Homework 8

Rong Sun

2024-11-24

Assignment Instructions

For each of the following problems,

(a) Define the problem: identify the problem class, detail the assumptions that are required (or that you are willing to make) to solve the problem, and describe the evaluation criteria you will use to evaluate results

(b) Define a solution: propose a model type, detail the selection criteria that you will use to choose a model, and describe the algorithm that will be implemented

(c) Implement the selected algorithm (using training/testing environments and k-fold cross validation as applicable) and evaluate model performance diagnostics, presenting visuals as needed.

(d) Interpret your results, presenting visuals as needed.

Please submit both a .Rmd file (you may work directly in this file), as well as a knitted .pdf to Canvas.

The NBA Draft

Every year, many aspiring college basketball players enter the NBA draft, a process by which teams in the NBA select new players for their teams. Teams will take turns, across two rounds of drafting (or recruiting) players to their roster – ordinarily, the "good" players are thus selected first, and the "bad" players remain to be selected.

Provided to you is a dataset, `nba_draft`, in the form of an .xlsx file, containing NBA draft results and player characteristics from the past 5 years of the NBA draft (data obtained from: https://www.basketball-reference.com/draft/) that you will use to answer the following questions.

You may use additional data that you find if you would like, just make sure that you submit those additional data files to Canvas along with your assignment. If any modifications are made to the data files themselves (i.e., not using code), please explicitly write out what you changes you have made your assignment files.

1. Draft Order

The order of the NBA draft is of great interest to the public, as players who are selected earlier on in the draft are typically expected to become star players across their careers in the NBA. Construct an approach by which the order of the 2024 NBA draft can be predicted based on 2019-2023 data, according to the following schematic:

(a) Define the problem

Problem Class:

This is a supervised learning problem that seeks to predict the NBA draft order (`pick`) for the 2024 draft class. The objective is to use historical player performance data and characteristics from the 2019–2023 drafts to build a predictive model, which will be provided with new player data from the 2024 draft class (excluding their draft order). The draft order (`pick`) is treated as a continuous target variable, meaning the model must generate numerical predictions corresponding to draft positions.

1. Data Provided to the Algorithm:

   - Training and Testing Data: The dataset includes historical NBA draft data from 2019–2023, split into training and testing sets to evaluate model performance.
     - Training Data: Historical NBA draft data from 2019–2023 is used to train the model. This data includes player performance metrics, advanced statistics, and corresponding draft positions (`pick`), which serve as the labeled target variable. The training data is a subset of the 2019–2023 dataset, with the remaining data reserved for testing.
     - Testing Data: A portion of the 2019–2023 data is withheld during training and used as the testing set to evaluate the model's performance. This ensures that the model is evaluated on data it has not seen during training, providing an unbiased estimate of its performance.
   - Prediction Data (2024 Draft Class): The 2024 draft class dataset includes player performance metrics and characteristics but excludes the actual draft order (`pick`). After the model is trained and validated using 2019–2023 data, it is applied to the 2024 data to predict draft positions for these players based on patterns learned from the historical data. This dataset is used for supervised prediction, where the model applies the relationships identified in the labeled training data.

2. Desired Conclusions:

   - The goal is to predict the draft position (`pick`) of each player in the 2024 NBA draft based on patterns observed in the 2019–2023 drafts. By identifying the relationships between player performance metrics and draft outcomes, the model will generate accurate predictions of the draft position (`pick`) for the 2024 class, providing insights into the draft order based on each player's performance and characteristics.
   - The conclusion will focus on ranking the 2024 draft class, identifying which players are likely to be selected earlier or later. This ranking reflects how players' attributes and achievements align with the patterns observed in the 2019–2023 drafts, enabling a better understanding of how teams might rank players based on their performance and characteristics, as well as how the draft order is shaped by historical trends.

3. Behavior of Algorithm:

   - The algorithm learns from historical data (2019–2023) by mapping input features (e.g., player performance metrics and advanced statistics) to the draft order (`pick`) as the target variable.
   - During testing, the model's accuracy is evaluated on unseen data from the 2019–2023 drafts to ensure it generalizes well.
   - Finally, the trained model is applied to the 2024 draft class to predict draft positions based on the relationships it learned, effectively capturing how teams historically value various player attributes when making draft decisions.

