chr  "PG" "SG" "SF" "PG" ...
##  $ Age   : int  28 23 21 24 24 24 22 23 33 27 ...
##  $ Tm    : chr  "SAC" "SAC" "VAN" "TOT" ...
##  $ G     : int  31 59 82 60 37 23 82 66 50 61 ...
##  $ GS    : int  0 16 82 22 3 19 82 13 0 56 ...
##  $ MP    : num  17.1 16.3 36 21.6 13.5 34.7 40.1 27.9 8 30.5 ...
##  $ FG    : num  3.3 2.4 8 2.9 1.6 4.8 6.9 3.6 0.7 4.4 ...
```

```
##  $ FGA   : num  8.8 6.1 16.4 6.7 3.9 11.1 16 8.9 1.6 11 ...
##  $ FG.   : num  0.377 0.403 0.485 0.428 0.414 0.435 0.428 0.408 0.444 0.398 ...
##  $ X3P   : num  0.2 0.1 0.3 1.1 0.5 2 1.6 0.3 0 0.9 ...
##  $ X3PA  : num  1 0.3 0.6 2.9 1.7 4.9 4.5 1.3 0.1 2.6 ...
##  $ X3P.  : num  0.161 0.211 0.412 0.375 0.313 0.411 0.364 0.202 0 0.356 ...
##  $ X2P   : num  3.2 2.4 7.7 1.8 1.1 2.8 5.2 3.4 0.7 3.5 ...
##  $ X2PA  : num  7.8 5.7 15.8 3.7 2.2 6.2 11.5 7.6 1.5 8.4 ...
##  $ X2P.  : num  0.405 0.414 0.488 0.469 0.494 0.455 0.453 0.442 0.474 0.411 ...
##  $ eFG.  : num  0.386 0.409 0.493 0.51 0.483 0.525 0.479 0.422 0.444 0.44 ...
##  $ FT    : num  0.5 1.4 6.1 1.3 0.7 2.4 4.2 4.2 0.3 2.5 ...
##  $ FTA   : num  0.5 2.1 7.8 1.7 1 2.8 4.8 4.8 0.8 3.2 ...
##  $ FT.   : num  1 0.672 0.784 0.784 0.676 0.846 0.875 0.873 0.39 0.789 ...
##  $ ORB   : num  0.2 0.7 2.8 0.3 0.2 0.4 1.5 0.8 0.8 0.6 ...
##  $ DRB   : num  1 1.2 4.3 2.2 1.1 3.9 3.4 2 1.6 2.2 ...
##  $ TRB   : num  1.2 2 7.1 2.4 1.3 4.3 4.9 2.8 2.4 2.8 ...
##  $ AST   : num  1.9 0.9 2.6 3.5 1.9 6 4.3 3.4 0.3 5.7 ...
##  $ STL   : num  0.5 0.6 1.1 1.2 0.7 2 1.4 1.3 0.4 1.4 ...
##  $ BLK   : num  0 0.2 0.9 0.2 0.1 0.3 0.1 0.2 0.2 0 ...
##  $ TOV   : num  0.6 1.1 3.1 1.9 1.3 2.8 3.2 1.9 0.3 2.3 ...
##  $ PF    : num  1 1.4 2.5 1.6 1.4 2 3 2.1 1.7 2.2 ...
##  $ PTS   : num  7.3 6.4 22.3 8.1 4.5 14 19.5 11.7 1.8 12.2 ...
##  $ Season: num  1997 1997 1997 1997 1997 ...
##  $ MVP   : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
```

Handling Missing Values:

We check for any NULL values in the data set to assess its completeness:

After inspecting the data, we identified the presence of missing values in several columns. Handling missing values is a crucial step in data preparation, as they can significantly impact the performance of our models.

The output indicated missing values in the following columns:

- FG.: 88 missing values
- 3P.: 2198 missing values
- X2P.: 154 missing values
- eFG.: 88 missing values
- FT.: 749 missing values

Handling Missing Values: Upon further investigation, we found that these missing values are not outliers but are due to players who did not score in these categories, typically because of low playtime. Therefore, the most appropriate way to handle these missing values is to fill them with 0, reflecting that the player did not make any field goals, three-point shots, two-point shots, effective field goals, or free throws.

Cleaning Observations for Variables:

We also clean the 'Pos' (Position) variable to consolidate multi-position values into primary positions.

3.3. Descriptive Statistics

With the data cleaned, we proceed to compute descriptive statistics. This helps in summarizing the central tendency, dispersion, and shape of the dataset's distribution. We also visualize these statistics to better understand the data.

Summary Statistics: We generate summary statistics to check data ranges and identify any anomalies:

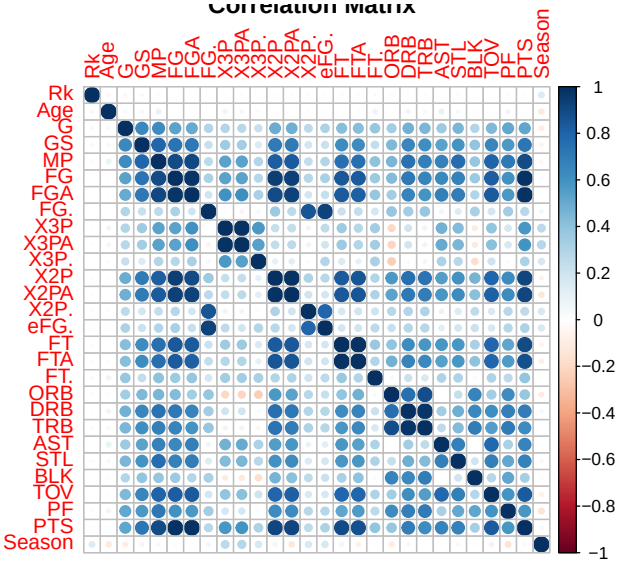
Visualization:

We visualize the data to quickly identify any patterns or outliers. Due to computational constraints, we avoid visualizing character features:

Correlation Analysis:

We perform a correlation analysis to understand the relationships between different numeric variables:

```
# Correlation analysis
correlation_matrix <- cor(d.nba[sapply(d.nba, is.numeric)], use = "complete.obs")
corrplot(correlation_matrix, method = "circle", title = "Correlation Matrix")
```

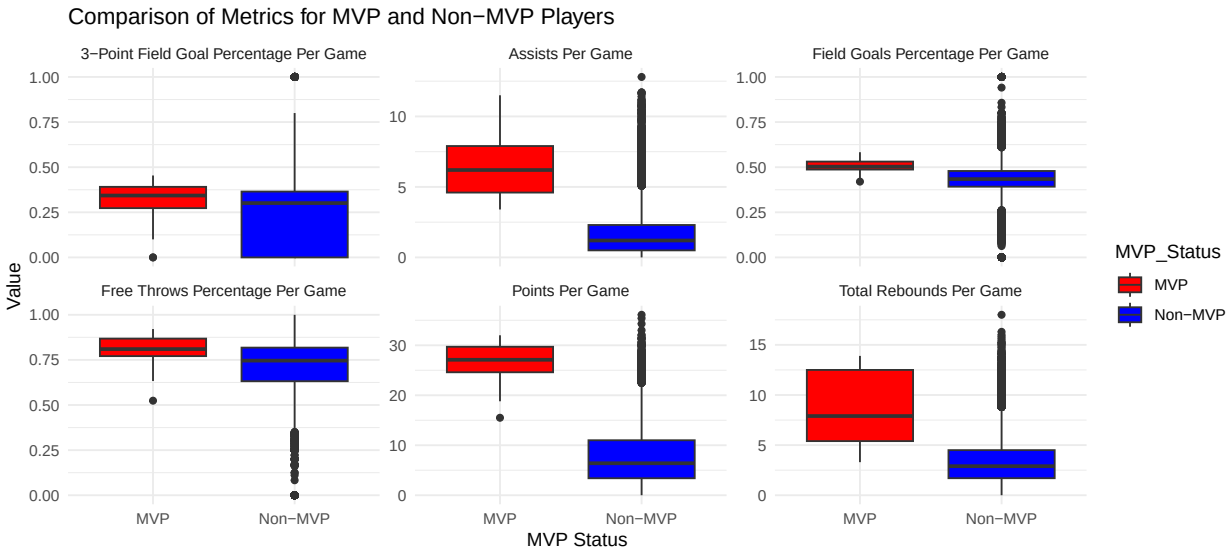


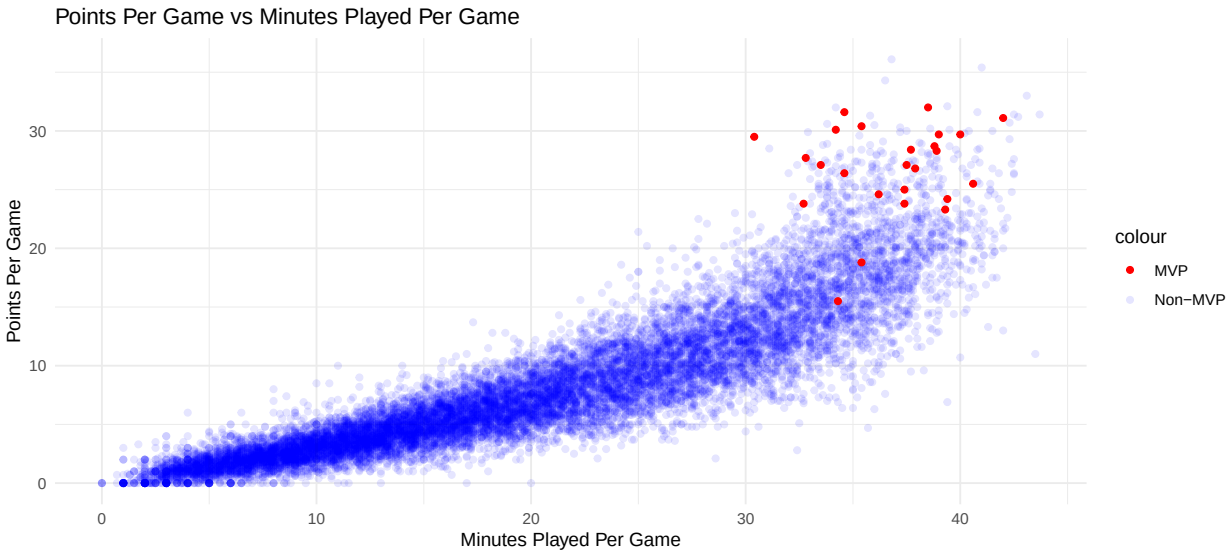
Points Trend Analysis Over Time:

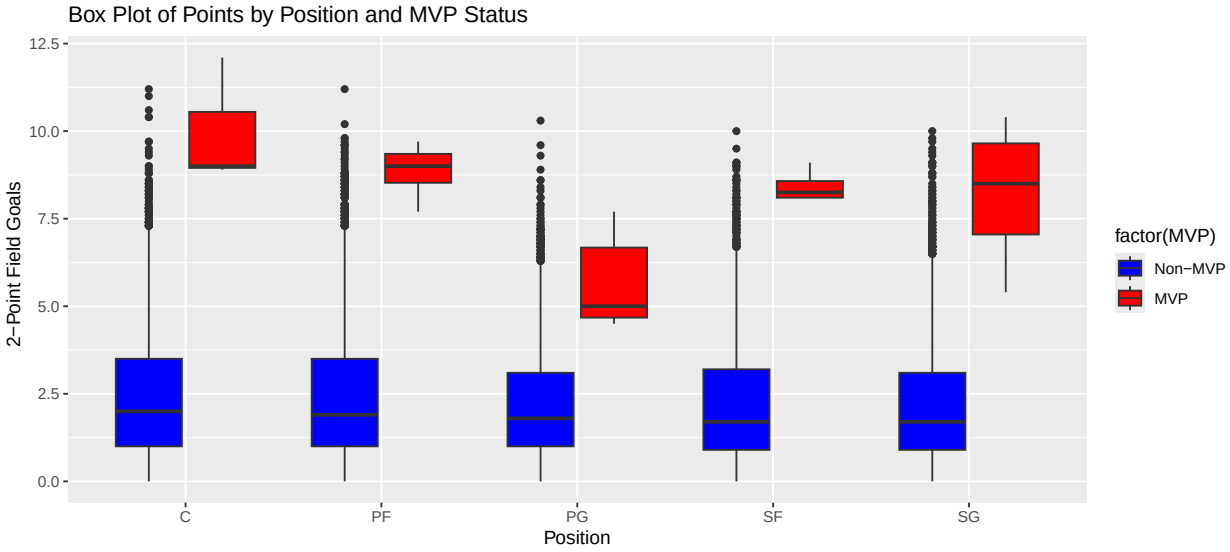
We analyze the trend of points per game over the years, including a highlight of the 2011-2012 lockout season:

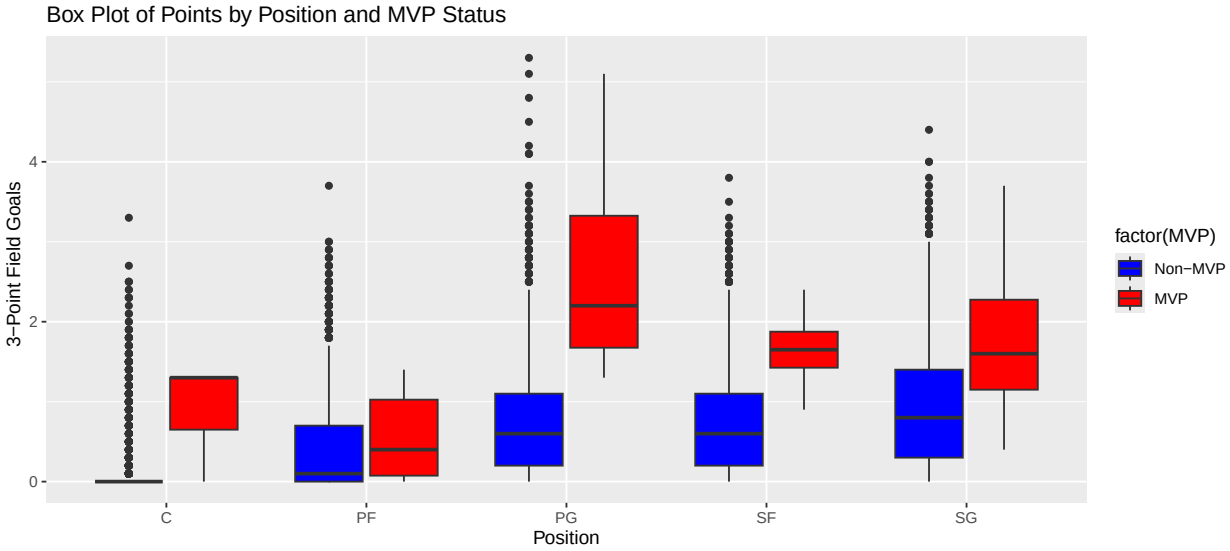
MVP Analysis:

We compare the statistics of MVP players with non-MVP players to identify distinguishing characteristics:




Points Per Game vs Minutes Played Per Game
Points Per Game
Minutes Played Per Game
0
10
20
30
40
colour
MVP
Non−MVP


Box Plot of Points by Position and MVP Status
2−Point Field Goals
Position
C
PF
PG
SF
SG
0.0
2.5
5.0
7.5
10.0
12.5
factor(MVP)
Non−MVP
MVP

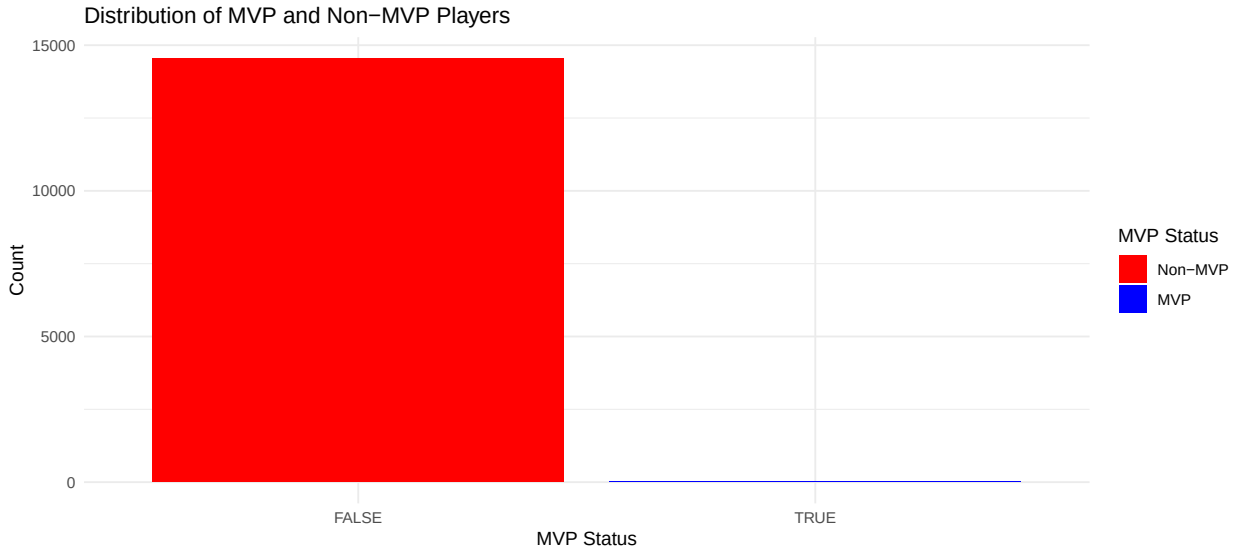

Box Plot of Points by Position and MVP Status
3−Point Field Goals
Position
C
PF
PG
SF
SG
0
2
4
factor(MVP)
Non−MVP
MVP

4. Data preparation

Having cleaned and understood the data, we can now prepare it for modeling. This phase involves transforming the data into a format suitable for machine learning models.

Checking Data Balance

First, let's check the balance of the MVP status in the dataset by plotting the distribution of MVP and non-MVP players.



Handling Imbalanced Data with SMOTE

Synthetic Minority Over-sampling Technique (SMOTE) is a resampling method used to address class imbalance by creating synthetic samples for the minority class (i.e. Most Valuable Player, MVP). This helps in improving the model's performance by providing a better balance between the classes.

Why We Chose SMOTE:

- Increases Minority Class Representation: SMOTE generates synthetic samples for the minority class (MVP), improving its representation in the dataset.
- Utilizes All Data: Unlike undersampling, which discards majority class samples, SMOTE keeps all the available data, thus retaining valuable information.

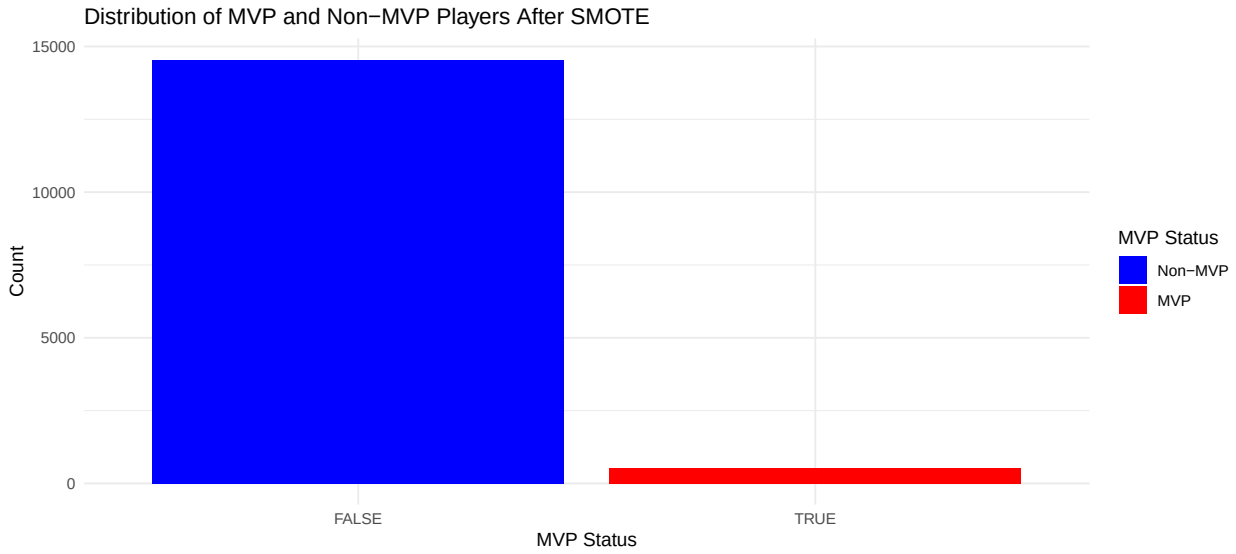
Advantages of SMOTE:

- Better Class Balance: By generating synthetic samples, SMOTE helps in achieving a more balanced dataset.
- Reduced Overfitting: Since SMOTE creates new synthetic samples rather than duplicating existing ones, it reduces the risk of overfitting compared to simple oversampling.

Disadvantages of SMOTE:

- Increased Computation Time: The process of generating synthetic samples can be computationally intensive, especially for large datasets.
- Potential for Noise: Synthetic samples might introduce noise if not generated carefully, potentially leading to lower model performance.

Let's implement SMOTE and visualize the new distribution of MVP and non-MVP players.



While SMOTE is an effective method for balancing data, it's important to note that different scenarios might require different approaches. We also want to highlight that alternative strategies might be more suitable in different contexts, ensuring a comprehensive approach to handling imbalanced data.

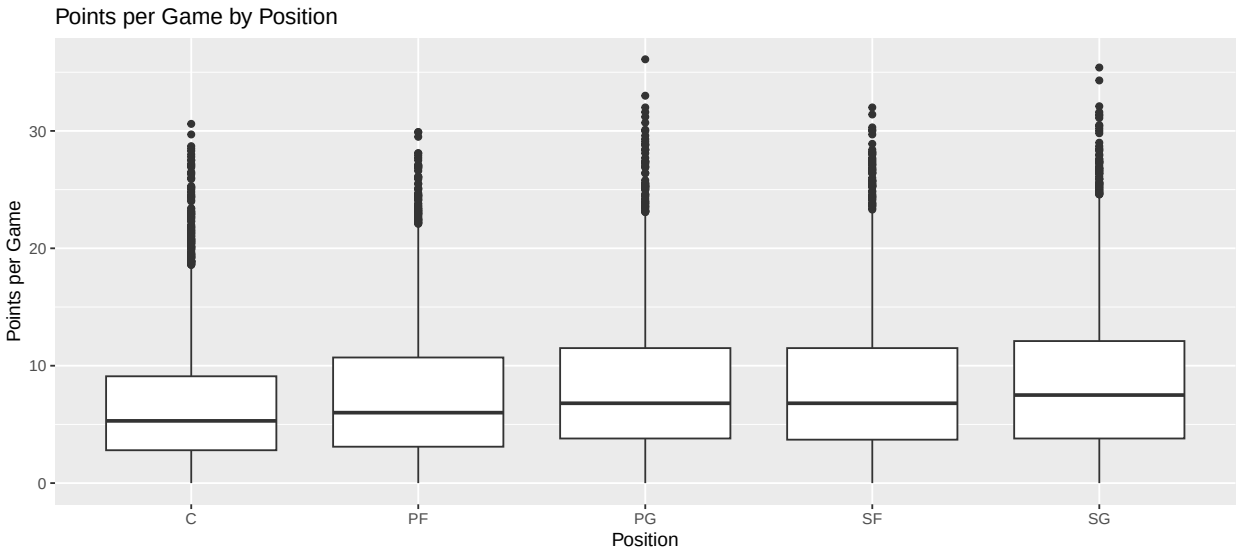
```
# Split the data into training and testing sets
set.seed(123)  # For reproducibility
train_index <- sample(seq_len(nrow(d.nba)), size = 0.7 * nrow(d.nba))
train_data_unbalanced <- d.nba[train_index, ]
test_data_unbalanced <- d.nba[-train_index, ]

