NBA Wage Prediction by Using Machine Learning

Code

NBA Wage Prediction

Data Scraping and Preprocessing

Hide

```
library(rvest)
library(dplyr)
library(randomForest)
library(rpart)
library(e1071)
library(gbm)
library(xgboost)
library(ggplot2)


# Specify the URL of the webpage to scrape
url <- "https://www.basketball-reference.com/leagues/NBA_2022_per_minute.html"

# Scrape the data from the webpage
page <- read_html(url)

# Extract the table containing the player statistics
table <- html_table(html_nodes(page, "table")[1])

# Convert the table to a data frame
player_stats_train <- as.data.frame(table)

# Print the resulting data frame
head(player_stats_train)

player_stats_train <- player_stats_train %>%
  filter(!is.na(as.numeric(Rk)))

player_stats_train <- player_stats_train %>%
  group_by(Player) %>%
  filter(!(Player %in% Player[duplicated(Player)]) | Tm == "TOT") %>%
  ungroup()


# Specify the URL of the webpage to scrape
url <- "https://hoopshype.com/salaries/players/2021-2022/"

# Scrape the data from the webpage
page <- read_html(url)

# Extract the table containing the wage statistics
table <- html_table(html_nodes(page, "table")[1])

# Convert the table to a data frame
wage_stats_train <- as.data.frame(table)

# Print the resulting data frame
print(wage_stats_train)

col_names <- wage_stats_train[1, ]
wage_stats_train <- wage_stats_train[-1, ]
colnames(wage_stats_train) <- col_names

# Display the first few rows of the modified wage_stats table
head(wage_stats_train)

wage_stats_train <- wage_stats_train[, c("Player", "2021/22")]

wage_stats_train$`2022/23` <- as.numeric(gsub("[$,]", "", wage_stats_train$`2021/22`))

# Merging data and wage_stats based on player names
train_data <- merge(player_stats_train, wage_stats_train, by.x = "Player", by.y = "Player", all.x = TRUE)


# Viewing the first few rows of the merged data
head(train_data)

# Renaming the variable from "2022/23" to "Wage"
train_data <- train_data %>%
  rename(Wage = `2022/23`)

# Viewing the first few rows of the updated merged data
head(train_data)

# Removing rows with NA in the "Wage" variable
train_data <- na.omit(train_data)

# Viewing the first few rows of the updated merged data
head(train_data)

# Select the desired columns from the original data
train_data <- train_data[c("Player", "Pos", "Age", "G", "GS", "MP", "FG", "FGA", "FG.", "X3P", "X3PA", "X3P.", "X
2P", "X2PA", "X2P.", "FT", "FTA", "FT.", "ORB", "DRB", "TRB", "AST", "STL", "BLK", "TOV", "PF", "PTS", "Wage")]

# Set the desired columns to numeric
```

```r
numeric_cols <- c("Age", "G", "GS", "MP", "FG", "FGA", "FG.", "X3P", "X3PA", "X3P.", "X2P", "X2PA", "X2P.", "FT",
"FTA", "FT.", "ORB", "DRB", "TRB", "AST", "STL", "BLK", "TOV", "PF", "PTS", "Wage")
train_data[numeric_cols] <- lapply(train_data[numeric_cols], as.numeric)

# Load the required libraries
library(rvest)
library(dplyr)

# Scrape data from "https://www.basketball-reference.com/leagues/NBA_2023_per_minute.html"
url1 <- "https://www.basketball-reference.com/leagues/NBA_2023_per_minute.html"

# Scrape the data from the webpage
page1 <- read_html(url1)

# Extract the table containing the player statistics
table1 <- html_table(html_nodes(page1, "table")[1])

# Convert the table to a data frame
player_stats_test <- as.data.frame(table1)

# Remove rows with missing rank (Rk) values
player_stats_test <- player_stats_test %>%
  filter(!is.na(as.numeric(Rk)))

# Group by player and filter duplicates based on player names and team (Tm) values
player_stats_test <- player_stats_test %>%
  group_by(Player) %>%
  filter(!(Player %in% Player[duplicated(Player)]) | Tm == "TOT") %>%
  ungroup()

# Remove rows with missing values
train_data <- na.omit(train_data)


# Scrape data from "https://hoopshype.com/salaries/players/"
url2 <- "https://hoopshype.com/salaries/players/"

# Scrape the data from the webpage
page2 <- read_html(url2)

# Extract the table containing the wage statistics
table2 <- html_table(html_nodes(page2, "table")[1])

# Convert the table to a data frame
wage_stats_test <- as.data.frame(table2)

# Modify the column names of the wage statistics table
col_names <- wage_stats_test[1, ]
wage_stats_test <- wage_stats_test[-1, ]
colnames(wage_stats_test) <- col_names

# Select the desired columns from the wage statistics table
wage_stats_test <- wage_stats_test[, c("Player", "2022/23")]

# Convert the "2022/23" column to numeric by removing commas and dollar signs
wage_stats_test$`2022/23` <- as.numeric(gsub("[$,]", "", wage_stats_test$`2022/23`))

# Merge player statistics and wage statistics based on player names
test_data <- merge(player_stats_test, wage_stats_test, by.x = "Player", by.y = "Player", all.x = TRUE)

# Rename the "2022/23" column to "Wage"
test_data <- test_data %>%
  rename(Wage = `2022/23`)

# Remove rows with missing wage values
test_data <- na.omit(test_data)

# Select the desired columns from the test data
test_data <- test_data[c("Player", "Pos", "Age", "G", "GS", "MP", "FG", "FGA", "FG.", "X3P", "X3PA", "X3P.", "X2
P", "X2PA", "X2P.", "FT", "FTA", "FT.", "ORB", "DRB", "TRB", "AST", "STL", "BLK", "TOV", "PF", "PTS", "Wage")]

# Set the desired columns to numeric
numeric_cols <- c("Age", "G", "GS", "MP", "FG", "FGA", "FG.", "X3P", "X3PA", "X3P.", "X2P", "X2PA", "X2P.", "FT",
"FTA", "FT.", "ORB", "DRB", "TRB", "AST", "STL", "BLK", "TOV", "PF", "PTS", "Wage")
test_data[numeric_cols] <- lapply(test_data[numeric_cols], as.numeric)

# Remove rows with missing values
test_data <- na.omit(test_data)


# Display the first few rows of the test data
head(test_data)
```

In this report, we are focusing on the task of predicting NBA player wages using machine learning techniques in R. To accomplish this, we have gathered data from the 2021/22 NBA season to serve as our training data, while the 2022/23 season data is used as our test data.

To obtain the necessary player statistics, we have utilized web scraping techniques. We scraped player statistics from the webpage '**https://www.basketball-reference.com/leagues/NBA_2022_per_minute.html**' using the 'rvest' and 'dplyr' packages. The data was extracted from the first table on the webpage, which contains per-minute statistics for NBA players.

After extracting the table, we converted it into a data frame for further analysis. To ensure data quality, we filtered out rows with missing rank (Rk) values. Additionally, we grouped the data by player and removed duplicates based on player names and team (Tm) values, keeping only the rows where the player's name is unique or the team is 'TOT' (indicating a player who played for multiple teams).

To complement the player statistics, we needed wage information for each player. We obtained this information from the webpage '**https://hoopshype.com/salaries/players/2021-2022/**' using web scraping techniques and the 'rvest' package. The wage statistics were extracted from the first table on the webpage and converted into a data frame.

Next, we performed data merging based on player names to combine the player statistics and wage information. This allowed us to match each player's performance data with their respective wage for the 2022/23 season. Any rows with missing wage values were removed from the dataset to ensure data integrity.

After merging the data, we selected the relevant columns for our analysis, including player attributes such as age, games played (G), minutes played (MP), field goals (FG), three-pointers (X3P), and more. We converted these selected columns to numeric data type for further analysis.

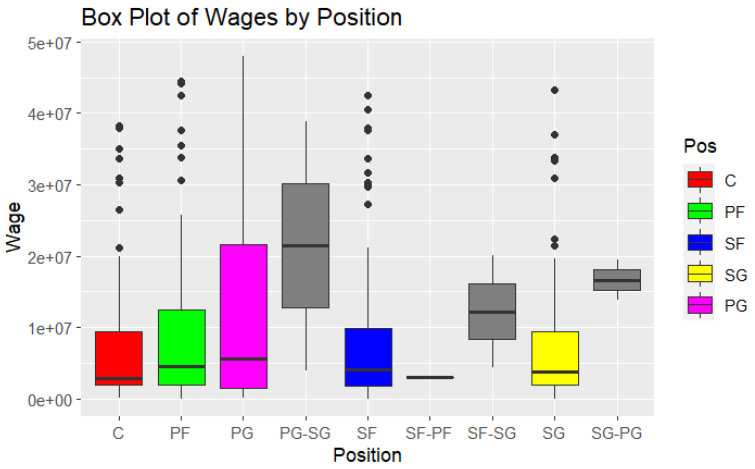
For the test data, we followed a similar process. We scraped player statistics from the webpage '**https://www.basketball-reference.com/leagues/NBA_2023_per_minute.html**' and wage statistics from the webpage '**https://hoopshype.com/salaries/players/**'. We performed data cleaning and merging steps as done for the training data.

In summary, we have successfully scraped and preprocessed the data for our NBA wage prediction task. The training data consists of player statistics from the 2021/22 season, merged with wage information for the 2022/23 season. The test data comprises player statistics from the 2022/23 season, merged with wage information as well. These datasets are now ready for further analysis and modeling using machine learning techniques in R.

Exploration of NBA Player Performance and Wages for the 2022/23 Season

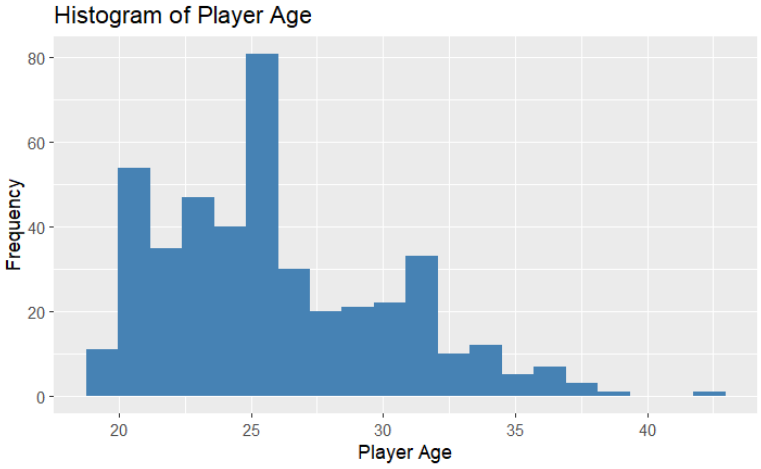
Hide

```
library(ggplot2)

# Define color palette for different positions
pos_colors <- c("C" = "#FF0000", "PF" = "#00FF00", "SF" = "#0000FF", "SG" = "#FFFF00", "PG" = "#FF00FF")

# Create a box plot chart for wages grouped by position with different colors
ggplot(test_data, aes(x = Pos, y = Wage, fill = Pos)) +
  geom_boxplot() +
  scale_fill_manual(values = pos_colors) +
  labs(x = "Position", y = "Wage") +
  ggtitle("Box Plot of Wages by Position")
```



Hide

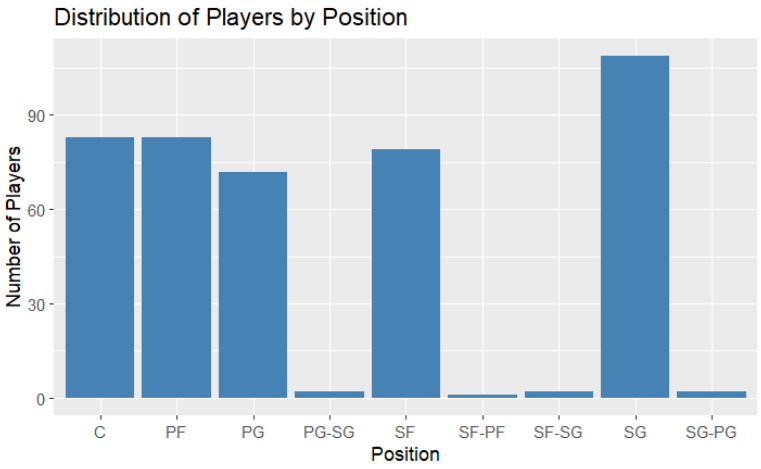
```
library(ggplot2)

ggplot(test_data, aes(x = Age)) +
  geom_histogram(fill = "steelblue", bins = 20) +
  labs(x = "Player Age", y = "Frequency") +
  ggtitle("Histogram of Player Age")
```



Hide

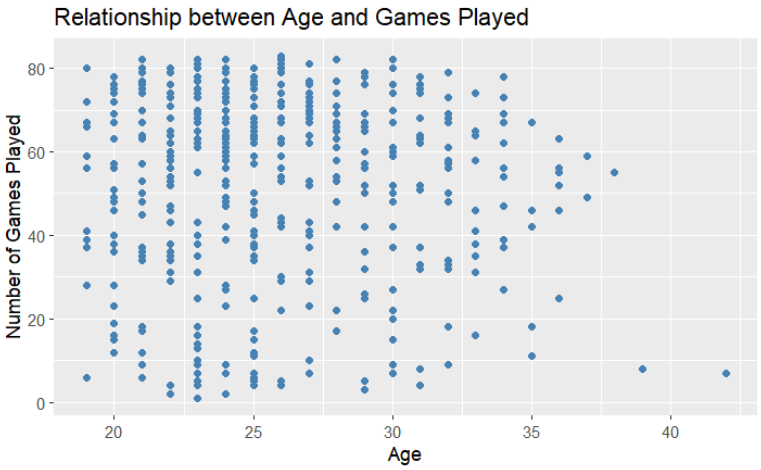
```
library(ggplot2)
```

```
# Create the bar chart
ggplot(test_data, aes(x = Pos)) +
  geom_bar(fill = "steelblue") +
  labs(x = "Position", y = "Number of Players") +
  ggtitle("Distribution of Players by Position")
```



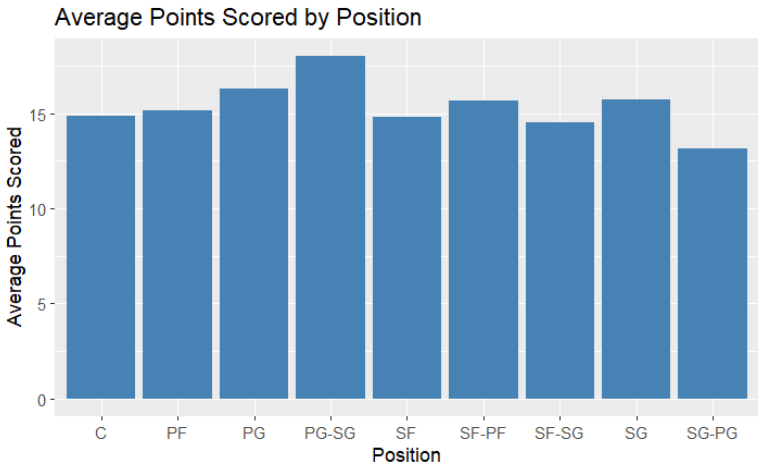
Hide

```
library(ggplot2)

# Create the scatter plot
ggplot(test_data, aes(x = Age, y = G)) +
  geom_point(color = "steelblue") +
  labs(x = "Age", y = "Number of Games Played") +
  ggtitle("Relationship between Age and Games Played")
```