Assumptions:

1. Source of Data:

   - The dataset includes player characteristics and performance metrics that are most relevant to NBA draft decisions. These features are organized into multiple categories:

- Totals: Cumulative statistics like Games Played (G), Minutes Played (MP), Points (PTS), Total Rebounds (TRB), and Assists (AST);
- Shooting Percentages: Efficiency metrics such as Field Goal Percentage (FG%), 3-Point Percentage (3P%), and Free Throw Percentage (FT%);
- Per Game Statistics: Average performance metrics like Minutes Played Per Game (MP), Points Per Game (PTS), Total Rebounds Per Game (TRB), and Assists Per Game (AST);
- Advanced Statistics: Metrics such as Win Shares (WS), Win Shares Per 48 Minutes (WS/48), Box Plus/Minus (BPM), and Value over Replacement Player (VORP), which provide deeper insights into a player's overall impact and value.

- The draft order is primarily influenced by these measurable features, which reflect players' performance, efficiency, and value during games. Unmeasured factors, such as team preferences or off-court considerations, are assumed to have a minor impact and do not significantly bias predictions.

2. Assumptions about the Problem:

- Consistency in Data: Relationships between performance metrics and draft positions are assumed stable over time, making historical data (2019–2023) a reliable foundation for predicting the 2024 draft positions. The included metrics align with commonly analyzed player attributes in professional scouting and decision-making.
- Evaluation Criteria Stability: The relationship between the input features (e.g., performance metrics) and the draft order is assumed to be stable across years. This implies that evaluation criteria for players have not significantly changed from 2019–2023 and will remain consistent in 2024.
- Handling of Missing Data: Missing values in the dataset are addressed through imputation (e.g., replaced with medians) to maintain data integrity. It is assumed that such imputation methods adequately capture the underlying data distribution and do not introduce biases or distort the predictions.
- Feature Relevance: The model assumes that the dataset includes all relevant features necessary for predicting draft order, and that irrelevant or noisy features are minimal, ensuring reliable model performance.

Evaluation Criteria:

1. Goal of the Prediction Model:

- The goal is to predict the draft order (a continuous variable, `pick`) of each player in the 2024 NBA draft as accurately as possible. This prediction is based on relationships between player performance metrics and the draft outcomes observed in the historical 2019–2023 data.

2. Query Evaluation:

- Individual predictions for each player's draft position will be evaluated using the Mean Absolute Error (MAE) to measure how close the predicted draft positions are to the actual positions (i.e., the average magnitude of prediction errors between predicted and actual draft positions). Lower MAE values indicate better performance, as it reflects a closer alignment between predicted and actual values.

3. Model Performance:

The overall model performance will be evaluated using Mean Absolute Error (MAE) for prediction accuracy, Spearman Correlation for rank-order consistency, and visual tools like scatter and residual plots to assess model fit and bias.

- Mean Absolute Error (MAE): This is the primary evaluation metric for prediction accuracy. It measures prediction accuracy at the individual query level, helping to assess how close the predicted draft positions are to the true positions in the 2024 data.

- Spearman Correlation: This secondary metric evaluates the rank-order consistency between predicted and actual draft positions, measuring how well the model's predicted draft positions align with the actual draft order of 2024 (i.e., the model's ability to rank players correctly in relation to each other). A higher Spearman correlation suggests better rank-order agreement.
- Visualizations: Scatter plots (actual vs. predicted values) and residual plots will provide insights into model performance and potential biases. These visual tools help assess how well the model fits the data and whether systematic errors exist.

(b) Define a solution

Model Type:

1. Model Selection:

- Random Forest Regression Model will be used to understand the relationships between the input features (e.g., player performance metrics) and the target variable (`pick`, draft position) and predict the NBA draft order for the 2024 class. This regression model will be trained to learn patterns from the input data (2019–2023 player performance metrics and characteristics) and will be applied to make predictions for unseen players in the 2024 draft class.

2. Aspects of the Data to Be Modeled:

The model will focus on capturing the relationships between a range of input features that are critical in determining draft order, such as:

- Totals: Aggregated statistics such as total points, total assists, and total rebounds are included to reflect a player's cumulative contributions throughout their career or season. These features highlight the overall productivity and ability to make a sustained impact.
- Shooting Percentages: Efficiency metrics, including field goal percentage (FG%), 3-Point field goal percentage (3P%), and free throw percentage (FT%), assess a player's scoring accuracy and shooting proficiency. These metrics are crucial for evaluating offensive efficiency and shooting reliability.
- Per Game Averages: Metrics such as average points per game, average rebounds per game, and average assists per game provide insights into a player's consistency and impact on a game-by-game basis. These features capture how consistently a player performs regardless of total playing time.
- Advanced Statistics: High-level metrics like Win Shares (WS), Win Shares per 48 minutes (WS/48), Box Plus/Minus (BPM), and Value Over Replacement Player (VORP) are incorporated to measure a player's overall impact on team success and their relative contribution compared to average players. These statistics go beyond basic box score metrics to capture a player's value and efficiency.