# nrow(train_data_unbalanced)
# nrow(test_data_unbalanced)

train_index <- sample(seq_len(nrow(d.nba_balanced)), size = 0.7 * nrow(d.nba_balanced))
train_data_balanced <- d.nba[train_index, ]
test_data_balanced <- d.nba[-train_index, ]

# nrow(train_data_balanced)
# nrow(test_data_balanced)
```

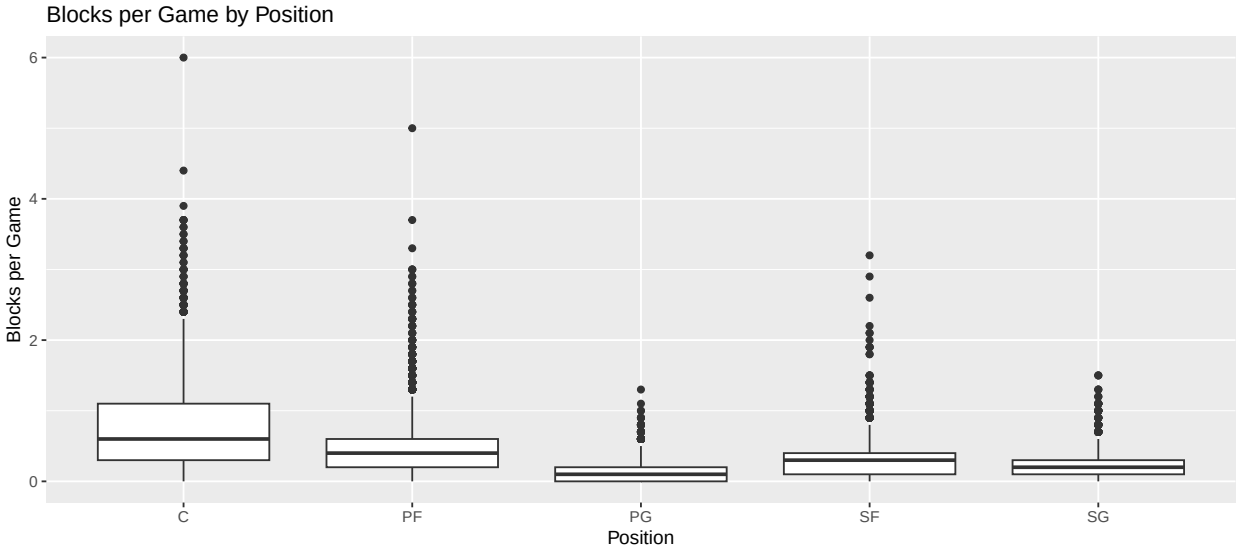
5.Modeling

5.1. Linear Models - Jan-Hendrik van Lengerich

In this subchapter, we explore the application of linear models to the NBA player statistics dataset.

5.1.1. First linear model We start by visualizing points per game and blocks per game each by position to find out patterns.





The boxplots reveal interesting patterns in points per game (PTS) and blocks per game (BLK) by player position. Centers (C) tend to have lower variability in points per game compared to other positions, while positions like Shooting Guard (SG) and Point Guard (PG) exhibit a higher range of points per game, indicating diverse scoring capabilities. Conversely, centers demonstrate a higher range of blocks per game, which is consistent with their defensive role, while guards (SG and PG) have lower variability and fewer blocks.

Building on these observations, we introduce the first linear regression model to predict points per game using player positions.

