A Comparison of the Ability of Regression Models to Predicting Total Points BY NBA Player

Professor - Aaron Nielsen

STAT 451 - Sports Statistics and Analytics II Spring 2024

Colorado State University

May 3, 2024

By Alyaqadhan Al-Fahdi

Table of Contents

1 Abstract

In this project, we looked at a data from NBA Statistics in 2023 to see if we could predict how many points players would score in the next season. We used a smart statistics trick called Principal Component Analysis to make the data easier to work with by focusing on the most important factors affecting the total points. Then, we tried out different ways of predicting scores, like making many decision trees with Random Forest, looking at the nearest neighbors with KNN, and drawing the best line with SVM. In the end, the SVM way was the best at predicting the player's scores. The SVM model outperformed others, achieving the lowest RMSE (0.088) and the highest R-squared value (0.991), indicating its superior predictive ability. These results show the possibility of using statistics and machine learning to predict sports performance and provide insight into the significance of proper data processing and modeling selection in predicting athletic performance. The work will be beneficial to teams and sports analysts in their decision-making process that is based on data.

2 Introduction

In the world of professional basketball, particularly the NBA, being able to predict how well players will perform is very important. This not only helps teams to strategize and win games, but also keeps fans engaged and helps those playing fantasy sports to make better decisions. The goal of this project is to explore whether data from past performances can be used to forecast how many points NBA players will score in future games. The main source of data for this study will be the NBA Players Stats for the 2023 season, available on Kaggle (NBA Players Stats (2023 Season), 2023) The following study uses regression methods to evaluate how players have performed to accurately predict their future scoring. These regression methods are compared to find which are most efficacious based on accuracy by The Root Mean Squared Error (RMSE) and R-squared that is a statistical measure that represents the proportion of the variance for a dependent variable that is explained by an independent variable.

3 The Data

This dataset provides comprehensive statistics for each player, covering various aspects of their performance across 539 players and 30 different predictors. These predictors include player names, positions, teams, age, and detailed in-game statistics such as field goals made, three-point shot statistics, rebounds, assists, and more. There are three categorical variables that are player names, positions and team, and the rest of them are countians variables. The main focus will be on predicting the 'Total Points' scored by the players. To address the issue of missing data, specifically player positions, this study will verify and fill in gaps using reliable online sources (Code section in Appendix).

4 Exploratory Data Analysis

Before we start making predictions, it is important to first look closely at the data we have from NBA players during the 2023 season. Exploratory data analysis helps us understand what the data looks like and what it tells us. We will look at different types of charts and graphs to see trends and patterns. This step is crucial because it helps us make sure our data is good and gives us ideas about what might affect a player's scoring. First, we are interested in the correlation between the continuous predictor variables and response variable (Total Points).

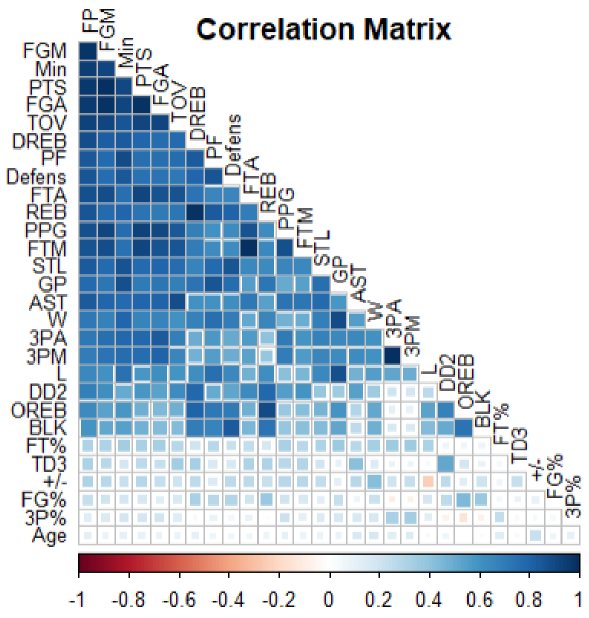


Figure 1

The correlation matrix (Figure 1) shows that most variables have highly correlated between them and the Total Points due to the count variables. For the categorical variables we can see if the different factors have different averages of the Total Points.