These features provide a detailed representation of player performance and attributes, ensuring comprehensive modeling.

3. Prediction Usage:

- The trained model will use these features to predict draft positions (`pick`) for players in the 2024 NBA draft class. The predictions aim to provide actionable insights by translating player performance metrics into expected draft outcomes. Key applications include:
  - Draft Strategy: Teams can identify players who are likely to be undervalued or overvalued based on predicted draft positions versus their actual performance metrics.

    - Performance Benchmarking: The predictions allow teams and analysts to benchmark players against others in the draft class, providing a data-driven perspective on player rankings.
    - Decision Support: The model helps decision-makers understand how various attributes contribute to a player's draft potential, offering valuable guidance in scouting and selection processes.

Model Class:

1. Parametric Class:

- The Random Forest Regression algorithm, a non-parametric ensemble tree-based model class, will be used. It constructs multiple decision trees during training, combining their outputs for robust and accurate predictions. The regression version of Random Forest averages individual tree predictions to estimate the draft order. This is a regression task because the target variable (draft pick) is numeric and continuous, not categorical. Unlike parametric models like linear regression, Random Forest does not assume a specific data distribution (e.g., normality) and can model non-linear relationships effectively.

2. Model Selection Criteria:

The specific Random Forest regression model will be chosen based on optimization of hyperparameters (e.g., `ntree`, `mtry`, `min.node.size`) to minimize prediction error and maximize ranking consistency. A specific Random Forest model will be selected based on the following criteria:

- Primary Evaluation Metric: Mean Absolute Error (MAE) to evaluate the precision of predicted draft orders (i.e., prediction accuracy). MAE directly measures the average magnitude of prediction errors in the same units as the target variable (e.g., draft order). This makes it highly interpretable for assessing how close the model's predictions are to actual draft positions.
- Secondary Evaluation Metric: Spearman Correlation to assess how well predicted ranks align with actual ranks (i.e., rank-order consistency between predicted and actual draft positions). While MAE measures precision, Spearman correlation evaluates the model's effectiveness in predicting the overall order.
- Model Interpretability: Feature importance scores provided by Random Forest help identify which player attributes most influence the draft order, enabling actionable recommendations for scouts and analysts.

Random Forest Regression is preferred for its versatility in handling both numeric and categorical data, resistance to overfitting by averaging predictions across multiple trees, and its ability to model complex, non-linear interactions between features.

3. Error Metric Selection (Choose MAE for NBA Draft Predictions):

- Interpretability: MAE gives a straightforward interpretation by measuring the average absolute error in predicted draft positions. In this case, we can easily understand how much, on average, the predictions are off from the actual draft positions.
- Robustness to Outliers: While extreme outliers may exist (e.g., predicting a player's draft position very far from the actual), MAE doesn't disproportionately penalize large errors as RMSE does. This is useful in cases where large deviations may be rare or less critical, and we want to treat all errors equally.
- Alignment with the Problem's Objective: The goal of the NBA draft prediction is to be as close as possible to the actual draft order. Since we are interested in the absolute difference between the predicted and actual draft positions, MAE is a more natural fit than RMSE, which tends to penalize large errors more heavily.
- Predicting Order, Not Magnitude: In NBA draft predictions, it's the relative positions (the order) that matter more than the magnitude of the errors. For example, being off by one pick can be just as important as being off by 10 picks, depending on the context. MAE treats all deviations uniformly, making it a more appropriate metric when order consistency matters.

Algorithm:

The computational process for fitting the Random Forest Regression model involves the following steps:

Step 1: Data Preparation

- Handle Missing Data: Impute missing values using the median for numeric features to ensure completeness of the dataset.
- Feature Engineering: Focus on ensuring the representativeness of the selected features. If necessary, apply scaling (e.g., centering and standardization) to improve comparability across features.
- Train-Test Split: Split the data from 2019–2023 into an 80:20 ratio for training and testing to evaluate the model's generalization ability. The 2024 draft data is reserved exclusively for prediction.