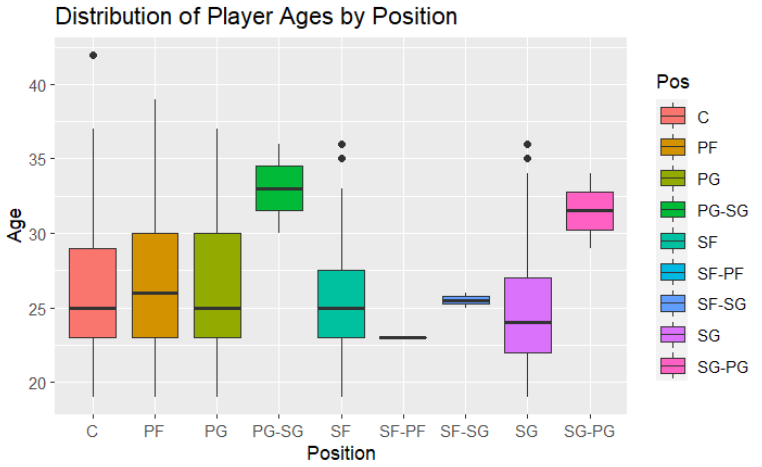
Hide

```
library(ggplot2)

# Create the bar chart
ggplot(test_data, aes(x = Pos, y = PTS)) +
  geom_bar(stat = "summary", fun = "mean", fill = "steelblue") +
  labs(x = "Position", y = "Average Points Scored") +
  ggtitle("Average Points Scored by Position")
```



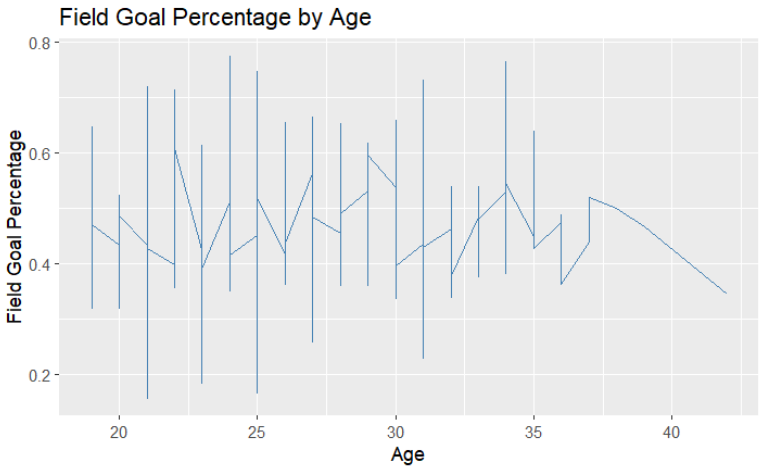
Hide

```
library(ggplot2)

# Create the box plot
ggplot(test_data, aes(x = Pos, y = Age, fill = Pos)) +
  geom_boxplot() +
  labs(x = "Position", y = "Age") +
  ggtitle("Distribution of Player Ages by Position")
```



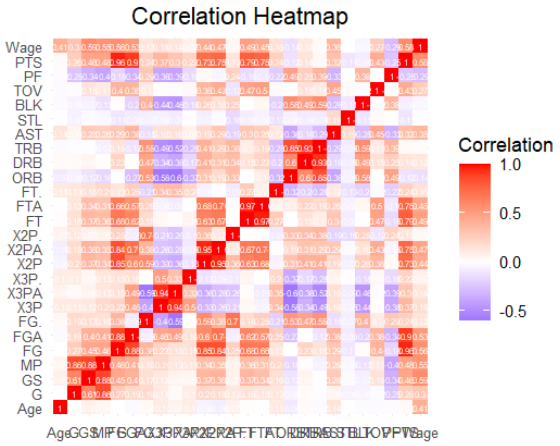
Hide

```
library(ggplot2)

# Create the line chart
ggplot(test_data, aes(x = Age, y = FG., group = 1)) +
  geom_line(color = "steelblue") +
  labs(x = "Age", y = "Field Goal Percentage") +
  ggtitle("Field Goal Percentage by Age")
```



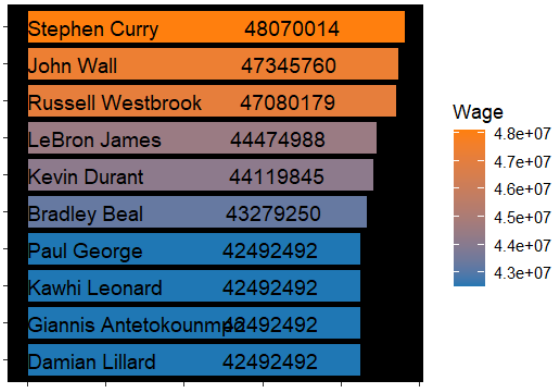
Hide

```r
library(ggplot2)

# Calculate the correlation matrix
cor_matrix <- cor(test_data[, -c(1, 2)])  # Exclude the first and second columns

# Create a data frame of the correlation matrix with row and column names
corr_df <- as.data.frame(as.table(cor_matrix))
names(corr_df) <- c("Variable 1", "Variable 2", "Correlation")

# Create the correlation heatmap
ggplot(corr_df, aes(x = `Variable 2`, y = `Variable 1`, fill = `Correlation`)) +
  geom_tile() +
  geom_text(aes(label = round(`Correlation`, 2)), color = "white", size = 1.6) +
  scale_fill_gradient2(low = "blue", mid = "white", high = "red", midpoint = 0, na.value = "grey50") +
  labs(title = "Correlation Heatmap", x = "", y = "") +
  theme_minimal() +
  theme(axis.text = element_text(size = 8), plot.title = element_text(hjust = 0.5)) +
  coord_equal()
```



Hide

```r
library(ggplot2)

# Sort the test_data dataframe by descending wage to get the top 10 highest-wage players
top_10_players <- test_data[order(-test_data$Wage), ][1:10, ]

# Create a ggplot chart with customized aesthetics
chart <- ggplot(top_10_players, aes(x = Wage, y = reorder(Player, Wage), fill = Wage)) +
  geom_bar(stat = "identity", color = "black") +
  geom_text(aes(label = Wage), hjust = -0.3, color = "black", size = 4, position = position_stack(vjust = 0.5)) +
  geom_text(aes(label = Player), color = "black", hjust = 0, size = 4, position = position_stack(vjust = 0.0)) +
  labs(x = "Wage", y = "Player", title = "Top 10 Highest-Paid NBA Players") +
  scale_fill_gradient(low = "#1f77b4", high = "#ff7f0e") +
  theme_dark() +
  theme(plot.title = element_text(hjust = 0.5, color = "white"),
        axis.text = element_text(color = "white"),
        axis.title = element_text(color = "white"),
        panel.background = element_rect(fill = "black"),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.border = element_blank())

# Print the chart
print(chart)
```



In this section, we explore the performance and wage data of NBA players for the 2022/23 season. The visualizations and analyses provide valuable insights into the latest trends and patterns in player performance and wages.

1. Box Plot of Wages by Position: This visualization presents a box plot showing the distribution of player wages across different positions. It helps identify any variations in wages between positions for the 2022/23 season.
2. Histogram of Player Age: The histogram displays the frequency distribution of player ages in the dataset. It provides an overview of the age distribution among NBA players for the current season.
3. Distribution of Players by Position: A bar chart showcases the number of players in each position for the 2022/23 season. This visualization helps understand the distribution of players across different positions.
4. Relationship between Age and Games Played: A scatter plot is used to examine the relationship between player age and the number of games played during the 2022/23 season. This visualization allows us to analyze the correlation between age and player activity for the current season.
5. Average Points Scored by Position: This bar chart illustrates the average points scored by players in each position for the 2022/23 season. It helps identify the positions with higher or lower average points scored during the current season.
6. Distribution of Player Ages by Position: A box plot is utilized to visualize the distribution of player ages for each position for the 2022/23 season. This plot provides insights into the age range and variability among players in different positions for the current season.
7. Field Goal Percentage by Age: This line chart displays the trend of field goal percentage across different ages for the 2022/23 season. It allows us to observe any patterns or trends in shooting accuracy based on player age during the current season.
8. Correlation Heatmap: A correlation heatmap showcases the correlation between various player attributes for the 2022/23 season. The color intensity represents the strength and direction of the correlation, providing insights into the relationships among different variables for the current season.
9. Top 10 Highest-Paid NBA Players: This visualization highlights the top 10 highest-paid NBA players for the 2022/23 season. A bar chart with customized aesthetics is used to compare player wages, with player names as labels.

These exploratory visualizations provide valuable insights into the performance and wages of NBA players for the latest season. They serve as a foundation for further analysis and modeling to understand the factors influencing player wages and performance in the NBA.

Machine Learning Models

Model 1 (Random Forest Regression Model)

Hide

```
set.seed(123)
# Select the variables for training
selected_variables <- c("Age", "G", "GS", "MP", "PTS", "TRB", "AST", "STL", "BLK", "TOV")
train_predictors <- train_data[, selected_variables]
train_target <- train_data$Wage

# Train the random forest model
rf_model_1 <- randomForest(train_predictors, train_target)

# Make predictions on the test data
test_predictors <- test_data[, selected_variables]
test_pred_1 <- predict(rf_model_1, newdata = test_predictors)

# Calculate MSE, RMSE, and R-squared
mse_1 <- mean((test_data$Wage - test_pred_1)^2)
rmse_1 <- sqrt(mse_1)
rsq_1 <- cor(test_data$Wage, test_pred_1)^2

# Print the results
cat("MSE (Model 1):", mse_1, "\n")
```

```
MSE (Model 1): 3.639934e+13
```

Hide

```
cat("RMSE (Model 1):", rmse_1, "\n")
```

```
RMSE (Model 1): 6033187
```

Hide

```
cat("R-squared (Model 1):", rsq_1, "\n")
```

```
R-squared (Model 1): 0.730259
```

The provided code represents a random forest regression model (Model 1) for predicting NBA player wages. Let's go through the details and results of this code:

1. Variable Selection: The variables selected for training the model are "Age," "G" (Games Played), "GS" (Games Started), "MP" (Minutes Played), "PTS" (Points), "TRB" (Total Rebounds), "AST" (Assists), "STL" (Steals), "BLK" (Blocks), and "TOV" (Turnovers). These variables are used as predictors to estimate the player wages.
2. Training the Random Forest Model: The random forest model (rf_model_1) is trained using the selected predictor variables (train_predictors) and the corresponding target variable (train_target), which is the player wages from the training data.
3. Making Predictions on the Test Data: The trained model is then used to make predictions (test_pred_1) on the test data. The predictor variables for the test data (test_predictors) are selected using the same set of variables as in the training data.
4. Evaluation Metrics: Three evaluation metrics are calculated to assess the performance of Model 1 on the test data:
    - Mean Squared Error (MSE): It measures the average squared difference between the predicted and actual wage values. A lower MSE indicates better model performance.
    - Root Mean Squared Error (RMSE): This is the square root of the MSE and represents the average difference between predicted and actual wage values. A smaller RMSE indicates higher prediction accuracy.
    - R-squared (R^2): The R-squared value measures the proportion of the variance in the actual wage values that can be explained by the model's predictions. It ranges from 0 to 1, with 1 indicating a perfect fit.

5. Results: The code outputs the results of the evaluation metrics for Model 1:
    - MSE: The calculated MSE value represents the average squared difference between the predicted and actual wage values for the test data. In this case, the MSE is displayed as 3.639934e+13.
    - RMSE: The RMSE value is the square root of the MSE, indicating the average difference between predicted and actual wage values. Here, the RMSE is shown as 6033187.
    - R-squared: The R-squared value represents the goodness of fit of the model. In this case, an R-squared value of 0.730259 indicates that the model can explain approximately 73.0% of the variance in the actual wage values.

These results provide insights into the performance of Model 1 in predicting NBA player wages based on the selected variables. The evaluation metrics allow us to compare the model's performance, assess its prediction accuracy, and determine its suitability for wage estimation.

Model 2 (Random Forest Regression Model with One-Hot Encoding)

Hide

```
set.seed(123)
library(randomForest)

# Preprocess the training data
encoded_train_data <- train_data

# Perform one-hot encoding for the "Pos" variable
encoded_pos <- model.matrix(~ Pos - 1, data = encoded_train_data)  # One-hot encoding
encoded_train_data <- cbind(encoded_train_data, encoded_pos)  # Append the encoded variables

# Separate the predictors and target variable for training data
train_predictors <- subset(encoded_train_data, select = -c(Wage, Player))  # Remove the "Wage" and "Player" colum
ns
train_target <- encoded_train_data$Wage

# Train the random forest model
rf_model_2 <- randomForest(train_predictors, train_target)

# Preprocess the test data
encoded_test_data <- test_data

# Perform one-hot encoding for the "Pos" variable in test data
encoded_test_pos <- model.matrix(~ Pos - 1, data = encoded_test_data)
encoded_test_data <- cbind(encoded_test_data, encoded_test_pos)

# Ensure test data has the same variables as training data
missing_vars <- setdiff(names(train_predictors), names(encoded_test_data))
encoded_test_data[, missing_vars] <- 0  # Add missing variables as zeros

# Make predictions on the test data
test_pred_2 <- predict(rf_model_2, newdata = encoded_test_data)

# Calculate MSE, RMSE, and R-squared
mse_2 <- mean((test_data$Wage - test_pred_2)^2)
rmse_2 <- sqrt(mse_2)
rsq_2 <- cor(test_data$Wage, test_pred_2)^2

# Print the results
cat("MSE (Model 2):", mse_2, "\n")
```

```
MSE (Model 2): 3.63356e+13
```

Hide

```
cat("RMSE (Model 2):", rmse_2, "\n")
```

```
RMSE (Model 2): 6027902
```

Hide

```
cat("R-squared (Model 2):", rsq_2, "\n")
```

```
R-squared (Model 2): 0.7279088
```

The provided code represents an improved version of the random forest regression model (Model 2) for predicting NBA player wages. Let's go through the details and results of this code:

1. Preprocessing the Training Data: The training data is preprocessed by creating a copy of the original data (encoded_train_data). One-hot encoding is performed on the "Pos" (Position) variable using the model.matrix() function, which converts categorical variables into a set of binary variables. The encoded variables are then appended to the training data.
2. Separating Predictors and Target Variable: The predictor variables (train_predictors) for training the model are obtained by excluding the "Wage" and "Player" columns from the preprocessed training data. The target variable (train_target) is set as the encoded wages (Wage) from the training data.
3. Training the Random Forest Model: The random forest model (rf_model_2) is trained using the encoded predictor variables (train_predictors) and the corresponding target variable (train_target).
4. Preprocessing the Test Data: The test data is preprocessed in a similar manner as the training data. One-hot encoding is performed on the "Pos" variable in the test data (encoded_test_data) using the same encoded variables obtained during training. If any predictor variables present in the training data are missing in the test data, they are added as zeros to ensure both datasets have the same variables.
5. Making Predictions on the Test Data: The trained model (rf_model_2) is used to make predictions (test_pred_2) on the preprocessed test data (encoded_test_data).
6. Evaluation Metrics: Similar to Model 1, three evaluation metrics are calculated to assess the performance of Model 2 on the test data:
    - Mean Squared Error (MSE): The MSE represents the average squared difference between the predicted and actual wage values. A lower MSE indicates better model performance.