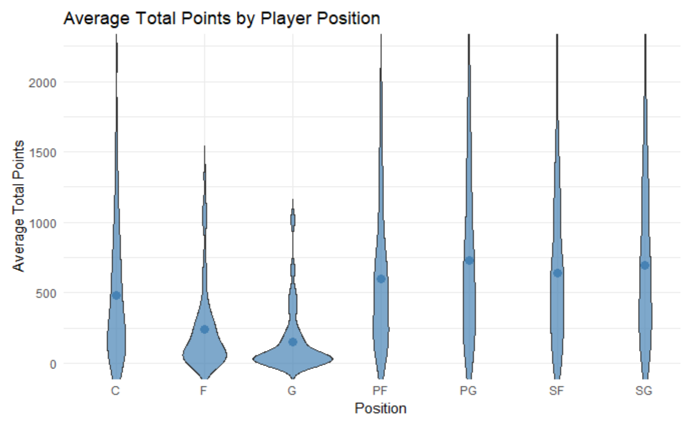


Figure 2

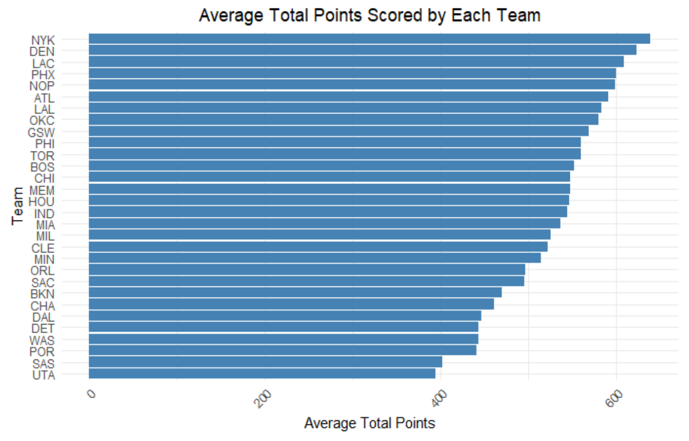


Figure 3

As we see from there the violin plot (Figure 2) and the horizontal bar chart (Figure 3), there are different on averages total points per different positions and different teams, but there is overlapping a lot on them, and this may make the models have high error. So, from the exploratory data analysis, it is very difficult to select main effect on the total points. Due to the highly correlated on the continuous variables and no significant different on average points by different teams and different position.

5 Principal Component Analysis (PCA)

Principal component analysis is a way to make a big set of data simpler to use. It works by taking lots of information and squishing it down into just a few pieces that still have most of the important stuff. This is helpful when we have tons of details but want to focus on just the key parts. Employing PCA is particularly useful not only for simplifying data but also for selecting features by clearing those that are highly correlated, especially with continuous variables, therefore enhancing the efficiency and performance of predictive models. (Jaadi, 2024)

By removing variables that are smaller correlation with total points and removing the categorical variables (Teams, Positions), there are still a problem that select the variables from highly correlated continuous variables. PCA is an important method here to select the variables. We divide data into x (games played, victories, time on court, field goals, three-point attempts and successes, defensive contributions, and several other relevant statistics) and y (Total points) and apply the PCA on the x dataset with standardization.

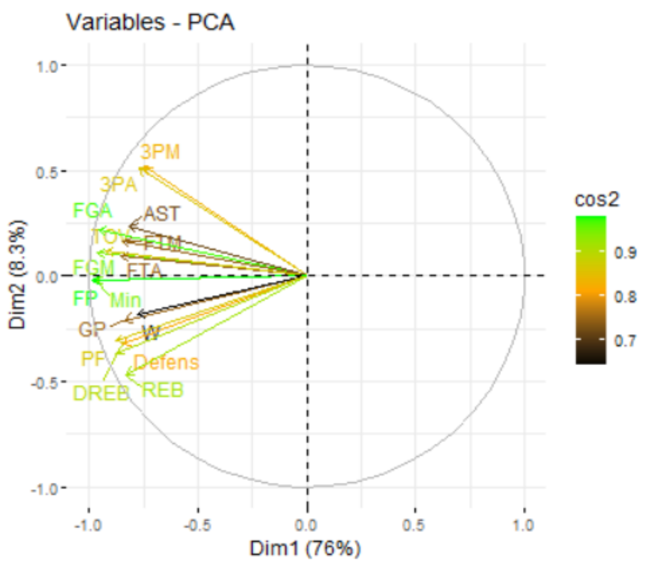


Figure 4

The PCA biplot (Figure 4) provided demonstrates how various basketball statistics are associated with the two main components that were extracted from the data and the squared cosine (cos2), which shows the quality of the representation of the variables on the principal components. These components are combinations of the original variables that represent new axes of maximum variance. The first component (Dim1) accounts for 76% of the data's variability, indicating it captures a large part of the information. The second component (Dim2) captures another 8.3%. The variables field goals attempted (FGA), minutes played (Min), field goals made (FGM), total rebounds (REB), and defensive rebounds (DREB) are all indicated by vectors pointing in a similar direction on the left side of the plot. Their location and directionality suggest they share a strong relationship and add significantly to the first principal component.