Step 2: Model Training

- Model Implementation: The `ranger` package is used for computational efficiency in training Random Forest models on large datasets.
- Initialization (Set initial parameters for the Random Forest model):
  - `ntree = 500`: The number of decision trees (default is 500).
  - `mtry = floor(number of predictors / 3)`: The number of features sampled at each split for regression (default is the floor of 1/3rd of the total predictors).
- Cross-validation Procedure: Perform 10-fold cross-validation on the training data to evaluate the model performance by repeatedly training on 90% of the data and testing on the remaining 10%. It ensures that the model generalizes well and minimizes overfitting risks.

Step 3: Hyperparameter Tuning

- Perform a grid search to optimize hyperparameters:
  - `ntree`: Experiment with different values for the number of trees (e.g., 100, 500, and 1000) to assess which provides the best model performance.
  - `mtry`: Vary the number of features to be considered at each split. Start with the default value and explore a range from 2 to the total number of features in the dataset.
  - `min.node.size`: Tune the minimum number of samples required at leaf nodes. This parameter helps control the tree depth and can prevent overfitting. Common values are 5, 10, and 15.
- Model Evaluation Metrics: Evaluate performance for each combination of hyperparameters using Mean Absolute Error (MAE) during cross-validation. MAE ensures the model is selected based on its precision in predicting draft positions.

Step 4: Model Evaluation

Evaluate the trained models on the test data (20% subset of 2019–2023 data) to identify the best-performing model:

- Mean Absolute Error (MAE): Measure the prediction accuracy by quantifying the average difference between predicted and actual draft positions. The model with the lowest MAE on the test data is selected as the finalized model.
- Spearman Correlation: Assess the rank-order consistency between predicted and actual draft positions, providing additional insight into model performance.

Generate diagnostic plots to further analyze model performance:

- Scatter Plots: Compare actual vs. predicted draft positions to visually evaluate the model's accuracy.
- Residual Plots: Analyze residuals (prediction errors) to detect potential biases or areas where the model may underperform.

Step 5: Prediction and Analysis

- Apply the Final Model to 2024 Data: Use the best-performing model (selected with the lowest MAE on the test data) to predict draft orders for the 2024 dataset. This ensures that the selected model is both accurate and generalizable.
- Save Predictions for Further Analysis: Store the predicted draft positions for the 2024 draft class, enabling further evaluation and comparison with actual draft outcomes.
- Feature Importance Analysis: Analyze which player attributes most influence the predicted draft outcomes by examining feature importance scores. This helps interpret the model and identify the metrics most predictive of draft positions.

(c) Implement the selected algorithm evaluate model performance diagnostics, presenting visuals as needed.