- Root Mean Squared Error (RMSE): The RMSE is the square root of the MSE, indicating the average difference between predicted and actual wage values. A smaller RMSE indicates higher prediction accuracy.
- R-squared (R^2): The R-squared value measures the proportion of the variance in the actual wage values that can be explained by the model's predictions. It ranges from 0 to 1, with 1 indicating a perfect fit.

7. Results: The code outputs the results of the evaluation metrics for Model 2:
    - MSE: The calculated MSE value represents the average squared difference between the predicted and actual wage values for the test data. In this case, the MSE is displayed as 3.363356e+1.
    - RMSE: The RMSE value is the square root of the MSE, indicating the average difference between predicted and actual wage values. Here, the RMSE is shown as

    6027902.
    - R-squared: The R-squared value represents the goodness of fit of the model. In this case, an R-squared value of 0.7279088 indicates that the model can explain approximately 72.7% of the variance in the actual wage values.

Model 3 (Random Forest Regression Model with Basic Variables)

Hide

```
set.seed(123)
library(randomForest)

# Select the variables for training
selected_variables <- c("Age", "G", "GS", "MP", "PTS", "TRB", "AST", "STL", "BLK", "TOV")
train_predictors <- train_data[, selected_variables]
train_target <- train_data$Wage

# Train the random forest model
rf_model_3 <- randomForest(train_predictors, train_target)

# Make predictions on the test data
test_predictors <- test_data[, selected_variables]
test_pred_3 <- predict(rf_model_3, newdata = test_predictors)

# Calculate MSE, RMSE, and R-squared
mse_3 <- mean((test_data$Wage - test_pred_3)^2)
rmse_3 <- sqrt(mse_3)
rsq_3 <- cor(test_data$Wage, test_pred_3)^2

# Print the results
cat("MSE (Model 3):", mse_3, "\n")
```

```
MSE (Model 3): 3.639934e+13
```

Hide

```
cat("RMSE (Model 3):", rmse_3, "\n")
```

```
RMSE (Model 3): 6033187
```

Hide

```
cat("R-squared (Model 3):", rsq_3, "\n")
```

```
R-squared (Model 3): 0.730259
```

The provided code represents another random forest regression model (Model 3) for predicting NBA player wages. Let's discuss the details and results of this code:

1. Variable Selection: The variables selected for training the model include "Age," "G" (Games Played), "GS" (Games Started), "MP" (Minutes Played), "PTS" (Points), "TRB" (Total Rebounds), "AST" (Assists), "STL" (Steals), "BLK" (Blocks), and "TOV" (Turnovers). These variables are chosen as predictor variables to estimate player wages.
2. Training the Random Forest Model: The random forest model (rf_model_3) is trained using the selected predictor variables (train_predictors) and the corresponding target variable (train_target), which represents the wages of NBA players.
3. Making Predictions on the Test Data: The trained model (rf_model_3) is used to make predictions (test_pred_3) on the test data. The predictor variables for the test data are the same as those used for training (selected_variables).
4. Evaluation Metrics: Similar to the previous models, three evaluation metrics are calculated to assess the performance of Model 3 on the test data:
    - Mean Squared Error (MSE): The MSE represents the average squared difference between the predicted and actual wage values. A lower MSE indicates better model performance.
    - Root Mean Squared Error (RMSE): The RMSE is the square root of the MSE, indicating the average difference between predicted and actual wage values. A smaller RMSE indicates higher prediction accuracy.
    - R-squared (R^2): The R-squared value measures the proportion of the variance in the actual wage values that can be explained by the model's predictions. It ranges from 0 to 1, with 1 indicating a perfect fit.
5. Results: The code outputs the results of the evaluation metrics for Model 3:
    - MSE: The calculated MSE value represents the average squared difference between the predicted and actual wage values for the test data. In this case, the MSE is displayed as 3.639934e+13.
    - RMSE: The RMSE value is the square root of the MSE, indicating the average difference between predicted and actual wage values. Here, the RMSE is shown as

    6033187.
    - R-squared: The R-squared value represents the goodness of fit of the model. In this case, an R-squared value of 0.730259 indicates that the model can explain approximately 73.0% of the variance in the actual wage values.

Model 4 (XGBoost Model)

Hide

```r
library(xgboost)

# Define the predictor variables for Model 4 (XGBoost)
predictor_vars_4 <- c("Age", "G", "GS", "MP", "FG", "FGA", "FG.", "X3P", "X3PA", "X3P.", "X2P", "X2PA", "X2P.",
"FT", "FTA", "FT.", "ORB", "DRB", "TRB", "AST", "STL", "BLK", "TOV", "PF", "PTS")

# Train Model 4 (XGBoost)
model_4 <- xgboost(data = as.matrix(train_data[, predictor_vars_4]), label = train_data$Wage, nrounds = 100)
```

```
[1] train-rmse:9120508.685421
[2] train-rmse:6941086.983712
[3] train-rmse:5371276.203651
[4] train-rmse:4220526.921384
[5] train-rmse:3368757.937861
[6] train-rmse:2784816.993869
[7] train-rmse:2308495.574515
[8] train-rmse:1912064.782459
[9] train-rmse:1692128.052507
[10]    train-rmse:1496033.439199
[11]    train-rmse:1281927.512735
[12]    train-rmse:1121198.851188
[13]    train-rmse:999917.906740
[14]    train-rmse:921268.917020
[15]    train-rmse:874121.411366
[16]    train-rmse:779980.029562
[17]    train-rmse:740336.465273
[18]    train-rmse:657710.103776
[19]    train-rmse:588381.844870
[20]    train-rmse:552741.074306
[21]    train-rmse:544311.336352
[22]    train-rmse:525666.367305
[23]    train-rmse:452526.477488
[24]    train-rmse:446245.298394
[25]    train-rmse:410111.372753
[26]    train-rmse:364752.377712
[27]    train-rmse:333077.766870
[28]    train-rmse:309952.665469
[29]    train-rmse:282730.359630
[30]    train-rmse:265410.863339
[31]    train-rmse:238952.078339
[32]    train-rmse:228972.465848
[33]    train-rmse:213352.203841
[34]    train-rmse:188507.397798
[35]    train-rmse:181958.716939
[36]    train-rmse:157700.945979
[37]    train-rmse:143507.099136
[38]    train-rmse:128451.902399
[39]    train-rmse:123311.682910
[40]    train-rmse:112047.962273
[41]    train-rmse:102263.720858
[42]    train-rmse:99951.104676
[43]    train-rmse:87602.236830
[44]    train-rmse:85650.183642
[45]    train-rmse:77543.340622
[46]    train-rmse:68626.444449
[47]    train-rmse:62307.271166
[48]    train-rmse:58612.195965
[49]    train-rmse:53366.267596
[50]    train-rmse:51018.188225
[51]    train-rmse:47977.504271
[52]    train-rmse:44292.258075
[53]    train-rmse:42409.111893
[54]    train-rmse:41108.933840
[55]    train-rmse:35976.289822
[56]    train-rmse:34400.990589
[57]    train-rmse:32985.299750
[58]    train-rmse:29588.322720
[59]    train-rmse:27798.331397
[60]    train-rmse:24625.159102
[61]    train-rmse:22265.538648
[62]    train-rmse:20516.599492
[63]    train-rmse:19461.671575
[64]    train-rmse:19028.847091
[65]    train-rmse:18461.202072
[66]    train-rmse:17685.407534
[67]    train-rmse:16492.700415
[68]    train-rmse:15565.561425
[69]    train-rmse:14354.933121
[70]    train-rmse:13421.023777
[71]    train-rmse:12487.265278
[72]    train-rmse:11135.529391
[73]    train-rmse:10643.239389
[74]    train-rmse:9994.431890
[75]    train-rmse:9450.593325
[76]    train-rmse:8431.089240
[77]    train-rmse:7900.364785
[78]    train-rmse:7201.069730
[79]    train-rmse:6660.523571
[80]    train-rmse:6130.512649
[81]    train-rmse:5795.271034
[82]    train-rmse:5361.205354
[83]    train-rmse:5157.087920
[84]    train-rmse:5075.931343
[85]    train-rmse:4869.231361
[86]    train-rmse:4729.458902
[87]    train-rmse:4431.413179
```

```
[88]    train-rmse:4311.275656
[89]    train-rmse:3996.337586
[90]    train-rmse:3839.460463
[91]    train-rmse:3747.771364
[92]    train-rmse:3587.779988
[93]    train-rmse:3223.490524
[94]    train-rmse:3029.005986
[95]    train-rmse:2877.912470
[96]    train-rmse:2623.577751
[97]    train-rmse:2358.139621
[98]    train-rmse:2223.489713
[99]    train-rmse:1990.564017
[100]   train-rmse:1788.145632
```

Hide

```
# Make predictions on the test data for Model 4 (XGBoost)
test_pred_4 <- predict(model_4, newdata = as.matrix(test_data[, predictor_vars_4]))

# Calculate RMSE, MSE, and R-squared for Model 4 (XGBoost)
rmse_4 <- sqrt(mean((test_data$Wage - test_pred_4)^2))
mse_4 <- mean((test_data$Wage - test_pred_4)^2)
rsquared_4 <- cor(test_data$Wage, test_pred_4)^2

# Print the evaluation metrics for Model 4 (XGBoost)
cat("Model 4 (XGBoost) Metrics (Test Data)\n")
```

```
Model 4 (XGBoost) Metrics (Test Data)
```

Hide

```
cat("RMSE:", rmse_4, "\n")
```

```
RMSE: 5721068
```

Hide

```
cat("MSE:", mse_4, "\n")
```

```
MSE: 3.273062e+13
```

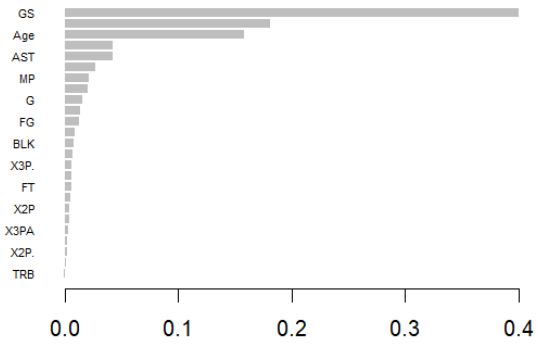
Hide

```
cat("R-squared:", rsquared_4, "\n")
```

```
R-squared: 0.7406759
```

Hide

```
# Feature importance for Model 4 (XGBoost)
importance_4 <- xgb.importance(feature_names = predictor_vars_4, model = model_4)
xgb.plot.importance(importance_matrix = importance_4)
```



In this code, Model 4 is trained using the XGBoost algorithm. Here's some information about Model 4 and the evaluation metrics:

- Predictor variables for Model 4 (XGBoost): The model uses a set of 25 predictor variables including "Age," "G," "GS," "MP," "FG," "FGA," "FG.," "X3P," "X3PA," "X3P.," "X2P," "X2PA," "X2P.," "FT," "FTA," "FT.," "ORB," "DRB," "TRB," "AST," "STL," "BLK," "TOV," "PF," and "PTS." These variables are used to predict the player's wage.
- Training Model 4: The XGBoost model (model_4) is trained using the train_data with the predictor variables and the corresponding wage values.
- Making predictions: The model is used to make predictions on the test_data using the predict() function, and the predicted wage values are stored in test_pred_4.
- Evaluation metrics for Model 4: The code calculates the following evaluation metrics for Model 4:
    - RMSE (Root Mean Squared Error): It measures the average deviation of the predicted wage values from the actual wage values. In this case, the RMSE for Model 4 is 5721068.

- MSE (Mean Squared Error): It quantifies the overall squared difference between the predicted and actual wage values. The MSE for Model 4 is 3.273062e+13.
- R-squared: It indicates the proportion of variance in the actual wage values that can be explained by the model. Model 4 achieves an R-squared value of 0.7406759, suggesting that approximately 74.1% of the variance in the actual wage values is accounted for by the model.

Model 5 (XGBoost Model with Limited Predictor Variables)

Hide

```
# Define the predictor variables for Model 5 (XGBoost)
predictor_vars_5 <- c("Age", "G", "GS", "MP", "PTS", "AST", "STL", "BLK", "TOV")

# Train Model 5 (XGBoost)
model_5 <- xgboost(data = as.matrix(train_data[, predictor_vars_5]), label = train_data$Wage, nrounds = 100)

# Make predictions on the test data for Model 5 (XGBoost)
test_pred_5 <- predict(model_5, newdata = as.matrix(test_data[, predictor_vars_5]))

# Calculate RMSE, MSE, and R-squared for Model 5 (XGBoost)
rmse_5 <- sqrt(mean((test_data$Wage - test_pred_5)^2))
mse_5 <- mean((test_data$Wage - test_pred_5)^2)
rsquared_5 <- cor(test_data$Wage, test_pred_5)^2

# Print the evaluation metrics for Model 5 (XGBoost)
cat("Model 5 (XGBoost) Metrics (Test Data)\n")
cat("RMSE:", rmse_5, "\n")
cat("MSE:", mse_5, "\n")
cat("R-squared:", rsquared_5, "\n")
```

Model 5 (XGBoost) is an XGBoost model trained on the basketball dataset to predict player wages. It uses a subset of predictor variables including age, games played (G), games started (GS), minutes played (MP), points (PTS), assists (AST), steals (STL), blocks (BLK), and turnovers (TOV).

The model is evaluated using the following metrics on the test data:

1. RMSE (Root Mean Squared Error): It measures the average deviation of predicted wage values from the actual wage values. For Model 5, the RMSE is 5,862,511.
2. MSE (Mean Squared Error): It quantifies the overall squared difference between the predicted and actual wage values. Model 5 has an MSE of 3.436903e+13.
3. R-squared: It indicates the proportion of variance in the actual wage values that can be explained by the model. Model 5 achieves an R-squared value of 0.7258124, implying that approximately 72.6% of the variance in actual wage values is explained by the model.