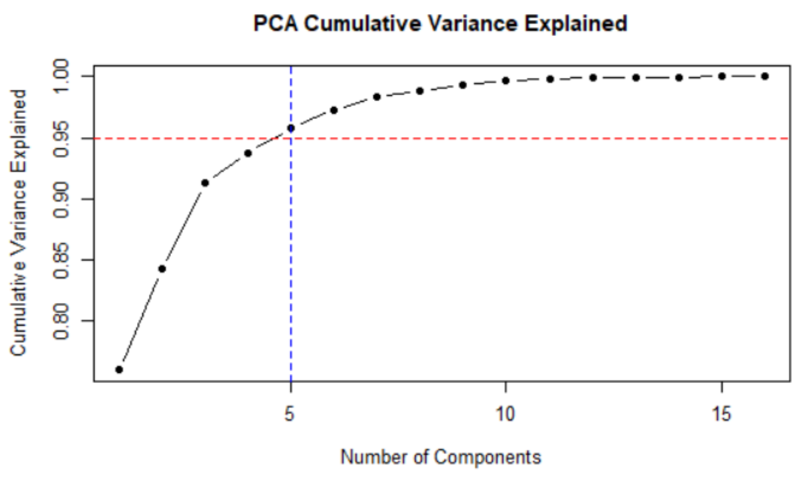


Figure 5

The chart (Figure 5) shows the cumulative variance explained by the principal components of a PCA. The x-axis indicates the number of components, while the y-axis shows the cumulative percentage of the variance explained. Around 5 principal components are sufficient to explain 95% of the variance, as the curve flattens out beyond this point, indicating that additional components contribute less to explaining the data. From (Table 3 in the Appendix), the first principal component (PC1) shows strong negative loadings across multiple variables, indicating a general inverse relationship. The second principal component (PC2) is heavily influenced by three-point shooting metrics such as three-point makes (3PM) and attempts (3PA), and it shows a negative correlation with defensive metrics like defensive rebounds (DREB) and total rebounds (REB). The third component (PC3) features positive loadings for wins (W) and strong negative loadings for free throws made (FTM) and attempted (FTA), suggesting an emphasis on successful outcomes that exclude free throws. The fourth component (PC4) is distinguished by a significant positive loading on assists (AST), highlighting teamwork and passing efficiency. Finally, the fifth component (PC5) presents negative loadings for wins (W) alongside a mixed set of positive and negative loadings on other variables, reflecting a complex interaction of factors affecting game outcomes.

6 Models and Results

6.1 Models Setup

To ensure a thorough and methodical approach to our predictive modeling, we took the following steps for each of the models we employed:

- Data Splitting: Divide our dataset into two parts using the subset that includes around 5 principal components. 80% of the data was used as the training set, which is the portion of the data that the models learn from. The remaining 20% was reserved as the test set, which we used to evaluate the performance of the models.
- Cross-Validation and Hyperparameter Tuning: For each model, we applied cross-validation techniques, which split the training set into smaller parts to train and validate the model multiple times. This process ensures that the model is stable and performs well not just on one subset of the data but across various subsets. Additionally, we applied a grid search approach for hyperparameter tuning. This means working through multiple combinations of the parameters that dictate the model's structure and learning process, seeking to find the combination that results in the best predictive performance.
- Evaluation Metrics: After training the models and tuning their hyperparameters, we evaluated their performance on the test set using two key metrics: the Root Mean Squared Error (RMSE) and the Coefficient of Determination (R^2). The RMSE gives us a measure of the average error between the model's predictions and the actual outcomes, so lower values indicate a better fit. The R^2 metric indicates the proportion of variance in the dependent variable that is predictable from the independent variables, a higher R^2 suggests a model that accounts for a greater portion of the variance.

6.2 Linear Regression

Linear regression is like finding the best-fitting line through a set of points on a graph. It predicts one thing (like how many points an NBA player will score) based on other things we know (like minutes played and shots taken). The line is drawn in such a way that the differences between the actual points and the line are as small as possible. There are some assumptions we should care to get best firring tine from linear regression, including the Variances of error terms (residuals) should be consistent across independent variables and Normal distributions are assumed for the residuals (the differences between the observed and predicted values) (Statistics Solutions, n.d).

6.2.1 Result from Linear Regression

When we first tried predicting NBA player points using linear regression, the model had some issues. The QQ plot (Figure 8 in Appendix) showed that the data had more extreme values than normal, and the plot of the errors (Figure 7 in Appendix) looked like a cone, meaning the model wasn't consistent across all values. These problems can make the predictions less reliable. Despite this, our model's R-squared 0.088 was very high, suggesting it was mostly on track, and the RMSE 0.99 was low, meaning the predictions were not too far off from the real scores. We applied a logarithmic transformation to the target variable (points scored) to address the assumption violations observed previously. However, it seems the assumptions were still not fully met (Figures 9 and 10 in Appendix). Despite this, the model's R-squared 0.596 remained high. The RMSE 0.758 increased compared to our first model, indicating that the average difference between the predicted and actual scores was larger.

6.3 Random Forest (RF)

Random Forest is a powerful machine learning modeling technique that works by building a 'forest' of many deciding trees. Each tree is created from a random selection of data points and variables and makes its own predictions. The final output of the Random Forest model is the average of all the trees' predictions, which tends to be more accurate and robust than any single tree (IBM, n.d).

6.3.1 Result from Random Forest

Random Forest model to predict NBA player points, using cross-validation and grid search to optimize its settings. The model training involved setting up a control with five-fold cross-validation and testing different settings for the number of predictors. After training with these parameters and building 100 trees, we evaluated the model's effectiveness on a test dataset. The model demonstrated excellent predictive accuracy, with a very low Root Mean Squared Error (RMSE) of 0.136 and a high R-squared value of 0.979.

6.4 K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) is an easy machine learning method that predicts the outcome for a new data point based on the most common outcomes among its nearest neighbors in the dataset. When deciding the prediction for a new point, KNN looks at the 'C' closest neighbors from the data it has been trained on. The final prediction is typically the average for regression (Singh, 2024).