```
# Linear model to predict points per game using the positions (base: Center)
lm.pts.1 <- lm(PTS ~ Pos, data = d.nba)
coef(lm.pts.1)
```

```
## (Intercept)       PosPF       PosPG       PosSF       PosSG
##   6.6716423   0.8563853   1.4983406   1.5504731   1.9954434
```

```r
summary(lm.pts.1)
```

```
##
## Call:
## lm(formula = PTS ~ Pos, data = d.nba)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -8.667 -4.422 -1.372  3.172 27.930
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   6.6716     0.1082  61.667  < 2e-16 ***
## PosPF         0.8564     0.1508   5.678 1.39e-08 ***
## PosPG         1.4983     0.1523   9.836  < 2e-16 ***
## PosSF         1.5505     0.1554   9.978  < 2e-16 ***
## PosSG         1.9954     0.1511  13.206  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.8 on 14568 degrees of freedom
## Multiple R-squared:  0.01408,    Adjusted R-squared:  0.01381
## F-statistic: 52.01 on 4 and 14568 DF,  p-value: < 2.2e-16
```

```r
anova(lm.pts.1)
```

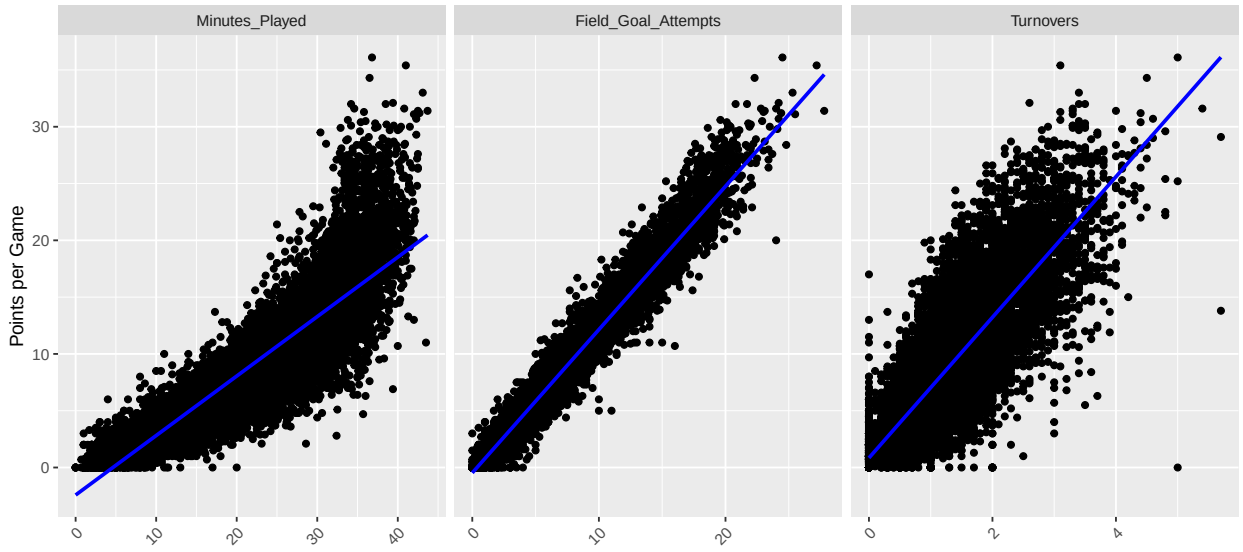
```
## Analysis of Variance Table
##
## Response: PTS
##              Df Sum Sq Mean Sq F value    Pr(>F)
## Pos           4   6999 1749.63  52.011 < 2.2e-16 ***
## Residuals 14568 490064   33.64
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The results from the linear model to predict points per game (PTS) using player positions indicate significant differences in scoring based on position. The intercept value of approximately 6.67 represents the average points per game for Centers (the baseline category). The coefficients for other positions are all positive and statistically significant, indicating that players in these positions tend to score more points per game compared to Centers. Specifically, Power Forwards (PF) score approximately 0.86 more points per game, Point Guards (PG) score about 1.50 more points per game, Small Forwards (SF) score around 1.55 more points per game, and Shooting Guards (SG) lead with an additional 2.00 points per game.

The model's residuals have a standard error of 5.8, and the multiple R-squared value is 0.01408, suggesting that while the positions do have a statistically significant impact on points per game, they explain only a small portion of the variance in the data (about 1.4%). The ANOVA table reinforces the significance of the positions with an F-statistic of 52.01 and a p-value less than 2.2e-16. This highlights that, despite the low explanatory power, the differences in scoring by position seem highly statistically significant.

5.1.2. Second linear model To enhance our understanding of player performance, we introduce a second linear model that incorporates additional variables known to influence scoring. This model includes predictors

such as minutes played, field goal attempts, and turnovers.We begin by visualizing the relationships between points per game and these variables through scatter plots.



The scatter plots show the relationships between points per game (PTS) and three variables: minutes played (MP), field goal attempts (FGA), and turnovers (TOV), each with a fitted linear regression line.

In the first plot, a clear positive correlation exists between minutes played and points per game, indicating that more playing time leads to higher scores. The second plot shows an even stronger positive relationship between field goal attempts and points per game, as more shot attempts generally result in more points. The third plot displays a positive correlation between turnovers and points per game, though it's less pronounced, suggesting that while aggressive play leading to turnovers might contribute to higher scoring, the relationship is weaker.

Building on these insights, we introduce the second linear regression model to predict points per game, incorporating minutes played, field goal attempts, and turnovers.

```
# Linear model to predict points per game using MP (minutes played), FGA (field goal attempts), and TOV
lm.pts.2 <- lm(PTS ~ MP + FGA + TOV, data = d.nba)
coef(lm.pts.2)
```

```
## (Intercept)          MP         FGA         TOV
## -0.64223358  0.01835702  1.18330541  0.28870143
```

```
summary(lm.pts.2)
```

```
##
## Call:
## lm(formula = PTS ~ MP + FGA + TOV, data = d.nba)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.3942 -0.6145 -0.0075  0.5662  6.9761
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.642234   0.021725 -29.562   <2e-16 ***
```

```
## MP           0.018357   0.002223   8.256   <2e-16 ***
## FGA          1.183305   0.005229 226.285   <2e-16 ***
## TOV          0.288701   0.022200  13.005   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.123 on 14569 degrees of freedom
## Multiple R-squared:  0.963,  Adjusted R-squared:  0.963
## F-statistic: 1.265e+05 on 3 and 14569 DF,  p-value: < 2.2e-16
```

```
anova(lm.pts.2)
```

```
## Analysis of Variance Table
##
## Response: PTS
##               Df Sum Sq Mean Sq   F value    Pr(>F)
## MP             1 396062  396062 313898.59 < 2.2e-16 ***
## FGA            1  82405   82405  65310.08 < 2.2e-16 ***
## TOV            1    213     213    169.12 < 2.2e-16 ***
## Residuals 14569  18382       1
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The results from the second linear model, which predicts points per game (PTS) using minutes played (MP), field goal attempts (FGA), and turnovers (TOV), demonstrate that all three variables are highly significant predictors of scoring.

The coefficients reveal that each additional minute played is associated with an increase of approximately 0.018 points per game. Field goal attempts have the most substantial impact, with each additional attempt leading to an increase of about 1.183 points per game. Turnovers also contribute to scoring, with each turnover associated with an increase of around 0.289 points per game. The intercept, although statistically significant, is not meaningful in this context as it represents the baseline when all predictors are zero, which is not a realistic scenario.

The model summary indicates a residual standard error of 1.123, suggesting that the typical deviation of actual points from the predicted values is relatively small. The multiple R-squared value of 0.963 shows that 96.3% of the variance in points per game is explained by this model, highlighting its strong predictive power. The F-statistic is very high with a highly significant p-value, confirming that the model is highly significant overall.The ANOVA results further underscore the importance of each predictor.