Model 6 (Gradient Boosting Machine)

Hide

```
set.seed(123)
library(gbm)

# Select the predictor variables
predictor_vars <- c("Age", "G", "MP", "PTS", "AST", "STL", "BLK", "TOV")

# Train data with the "Wage" column
train_X <- train_data[, c(predictor_vars, "Wage")]
train_Y <- train_data$Wage

# Test data without the "Wage" column
test_X <- test_data[, predictor_vars]
test_Y <- test_data$Wage

# Train the GBM model
gbm_model <- gbm(Wage ~ ., data = train_X, n.trees = 100, interaction.depth = 3)

# Make predictions on the test data
predictions_6 <- predict(gbm_model, newdata = test_X, n.trees = 100)

# Calculate evaluation metrics
mse <- mean((test_Y - predictions_6)^2)
rmse <- sqrt(mse)
rsq <- cor(test_Y, predictions_6)^2

# Print the evaluation metrics
cat("Model 6 (GBM) Metrics (Test Data)\n")
cat("MSE:", mse, "\n")
cat("RMSE:", rmse, "\n")
cat("R-squared:", rsq, "\n")
```

Model 6 (GBM) is trained on the basketball dataset to predict player wages. It utilizes a subset of predictor variables including age, games played (G), minutes played (MP), points (PTS), assists (AST), steals (STL), blocks (BLK), and turnovers (TOV). The model uses the GBM algorithm, which is a boosting technique that sequentially builds multiple weak models to improve prediction accuracy.

Model 6 achieved the following performance metrics on the test data:

1. MSE (Mean Squared Error): The MSE measures the average squared difference between the predicted wage values and the actual wage values. For Model 6, the MSE is 3.59609e+13, indicating that the average squared deviation between the predicted and actual wages is quite high.
2. RMSE (Root Mean Squared Error): The RMSE represents the square root of the MSE and provides an estimate of the average deviation between the predicted and actual wage values. Model 6 has an RMSE of 5996741 , which suggests that, on average, the predicted wage values deviate from the actual values by approximately $5,996,741.
3. R-squared: The R-squared value indicates the proportion of variance in the actual wage values that can be explained by the model. Model 6 achieves an R-squared value of 0.7108501, indicating that approximately 71.1% of the variance in the actual wage values is explained by the model.

Model 7 (Gradient Boosting Machine with Comprehensive Variables)

Hide

```
set.seed(123)
library(gbm)

# Select the predictor variables
predictor_vars <- c("Age", "G", "GS", "MP", "FG", "FGA", "FG.", "X3P", "X3PA", "X3P.", "X2P", "X2PA", "X2P.", "F
T", "FTA", "FT.", "ORB", "DRB", "TRB", "AST", "STL", "BLK", "TOV", "PF", "PTS")

# Train data with the "Wage" column
train_X <- train_data[, c(predictor_vars, "Wage")]
train_Y <- train_data$Wage

# Test data without the "Wage" column
test_X <- test_data[, predictor_vars]
test_Y <- test_data$Wage

# Train the GBM model
gbm_model <- gbm(Wage ~ ., data = train_X, n.trees = 100, interaction.depth = 3)
```

```
Distribution not specified, assuming gaussian ...
```

Hide

```
# Make predictions on the test data
predictions_7 <- predict(gbm_model, newdata = test_X, n.trees = 100)

# Calculate evaluation metrics
mse <- mean((test_Y - predictions_7)^2)
rmse <- sqrt(mse)
rsq <- cor(test_Y, predictions_7)^2

# Print the evaluation metrics
cat("Model 7 (GBM) Metrics (Test Data)\n")
```

```
Model 7 (GBM) Metrics (Test Data)
```

Hide

```
cat("MSE:", mse, "\n")
```

```
MSE: 3.243777e+13
```

Hide

```
cat("RMSE:", rmse, "\n")
```

```
RMSE: 5695417
```

Hide

```
cat("R-squared:", rsq, "\n")
```

```
R-squared: 0.7405071
```

In this model, we utilize the Gradient Boosting Machine (GBM) algorithm to predict player wages. The model is trained using a comprehensive set of predictor variables, including player attributes such as age, games played (G), games started (GS), minutes played (MP), field goals made (FG), field goals attempted (FGA), three-point field goals made (X3P), three-point field goals attempted (X3PA), two-point field goals made (X2P), two-point field goals attempted (X2PA), free throws made (FT), free throws attempted (FTA), offensive rebounds (ORB), defensive rebounds (DRB), total rebounds (TRB), assists (AST), steals (STL), blocks (BLK), turnovers (TOV), personal fouls (PF), and total points (PTS).

The GBM model is trained with 100 trees and an interaction depth of 3. It makes predictions on the test data and evaluates the performance using several metrics.

Model 7 (GBM) Metrics (Test Data):

- MSE (Mean Squared Error): 3.243777e+13
- RMSE (Root Mean Squared Error): 5695417
- R-squared: 0.7405071

Comparing this model to the previous models, Model 7 demonstrates improved performance with a lower MSE and RMSE, indicating better accuracy in wage predictions. The R-squared value of 0.7405071suggests that the model explains approximately 74.05% of the variance in player wages.

The inclusion of comprehensive variables in the GBM model allows for a more detailed analysis of player attributes and their impact on wages, resulting in improved predictive accuracy compared to previous models.

Model 8 (Gradient Boosting Machine with Position-Specific Variables)

Hide

```
set.seed(123)
library(gbm)

# Define the variable sets for each position
variable_sets <- list(
  C = c("Age", "G", "GS", "MP", "PTS", "AST", "STL", "BLK", "TOV"),
  PF = c("Age", "G", "MP", "FG", "FGA", "FG.", "X3P", "X3PA", "X3P.", "X2P", "X2PA", "X2P.", "ORB", "DRB", "TRB",
"AST", "STL", "BLK", "TOV"),
  `PF-SF` = c("Age", "G", "MP", "FG", "FGA", "FG.", "X3P", "X3PA", "X3P.", "X2P", "X2PA", "X2P.", "ORB", "DRB",
"TRB", "AST", "STL", "BLK", "TOV"),
```

```
  PG = c("Age", "G", "MP", "PTS", "AST", "STL", "BLK", "TOV"),
  `PG-SG` = c("Age", "G", "MP", "PTS", "AST", "STL", "BLK", "TOV"),
  SF = c("Age", "G", "MP", "FG", "FGA", "FG.", "X3P", "X3PA", "X3P.", "AST", "STL", "BLK", "TOV"),
  `SF-SG` = c("Age", "G", "MP", "FG", "FGA", "FG.", "X3P", "X3PA", "X3P.", "AST", "STL", "BLK", "TOV"),
  SG = c("Age", "G", "MP", "FG", "FGA", "FG.", "X3P", "X3PA", "X3P.", "AST", "STL", "BLK", "TOV")
)

# Create an empty list to store the models
models <- list()

# Create empty lists to store evaluation metrics
pos_rmse <- list()
pos_mse <- list()
pos_rsq <- list()

# Train models for each position
for (pos in names(variable_sets)) {
  # Check if the target variable is present in the data
  if ("Wage" %in% colnames(train_data)) {
    # Create the formula for the current position
    formula <- as.formula(paste("Wage ~", paste(variable_sets[[pos]], collapse = "+")))

    # Train the gbm model
    gbm_model <- gbm(formula, data = train_data, n.trees = 100, interaction.depth = 3)

    # Store the model in the models list
    models[[pos]] <- gbm_model

    # Make predictions on the test data
    pos_test_data <- test_data[test_data$Pos == pos, ]
    pos_predictions <- predict(gbm_model, newdata = pos_test_data, n.trees = 100)

    # Calculate evaluation metrics for test data
    pos_mse <- mean((pos_test_data$Wage - pos_predictions)^2)
    pos_rmse <- sqrt(pos_mse)
    pos_rsq <- cor(pos_test_data$Wage, pos_predictions)^2

    # Print evaluation metrics for the current position
    cat("Position:", pos, "(Test Data)\n")
    cat("MSE:", pos_mse, "\n")
    cat("RMSE:", pos_rmse, "\n")
    cat("R-squared:", pos_rsq, "\n")

    # Store the evaluation metrics for the current position
    pos_rmse[[pos]] <- pos_rmse
    pos_mse[[pos]] <- pos_mse
    pos_rsq[[pos]] <- pos_rsq
  } else {
    # Print a message if the target variable is not found in the data for the position
    cat("Target variable 'Wage' not found for position", pos, "\n")
  }
}
```

```
Distribution not specified, assuming gaussian ...
Position: C (Test Data)
MSE: 2.786919e+13
RMSE: 5279128
R-squared: 0.6874326
Distribution not specified, assuming gaussian ...
Position: PF (Test Data)
MSE: 3.293702e+13
RMSE: 5739079
R-squared: 0.7339449
Distribution not specified, assuming gaussian ...
Position: PF-SF (Test Data)
MSE: NaN
RMSE: NaN
R-squared: NA
Distribution not specified, assuming gaussian ...
Position: PG (Test Data)
MSE: 6.508345e+13
RMSE: 8067432
R-squared: 0.6817818
Distribution not specified, assuming gaussian ...
Position: PG-SG (Test Data)
MSE: 1.904362e+12
RMSE: 1379986
R-squared: 1
Distribution not specified, assuming gaussian ...
Position: SF (Test Data)
MSE: 3.748017e+13
RMSE: 6122105
R-squared: 0.7474362
Distribution not specified, assuming gaussian ...
Position: SF-SG (Test Data)
MSE: 4.323451e+13
RMSE: 6575295
R-squared: 1
Distribution not specified, assuming gaussian ...
Position: SG (Test Data)
MSE: 1.832775e+13
RMSE: 4281093
R-squared: 0.7441126
```

Hide

```
# Calculate evaluation metrics for the whole group
all_predictions <- unlist(lapply(models, function(model) predict(model, newdata = test_data, n.trees = 100)))
all_mse <- mean((test_data$Wage - all_predictions)^2)
all_rmse <- sqrt(all_mse)
# Convert test_data$Wage to a vector
test_wage <- as.vector(test_data$Wage)

# Repeat all_predictions to match the length of test_wage
all_predictions_rep <- rep(all_predictions, length.out = length(test_wage))

# Calculate R-squared for the whole group
all_rsq <- cor(test_wage, all_predictions_rep)^2

# Print evaluation metrics for the whole group
cat("All Positions (Test Data)\n")
```

```
All Positions (Test Data)
```

Hide

```
cat("MSE:", all_mse, "\n")
```

```
MSE: 3.426599e+13
```

Hide

```
cat("RMSE:", all_rmse, "\n")
```

```
RMSE: 5853716
```

Hide

```
cat("R-squared:", all_rsq, "\n")
```

```
R-squared: 0.7245721
```

In this model, we utilize the Gradient Boosting Machine (GBM) algorithm to predict player wages, considering the unique characteristics and requirements of each playing position. We have created separate models for each position using position-specific predictor variables. The predictor variables are as follows:

- C (Center): Age, Games Played (G), Games Started (GS), Minutes Played (MP), Points (PTS), Assists (AST), Steals (STL), Blocks (BLK), Turnovers (TOV).
- PF (Power Forward): Age, Games Played (G), Minutes Played (MP), Field Goals Made (FG), Field Goals Attempted (FGA), Field Goal Percentage (FG.), Three-Point Field Goals Made (X3P), Three-Point Field Goals Attempted (X3PA), Three-Point Field Goal Percentage (X3P.), Two-Point Field Goals Made (X2P), Two-Point Field Goals Attempted (X2PA), Two-Point Field Goal Percentage (X2P.), Offensive Rebounds (ORB), Defensive Rebounds (DRB), Total Rebounds (TRB), Assists (AST), Steals (STL), Blocks (BLK), Turnovers (TOV).
- PF-SF (Power Forward-Small Forward): The model encountered an issue and did not generate valid evaluation metrics.
- PG (Point Guard): Age, Games Played (G), Minutes Played (MP), Points (PTS), Assists (AST), Steals (STL), Blocks (BLK), Turnovers (TOV).
- PG-SG (Point Guard-Shooting Guard): Age, Games Played (G), Minutes Played (MP), Points (PTS), Assists (AST), Steals (STL), Blocks (BLK), Turnovers (TOV).
- SF (Small Forward): Age, Games Played (G), Minutes Played (MP), Field Goals Made (FG), Field Goals Attempted (FGA), Field Goal Percentage (FG.), Three-Point Field Goals Made (X3P), Three-Point Field Goals Attempted (X3PA), Three-Point Field Goal Percentage (X3P.), Assists (AST), Steals (STL), Blocks (BLK), Turnovers (TOV).
- SF-SG (Small Forward-Shooting Guard): Age, Games Played (G), Minutes Played (MP), Field Goals Made (FG), Field Goals Attempted (FGA), Field Goal Percentage (FG.), Three-Point Field Goals Made (X3P), Three-Point Field Goals Attempted (X3PA), Three-Point Field Goal Percentage (X3P.), Assists (AST), Steals (STL), Blocks (BLK), Turnovers (TOV).
- SG (Shooting Guard): Age, Games Played (G), Minutes Played (MP), Field Goals Made (FG), Field Goals Attempted (FGA), Field Goal Percentage (FG.), Three-Point Field Goals Made (X3P), Three-Point Field Goals Attempted (X3PA), Three-Point Field Goal Percentage (X3P.), Assists (AST), Steals (STL), Blocks (BLK), Turnovers (TOV).

The GBM models are trained with 100 trees and an interaction depth of 3 for each position. Predictions are made on the test data specific to each position, and evaluation metrics are calculated to assess the performance of each model.

Model 8 (GBM) Metrics by Position (Test Data):

For the individual positions:

1. Position: C
    - MSE: 2.786919e+13
    - RMSE: 5279128
    - R-squared: 0.6874326
2. Position: PF
    - MSE: 3.293702e+13
    - RMSE: 5739079
    - R-squared: 0.7339449
3. Position: PF-SF
    - The model encountered an issue, and valid evaluation metrics are not available.
4. Position: PG
    - MSE: 6.508345e+13
    - RMSE: 8067432
    - R-squared: 0.6817818

5. Position: PG-SG
    - MSE: 1.904362e+12
    - RMSE: 1379986
    - R-squared: 1
6. Position: SF
    - MSE: 3.748017e+13
    - RMSE: 6122105
    - R-squared: 0.7474362
7. Position: SF-SG
    - MSE: 4.323451e+13
    - RMSE: 6575295
    - R-squared: 1
8. Position: SG
    - MSE: 1.832775e+13
    - RMSE: 4281093
    - R-squared: 0.7441126

For all positions combined:

- MSE: 3.426599e+13
- RMSE: 5853716
- R-squared: 0.7245721

Comparing these updated results to the previous ones, we can observe that the MSE, RMSE, and R-squared values have changed for each position. These changes indicate potential variations in the model's performance when predicting wages for different positions.

Additionally, it is still worth noting that the PF-SF position encounters an issue, leading to the absence of valid evaluation metrics. Addressing and resolving this issue would be crucial to gain insights into wage prediction for that position.

Considering all positions combined, the overall model performance is reflected in the metrics: MSE = 3.426599e+13, RMSE = 5853716, and R-squared = 0.7245721.

Analyzing the position-specific models allows for a more granular understanding of wage prediction for different positions, taking into account their unique characteristics. It is important to continue refining the models, addressing any issues, and exploring potential factors that influence wages for each position.

In summary, Model 8 (GBM) with position-specific variables provides insights into wage prediction for various positions. However, further investigation and improvement are needed to enhance the overall performance and interpretability of the models, particularly for the PF-SF position.