6.4.1 Result from K-Nearest Neighbors:

For the K-Nearest Neighbors (KNN) model, I set up a training control with five-fold cross-validation and performed a grid search to fine-tune the hyperparameters, specifically testing different numbers of neighbors (k values of 1, 5, 10, 15, 2. After training the KNN model using these parameters, we predicted NBA player points on a test dataset. The model achieved a Root Mean Squared Error (RMSE) of 0.13266 and an R-squared value of 0.98193.

6.5 Support Vector Machine (SVM)

Support Vector Machine (SVM) with a linear kernel is a type of machine learning algorithm that tries to find the best boundary (hyperplane) that divides different predicted values in a dataset. It does this by maximizing the margin, which is the distance between the hyperplane and the nearest data points from each value. In the context of regression, like predicting NBA player points, the linear SVM aims to fit the best flat line that minimizes errors.

6.5.1 Result from Support Vector Machine

For the Support Vector Machine (SVM) with a linear kernel, we used a training approach that involved cross-validation with five folds and a grid search to optimize the C parameter (regularization strength), testing values of 0.1, 1, 10, 100, and 150. This setup helps determine the best configuration to minimize overfitting while maximizing prediction accuracy. After setting up with C (1) and training the SVM model (Table 4 in Appendix), we evaluated its performance on a test set. The model exhibited excellent predictive accuracy, achieving a Root Mean Squared Error (RMSE) of approximately 0.08889 and an R-squared value of 0.9911.

7 Comparing the Models

Table 1: Comparison of Model Performances

	Performance Metrics	
Model	**RMSE**	**R-squared**
Linear Regression	0.758	0.596
Random Forest	0.136	0.979
K-Nearest Neighbors	0.132	0.982
Support Vector Machine	**0.088**	**0.991**

Note: The RMSE and R-squared are calculated in standardizing output

From table 1, the Support Vector Machine (SVM) model shows the best performance with the lowest RMSE and the highest R-squared value, implying it has the highest predictive accuracy and consistency among the evaluated models. Linear Regression has the highest RMSE, suggesting less accuracy in its predictions, as a relatively high R-squared value. The Random Forest and K-Nearest Neighbors models also show strong performance, with low RMSE.

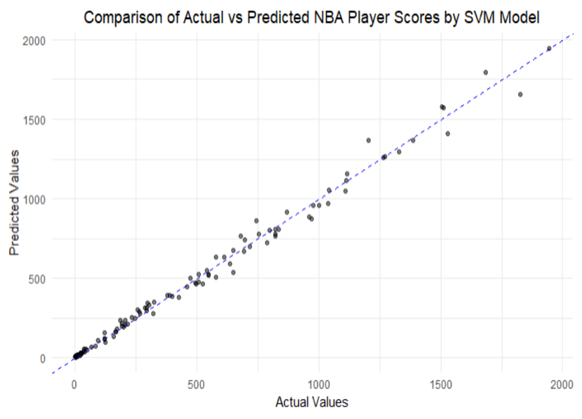
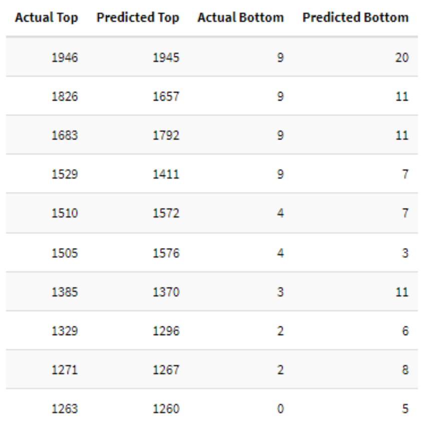


Figure 6

Table 2 : Side-by-Side Comparison of Actual vs Predicted NBA Player Scores by SVM Model



Actual Top	Predicted Top	Actual Bottom	Predicted Bottom
1946	1945	9	20
1826	1657	9	11
1683	1792	9	11
1529	1411	9	7
1510	1572	4	7
1505	1576	4	3
1385	1370	3	11
1329	1296	2	6
1271	1267	2	8
1263	1260	0	5

From the scatter plot (Figure 6) and Table 2, the SVM model shows a strong capability in predicting NBA player scores. The scatter plot shows a close straight between predicted and actual values, and the table 2 offers a detailed numerical comparison, further confirming the model's accuracy. Overall, the SVM's predictions closely match the actual scores, with both visual and tabular evidence suggesting that the model is effective and reliable for this task.

8 Conclusion

In conclusion, this report has explored the application of various statistical and machine learning methods to predict NBA player performance (total points scored). The study drew on an extensive dataset from the 2023 NBA season, navigating through initial data challenges such as the need for dimensionality reduction. Exploratory Data Analysis revealed complex patterns in player scoring, with no clear distinction based on positions or team performance and continuous statistics. The use of PCA proved valuable in reducing these complexities into principal components that captured the most important variance in the dataset. SVM is more Effective than other models. The Support Vector Machine with a linear kernel showed up as the most effective model with an RMSE of 0.088 and an R-squared of 0.991. The results highlight the value of using advanced statistical techniques in sports analytics by providing information that may influence fan interaction and team tactics. They also stress how crucial it is to choose the right model and prepare data appropriately for prediction jobs.

9 Improvements/Future Research

In order to further improve forecast accuracy, we recommend the continued improvement of the SVM model with big data that contains more seasons and exploring the use of additional factors, such as player health for even more accurate predictions. Also, we can investigate more models like Neural Network.