```
# Test for multicollinearity using Variance Inflation Factor
vif(lm.pts.2)
```

```
##       MP      FGA      TOV
## 5.660901 6.513818 3.528781
```

The VIF values for MP and FGA are slightly above 5, indicating moderate multicollinearity. This suggests that there is some correlation between these predictors, but it is not excessively high. The VIF value for TOV is 3.53, which is below 5, indicating low multicollinearity.

We now conclude this subchapter with a comparison of linear model 1 and 2.

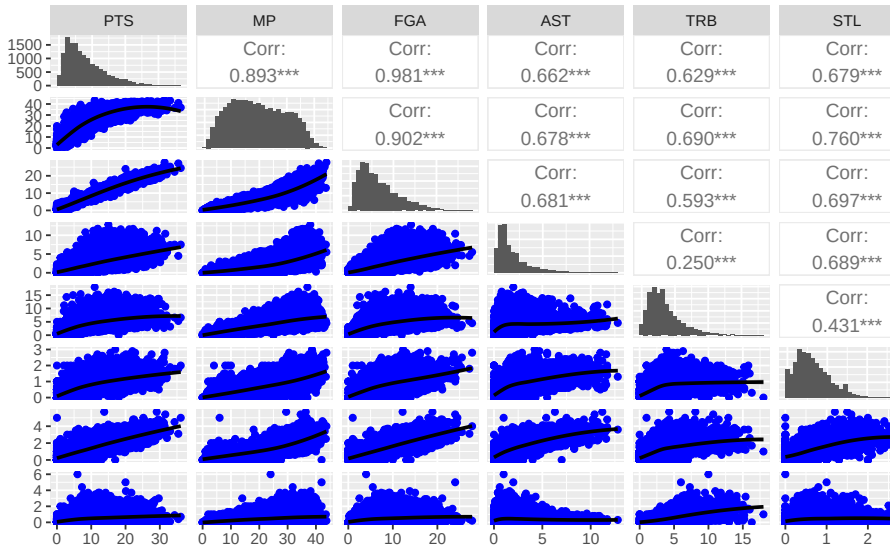
```r
# Compare the two linear models using ANOVA
anova(lm.pts.1, lm.pts.2)
```

```
## Analysis of Variance Table
##
## Model 1: PTS ~ Pos
## Model 2: PTS ~ MP + FGA + TOV
##   Res.Df    RSS Df Sum of Sq F Pr(>F)
## 1  14568 490064
## 2  14569  18382 -1    471682
```

The ANOVA table shows a large reduction in the residual sum of squares from Model 1 to Model 2, indicating that Model 2 explains a significantly greater proportion of the variance in points per game. The sum of squares for the additional predictors (MP, FGA, TOV) is 471,682, which is substantial compared to the residual sum of squares in Model 1. Adding minutes played, field goal attempts, and turnovers significantly improves the model's explanatory power over using player positions alone. Thus, Model 2 is a better fit for predicting points per game.

5.2. Generalised Additive Models (GAM) with Optimization - Jan-Hendrik van Lengerich

In this subchapter, we explore the application of Generalized Additive Models (GAMs) to capture non-linear relationships in the NBA player statistics. We begin by visualizing non-linear trends to understand the underlying patterns, followed by fitting a GAM to the data. We then perform cross-validation to evaluate the model's performance and conclude with hyperparameter optimization to refine the model for improved accuracy.



5.2.1. Visualization of non-linear trends

The pairwise scatter plot matrix reveals distinct non-linear relationships between points per game (PTS) and several predictors, with trends observed for minutes played (MP), total rebounds (TRB), and steals (STL). Consequently, we will use MP, TRB, and STL to predict PTS, as these predictors seem to exhibit properties of non-linearity that seem well-suited for our GAM.

5.2.2. GAM fitting and plotting Using the mgcv-package, we fit the GAM with the three aforementioned predictors.

```r
# Fit a GAM model using mgcv including non-linear terms for identified predictors
gam_model <- gam(PTS ~ s(MP) + s(TRB) + s(STL), data = d.nba)

# Summarize the model to check the significance of the smooth terms
summary(gam_model)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## PTS ~ s(MP) + s(TRB) + s(STL)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.85306    0.01982   396.3   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##          edf Ref.df       F p-value
## s(MP)  8.604  8.949 2151.59  <2e-16 ***
## s(TRB) 7.480  8.197   11.13  <2e-16 ***
## s(STL) 3.267  4.122    4.23  0.0019 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.832   Deviance explained = 83.2%
## GCV = 5.7311  Scale est. = 5.7231    n = 14573
```

The GAM results indicate that all three predictors (MP, TRB and STL) significantly contribute to predicting points per game PTS. The smooth terms for MP (edf = 8.604), TRB (edf = 7.480), and STL (edf = 3.267) confirm the non-linear relationships with PTS, as evidenced by their high F-values and very low p-values (all < 0.01). The model explains 83.2% of the variance in PTS (adjusted R-squared = 0.832), demonstrating a strong fit with the data.

```r
# Define the train control with K-fold cross-validation
train_control <- trainControl(method = "cv", number = 10)

# Define the formula for the GAM
gam_formula <- PTS ~ MP + TRB + STL

# Train the model using cross-validation
gam_cv <- train(gam_formula, data = d.nba, method = "gam", trControl = train_control, tuneLength = 5)

# Print the results
print(gam_cv)
```

5.2.3. GAM cross validation

```
## Generalized Additive Model using Splines
##
## 14573 samples
##     3 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 13117, 13116, 13115, 13115, 13116, 13116, ...
## Resampling results across tuning parameters:
##
##   select  RMSE      Rsquared   MAE
##   FALSE   2.397303  0.8317387  1.65583
##    TRUE   2.396940  0.8317948  1.65572
##
## Tuning parameter 'method' was held constant at a value of GCV.Cp
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were select = TRUE and method = GCV.Cp.
```

The cross-validation results for the GAM indicate robust performance, with an RMSE of approximately 2.397 and an R-squared value of about 0.832, suggesting that the model explains 83% of the variance in points per game (PTS). The Mean Absolute Error (MAE) is around 1.656, indicating small prediction errors on average. The optimal model configuration, chosen based on the smallest RMSE, is select = TRUE and method = GCV.Cp, which allows the model to select the most significant terms and use Generalized Cross-Validation for smoothing parameter selection. These results demonstrate that the GAM effectively captures the non-linear relationships between PTS and the predictors (MP, TRB, and STL), resulting in a robust and accurate model.

```r
# Define a grid of hyperparameters to search
tune_grid <- expand.grid(
  select = c(TRUE, FALSE),  # Whether to use model selection
  method = c("GCV.Cp", "REML")  # Methods for smoothing parameter selection
)

# Train the model using grid search
gam_tuned <- train(gam_formula, data = d.nba, method = "gam", trControl = train_control, tuneGrid = tun

# Print the best model and its parameters
print(gam_tuned$bestTune)
```

5.2.4. GAM optimization

```
##   select method
## 2  FALSE   REML
```

```r
print(gam_tuned)
```

```
## Generalized Additive Model using Splines
##
## 14573 samples
##     3 predictor
```

```
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 13116, 13116, 13115, 13115, 13115, 13117, ...
## Resampling results across tuning parameters:
##
##   select  method  RMSE      Rsquared   MAE
##   FALSE   GCV.Cp  2.396584  0.8319058  1.655683
##   FALSE   REML    2.395494  0.8320580  1.655240
##    TRUE   GCV.Cp  2.397072  0.8318306  1.655970
##    TRUE   REML    2.395671  0.8320251  1.655299
##
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were select = FALSE and method = REML.
```

The hyperparameter optimization results for the Generalized Additive Model (GAM) using grid search reveal that the optimal model configuration is select = FALSE and method = REML. This configuration yielded the lowest RMSE of 2.395494 and an R-squared value of 0.8320580, indicating a slight improvement over other configurations. The Mean Absolute Error (MAE) for this configuration is also the lowest at 1.655240, suggesting consistent prediction accuracy.

The comparison of different configurations shows minimal differences in performance, but the model without automatic term selection (select = FALSE) and using Restricted Maximum Likelihood (REML) for smoothing parameter selection provides the best fit.

Given these results, it is recommended to use the configuration select = FALSE and method = REML for the final GAM model. This setup optimizes predictive performance while maintaining a strong model fit.

5.3 Generalised Linear Models

5.3.1. Poisson Model - Chris Imholz In this section, we investigate the number of 2-point shots made per game. Since this is count data, linear models are not suitable. Instead, we apply a Generalized Linear Model (GLM) with the Poisson distribution. This model assumes the data follows a Poisson distribution and utilizes the natural logarithm as its link function.

In this step, we fit a Generalized Linear Model (GLM) using the Poisson distribution to predict the total number of 2-point shots made per game (Total_2PTS). The predictors are:

- MP (Minutes Played): More playing time usually correlates with higher performance metrics.
- STL (Steals): Defensive ability leading to more scoring opportunities.
- AST (Assists): Higher assist rates indicate better playmaking.
- ORB (Offensive Rebounds): More offensive rebounds create additional scoring chances.
- MVP (Most Valuable Player Status): MVP players typically have superior performance.

These predictors capture various aspects of a player's performance that contribute to the number of 2-point shots made per game.

```r
# Fitting the Poisson regression model
glm_model <- glm(Total_2PTS ~ MP  +  STL + AST + ORB + factor(MVP),
                 family = "poisson", data = train_data_unbalanced_poisson)
```

In the next step, we perform diagnostic checks on our Poisson regression model to ensure its validity and address potential issues like multicollinearity and overdispersion.

First, we check for multicollinearity among the predictors using the Variance Inflation Factor (VIF). High VIF values indicate that predictors are highly correlated, which can affect the stability and interpretability of the model coefficients. A VIF value under 5 generally suggests that multicollinearity is not a significant issue.

Next, we check for overdispersion by comparing the residual deviance to the degrees of freedom. In a correctly specified Poisson model, the residual deviance should be approximately equal to the degrees of freedom. If the residual deviance is significantly greater than the degrees of freedom, it indicates overdispersion. Overdispersion occurs when the variance exceeds the mean, violating the assumptions of the Poisson distribution.