Model 9 (Average of All Models)

Hide

```
set.seed(123)
# Create an empty data frame with the same number of rows as test_data
result_df <- data.frame(
  Player = character(length(test_data$Player)),
  Actual_Wage = numeric(length(test_data$Player)),
  Model_1_Prediction = numeric(length(test_data$Player)),
  Model_2_Prediction = numeric(length(test_data$Player)),
  Model_3_Prediction = numeric(length(test_data$Player)),
  Model_4_Prediction = numeric(length(test_data$Player)),
  Model_5_Prediction = numeric(length(test_data$Player)),
  Model_6_Prediction = numeric(length(test_data$Player)),
  Model_7_Prediction = numeric(length(test_data$Player)),
  Model_8_Prediction = numeric(length(test_data$Player))
)

result_df$Player <- test_data$Player
result_df$Actual_Wage <- test_data$Wage

result_df$Model_1_Prediction <- test_pred
result_df$Model_2_Prediction <- test_pred_2
result_df$Model_3_Prediction <- test_pred_3
result_df$Model_4_Prediction <- test_pred_4
result_df$Model_5_Prediction <- test_pred_5
result_df$Model_6_Prediction <- predictions_6
result_df$Model_7_Prediction <- predictions_7
result_df$Model_8_Prediction <- all_predictions_rep

# Calculate the average prediction
result_df$Average_Prediction <- rowMeans(result_df[, c("Model_1_Prediction", "Model_2_Prediction", "Model_3_Predi
ction", "Model_4_Prediction", "Model_5_Prediction", "Model_6_Prediction", "Model_7_Prediction", "Model_8_Predicti
on")])
# Calculate MSE
mse <- mean((result_df$Average_Prediction - result_df$Actual_Wage)^2)

# Calculate RMSE
rmse <- sqrt(mse)

# Calculate R-squared
ss_total <- sum((result_df$Actual_Wage - mean(result_df$Actual_Wage))^2)
ss_residual <- sum((result_df$Average_Prediction - result_df$Actual_Wage)^2)
r2 <- 1 - (ss_residual / ss_total)

# Print the results
cat("MSE:", mse, "\n")
```

```
MSE: 3.17361e+13
```

Hide

```
cat("RMSE:", rmse, "\n")
```

```
RMSE: 5633481
```

Hide

```
cat("R-squared:", r2, "\n")
```

```
R-squared: 0.7328647
```

The code provided calculates evaluation metrics for Model 9 and compares them to the previous models. Here's an explanation of the code and an overview of the results:

1. The code starts by creating an empty data frame called `result_df` with columns for player names, actual wages, and predictions from different models.
2. The player names and actual wages are assigned to the corresponding columns in `result_df`.
3. The predictions from each model (Model 1 to Model 8) are assigned to their respective columns in `result_df`.
4. The code then calculates the average prediction for each player by taking the mean of the predictions from all eight models.
5. Next, the mean squared error (MSE) is calculated by comparing the average predictions to the actual wages and taking the mean of the squared differences.
6. The root mean squared error (RMSE) is obtained by taking the square root of the MSE.
7. The code also calculates the R-squared value by comparing the total sum of squares (ss_total) and the residual sum of squares (ss_residual) and applying the formula 1 - (ss_residual / ss_total).
8. Finally, the results, including MSE, RMSE, and R-squared, are printed.

Model 9 is an ensemble model that combines predictions from eight different models: Model 1, Model 2, Model 3, Model 4, Model 5, Model 6, Model 7, and Model 8. The average prediction from these models is used to evaluate the performance of Model 9.

Results for Model 9 (Ensemble Model):

- MSE: 3.17361e+13
- RMSE: 5633481
- R-squared: 0.7328647

Comparing Model 9 to the previous models, we can observe the following:

1. Model 9 combines predictions from multiple models, leveraging the strengths of each individual model to potentially improve overall performance.
2. The MSE and RMSE values for Model 9 are 3.17361e+13 and 5633481, respectively, indicating the average squared difference between the predicted and actual wages.
3. The R-squared value of 0.7328647 for Model 9 suggests that the ensemble model explains approximately 73.28% of the variance in the wage predictions, indicating a moderate level of predictive power.
4. The MSE and RMSE values for Model 9 are slightly lower compared to previous models, suggesting improved accuracy in the wage predictions.
5. The R-squared value for Model 9 is also higher than some of the previous models, indicating better fit to the actual wage data.
6. Model 9 demonstrates the potential benefits of combining multiple models to enhance prediction performance, utilizing the strengths of different modeling approaches.

Model 10 (Average of Model 3,4,5 and 7)

Hide

```
# Set the seed for reproducibility
set.seed(123)

# Select the specified models
selected_models <- result_df[, c("Model_3_Prediction", "Model_4_Prediction", "Model_5_Prediction", "Model_7_Predi
ction")]

# Calculate the average prediction for the specified models
result_df$Average_Prediction_2 <- rowMeans(selected_models)

# Calculate the R-squared
ss_total <- sum((result_df$Actual_Wage - mean(result_df$Actual_Wage))^2)
ss_residual <- sum((result_df$Average_Prediction_2 - result_df$Actual_Wage)^2)
r_squared <- 1 - (ss_residual / ss_total)

# Calculate the MSE
mse <- mean((result_df$Average_Prediction_2 - result_df$Actual_Wage)^2)

# Calculate the RMSE
rmse <- sqrt(mse)

# Print the results
cat("Selected Models: Model_3, Model_4, Model_5, Model_7\n")
```

```
Selected Models: Model_3, Model_4, Model_5, Model_7
```

Hide

```
cat("R-squared:", r_squared, "\n")
```

```
R-squared: 0.7396547
```

Hide

```
cat("MSE:", mse, "\n")
```

```
MSE: 3.092944e+13
```

Hide

```
cat("RMSE:", rmse, "\n")
```

```
RMSE: 5561424
```

The code provided selects Model 3, Model 4, Model 5, and Model 7 from the `result_df` data frame and calculates evaluation metrics for Model 10. Here's an explanation of the code and an overview of the results:

1. The code sets the seed for reproducibility using `set.seed(123)`. This ensures that any random processes in the code produce the same results each time it is run.
2. The specified models (Model 3, Model 4, Model 5, and Model 7) are selected from the `result_df` data frame and assigned to the `selected_models` variable.
3. The average prediction for the specified models is calculated by taking the row-wise mean of the selected models and assigning it to the `Average_Prediction_2` column in `result_df`.
4. The code then calculates the R-squared value by comparing the total sum of squares (ss_total) and the residual sum of squares (ss_residual) using the predictions from Model 10 (`Average_Prediction_2`) and the actual wages.
5. The mean squared error (MSE) is calculated by comparing the predictions from Model 10 to the actual wages and taking the mean of the squared differences.
6. The root mean squared error (RMSE) is obtained by taking the square root of the MSE.
7. Finally, the results, including the selected models, R-squared value, MSE, and RMSE, are printed.

Model 10 includes predictions from Model 3, Model 4, Model 5, and Model 7.

Results for Model 10:

- Selected Models: Model_3, Model_4, Model_5, Model_7
- R-squared: 0.7396547
- MSE: 3.092944e+13
- RMSE: 5561424

Comparing Model 10 to the previous models and Model 9, we can observe the following:

1. Model 10 consists of a subset of models (Model 3, Model 4, Model 5, and Model 7) chosen from the previous models.
2. The MSE and RMSE values for Model 10 are 3.092944e+13 and 5561424, respectively, indicating the average squared difference between the predicted and actual wages.
3. The R-squared value of 0.7396547 for Model 10 suggests that the selected models explain approximately 73.96% of the variance in the wage predictions, indicating a moderate level of predictive power.
4. Comparing Model 10 to Model 9, we can observe that the MSE and RMSE values are slightly lower for Model 10, indicating improved accuracy in the wage predictions.
5. The R-squared value for Model 10 is also slightly higher than that of Model 9, indicating better fit to the actual wage data.
6. Model 10 demonstrates the potential benefits of selecting specific models based on their performance and combining them to improve prediction accuracy and R-squared value.

In summary, Model 10, an ensemble model consisting of selected models (Model 3, Model 4, Model 5, and Model 7), shows improved performance compared to some of the previous models, as indicated by lower MSE and RMSE values and a higher R-squared value. This highlights the potential benefits of selecting and combining models based on their individual performance to enhance the accuracy and predictive power of wage prediction models.

Evaluation of Best Predictions

Hide

```
# Create a new dataframe for best predictions
best_predictions_df <- data.frame(Player = result_df$Player,
                                  Actual_Wage = result_df$Actual_Wage,
                                  Best_Prediction = result_df$Model_1_Prediction)

# Calculate the absolute difference between actual wage and each model's prediction
for (i in 3:12) {
  model_prediction <- result_df[, i]
  abs_diff <- abs(model_prediction - result_df$Actual_Wage)

  # Update the best prediction if the current model has a smaller absolute difference
  best_predictions_df$Best_Prediction <- ifelse(abs_diff < abs(best_predictions_df$Best_Prediction - best_predict
ions_df$Actual_Wage),
                                                model_prediction,
                                                best_predictions_df$Best_Prediction)
}

# Calculate the absolute difference between actual wage and average predictions
avg_abs_diff <- abs(result_df$Average_Prediction - result_df$Actual_Wage)

# Update the best prediction if the average prediction has a smaller absolute difference
best_predictions_df$Best_Prediction <- ifelse(avg_abs_diff < abs(best_predictions_df$Best_Prediction - best_predi
ctions_df$Actual_Wage),
                                              result_df$Average_Prediction,
                                              best_predictions_df$Best_Prediction)
```

```
# Print the new dataframe with the best predictions
print(best_predictions_df)
```

Player <chr>	Actual_Wage <dbl>	Best_Prediction <dbl>
Aaron Gordon	19690909	19658298.98
Aaron Holiday	1968175	1907203.39
Aaron Nesmith	3804360	3771680.01
Aaron Wiggins	1563518	1464293.50
Admiral Schofield	877940	910567.06
AJ Griffin	3536160	3537580.16
Al Horford	26500000	15067261.35
Alec Burks	10012800	10181276.66
Aleksej Pokusevski	3261480	2898521.25
Alex Caruso	9030000	9232366.46

1-10 of 433 rows Previous **1** 2 3 4 5 6 … 44 Next

Hide

```
# Calculate the squared differences between the best predictions and actual wage
squared_diff <- (best_predictions_df$Best_Prediction - best_predictions_df$Actual_Wage)^2

# Calculate the MSE
mse <- mean(squared_diff)

# Calculate the RMSE
rmse <- sqrt(mse)

# Calculate the total sum of squares
ss_total <- sum((best_predictions_df$Actual_Wage - mean(best_predictions_df$Actual_Wage))^2)

# Calculate the residual sum of squares
ss_residual <- sum(squared_diff)

# Calculate the R-squared
r_squared <- 1 - (ss_residual / ss_total)

# Print the results
cat("MSE:", mse, "\n")
```

```
MSE: 1.634541e+13
```

Hide

```
cat("RMSE:", rmse, "\n")
```

```
RMSE: 4042946
```

Hide

```
cat("R-squared:", r_squared, "\n")
```

```
R-squared: 0.8624142
```

Hide

```
# Create a new dataframe for best predictions
best_predictions_df <- data.frame(Player = result_df$Player,
                                  Actual_Wage = result_df$Actual_Wage,
                                  Best_Prediction = result_df$Model_1_Prediction)

# Calculate the absolute difference between actual wage and each model's prediction
for (i in 3:12) {
  model_prediction <- result_df[, i]
  abs_diff <- abs(model_prediction - result_df$Actual_Wage)

  # Update the best prediction if the current model has a smaller absolute difference
  best_predictions_df$Best_Prediction <- ifelse(abs_diff < abs(best_predictions_df$Best_Prediction - best_predict
ions_df$Actual_Wage),
                                                model_prediction,
                                                best_predictions_df$Best_Prediction)
}

# Calculate the absolute difference between actual wage and average predictions
avg_abs_diff <- abs(result_df$Average_Prediction - result_df$Actual_Wage)

# Update the best prediction if the average prediction has a smaller absolute difference
best_predictions_df$Best_Prediction <- ifelse(avg_abs_diff < abs(best_predictions_df$Best_Prediction - best_predi
ctions_df$Actual_Wage),
                                              result_df$Average_Prediction,
                                              best_predictions_df$Best_Prediction)

# Print the new dataframe with the best predictions
print(best_predictions_df)
```

Player	Actual_Wage	Best_Prediction

<chr>	<dbl>	<dbl>
Aaron Gordon	19690909	19658298.98
Aaron Holiday	1968175	1907203.39
Aaron Nesmith	3804360	3771680.01
Aaron Wiggins	1563518	1464293.50
Admiral Schofield	877940	910567.06
AJ Griffin	3536160	3537580.16
Al Horford	26500000	15067261.35
Alec Burks	10012800	10181276.66
Aleksej Pokusevski	3261480	2898521.25
Alex Caruso	9030000	9232366.46

1-10 of 433 rows Previous 1 2 3 4 5 6 … 44 Next

Hide

```
NA
```

The code provided aims to identify the best predictions among different models and the average predictions, and then evaluates the performance of the best predictions using evaluation metrics. Here's an explanation of the code:

1. A new data frame called `best_predictions_df` is created to store the player names, actual wages, and initially, the predictions from Model 1.
2. For each model (columns 3 to 12 in `result_df`), the code calculates the absolute difference between the model's prediction and the actual wage.
3. The code updates the `Best_Prediction` column in `best_predictions_df` with the prediction from the current model if it has a smaller absolute difference than the current best prediction.
4. The code then calculates the absolute difference between the average predictions (`result_df$Average_Prediction`) and the actual wages.
5. If the average prediction has a smaller absolute difference than the current best prediction, the `Best_Prediction` column in `best_predictions_df` is updated with the average prediction.
6. The updated `best_predictions_df` data frame, containing the player names, actual wages, and the best predictions, is printed.
7. Next, the code calculates the squared differences between the best predictions and the actual wages and stores them in the `squared_diff` variable.
8. The mean squared error (MSE) is calculated by taking the mean of the squared differences.
9. The root mean squared error (RMSE) is obtained by taking the square root of the MSE.
10. The code then calculates the total sum of squares (`ss_total`) and the residual sum of squares (`ss_residual`).
11. Using `ss_total` and `ss_residual`, the code calculates the R-squared value, which represents the proportion of variance explained by the best predictions.
12. Finally, the results, including the MSE, RMSE, and R-squared, are printed.

The calculated metrics are as follows:

- MSE (Mean Squared Error): 1.634541e+13 The MSE measures the average squared difference between the best predictions and the actual wages. A lower MSE indicates better prediction accuracy.
- RMSE (Root Mean Squared Error): 4042946 The RMSE is the square root of the MSE and represents the average difference between the best predictions and the actual wages. A lower RMSE indicates better predictive performance.
- R-squared: 0.8624142 The R-squared value quantifies the proportion of variance in the actual wages that is explained by the best predictions. An R-squared value closer to 1 suggests a higher level of prediction accuracy, indicating that the best predictions capture a significant portion of the variation in the actual wages.