10 References

Jaadi, Z. (2024, February 23). *A Step-by-Step Explanation of Principal Component Analysis (PCA)*. Built In. https://builtin.com/data-science/step-step-explanation-principal-component-analysis

NBA Players stats(2023 season). (2023, August 4). Kaggle. https://www.kaggle.com/datasets/amirhosseinmirzaie/nba-players-stats2023-season

NBA Player Stat Leaders, 2023-24 Postseason - ESPN. (n.d.). ESPN. https://www.espn.com/nba/stats

Singh, A. (2024, February 13). *KNN algorithm: Introduction to K-Nearest Neighbors Algorithm for Regression*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2018/08/k-nearest-neighbor-introduction-regression-python/

Statistics Solutions. (2024, April 3). *Assumptions of multiple linear regression - Statistics solutions*. https://www.statisticssolutions.com/free-resources/directory-of-statistical-analyses/assumptions-of-multiple-linear-regression/

What is Random Forest? | IBM. (n.d.). https://www.ibm.com/topics/random-forest

11 Appendix

Figure 7

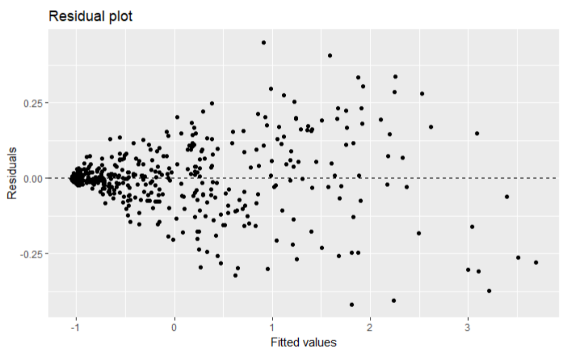


Figure 8

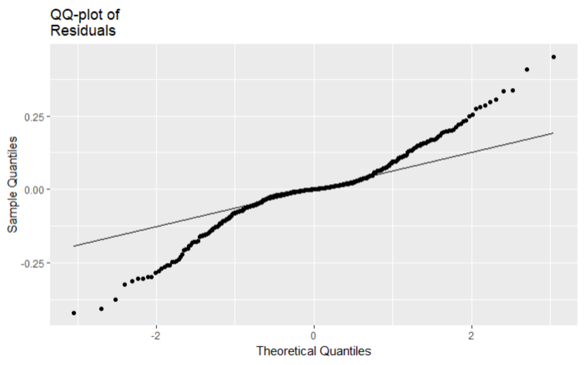


Figure 9

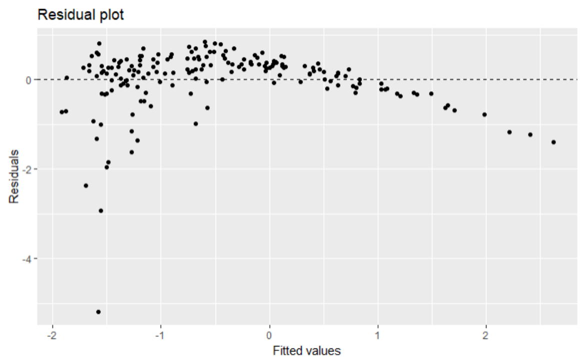
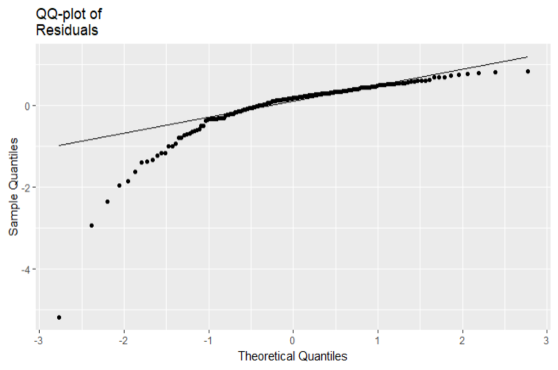