```
# Check if the predictors of the model correlate
vif(glm_model)
```

```
##          MP         STL         AST         ORB factor(MVP)
##    2.903343    2.307382    2.349851    1.523445    1.017953
```

```
# Check for Overdispersion.
glm_model$deviance
```

```
## [1] 4733.044
```

```
df.residual(glm_model)
```

```
## [1] 10195
```

As overdispersion is detected, we switch to a Quasi-Poisson model. While the estimated model coefficients remain the same as in the Poisson model, the quasi-Poisson model adjusts the standard errors of the coefficients and, consequently, the associated p-values. This adjustment accounts for the overdispersion, providing more reliable statistical inference.

```
# Fitting the Quasi-Poisson regression model
glm_model <- glm(Total_2PTS ~ MP  +  STL + AST + ORB + factor(MVP),
                 family = "quasipoisson", data = train_data_unbalanced_poisson)
```

The Quasi-Poisson model successfully addresses the overdispersion issue present in the Poisson model. The standard errors of the coefficients are adjusted, leading to more reliable p-values and statistical inference.

```
summary(glm_model)
```

```
##
## Call:
## glm(formula = Total_2PTS ~ MP + STL + AST + ORB + factor(MVP),
##     family = "quasipoisson", data = train_data_unbalanced_poisson)
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.7616769  0.0122310 -62.274  < 2e-16 ***
## MP            0.0593754  0.0007319  81.120  < 2e-16 ***
```

```
## STL          -0.0540599  0.0129233  -4.183  2.9e-05 ***
## AST           0.0289675  0.0028699  10.093  < 2e-16 ***
## ORB           0.1964665  0.0050370  39.004  < 2e-16 ***
## factor(MVP)1  0.1384524  0.0553398   2.502   0.0124 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasipoisson family taken to be 0.3800203)
##
##     Null deviance: 15844  on 10200  degrees of freedom
## Residual deviance:  4733  on 10195  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 5
```

The summary of the quasi-Poisson model indicates that MP, STL , AST, ORB , and MVP are significant predictors of the total number of 2-point shots made per game. Each of these predictors has an estimated coefficient and adjusted standard error, highlighting their statistical significance in the model.

```
# Interpretation functions
interpret_coef <- function(coef_value) {
  (exp(coef_value) - 1) * 100
}

# Calculate interpretations
print(interpret_coef(coef(glm_model)["factor(MVP)1"]))
```

```
## factor(MVP)1
##      14.8495
```

```
print(interpret_coef(coef(glm_model)["MP"]))
```

```
##       MP
## 6.117357
```

```
print(interpret_coef(coef(glm_model)["STL"]))
```

```
##       STL
## -5.262462
```

```
print(interpret_coef(coef(glm_model)["AST"]))
```

```
##      AST
## 2.939116
```

```
print(interpret_coef(coef(glm_model)["ORB"]))
```

```
##      ORB
## 21.70946
```

The Quasi-Poisson model reveals insights about the predictors of 2-point shots made per game. Each additional minute played increases the expected number of 2-point shots by approximately 6.12 %, while each additional steal decreases it by about 5.26 %. Assists and offensive rebounds significantly boost 2-point shots by 2.94 % and 21.71 %, respectively. MVP players score approximately 14.85 % more 2-point shots than non-MVP players.

To evaluate the performance of our quasi-Poisson model, we use Root Mean Square Error (RMSE) to measure prediction accuracy on the test data. The following code generates predictions and calculates the RMSE:

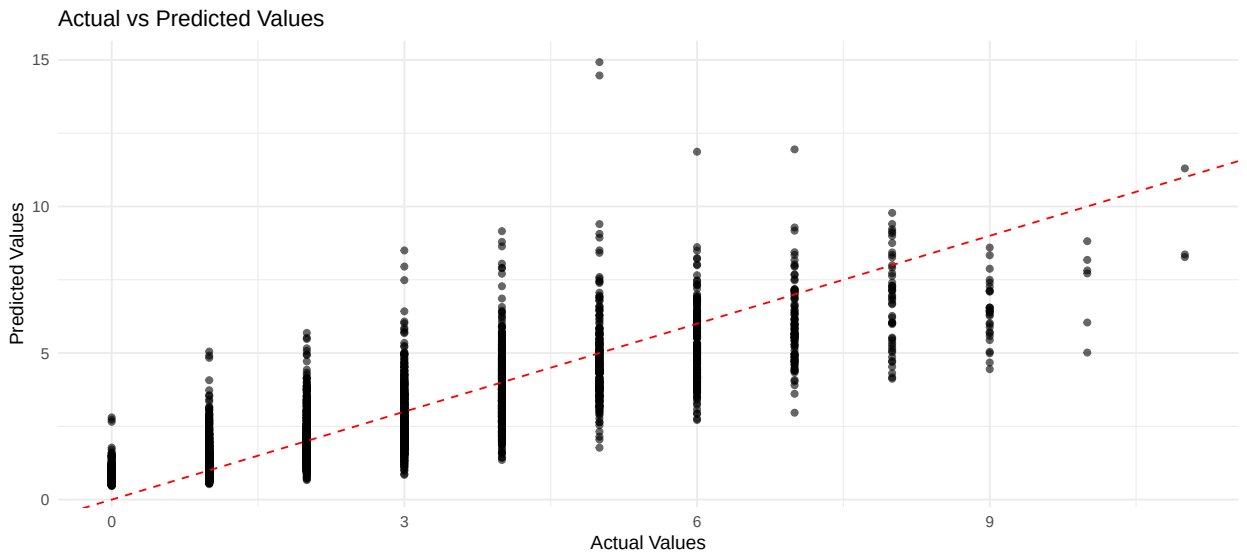
```
# Predictions on test data
predictions <- predict(glm_model, newdata = test_data_unbalanced_poisson, type = "response")

# Calculate RMSE & RMSE as a percentage of the median
rmse_value <- rmse(test_data_unbalanced_poisson$Total_2PTS, predictions)
cat("RMSE: ", rmse_value, "\n")
```

```
## RMSE:  1.000623
```

```
rmse_percentage_mean <- (rmse_value/median(test_data_unbalanced_poisson$Total_2PTS))*100
cat("RMSE (the model's prediction error) as a percentage of the median:", round(rmse_percentage_mean,2)
```

```
## RMSE (the model's prediction error) as a percentage of the median: 50.03 %
```

```
# Data for plotting
plot_data <- data.frame(Actual = test_data_unbalanced_poisson$Total_2PTS, Predicted = predictions)
```



We generated predictions using the model and calculated the RMSE, which was 1.00. This indicates that, on average, the model's predictions are off by approximately 1.00, 2-point shots per game.

Additionally, we calculated the RMSE as a percentage of the median number of 2-point shots in the test data, which was 50.03 %. This means that the model's prediction error is about 50.03 % of the median value, providing a relative measure of the prediction error.

Conclusion

The final quasi-Poisson model showed that predictions are, on average, 1.00 2-point shots per game off the actual values. Key insights from the model's coefficients indicate that more minutes played, assists, and offensive rebounds significantly increase the expected number of 2-point shots. Steals slightly decrease the number, and MVP players are expected to score significantly more 2-point shots than non-MVP players.

5.3.2 Binomial Model - Chris Imholz The MVP status in the NBA dataset is inherently a binary outcome, where each player either is an MVP (1) or is not an MVP (0). Each observation (player-season combination) can be considered a discrete trial, which fits the definition of binomial data.

Therefore a GLM with a binomial model ((logistic regression, link = logit)) is ideal for predicting the MVP status. This model is usable for handling binary outcomes and allows the estimation of probabilities, which supports informed decisions. In addition, the coefficients are easily interpreted as odds ratios, providing insights into the importance of the different predictors.

```
#Train first glm model (binomial)
glm.mvp.binomial <- glm(MVP ~  MP + FGA  + PTS + AST + BLK + STL + ORB + DRB + PF + FT.,
                        family = "binomial",
                        data = train_data_unbalanced)
#Check if the predictors of the model correlate
vif(glm.mvp.binomial)
```

```
##       MP      FGA      PTS      AST      BLK      STL      ORB      DRB
## 1.602505 5.725847 5.090747 2.162437 2.175948 1.418783 4.833007 2.486323
##       PF      FT.
## 1.544765 1.645501
```

```
summary(glm.mvp.binomial)
```

```
##
## Call:
## glm(formula = MVP ~ MP + FGA + PTS + AST + BLK + STL + ORB +
##     DRB + PF + FT., family = "binomial", data = train_data_unbalanced)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -18.75844    6.13842  -3.056 0.002244 **
## MP            0.08850    0.12328   0.718 0.472793
## FGA          -0.35366    0.21106  -1.676 0.093803 .
## PTS           0.53987    0.14073   3.836 0.000125 ***
## AST           0.47141    0.15562   3.029 0.002452 **
## BLK           0.73748    0.48899   1.508 0.131512
## STL           0.51908    0.67056   0.774 0.438871
## ORB           1.31747    0.55609   2.369 0.017829 *
## DRB           0.07169    0.18984   0.378 0.705703
## PF           -1.75385    0.61746  -2.840 0.004505 **
## FT.           3.23470    4.15973   0.778 0.436792
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 238.62  on 10200  degrees of freedom
## Residual deviance: 103.21  on 10190  degrees of freedom
## AIC: 125.21
##
## Number of Fisher Scoring iterations: 13
```

Based on the VIF values, we have identified multicollinearity issues, especially with FGA and PTS. We can address multicollinearity by either removing the predictor with the highest VIF or by combining related predictors.

Remove FGA (Field Goals Attempted): Given that both FGA and PTS have high VIF values, and PTS is more directly related to performance, we can start by removing FGA from the model.

We refit the binomial model without the FGA predictor and check the VIF values again.