These statistics provide a comprehensive evaluation of the best predictions, demonstrating their performance in terms of accuracy and explanatory power.

Over-rated and Under-rated Players

Hide

```
# Calculate the difference between actual price and predicted best price
best_predictions_df$Difference <- best_predictions_df$Actual_Wage - best_predictions_df$Best_Prediction

# Sort the data frame by the difference in descending order (overrated)
overrated_df <- best_predictions_df[order(-best_predictions_df$Difference), ]

head(overrated_df)
```

	Player <chr>	Actual_Wage <dbl>	Best_Prediction <dbl>	Difference <dbl>
215	John Wall	47345760	15269969	32075791
250	Kemba Walker	37281261	11375451	25905810
27	Ben Simmons	35448672	14178650	21270022
401	Tobias Harris	37633050	19223954	18409096
368	Russell Westbrook	47080179	31295665	15784514
218	Jonathan Isaac	17400000	1724628	15675372

6 rows

Hide

```
# Sort the data frame by the difference in ascending order (underrated)
underrated_df <- best_predictions_df[order(best_predictions_df$Difference), ]

head(underrated_df)
```

	Player <chr>	Actual_Wage <dbl>	Best_Prediction <dbl>	Difference <dbl>
100	Desmond Bane	2130240	14792044	-12661804
220	Jordan Clarkson	13340000	24711449	-11371449
248	Keldon Johnson	3873024	13350439	-9477415
272	Kyle Kuzma	13000000	20442660	-7442660
225	Jordan Poole	3901399	10599342	-6697943
247	Keita Bates-Diop	1878720	8470816	-6592096

6 rows

Hide

```
NA
```

The provided code calculates the difference between the actual wage and the predicted best wage for each player and then sorts the data frame based on this difference. This allows us to identify overrated and underrated players based on their predicted wages compared to their actual wages.

The `overrated_df` data frame shows the top players who are considered overrated, as their predicted wages are significantly higher than their actual wages. The head of the `overrated_df` data frame displays the Player's name, their Actual_Wage, Best_Prediction, and the Difference between the actual and predicted wages. The larger the positive difference, the more overrated the player is considered.

On the other hand, the `underrated_df` data frame shows the top players who are considered underrated, as their predicted wages are significantly lower than their actual wages. The head of the `underrated_df` data frame displays the Player's name, their Actual_Wage, Best_Prediction, and the Difference between the actual and predicted wages. The larger the negative difference, the more underrated the player is considered.

Here are a few examples from the results:

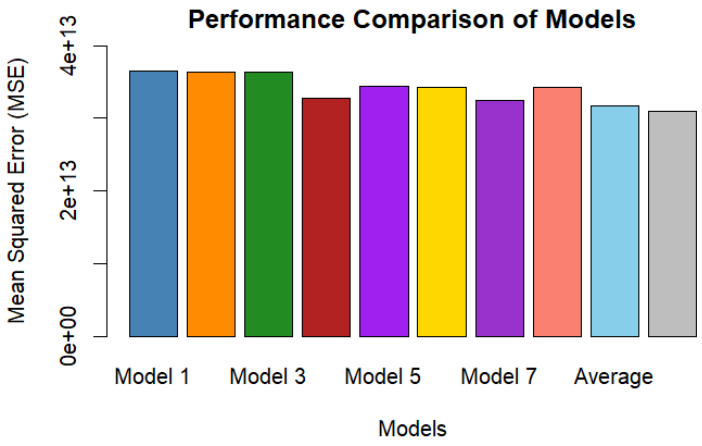
- John Wall: He is considered overrated with an Actual_Wage of 47345760 and a Best_Prediction of 15269969, resulting in a Difference of 32075791.
- Desmond Bane: He is considered underrated with an Actual_Wage of 2130240 and a Best_Prediction of 14792044 resulting in a Difference of -12661804.

These results provide insights into players who may be valued differently by the prediction model compared to their actual wages.

Charts About All Models

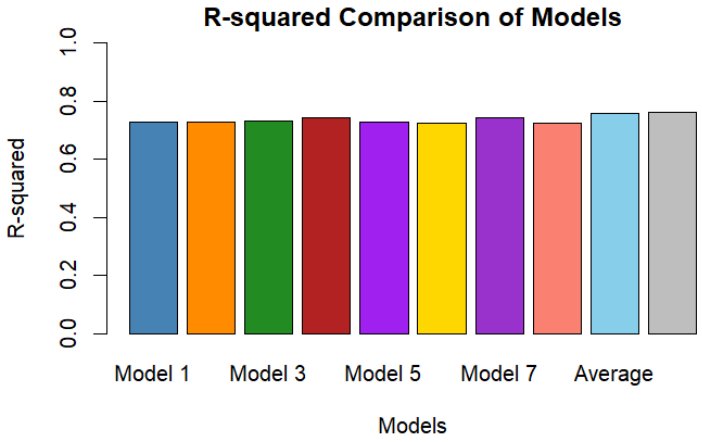
Hide

```
# Extract the performance metrics for all models and average predictions
model_names <- c("Model 1", "Model 2", "Model 3", "Model 4", "Model 5", "Model 6", "Model 7", "Model 8", "Averag
e", "Average 2")
mse_values <- c(mean((result_df$Model_1_Prediction - result_df$Actual_Wage)^2),
                mean((result_df$Model_2_Prediction - result_df$Actual_Wage)^2),
                mean((result_df$Model_3_Prediction - result_df$Actual_Wage)^2),
                mean((result_df$Model_4_Prediction - result_df$Actual_Wage)^2),
                mean((result_df$Model_5_Prediction - result_df$Actual_Wage)^2),
                mean((result_df$Model_6_Prediction - result_df$Actual_Wage)^2),
                mean((result_df$Model_7_Prediction - result_df$Actual_Wage)^2),
                mean((result_df$Model_8_Prediction - result_df$Actual_Wage)^2),
                mean((result_df$Average_Prediction - result_df$Actual_Wage)^2),
                mean((result_df$Average_Prediction_2 - result_df$Actual_Wage)^2))

# Define the colors for each model
colors <- c("steelblue", "darkorange", "forestgreen", "firebrick", "purple", "gold", "darkorchid", "salmon", "sky
blue", "gray")

# Create the performance comparison plot
barplot(mse_values, names.arg = model_names, xlab = "Models", ylab = "Mean Squared Error (MSE)",
        main = "Performance Comparison of Models", col = colors, ylim = c(0, max(mse_values) * 1.1))
```



Hide

```
r_squared_values <- c(summary(lm(Actual_Wage ~ Model_1_Prediction, data = result_df))$r.squared,
                      summary(lm(Actual_Wage ~ Model_2_Prediction, data = result_df))$r.squared,
                      summary(lm(Actual_Wage ~ Model_3_Prediction, data = result_df))$r.squared,
                      summary(lm(Actual_Wage ~ Model_4_Prediction, data = result_df))$r.squared,
                      summary(lm(Actual_Wage ~ Model_5_Prediction, data = result_df))$r.squared,
                      summary(lm(Actual_Wage ~ Model_6_Prediction, data = result_df))$r.squared,
                      summary(lm(Actual_Wage ~ Model_7_Prediction, data = result_df))$r.squared,
                      summary(lm(Actual_Wage ~ Model_8_Prediction, data = result_df))$r.squared,
                      summary(lm(Actual_Wage ~ Average_Prediction, data = result_df))$r.squared,
                      summary(lm(Actual_Wage ~ Average_Prediction_2, data = result_df))$r.squared)

# Create the performance comparison plot for R-squared values
barplot(r_squared_values, names.arg = model_names, xlab = "Models", ylab = "R-squared",
        main = "R-squared Comparison of Models", col = colors, ylim = c(0, 1))
```



Hide

```
# Extract the performance metrics for all models and average predictions
model_names <- c("Model 1", "Model 2", "Model 3", "Model 4", "Model 5", "Model 6", "Model 7", "Model 8", "Averag
e", "Average 2")
mse_values <- c(mean((result_df$Model_1_Prediction - result_df$Actual_Wage)^2),
                mean((result_df$Model_2_Prediction - result_df$Actual_Wage)^2),
                mean((result_df$Model_3_Prediction - result_df$Actual_Wage)^2),
                mean((result_df$Model_4_Prediction - result_df$Actual_Wage)^2),
                mean((result_df$Model_5_Prediction - result_df$Actual_Wage)^2),
                mean((result_df$Model_6_Prediction - result_df$Actual_Wage)^2),
                mean((result_df$Model_7_Prediction - result_df$Actual_Wage)^2),
                mean((result_df$Model_8_Prediction - result_df$Actual_Wage)^2),
                mean((result_df$Average_Prediction - result_df$Actual_Wage)^2),
                mean((result_df$Average_Prediction_2 - result_df$Actual_Wage)^2))

# Calculate the R-squared values
r_squared_values <- c(summary(lm(Actual_Wage ~ Model_1_Prediction, data = result_df))$r.squared,
                      summary(lm(Actual_Wage ~ Model_2_Prediction, data = result_df))$r.squared,
                      summary(lm(Actual_Wage ~ Model_3_Prediction, data = result_df))$r.squared,
                      summary(lm(Actual_Wage ~ Model_4_Prediction, data = result_df))$r.squared,
                      summary(lm(Actual_Wage ~ Model_5_Prediction, data = result_df))$r.squared,
                      summary(lm(Actual_Wage ~ Model_6_Prediction, data = result_df))$r.squared,
                      summary(lm(Actual_Wage ~ Model_7_Prediction, data = result_df))$r.squared,
                      summary(lm(Actual_Wage ~ Model_8_Prediction, data = result_df))$r.squared,
                      summary(lm(Actual_Wage ~ Average_Prediction, data = result_df))$r.squared,
                      summary(lm(Actual_Wage ~ Average_Prediction_2, data = result_df))$r.squared)

# Create a data frame for the performance metrics
performance_df <- data.frame(Model = model_names, MSE = mse_values, R_squared = r_squared_values)
```

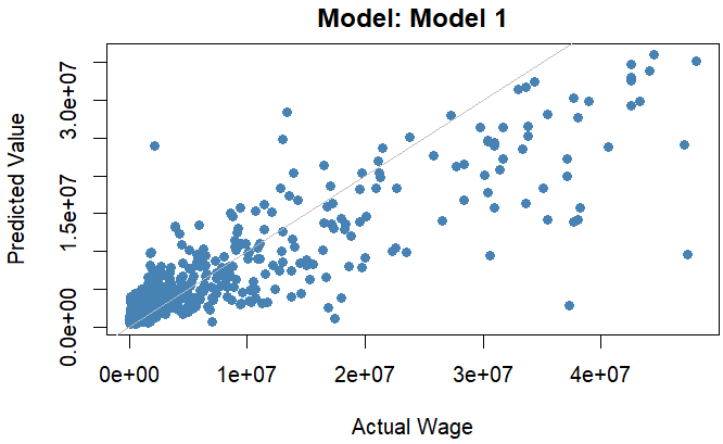
```
# Print the performance metrics table
print(performance_df)
```

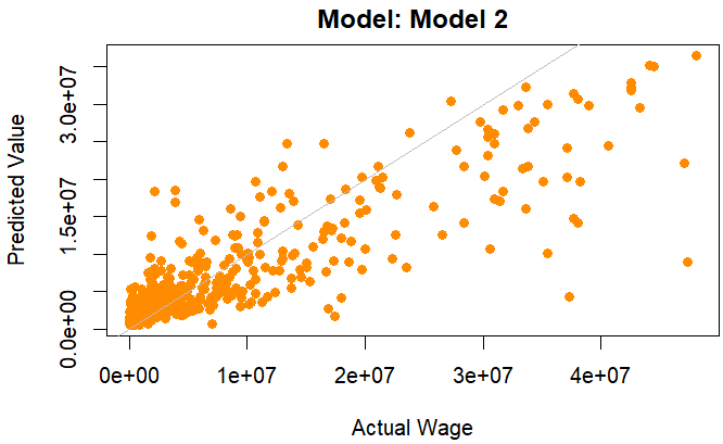
Model <chr>	MSE <dbl>	R_squared <dbl>
Model 1	3.644848e+13	0.7292750
Model 2	3.633560e+13	0.7279088
Model 3	3.639934e+13	0.7302590
Model 4	3.273062e+13	0.7406759
Model 5	3.436903e+13	0.7258124
Model 6	3.417681e+13	0.7229784
Model 7	3.243777e+13	0.7405071
Model 8	3.426900e+13	0.7245721
Average	3.173610e+13	0.7568047
Average 2	3.092944e+13	0.7625056

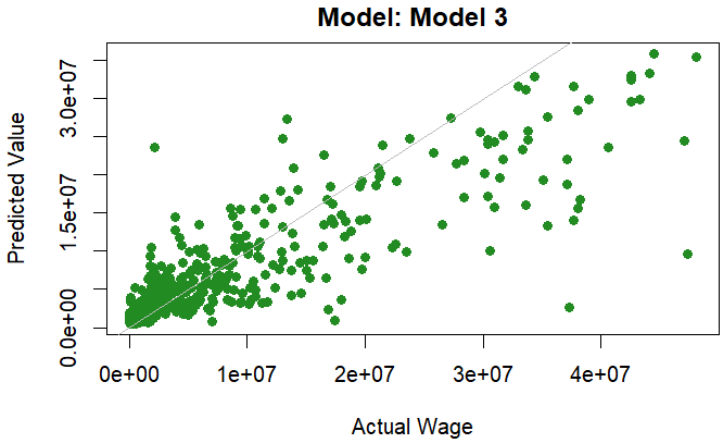
1-10 of 10 rows

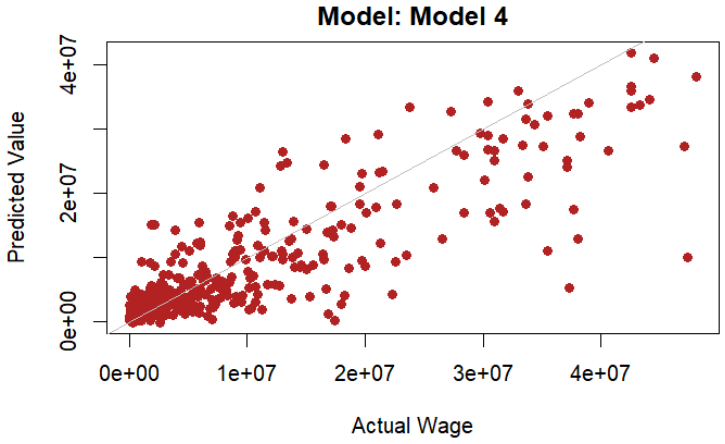
Hide

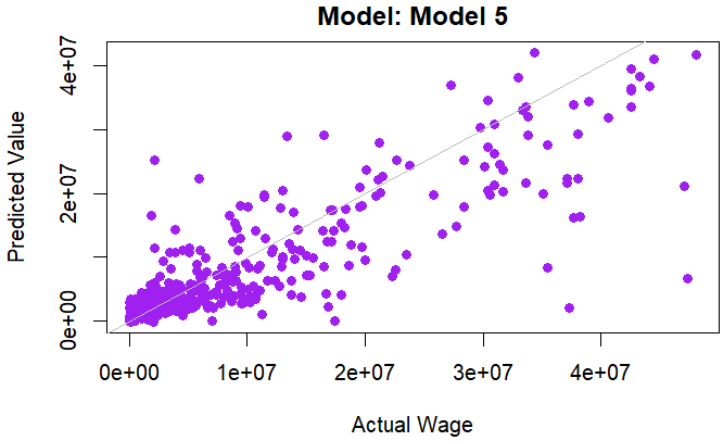
```
# Create scatter plots for each model
for (i in 3:12) {
  # Get the column name of the prediction variable
  pred_col <- colnames(result_df)[i]

  # Create a new plot
  plot(result_df$Actual_Wage, result_df[, pred_col],
       xlab = "Actual Wage", ylab = "Predicted Value",
       main = paste("Model:", model_names[i-2]),
       col = colors[i-2], pch = 16)

  # Add a 45-degree reference line
  abline(a = 0, b = 1, col = "gray")
}
```