Figure 10



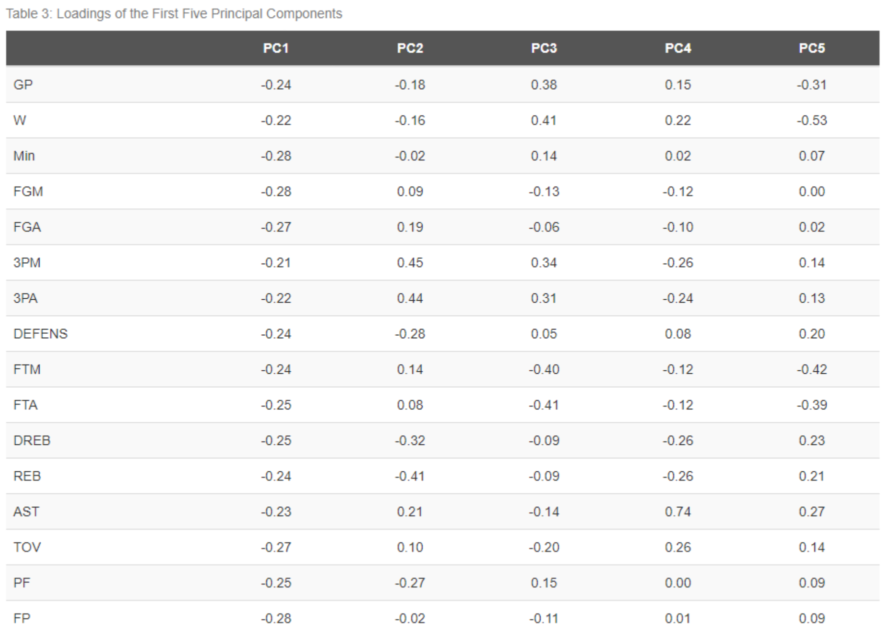
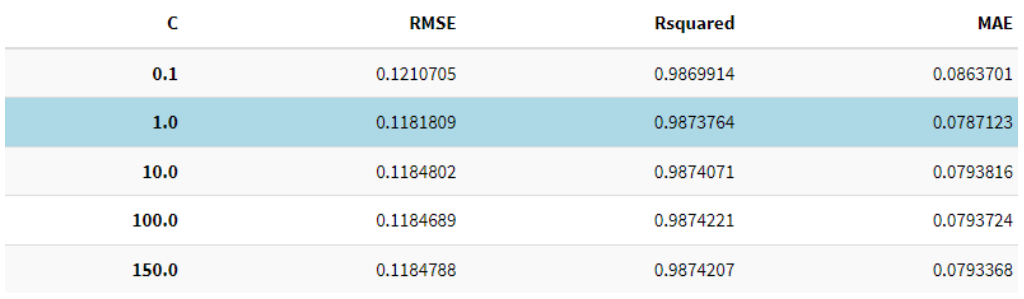


Table 3: Loadings of the First Five Principal Components

	PC1	PC2	PC3	PC4	PC5
GP	-0.24	-0.18	0.38	0.15	-0.31
W	-0.22	-0.16	0.41	0.22	-0.53
Min	-0.28	-0.02	0.14	0.02	0.07
FGM	-0.28	0.09	-0.13	-0.12	0.00
FGA	-0.27	0.19	-0.06	-0.10	0.02
3PM	-0.21	0.45	0.34	-0.26	0.14
3PA	-0.22	0.44	0.31	-0.24	0.13
DEFENS	-0.24	-0.28	0.05	0.08	0.20
FTM	-0.24	0.14	-0.40	-0.12	-0.42
FTA	-0.25	0.08	-0.41	-0.12	-0.39
DREB	-0.25	-0.32	-0.09	-0.26	0.23
REB	-0.24	-0.41	-0.09	-0.26	0.21
AST	-0.23	0.21	-0.14	0.74	0.27
TOV	-0.27	0.10	-0.20	0.26	0.14
PF	-0.25	-0.27	0.15	0.00	0.09
FP	-0.28	-0.02	-0.11	0.01	0.09

Table 4:Support Vector Machines with Linear Kernel - Summary



C	RMSE	Rsquared	MAE
0.1	0.1210705	0.9869914	0.0863701
1.0	0.1181809	0.9873764	0.0787123
10.0	0.1184802	0.9874071	0.0793816
100.0	0.1184689	0.9874221	0.0793724
150.0	0.1184788	0.9874207	0.0793368

11.1 The Code

```r
library(broom)
library(tidyverse)
library(ellipse)
library(RColorBrewer)
library(corrplot)
library(GGally)
library(reshape2)
library(fastDummies)
library(caret)
library(kernlab)
library(tibble)


nba_data <- read_csv("data/2023_nba_player_stats.csv")
```

In the DataFrame , there is a column named ‘Position’ that represents the position of each player. However, some rows in the ‘Position’ column have missing values (na). The goal of this code is to fill these missing values with the position.

From ESPN (NBA Player Stat Leaders, 2023-24 Postseason - ESPN, n.d.)