Model Application

```r
# Load required libraries
library(readxl)        # For reading Excel files
library(caret)         # For machine learning utilities
library(dplyr)         # Data manipulation
library(Metrics)       # For Mean Absolute Error (MAE) calculation
library(ggplot2)       # Data visualization
library(ranger)        # Efficient Random Forest implementation
library(corrplot)      # For correlation analysis
library(randomForest)  # Random Forest base package
library(rsample)       # For reproducible data splitting

# Step 1: Data Preparation ----------------------------------------------

# Load the dataset containing NBA draft data
nba_data <- read_excel("nba_draft.xlsx")

# 1.1 Filter Data by Year
# Split the dataset into:
# - Training and testing data (2019-2023)
# - Prediction data (2024)
nba_train_test <- nba_data %>% filter(year < 2024)  # 2019-2023 Training/testing data
nba_predict <- nba_data %>% filter(year == 2024)    # 2024 prediction data

# 1.2 Handle Missing Values
# Replace missing numeric values with the median for both train/test and prediction datasets
nba_train_test <- nba_train_test %>%
  mutate(across(where(is.numeric), ~ifelse(is.na(.), median(., na.rm = TRUE), .)))
nba_predict <- nba_predict %>%
  mutate(across(where(is.numeric), ~ifelse(is.na(.), median(., na.rm = TRUE), .)))

# 1.3 Train-Test Split
# Split the 2019-2023 data into training (80%) and testing (20%) sets
set.seed(1234)  # Set seed for reproducibility
# train_index <- createDataPartition(nba_train_test$pick, p = 0.8, list = FALSE)
# nba_train <- nba_train_test[train_index, ]  # Training data
# nba_test <- nba_train_test[-train_index, ]  # Testing data
data_split <- initial_split(nba_train_test, prop = 0.8, strata = "pick")  # Stratify by 'pick'
nba_train <- training(data_split)  # Training data
nba_test <- testing(data_split)    # Testing data

# 1.4 Prepare Features and Target Variables
# Exclude non-numeric columns and the target variable ('pick') to create feature matrices
X_train <- nba_train %>% dplyr::select(-pick, -player_name, -team, -college, -year)
y_train <- nba_train$pick  # Extract the target variable ('pick') for training

X_test <- nba_test %>% dplyr::select(-pick, -player_name, -team, -college, -year)
y_test <- nba_test$pick  # Extract the target variable ('pick') for testing

X_predict <- nba_predict %>% dplyr::select(-pick, -player_name, -team, -college, -year)
y_predict <- nba_predict$pick  # Extract the actual draft positions in 2024

# 1.5 Feature Scaling
# Center and scale numeric features for comparability
scaler <- preProcess(X_train, method = c("center", "scale"))  # Center and scale features
X_train <- predict(scaler, X_train)  # Apply scaling to the training set
X_test <- predict(scaler, X_test)  # Apply scaling to the test set
X_predict <- predict(scaler, X_predict)  # Apply scaling to the prediction set

# Ensure data is in data.frame format
X_train <- as.data.frame(X_train)
X_test <- as.data.frame(X_test)

# Impute missing values with the median of each column
X_train <- X_train %>%
  mutate(across(where(is.numeric), ~ifelse(is.na(.), median(., na.rm = TRUE), .)))
y_train <- ifelse(is.na(y_train), median(y_train, na.rm = TRUE), y_train)

# Step 2: Model Training -----------------------------------------------

# 2.1 Define Cross-Validation Strategy
# 10-fold cross-validation for robust model evaluation
control_rf <- trainControl(
  method = "cv",    # Cross-validation
  number = 10,      # Number of folds (10-fold cross-validation)
  verboseIter = FALSE  # Suppress detailed output
)

# 2.2 Default and Tuned Hyperparameters
# Default number of features sampled at each split (mtry)
default_mtry <- floor(ncol(X_train) / 3)  # Floor to ensure integer value

# Define a grid of hyperparameters for tuning
tune_grid <- expand.grid(
  # Number of features sampled at each split
  .mtry = c(default_mtry, seq(2, ncol(X_train), by = 2)),  # Features sampled
  .splitrule = "variance",          # Split rule for regression
  .min.node.size = c(5, 10, 15))   # Minimum samples in leaf nodes
```

```r
# Step 3: Hyperparameter Tuning ------------------------------------------

# Experiment with different numbers of trees: ntree values (100, 500, 1000)
ntree_values <- c(100, 500, 1000)  # Different ntree values to test
rf_results <- list()  # Store results for each ntree value

set.seed(1234)  # Ensures reproducible cross-validation folds

# Train models and validate with test data
for (ntree_val in ntree_values) {
  rf_tuned <- train(
    x = X_train, y = y_train,
    method = "ranger",           # Use the 'ranger' implementation
    metric = "MAE",              # Evaluate using Mean Absolute Error (MAE)
    trControl = control_rf,      # Cross-validation control
    tuneGrid = tune_grid,        # Hyperparameter grid
    importance = "impurity",     # Enable variable importance calculation
    num.trees = ntree_val        # Explicitly set ntree values for each iteration
  )

  # Validate the tuned model on the test set
  y_test_pred <- predict(rf_tuned, X_test)
  test_mae <- mae(y_test, y_test_pred)
  test_spearman <- cor(y_test, y_test_pred, method = "spearman")

  # Store the tuned model, test results, and ntree values
  rf_results[[as.character(ntree_val)]] <- list(
    model = rf_tuned,
    ntree = ntree_val,
    test_mae = test_mae,
    test_spearman = test_spearman
  )
}

# Step 4: Model Evaluation -----------------------------------------------

# Identify the finalized model based on the test MAE
# (Compare results for different ntree values)
final_model <- NULL  # Store the final model with the lowest MAE
final_mae <- Inf  # Initialize the final MAE as infinity
final_ntree <- NULL  # Store the final ntree value

for (ntree_val in names(rf_results)) {
  # Print results for each ntree value
  cat("Results for ntree =", ntree_val, ":\n")
  print(rf_results[[ntree_val]]$model)  # Print model results
  cat("\n")

  # Get the MAE of the current model
  current_mae <- rf_results[[ntree_val]]$test_mae

  # Check if this model has the lowest MAE so far
  if (current_mae < final_mae) {
    final_mae <- current_mae
    final_model <- rf_results[[ntree_val]]$model  # Update the finalized model