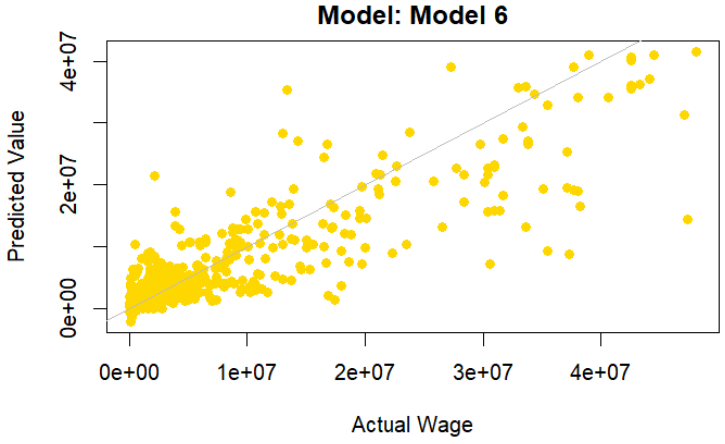


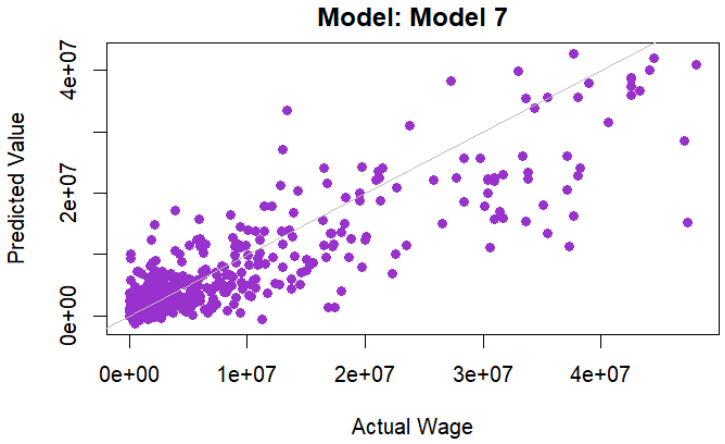


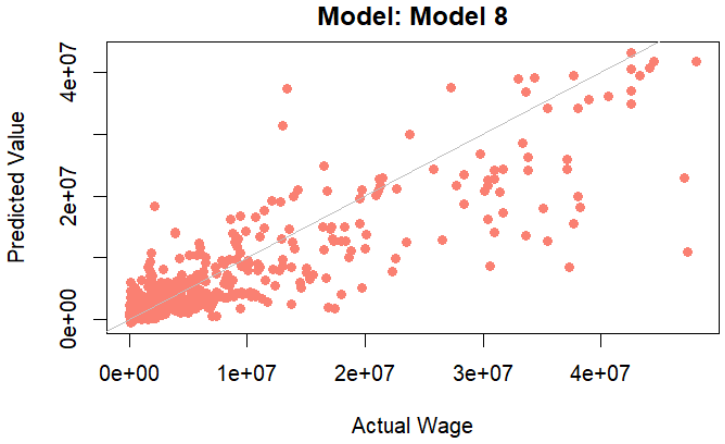

Model: Model 3
Predicted Value
0.0e+00
1.5e+07
3.0e+07
0e+00
1e+07
2e+07
3e+07
4e+07
Actual Wage

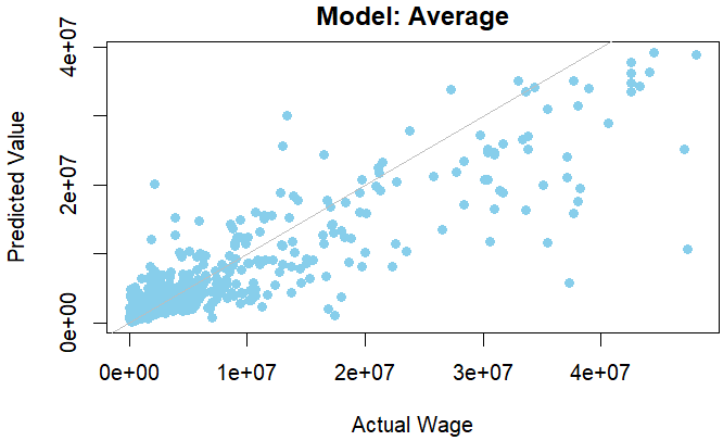

Model: Model 4
Predicted Value
0e+00
2e+07
4e+07
0e+00
1e+07
2e+07
3e+07
4e+07
Actual Wage

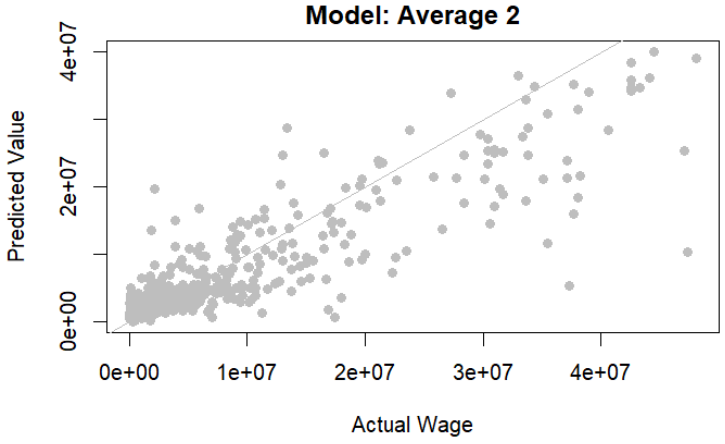

Model: Model 5
Predicted Value
0e+00
2e+07
4e+07
0e+00
1e+07
2e+07
3e+07
4e+07
Actual Wage


Model: Model 6
Predicted Value
0e+00
2e+07
4e+07
0e+00
1e+07
2e+07
3e+07
4e+07
Actual Wage


Model: Model 7
Predicted Value
0e+00
2e+07
4e+07
0e+00
1e+07
2e+07
3e+07
4e+07
Actual Wage

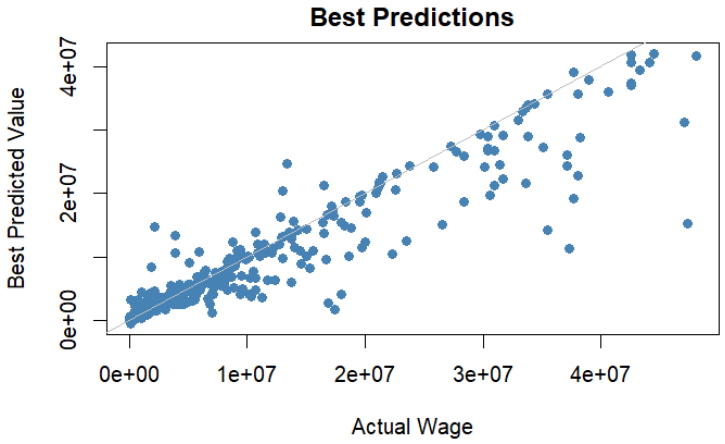

Model: Model 8
Predicted Value
0e+00
2e+07
4e+07
0e+00
1e+07
2e+07
3e+07
4e+07
Actual Wage


Model: Average
Predicted Value
0e+00
2e+07
4e+07
0e+00
1e+07
2e+07
3e+07
4e+07
Actual Wage


Model: Average 2
Predicted Value
0e+00
2e+07
4e+07
0e+00
1e+07
2e+07
3e+07
4e+07
Actual Wage

Hide

```
# Create scatter plot for best predictions
plot(best_predictions_df$Actual_Wage, best_predictions_df$Best_Prediction,
     xlab = "Actual Wage", ylab = "Best Predicted Value",
     main = "Best Predictions",
     col = "steelblue", pch = 16)

# Add a 45-degree reference line
abline(a = 0, b = 1, col = "gray")
```



Hide

```
# Create empty lists to store MSE values for each model
mse_train_list <- vector("list", length = 9)
mse_test_list <- vector("list", length = 9)

# Define the range of training sample sizes
sample_sizes <- seq(50, nrow(result_df), by = 50)

# Iterate over each model and average predictions
for (i in 1:9) {
  # Initialize empty vectors to store MSE values for the current model
  mse_train <- numeric()
  mse_test <- numeric()

  # Iterate over different training sample sizes
  for (size in sample_sizes) {
    # Subset the data for the current sample size
    subset_data <- result_df[1:size, ]

    if (i <= 8) {
      # Train the model using the subset data
      model <- lm(Actual_Wage ~ get(paste0("Model_", i, "_Prediction")), data = subset_data)

      # Make predictions on the training set
      train_predictions <- predict(model, subset_data)

      # Calculate the MSE on the training set
      mse_train <- c(mse_train, mean((subset_data$Actual_Wage - train_predictions)^2))

      # Make predictions on the full dataset
      test_predictions <- predict(model, result_df)

      # Calculate the MSE on the full dataset
      mse_test <- c(mse_test, mean((result_df$Actual_Wage - test_predictions)^2))
    } else {
      # Make predictions using the average prediction columns
      train_predictions <- subset_data[, grep("Average_Prediction", colnames(subset_data))]
      test_predictions <- result_df[, grep("Average_Prediction", colnames(result_df))]

      # Calculate the MSE on the training set
      mse_train <- c(mse_train, mean((subset_data$Actual_Wage - rowMeans(train_predictions))^2))

      # Calculate the MSE on the full dataset
      mse_test <- c(mse_test, mean((result_df$Actual_Wage - rowMeans(test_predictions))^2))
    }
  }

  # Store the MSE values in the corresponding lists
  mse_train_list[[i]] <- mse_train
  mse_test_list[[i]] <- mse_test
}

# Plot the learning curves for each model and average predictions
plot(NULL, xlim = c(0, max(sample_sizes)), ylim = c(0, max(unlist(mse_train_list, mse_test_list))),
     xlab = "Training Sample Size", ylab = "MSE", main = "Learning Curves - Mean Squared Error")

# Add learning curves for each model
colors <- rainbow(9)
```

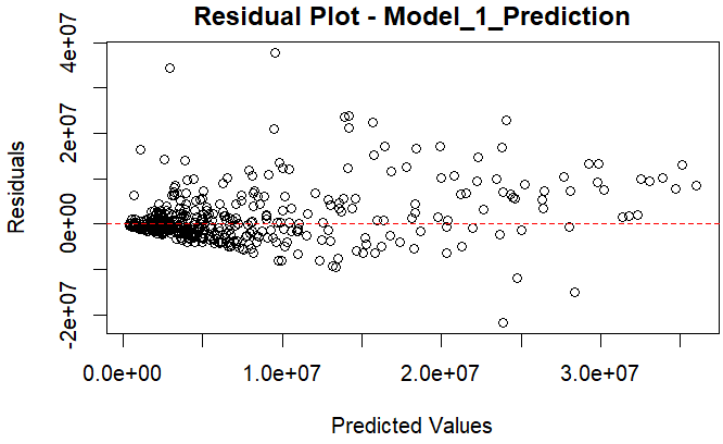
```
legend_labels <- c(paste0("Model ", 1:8), "Average", "Average 2")
for (i in 1:9) {
  lines(sample_sizes, mse_train_list[[i]], col = colors[i])
  lines(sample_sizes, mse_test_list[[i]], col = colors[i], lty = 2)
}
```

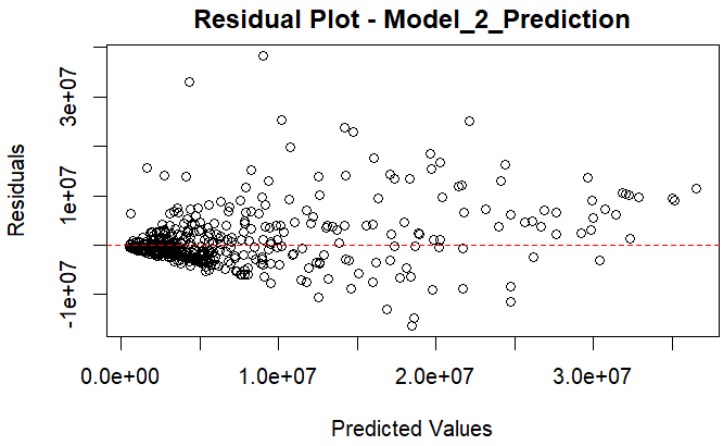
Hide

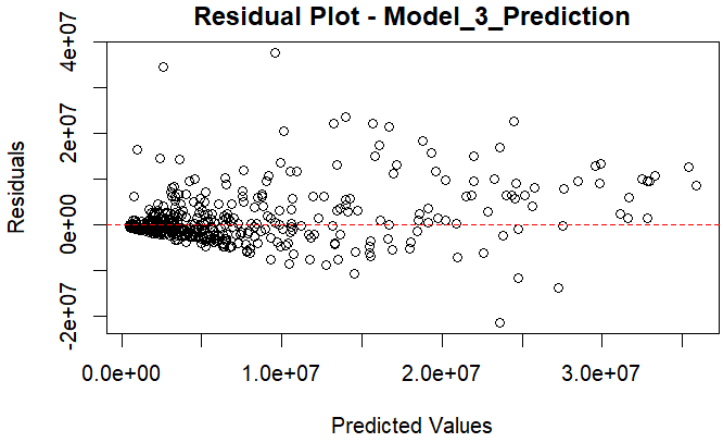
```
# Add legend
legend("topright", legend = legend_labels, col = colors, lty = c(1, 2), bty = "n")
```