Alondes williams = SG Deonte burton = SF Frank Jackson = G Michael Foster Jr.= F Sterling Brown = SF

```r
nba_data <- nba_data %>%
  mutate(POS = case_when(
    PName == "Alondes Williams" ~ "SG",
    PName == "Deonte Burton" ~ "SF",
    PName == "Frank Jackson" ~ "G",
    PName == "Michael Foster Jr." ~ "F",
    PName == "Sterling Brown" ~ "SF",
    TRUE ~ POS
  ))

#####EDA
nba_data <- nba_data %>%
  mutate(PPG = PTS / GP)

nba_data <- nba_data %>%
  mutate(Defens = BLK + STL)

nba_summary <- nba_data %>%
  group_by(POS) %>%
  summarise(
    Mean_PTS = mean(PTS, na.rm = TRUE),
    SE = sd(PTS, na.rm = TRUE) / sqrt(n())
  )

a <- ggplot(nba_data, aes(x = POS, y = PTS)) +
  geom_violin(trim = FALSE, alpha = 0.7, fill = "steelblue") + # Adjus
ted fill color here
  geom_point(data = nba_summary, aes(x = POS, y = Mean_PTS), color = "
steelblue", size = 3) +
  coord_cartesian(ylim = c(0, max(nba_data$PTS))) + # Ensure y-axis st
arts at 0
  labs(title = "Average Total Points by Player Position", x = "Positio
n", y = "Average Total Points") +
  theme_minimal() +
  theme(legend.position = "none")

b <- nba_data %>%
  top_n(10, PTS) %>%
  ggplot(aes(x = reorder(PName, FP), y = FP, fill = FP)) +
  geom_col(fill = "steelblue") +
  coord_flip() +
  labs(title = "Top 10 Players by Total Points", x = "", y = "Total Po
ints") +
  theme_minimal() +
  theme(legend.position = "none")

best_defending_players <- nba_data %>%
  arrange(desc(Defens)) %>%
  slice(1:10)
```

```
c<-ggplot(best_defending_players, aes(x = reorder(PName, Defens), y =
Defens, fill = Defens)) +
  geom_col(fill = "steelblue") +
  coord_flip() +
  labs(title = 'Top 10 Best Defending Players',
       x = 'Player Name',
       y = 'Defensive Performance (Combined Blocks and Steals)') +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5),
        text = element_text(size = 12),
        axis.title = element_text(color = "#333333"),
        axis.text = element_text(color = "#333333")) +
  theme(legend.position = "none")

team_avg_points <- nba_data %>%
  group_by(Team) %>%
  summarise(Avg_Total_Points = mean(PTS, na.rm = TRUE)) %>%
  mutate(Team = reorder(Team, Avg_Total_Points))


d<-ggplot(team_avg_points, aes(x = Team, y = Avg_Total_Points)) +
  geom_col(fill = "steelblue") +
  coord_flip() +
  labs(title = "Average Total Points Scored by Each Team",
       x = "Team",
       y = "Average Total Points") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        plot.title = element_text(hjust = 0.5))

data <- nba_data %>%
  select(Age, GP, W, L, Min, PTS, FGM, FGA, PPG, Defens,
         `FG%`, `3PM`, `3PA`, `3P%`, FTM, FTA, `FT%`, OREB,
         DREB, REB, AST, TOV, STL, BLK,
         PF, FP, DD2, TD3, `+/-`) %>%
  cor()

e<-corrplot(data, method = 'square', order = 'FPC', type = 'lower', di
ag = FALSE, tl.col = "black", tl.cex = 0.8)
title("Correlation Matrix", col.main = "black", font.main = 2)

sel_data <- nba_data %>%
  select(GP, W, Min, POS, FGM, FGA, `3PM`, `3PA`,Defens, FTM, FTA,
         DREB, REB, AST, TOV, PF, FP)
```

```
X <- sel_data %>%
  select(-POS)

Y <- nba_data %>%
  select(PTS)

####PCA

set.seed(451)
pca_result <- prcomp(X, scale. = TRUE)

library(factoextra)
fviz_eig(pca_result, addlabels = TRUE)

fviz_pca_var(pca_result, col.var = "cos2",
            gradient.cols = c("black", "orange", "green"),
            repel = TRUE)

fviz_cos2(pca_result, choice = "var", axes = 1:2)

explained_variance <- summary(pca_result)$importance[2,]
cumulative_variance <- cumsum(explained_variance)

num_pcs <- which.min(abs(cumulative_variance - 0.95))
print(num_pcs)

## PC5
##   5
plot(cumulative_variance, xlab = "Number of Components", ylab = "Cumul
ative Variance Explained",
     type = 'b', pch = 19, main = "PCA Cumulative Variance Explained")
abline(h = 0.95, col = "red", lty = 2)
abline(v = num_pcs, col = "blue", lty = 2)

X_pca <- pca_result$x
#y <- as.factor(y)

X_pca_selected <- X_pca[, 1:5]

y_scaled <- scale(Y)

### Split data

set.seed(451)
train_index <- createDataPartition(y_scaled, p = 0.8, list = FALSE)
x_train <- X_pca_selected[train_index, ]
y_train <- y_scaled[train_index]
x_test <- X_pca_selected[-train_index, ]
y_test <- y_scaled[-train_index]
```

```
train_control <- trainControl(method = "none")

set.seed(451)

LM


model <- train(x =x_train, y = log(y_train) , method = "lm", trControl
= train_control)

predictions <- predict(model, newdata = x_test)

predictions_exp <- exp(predictions)


rmse <- sqrt(mean((predictions_exp - y_test)^2))
paste("RMSE:", rmse)

## [1] "RMSE: 0.75823714773631"

rsq <- cor(predictions_exp, y_test)^2
paste("R-squared:", rsq)

## [1] "R-squared: 0.596080391206214"

mean_y <- mean(Y$PTS, na.rm = TRUE)
sd_y <- sd(Y$PTS, na.rm = TRUE)

preds = (predictions * sd_y) + mean_y
ys = (y_test * sd_y) + mean_y

residuals <- data.frame(x = fitted(model), y = residuals(model))


ggplot(residuals, aes(x, y)) +
geom_point() +
geom_hline(yintercept = 0, linetype = "dashed") +
labs(x = "Fitted values", y = "Residuals", title = "Residual plot")

ggplot(residuals, aes(sample = y)) +
stat_qq() +
stat_qq_line() +
labs(x = "Theoretical Quantiles", y = "Sample Quantiles", title = "QQ-
plot of
Residuals")

mean_y <- mean(Y$PTS, na.rm = TRUE)
sd_y <- sd(Y$PTS, na.rm = TRUE)
```

```
preds = (predictions * sd_y) + mean_y
ys = (y_test * sd_y) + mean_y
```