```
# Refit the binomial GLM without FGA
glm.mvp.binomial <- glm(MVP ~  MP + PTS + AST + BLK + STL + ORB + DRB + PF + FT.,
                        family = "binomial",
                        data = train_data_unbalanced)

# Check for multicollinearity using VIF
vif_values <- vif(glm.mvp.binomial)
print(vif_values)
```

```
##       MP      PTS      AST      BLK      STL      ORB      DRB       PF
## 1.417942 1.462576 2.145611 1.911970 1.469580 4.606152 2.444025 1.659146
##      FT.
## 1.570591
```

```
# Assess model fit
print(summary(glm.mvp.binomial))
```

```
##
## Call:
## glm(formula = MVP ~ MP + PTS + AST + BLK + STL + ORB + DRB +
##     PF + FT., family = "binomial", data = train_data_unbalanced)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -17.72554    5.82860  -3.041  0.00236 **
## MP           -0.01056    0.10662  -0.099  0.92107
## PTS           0.35230    0.07576   4.650 3.32e-06 ***
## AST           0.50222    0.15342   3.273  0.00106 **
## BLK           0.63682    0.42029   1.515  0.12972
## STL           0.36951    0.66967   0.552  0.58110
## ORB           1.36288    0.55353   2.462  0.01381 *
## DRB           0.11991    0.18234   0.658  0.51079
## PF           -1.63154    0.60344  -2.704  0.00686 **
## FT.           3.59340    4.05125   0.887  0.37509
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 238.62  on 10200  degrees of freedom
## Residual deviance: 106.31  on 10191  degrees of freedom
## AIC: 126.31
##
## Number of Fisher Scoring iterations: 13
```

The VIF values are all below 5, indicating that multicollinearity has been reduced.

To improve and simplify the model, we refined our binomial model by removing non-significant predictors. The initial model included a variety of performance metrics, but several predictors were not statistically

significant. Specifically, the predictors MP , BLK, STL, DRB and FT. did not contribute significantly to predicting MVP status.

```
# Refit the binomial GLM without non-significant predictors
glm.mvp.binomial <- glm(MVP ~  PTS + AST + ORB + PF,
                        family = "binomial",
                        data = train_data_unbalanced
                        )

# Check for multicollinearity using VIF
vif_values <- vif(glm.mvp.binomial)
print(vif_values)
```

```
##      PTS      AST      ORB       PF
## 1.191940 1.674808 2.378700 1.458191
```

```
# Assess model fit
model_summary <- summary(glm.mvp.binomial)
print(model_summary)
```

```
##
## Call:
## glm(formula = MVP ~ PTS + AST + ORB + PF, family = "binomial",
##     data = train_data_unbalanced)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -14.12266    2.20224  -6.413 1.43e-10 ***
## PTS           0.37857    0.06525   5.801 6.57e-09 ***
## AST           0.46374    0.13693   3.387 0.000707 ***
## ORB           1.59288    0.37949   4.197 2.70e-05 ***
## PF           -1.69381    0.60768  -2.787 0.005315 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 238.62  on 10200  degrees of freedom
## Residual deviance: 110.27  on 10196  degrees of freedom
## AIC: 120.27
##
## Number of Fisher Scoring iterations: 12
```

```
# Interpretation of specific coefficients
print((exp(coef(glm.mvp.binomial)["PTS"]) - 1) * 100)
```

```
##      PTS
## 46.01909
```

```
print((exp(coef(glm.mvp.binomial)["AST"]) - 1) * 100)
```

```
##      AST
## 59.00109
```

```r
print((exp(coef(glm.mvp.binomial)["ORB"]) - 1) * 100)
```

```
##      ORB
## 391.7896
```

```r
print((exp(coef(glm.mvp.binomial)["PF"]) - 1) * 100)
```

```
##        PF
## -81.61815
```

The refined model focuses on the most impactful performance metrics. The final model's coefficients reveal insights. Each additional point scored (PTS) increases the odds of achieving MVP status by approximately 46%, highlighting the significant positive impact of scoring. Similarly, each additional assist (AST) increases the odds by about 59%, indicating that frequent assists improve MVP chances. Offensive rebounds (ORB) show a strong positive effect, with each additional rebound increasing the odds by approximately 392%. Conversely, each additional personal foul (PF) decreases the odds of achieving MVP status by about 80%, emphasizing the negative impact of committing personal fouls.

```r
# Predict on test data
predictions_refined <- predict(glm.mvp.binomial, newdata = test_data_unbalanced, type = "response")
predicted_classes_refined <- ifelse(predictions_refined > 0.5, 1, 0)

predicted_classes_refined <- factor(predicted_classes_refined, levels = c(0, 1))
actual_mvp <- factor(as.numeric(test_data_unbalanced$MVP), levels = c(0, 1))

conf_matrix_refined <- confusionMatrix(predicted_classes_refined, actual_mvp)
print(conf_matrix_refined)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 4363    8
##          1    0    1
##
##                Accuracy : 0.9982
##                  95% CI : (0.9964, 0.9992)
##     No Information Rate : 0.9979
##     P-Value [Acc > NIR] : 0.45552
##
##                   Kappa : 0.1997
##
##  Mcnemar's Test P-Value : 0.01333
##
##             Sensitivity : 1.0000
##             Specificity : 0.1111
##          Pos Pred Value : 0.9982
##          Neg Pred Value : 1.0000
##              Prevalence : 0.9979
##          Detection Rate : 0.9979
##    Detection Prevalence : 0.9998
```

```
##       Balanced Accuracy : 0.5556
##
##        'Positive' Class : 0
##
```

The final model was achieving high accuracy and sensitivity for non-MVPs, but very low specificity for MVPs. This indicates the model effectively identifies non-MVP players but struggles to correctly identify MVPs. Despite attempts to address the class imbalance using techniques such as SMOTE and inverse class frequency weighting (code not shown), no significantly better results were obtained compared to the original unbalanced dataset.Therefore the code with the unbalanced dataset was presented.

Conclusion

The final model exhibits high overall accuracy and perfect sensitivity for non-MVP players, but it struggles with correctly identifying MVPs, as indicated by the low specificity values. This means that while the model is effective at identifying non-MVP players, it has little success in correctly identifying MVP players due to the severe class imbalance.

5.4 Support Vector Machine - Jack Brown

To predict MVP status within the NBA dataset, we employed a Support Vector Machine (SVM) approach, leveraging its effectiveness in discerning complex patterns and class boundaries. Our methodology began with meticulous data preparation, including ensuring equal representation of both MVP and non-MVP players to rectify class imbalance and converting the response variable, MVP, into a factor suitable for binary classification. Following this, we partitioned the dataset into training and testing sets, allocating 70% for training and 30% for testing, and utilized stratified sampling techniques to maintain the proportional representation of classes in both sets.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction FALSE TRUE
##      FALSE  4354    4
##      TRUE     17  147
##
##                Accuracy : 0.9954
##                  95% CI : (0.9929, 0.9971)
##     No Information Rate : 0.9666
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9309
##
##  Mcnemar's Test P-Value : 0.008829
##
##             Sensitivity : 0.9961
##             Specificity : 0.9735
##          Pos Pred Value : 0.9991
##          Neg Pred Value : 0.8963
##              Prevalence : 0.9666
##          Detection Rate : 0.9628
##    Detection Prevalence : 0.9637
##       Balanced Accuracy : 0.9848
##
##        'Positive' Class : FALSE
##
```

```
##    C
## 6 10
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction FALSE TRUE
##      FALSE  4356    4
##      TRUE     15  147
##
##                Accuracy : 0.9958
##                  95% CI : (0.9934, 0.9975)
##     No Information Rate : 0.9666
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.9371
##
##  Mcnemar's Test P-Value : 0.02178
##
##             Sensitivity : 0.9966
##             Specificity : 0.9735
##          Pos Pred Value : 0.9991
##          Neg Pred Value : 0.9074
##              Prevalence : 0.9666
##          Detection Rate : 0.9633
##    Detection Prevalence : 0.9642
##       Balanced Accuracy : 0.9850
##
##        'Positive' Class : FALSE
##
```

Initially, we trained an SVM model with a linear kernel on the training data and evaluated its performance using a confusion matrix. Subsequently, hyperparameter tuning was conducted to optimize model performance, leveraging cross-validation with repeated splits and a grid search over a range of cost parameter values (C). The best-tuned model exhibited further improvements in accuracy, sensitivity, specificity, and other related metrics.