    final_ntree <- ntree_val  # Update the finalized ntree value
  }
}
```

```
## Results for ntree = 100 :
## Random Forest
##
## 236 samples
##  14 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 213, 213, 213, 212, 213, 212, ...
## Resampling results across tuning parameters:
##
##   mtry  min.node.size  RMSE      Rsquared   MAE
##    2     5             12.48067  0.4539232  10.212974
##    2    10             12.49923  0.4516354  10.244335
##    2    15             12.50222  0.4524812  10.223348
##    4     5             12.48465  0.4532126  10.030797
##    4    10             12.33941  0.4655046  10.025215
##    4    15             12.31878  0.4698387  10.045482
##    6     5             12.68807  0.4373698  10.306596
##    6    10             12.34824  0.4626122   9.966170
##    6    15             12.39894  0.4612440  10.038936
##    8     5             12.39681  0.4617320  10.035224
##    8    10             12.53369  0.4533883  10.081833
##    8    15             12.35967  0.4649310  10.026655
##   10     5             12.48781  0.4580294  10.022943
##   10    10             12.43584  0.4579272  10.029346
##   10    15             12.50118  0.4554126  10.053532
##   12     5             12.54338  0.4514193   9.979279
##   12    10             12.52807  0.4514246  10.048988
##   12    15             12.46745  0.4555656   9.987202
##   14     5             12.57245  0.4493993  10.133602
##   14    10             12.66108  0.4416779  10.160850
##   14    15             12.63162  0.4448297  10.109876
##
## Tuning parameter 'splitrule' was held constant at a value of variance
## MAE was used to select the optimal model using the smallest value.
## The final values used for the model were mtry = 6, splitrule = variance
##  and min.node.size = 10.
##
## Results for ntree = 500 :
## Random Forest
##
## 236 samples
##  14 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 212, 212, 213, 212, 213, 212, ...
## Resampling results across tuning parameters:
##
##   mtry  min.node.size  RMSE      Rsquared   MAE
##    2     5             12.55604  0.4600497  10.120855
##    2    10             12.54107  0.4594997  10.142855
##    2    15             12.57437  0.4575969  10.191630
##    4     5             12.52639  0.4597845  10.048883
##    4    10             12.50488  0.4637115  10.099693
##    4    15             12.44380  0.4693681  10.046721
##    6     5             12.54141  0.4583744  10.064348
##    6    10             12.50154  0.4621359  10.024590
##    6    15             12.43577  0.4687089  10.011384
##    8     5             12.54538  0.4575729  10.060996
##    8    10             12.53276  0.4579535  10.056378
##    8    15             12.50551  0.4612261  10.033508
##   10     5             12.59041  0.4538036  10.078828
##   10    10             12.52031  0.4612157  10.032257
##   10    15             12.55249  0.4580409  10.021540
##   12     5             12.54790  0.4577225  10.027810
##   12    10             12.53006  0.4586379   9.987403
##   12    15             12.56483  0.4570108  10.030723
##   14     5             12.55711  0.4558493   9.981604
##   14    10             12.57572  0.4555476  10.020042
##   14    15             12.51340  0.4609477  10.001463
##
## Tuning parameter 'splitrule' was held constant at a value of variance
## MAE was used to select the optimal model using the smallest value.
## The final values used for the model were mtry = 14, splitrule = variance
##  and min.node.size = 5.
##
## Results for ntree = 1000 :
## Random Forest
##
## 236 samples
##  14 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 213, 212, 212, 212, 212, 212, ...
## Resampling results across tuning parameters:
##
##   mtry  min.node.size  RMSE      Rsquared   MAE
##    2     5             12.60223  0.4572458  10.26315
##    2    10             12.59773  0.4586165  10.28620
##    2    15             12.53792  0.4633760  10.22494
##    4     5             12.57472  0.4613181  10.15295
##    4    10             12.52896  0.4644347  10.13500
##    4    15             12.46232  0.4710452  10.10604
##    6     5             12.54657  0.4636629  10.08711
##    6    10             12.46643  0.4709830  10.05075
##    6    15             12.44391  0.4744824  10.04617
##    8     5             12.59995  0.4612774  10.14063
##    8    10             12.52387  0.4682582  10.06593
##    8    15             12.48542  0.4703937  10.03193
##   10     5             12.55809  0.4656773  10.06051
##   10    10             12.52167  0.4691880  10.04441
##   10    15             12.45997  0.4737892  10.00500
##   12     5             12.54262  0.4666203  10.01513
##   12    10             12.48883  0.4731955  10.00185
##   12    15             12.49681  0.4715534  10.02035
##   14     5             12.55785  0.4671170  10.01242
##   14    10             12.53759  0.4693275  10.04287
##   14    15             12.50007  0.4724348  10.01606
##
## Tuning parameter 'splitrule' was held constant at a value of variance
## MAE was used to select the optimal model using the smallest value.
## The final values used for the model were mtry = 12, splitrule = variance
##  and min.node.size = 10.
```

```
# Display the finalized model's performance on the test set
cat("\nFinal Model Summary:\n")
```

```
##
## Final Model Summary:
```

```
cat("Final Model Based on Test MAE: ntree =", final_ntree, "\n")
```

```
## Final Model Based on Test MAE: ntree = 500
```

```
cat("Final Model Test MAE:", final_mae, "\n")
```

```
## Final Model Test MAE: 11.10188
```

```
cat("Final Model Test Spearman Correlation:", rf_results[[final_ntree]]$test_spearman, "\n")
```

```
## Final Model Test Spearman Correlation: 0.6439236
```

```
# Step 5: Prediction and Analysis ----------------------------------------

# Use the finalized model to predict 2024 draft positions
y_2024_pred_rf <- predict(final_model, X_predict)  # Predict 2024 draft order
mae_test_rf <- mae(y_predict, y_2024_pred_rf)  # Calculate Mean Absolute Error (MAE)
# Calculate Spearman Correlation Coefficient
spearman_corr_test_rf <- cor(y_predict, y_2024_pred_rf, method = "spearman")

# Display prediction performance on the 2024 data
cat("\nPerformance on 2024 Data:\n")
```

```
##
## Performance on 2024 Data:
```