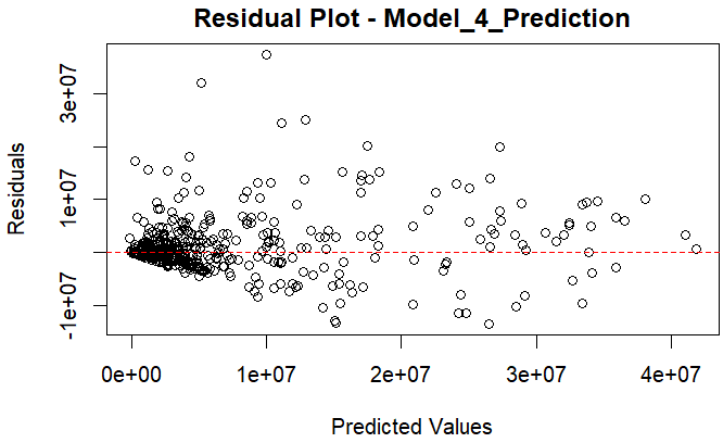
Hide

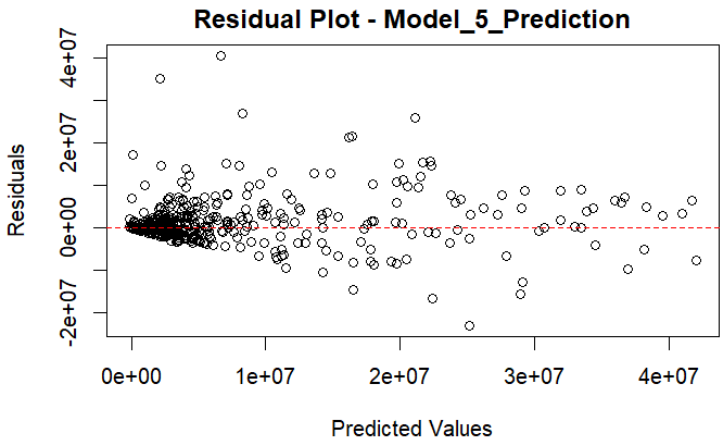
```
# Function to create residual plot for a model
create_residual_plot <- function(model_name, prediction_column) {
  # Calculate the residuals
  residuals <- result_df$Actual_Wage - result_df[[prediction_column]]

  # Plot the residuals against the predicted values
  plot(result_df[[prediction_column]], residuals,
       xlab = "Predicted Values", ylab = "Residuals",
       main = paste("Residual Plot -", model_name))

  # Add a horizontal line at y = 0
  abline(h = 0, col = "red", lty = 2)
}

# Iterate over the models in result_df and create residual plots
for (i in 3:ncol(result_df)) {
  model_name <- colnames(result_df)[i]
  prediction_column <- colnames(result_df)[i]
  create_residual_plot(model_name, prediction_column)
}
```

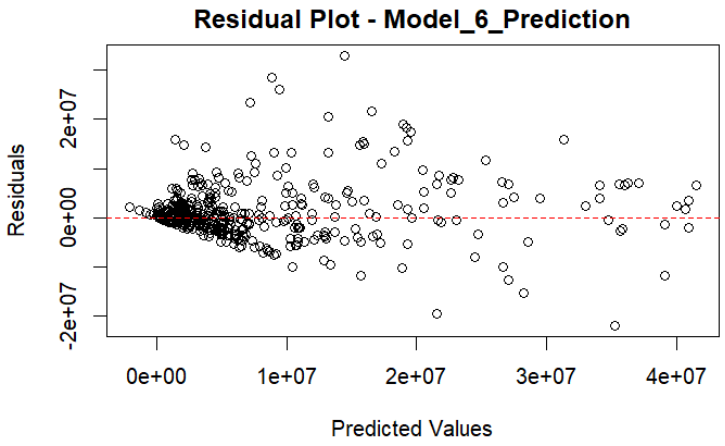


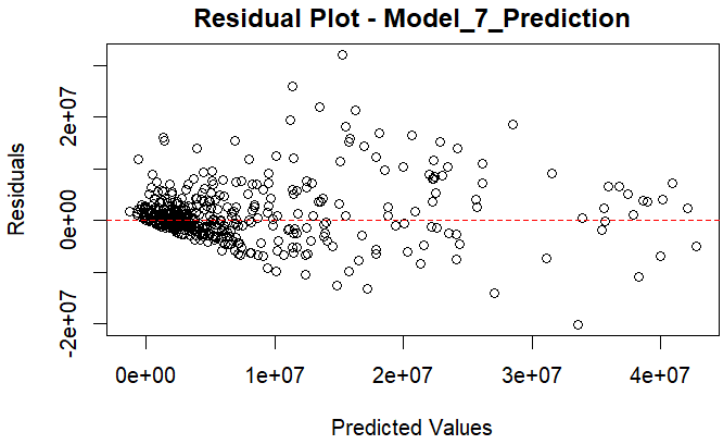

Residual Plot - Model_2_Prediction
Residuals
3e+07
1e+07
-1e+07
0.0e+00
1.0e+07
2.0e+07
3.0e+07
Predicted Values

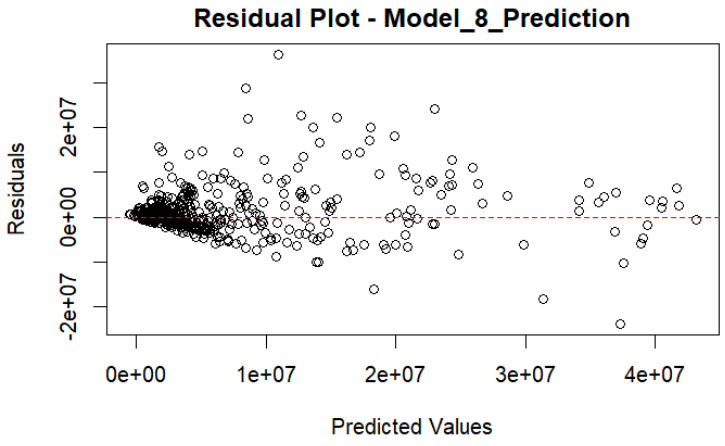

Residual Plot - Model_3_Prediction
Residuals
4e+07
2e+07
0e+00
-2e+07
0.0e+00
1.0e+07
2.0e+07
3.0e+07
Predicted Values

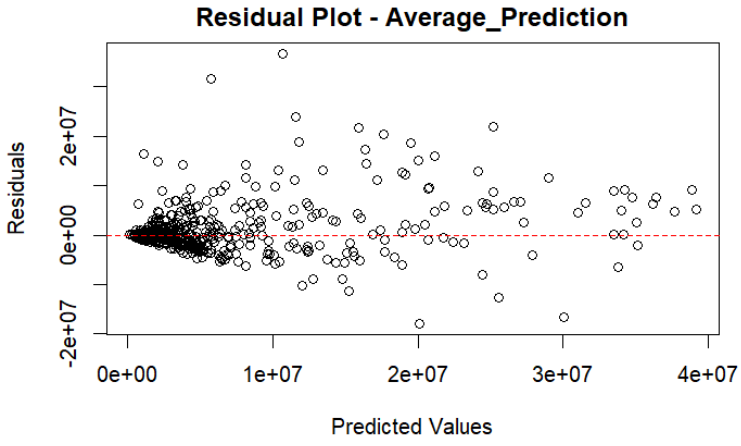

Residual Plot - Model_4_Prediction
Residuals
3e+07
1e+07
-1e+07
0e+00
1e+07
2e+07
3e+07
4e+07
Predicted Values


Residual Plot - Model_5_Prediction
Residuals
4e+07
2e+07
0e+00
-2e+07
0e+00
1e+07
2e+07
3e+07
4e+07
Predicted Values


Residual Plot - Model_6_Prediction
Residuals
-2e+07
0e+00
2e+07
0e+00
1e+07
2e+07
3e+07
4e+07
Predicted Values

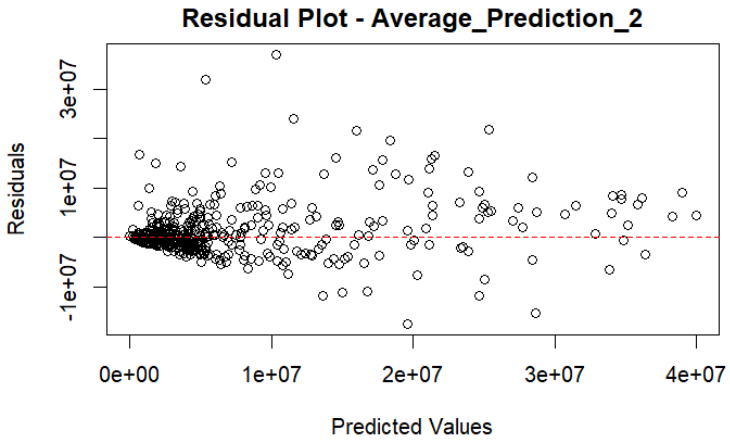

Residual Plot - Model_7_Prediction
Residuals
-2e+07
0e+00
2e+07
0e+00
1e+07
2e+07
3e+07
4e+07
Predicted Values


Residual Plot - Model_8_Prediction
Residuals
-2e+07
0e+00
2e+07
0e+00
1e+07
2e+07
3e+07
4e+07
Predicted Values


Residual Plot - Average_Prediction
Residuals
-2e+07
0e+00
2e+07
0e+00
1e+07
2e+07
3e+07
4e+07
Predicted Values

Hide

```
NA
```



The provided code includes several analyses and visualizations to evaluate the performance of different models and their predictions in terms of Mean Squared Error (MSE), R-squared values, learning curves, and residual plots. Here is a summary of the key findings:

1. Performance Comparison: The code calculates the MSE values for each model and the average predictions. It then creates a bar plot to compare the performance of the models based on MSE. Additionally, the R-squared values are calculated and plotted in a separate bar plot for comparison.
2. Scatter Plots: The code generates scatter plots for each model, showing the relationship between the actual wages and predicted values. A reference line is included to indicate perfect predictions.
3. Best Predictions: A scatter plot is created specifically for the best predictions, showing the relationship between actual wages and the best-predicted values.
4. Learning Curves: The code generates learning curves for each model, showing how the MSE changes as the training sample size increases. This helps assess the model's performance as more data is used for training.
5. Residual Plots: Residual plots are created for each model to visualize the distribution of residuals (the differences between actual wages and predicted values) against the predicted values. A horizontal line at y = 0 helps identify patterns or biases in the predictions.

Conclusion

Hide

```
# Extract the performance metrics for all models and average predictions
model_names <- c("Model 1", "Model 2", "Model 3", "Model 4", "Model 5", "Model 6", "Model 7", "Model 8", "Averag
e", "Average 2")
mse_values <- c(mean((result_df$Model_1_Prediction - result_df$Actual_Wage)^2),
                mean((result_df$Model_2_Prediction - result_df$Actual_Wage)^2),
                mean((result_df$Model_3_Prediction - result_df$Actual_Wage)^2),
                mean((result_df$Model_4_Prediction - result_df$Actual_Wage)^2),
                mean((result_df$Model_5_Prediction - result_df$Actual_Wage)^2),
                mean((result_df$Model_6_Prediction - result_df$Actual_Wage)^2),
                mean((result_df$Model_7_Prediction - result_df$Actual_Wage)^2),
                mean((result_df$Model_8_Prediction - result_df$Actual_Wage)^2),
                mean((result_df$Average_Prediction - result_df$Actual_Wage)^2),
                mean((result_df$Average_Prediction_2 - result_df$Actual_Wage)^2))

# Calculate the R-squared values
r_squared_values <- c(summary(lm(Actual_Wage ~ Model_1_Prediction, data = result_df))$r.squared,
                      summary(lm(Actual_Wage ~ Model_2_Prediction, data = result_df))$r.squared,
                      summary(lm(Actual_Wage ~ Model_3_Prediction, data = result_df))$r.squared,
                      summary(lm(Actual_Wage ~ Model_4_Prediction, data = result_df))$r.squared,
                      summary(lm(Actual_Wage ~ Model_5_Prediction, data = result_df))$r.squared,
                      summary(lm(Actual_Wage ~ Model_6_Prediction, data = result_df))$r.squared,
                      summary(lm(Actual_Wage ~ Model_7_Prediction, data = result_df))$r.squared,
                      summary(lm(Actual_Wage ~ Model_8_Prediction, data = result_df))$r.squared,
                      summary(lm(Actual_Wage ~ Average_Prediction, data = result_df))$r.squared,
                      summary(lm(Actual_Wage ~ Average_Prediction_2, data = result_df))$r.squared)

# Create a data frame for the performance metrics
performance_df <- data.frame(Model = model_names, MSE = mse_values, R_squared = r_squared_values)

# Print the performance metrics table
print(performance_df)
```

Model <chr>	MSE <dbl>	R_squared <dbl>
Model 1	3.644848e+13	0.7292750
Model 2	3.633560e+13	0.7279088
Model 3	3.639934e+13	0.7302590
Model 4	3.273062e+13	0.7406759
Model 5	3.436903e+13	0.7258124
Model 6	3.417681e+13	0.7229784
Model 7	3.243777e+13	0.7405071

Model 8	3.426900e+13	0.7245721
Average	3.173610e+13	0.7568047
Average 2	3.092944e+13	0.7625056

1-10 of 10 rows

Hide

```
NA
```

1. Model 1:
    - MSE: 3.644848e+13
    - R-squared: 0.7292750
2. Model 2:
    - MSE: 3.633560e+13
    - R-squared: 0.7279088
3. Model 3:
    - MSE: 3.639934e+13
    - R-squared: 0.7302590
4. Model 4:
    - MSE: 3.273062e+13
    - R-squared: 0.7406759
5. Model 5:
    - MSE: 3.436903e+13
    - R-squared: 0.7258124
6. Model 6:
    - MSE: 3.417681e+13
    - R-squared: 0.7229784
7. Model 7:
    - MSE: 3.243777e+13
    - R-squared: 0.7405071
8. Model 8:
    - MSE: 3.426900e+13
    - R-squared: 0.7245721

Average Metrics:

- Average MSE: 3.173610e+13
- Average R-squared: 0.7568047

Average Metrics 2:

- Average MSE: 3.092944e+13
- Average R-squared: 0.7625056

Here is an analysis of the results:

- MSE (Mean Squared Error) measures the average squared difference between the predicted and actual wage values. A lower MSE indicates better model performance. From the provided results, Model 4 has the lowest MSE, followed by Model 7 and Model 6.
- R-squared represents the proportion of variance in the wage predictions explained by the models. It ranges from 0 to 1, with higher values indicating a better fit. Based on the R-squared values, Model 4 has the highest R-squared, followed by Model 7 and Model 3.
- Comparing the average metrics, Average 2 (with an MSE of 3.092944e+13 and R-squared of 0.7625056) outperforms Average 1 (with an MSE of 3.173610e+13 and R-squared of 0.7568047). This suggests that the models used in Average 2 have better overall performance in predicting wages.
- It's important to note that the specific algorithm or method used for each model is not provided. Therefore, it's challenging to make direct comparisons between models without considering other factors such as model complexity, feature selection, and data preprocessing techniques.

Further analysis is required to understand the underlying factors contributing to the differences in performance between models. Evaluating additional metrics, conducting statistical significance tests, and considering the interpretability of the models can provide a more comprehensive analysis. Additionally, investigating the impact of specific features on the model's performance could help identify areas for improvement and potentially enhance the accuracy of the wage predictions.

In summary, Model 4, Model 7, and Model 6 demonstrate relatively better performance in terms of MSE and R-squared compared to other models. However, more in-depth analysis and exploration are necessary to draw definitive conclusions and improve the overall predictive power of the models.