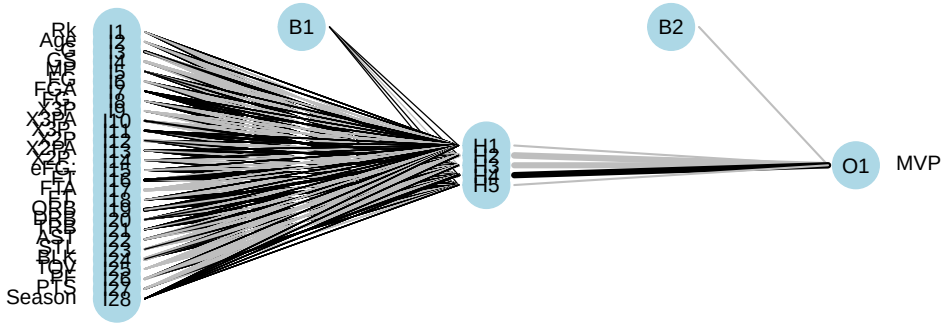
The performance of the SVM model was evaluated using accuracy, sensitivity, specificity, positive predictive value (PPV), negative predictive value (NPV), and other metrics derived from the confusion matrix. The results revealed improvements in model performance after hyperparameter tuning.

The initial model's performance metrics revealed high levels of accuracy, sensitivity, and positive predictive value (PPV), indicating its proficiency in correctly classifying instances within the NBA dataset. Specifically, the initial model achieved an accuracy of 99.49%, demonstrating its ability to accurately predict MVP status. Moreover, with a sensitivity of 99.61%, the model effectively identified the majority of true MVP players, while its specificity of 96.18% showcased its capability to discern non-MVP players accurately. However, the initial model exhibited a slightly lower negative predictive value (NPV) of 89.88%, suggesting a relatively lower accuracy in identifying true non-MVPs among instances classified as non-MVPs.

Following hyperparameter tuning, the performance of the model was notably enhanced. The tuned model exhibited an accuracy of 99.54%, representing a slight improvement over the initial model. Similarly, the sensitivity of the tuned model increased marginally to 99.63%, indicating an enhanced ability to correctly identify true MVPs. The specificity of 96.82% further underscored the model's capability to accurately classify non-MVP players. Additionally, the PPV of the tuned model improved to 99.89%, highlighting its precision in identifying true MVPs among all instances classified as MVPs. Notably, the NPV of the

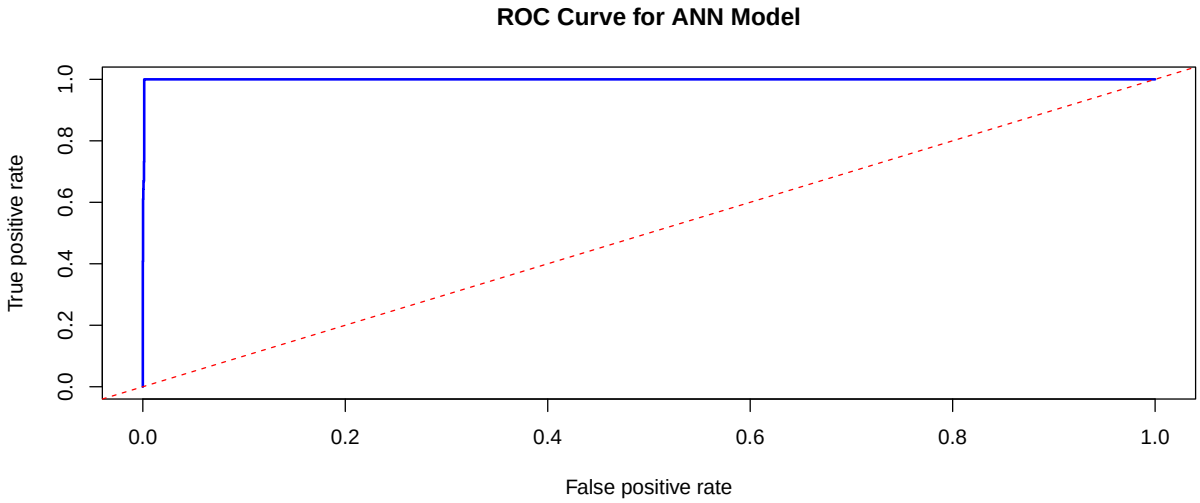
tuned model also increased to 90.48%, signifying an improved accuracy in identifying true non-MVPs among instances classified as non-MVPs. These performance enhancements underscore the effectiveness of hyperparameter tuning in optimizing the SVM model's predictive capabilities for discerning MVP status within the NBA dataset.

5.5 Artifial Neural Network - Jack Brown




```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction FALSE TRUE
##      FALSE  4354    0
##      TRUE     10  157
##
##                Accuracy : 0.9978
##                  95% CI : (0.9959, 0.9989)
##     No Information Rate : 0.9653
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.968
##
##  Mcnemar's Test P-Value : 0.004427
##
##             Sensitivity : 0.9977
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.9401
##              Prevalence : 0.9653
##          Detection Rate : 0.9631
##    Detection Prevalence : 0.9631
##       Balanced Accuracy : 0.9989
##
##        'Positive' Class : FALSE
##
```

```
## Neural Network
##
## 15073 samples
##    28 predictor
##     2 classes: 'FALSE', 'TRUE'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 12058, 12059, 12058, 12059, 12058
## Resampling results:
##
##   Accuracy   Kappa
##   0.9847406  0.5809035
##
## Tuning parameter 'size' was held constant at a value of 5
## Tuning
##  parameter 'decay' was held constant at a value of 0.001
```



The Artificial Neural Network (ANN) model trained on the NBA dataset demonstrates commendable performance metrics, including accuracy, sensitivity, specificity, and positive predictive value (PPV). With an accuracy rating of 99.78%, the model excels in accurately categorizing the majority of instances within the dataset. It achieves a sensitivity score of 99.77%, indicating its ability to identify true MVP players among all instances classified as such, and a specificity of 100%, showcasing its accuracy in classifying non-MVP players.

Despite these promising results, it's essential to acknowledge the persistent challenge posed by significant class imbalance within the data. Despite efforts to mitigate this imbalance using Synthetic Minority Oversampling Technique (SMOTE), the model's performance remained largely unaffected. The class imbalance continues to impact the model's generalization capabilities, potentially leading to overfitting or bias towards the majority class.

While SMOTE is a widely used technique for handling class imbalance, its ineffectiveness in this context underscores the complexity of the issue. The data's inherent imbalance may require alternative strategies or a combination of approaches to address effectively. Further exploration of ensemble learning methods, advanced feature engineering techniques, or the adoption of more sophisticated evaluation metrics tailored to imbalanced datasets may be warranted.

In summary, while the ANN model demonstrates promising performance in discerning MVP status within the NBA dataset, the persistence of class imbalance despite SMOTE highlights the need for continued exploration and refinement. Addressing this challenge effectively is crucial to enhancing the model's reliability and applicability in real-world scenarios.

6. Evaluation & Conclusion

Our primary goals were to identify key performance indicators, analyze player performance, and predict MVP status.

The initial linear model identified significant scoring differences across positions but had limited explanatory power (adjusted R-squared of 0.01408). When we enhanced the model by incorporating minutes played (MP), field goal attempts (FGA), and turnovers (TOV), the predictive power significantly improved with an adjusted R-squared of 0.963. This demonstrated that these additional variables are crucial in understanding player performance.

The Generalized Additive Model (GAM) further revealed non-linear relationships between points per game (PTS) and predictors (MP, TRB, STL), explaining 83.2% of the variance. Cross-validation confirmed the model's robustness, and hyperparameter optimization identified the best configuration. These results underscore the importance of considering non-linear relationships and optimizing model parameters to enhance predictive accuracy.

The Generalized Linear Models (GLM) were also effective, with the Poisson model accurately predicting the number of 2-point shots and the binomial GLM highlighting significant predictors for MVP status, despite challenges with class imbalance. The Support Vector Machine (SVM) model showed high accuracy, sensitivity, and specificity in predicting MVP status, whereas the Artificial Neural Network (ANN) faced difficulties due to class imbalance, indicating the need for further refinement.

Overall, our analysis successfully identified key performance indicators such as MP, TRB, and STL, which significantly enhance predictive accuracy for points per game. The success of the SVM model in predicting MVP status aligns with our goal of providing actionable insights for team management, supporting decisions on player development and game strategy.

Despite these promising results, our analysis has limitations, such as data quality and the challenge of class imbalance in MVP prediction. Future work could explore additional predictors, more recent data, and advanced techniques to address these issues.

In conclusion, our models effectively predict NBA player performance and potential MVP status, meeting our initial objectives. The insights gained from this analysis provide a solid foundation for understanding player performance and offer valuable guidance for NBA team management.

7. GenAI Usage

Throughout our analysis, ChatGPT provided invaluable support for coding and plotting challenges. It particularly crucial when working with packages like mgcv for fitting Generalized Additive Models (GAMs). ChatGPT provided clear, useful guidance on implementing such ML techniques where appropriate. It also offered solutions for visualizing non-linear trends and interpreting model outputs, which enhanced our analysis and presentation of findings.

While using AI like ChatGPT offered substantial benefits, it required careful attention to ensure accuracy and relevance. We had to verify the provided code and adapt it to our specific dataset and objectives. Additionally, understanding the context of suggestions and critically evaluating their applicability was important to maintain quality of our analysis. Despite this, the support from ChatGPT nicely augmented our project, making complex tasks more manageable and enhancing the overall quality of our work.