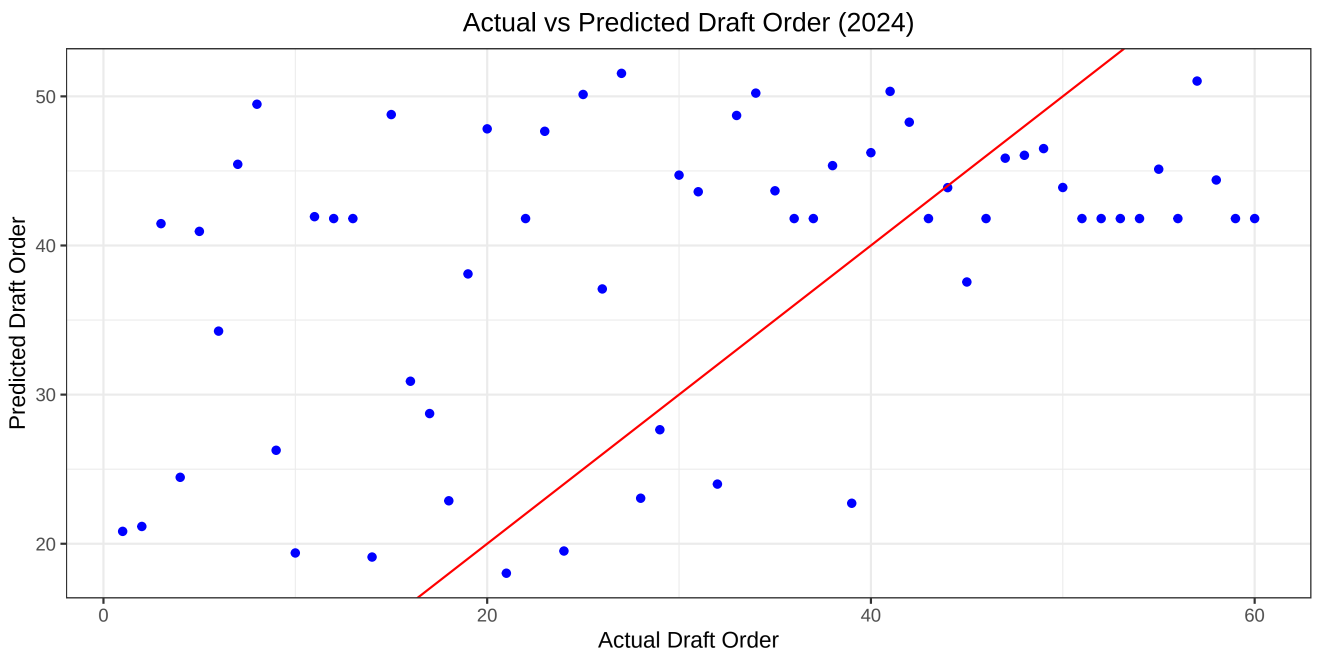
```
cat("2024 Draft Prediction Mean Absolute Error:", mae_test_rf, "\n")
```

```
## 2024 Draft Prediction Mean Absolute Error: 14.55034
```

```
cat("2024 Draft Prediction Spearman Correlation:", spearman_corr_test_rf, "\n")
```

```
## 2024 Draft Prediction Spearman Correlation: 0.3659773
```