RF

```
train_control <- trainControl(method = "cv", number = 5, search = "gri
d")

grid <- expand.grid(mtry = c(2, 4,6,8))

set.seed(451)
model_rf <- train(x = x_train, y = y_train,
                  method = "rf",
                  trControl = train_control,
                  tuneGrid = grid,
                  ntree = 100)
print(model_rf)
```

```
## Random Forest
##
## 432 samples
##   5 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 346, 346, 344, 348, 344
## Resampling results across tuning parameters:
##
##   mtry  RMSE       Rsquared   MAE
##   2     0.1897127  0.9678888  0.1255489
##   4     0.1709506  0.9744032  0.1117290
##   6     0.1778848  0.9725316  0.1155084
##   8     0.1756556  0.9732022  0.1151347
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 4.
```

```
print(model_rf)
```

```
## Random Forest
##
## 432 samples
##   5 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 346, 346, 344, 348, 344
## Resampling results across tuning parameters:
##
##   mtry  RMSE       Rsquared   MAE
##   2     0.1897127  0.9678888  0.1255489
##   4     0.1709506  0.9744032  0.1117290
##   6     0.1778848  0.9725316  0.1155084
##   8     0.1756556  0.9732022  0.1151347
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 4.
```

```
predictions <- predict(model_rf, newdata = x_test)

rmse <- sqrt(mean((predictions - y_test)^2))
paste("RMSE:", rmse)
```

```
## [1] "RMSE: 0.136249304799215"
```

```
rsq <- cor(predictions, y_test)^2
paste("R-squared:", rsq)
```

```
## [1] "R-squared: 0.979253949497207"
```
KNN

```
train_control <- trainControl(method = "cv", number = 5,search = "grid
")

grid <- expand.grid(k = c(1, 5, 10, 15, 20))
set.seed(451)
model_knn <- train(x = x_train, y = y_train,
                   method = "knn",
                   trControl = train_control,
                   tuneGrid = grid)

print(model_knn)
```

```
## k-Nearest Neighbors
##
## 432 samples
##   5 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 346, 346, 344, 348, 344
## Resampling results across tuning parameters:
##
##   k   RMSE       Rsquared   MAE
##    1  0.2073122  0.9597071  0.1364664
##    5  0.1766299  0.9724308  0.1155679
##   10  0.1736043  0.9745579  0.1154963
##   15  0.1847388  0.9736747  0.1221354
##   20  0.1935363  0.9730734  0.1254518
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 10.
```

```
predictions <- predict(model_knn, newdata = x_test)

rmse <- sqrt(mean((predictions - y_test)^2))
paste("RMSE:", rmse)
```

```
## [1] "RMSE: 0.132658900987549"
```

```
rsq <- cor(predictions, y_test)^2
paste("R-squared:", rsq)
```

```
## [1] "R-squared: 0.981927735729792"
```

#svm

```
train_control <- trainControl(method = "cv", number = 5,search = "grid
")

grid <- expand.grid(C = c(0.1, 1, 10, 100,150))
set.seed(451)
model_svm_linear <- train(x = x_train, y = y_train,
                          method = "svmLinear",
                          trControl = train_control,
                          tuneGrid = grid)

print(model_svm_linear)
```

```
## Support Vector Machines with Linear Kernel
##
## 432 samples
##   5 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 346, 346, 344, 348, 344
## Resampling results across tuning parameters:
##
##   C       RMSE       Rsquared   MAE
##     0.1  0.1210705  0.9869914  0.08637013
##     1.0  0.1181809  0.9873764  0.07871229
##    10.0  0.1184802  0.9874071  0.07938162
##   100.0  0.1184689  0.9874221  0.07937243
##   150.0  0.1184788  0.9874207  0.07933682
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was C = 1.
```

```
predictions <- predict(model_svm_linear, newdata = x_test)

rmse <- sqrt(mean((predictions - y_test)^2))
paste("RMSE:", rmse)
```

```
## [1] "RMSE: 0.0888874448537297"
```

```
rsq <- cor(predictions, y_test)^2
paste("R-squared:", rsq)
```

```
## [1] "R-squared: 0.991108905007208"
```

```
mean_y <- mean(Y$PTS, na.rm = TRUE)
sd_y <- sd(Y$PTS, na.rm = TRUE)

preds = (predictions * sd_y) + mean_y
ys = (y_test * sd_y) + mean_y

results <- data.frame(Actual = ys, Predicted = round(preds))

library(broom)
library(kableExtra)
results %>% select(1:2) %>% slice(1:15) %>% kable(booktabs=T)

ggplot(results, aes(x = Actual, y = Predicted)) +
  geom_point(alpha = 0.5) +
  geom_abline(intercept = 0, slope = 1, linetype = "dashed", color = "
blue") +
  theme_minimal() +
  labs(title = "Comparison of Actual vs Predicted NBA Player Scores by
SVM Model",
       x = "Actual Values",
       y = "Predicted Values") +
  theme(plot.title = element_text(hjust = 0.5))
```