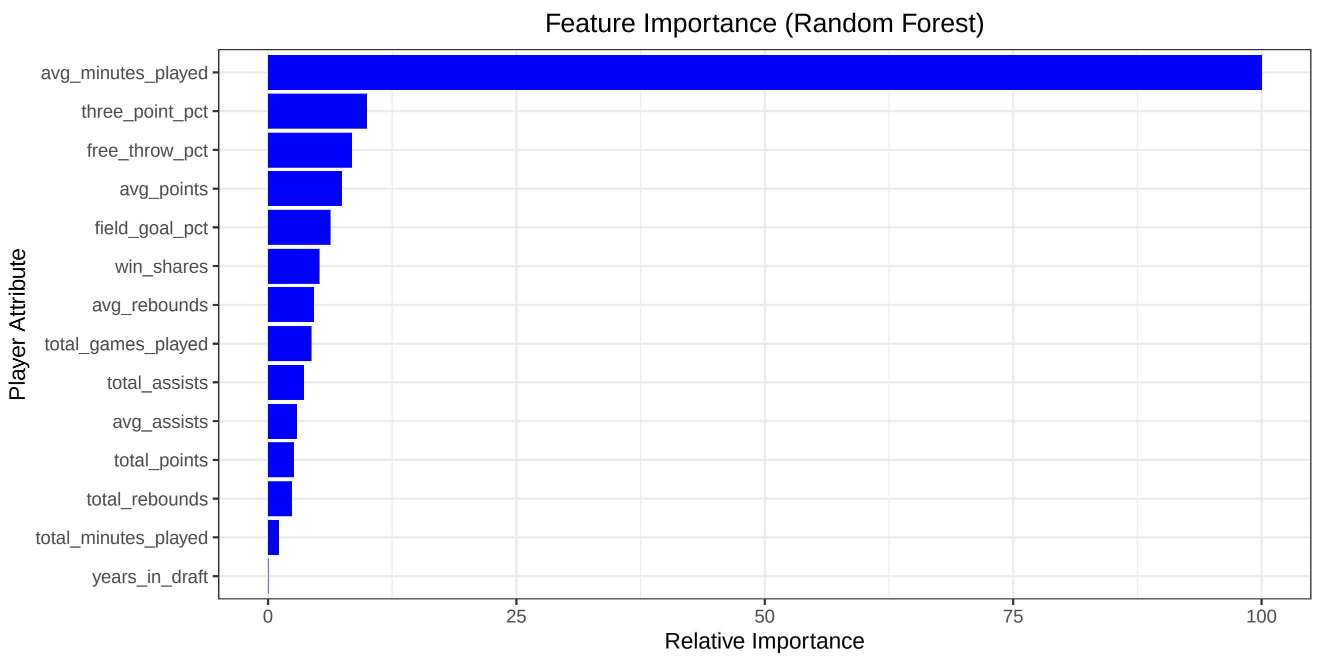
```r
# Add Predictions to the 2024 Data (nba_predict)
nba_predict <- nba_predict %>% mutate(predicted_pick_rf = round(y_2024_pred_rf))

# Save the predictions to a CSV file for 2024 draft
write.csv(nba_predict, "nba_draft_2024_predictions_rf_tuned.csv", row.names = FALSE)

# Visualization: Actual vs Predicted for 2024 Draft Picks
ggplot(data.frame(Actual = y_predict, Predicted = y_2024_pred_rf),
       aes(x = Actual, y = Predicted)) +
  geom_point(color = "blue") +
  geom_abline(slope = 1, intercept = 0, color = "red") +
  labs(
    title = "Actual vs Predicted Draft Order (2024)",
    x = "Actual Draft Order",
    y = "Predicted Draft Order"
  )
```



```r
###### Feature Importance Analysis ######
# Retrieve feature importance from the finalized model
feature_importance <- varImp(final_model, scale = TRUE)

# Print feature importance values
print(feature_importance)
```

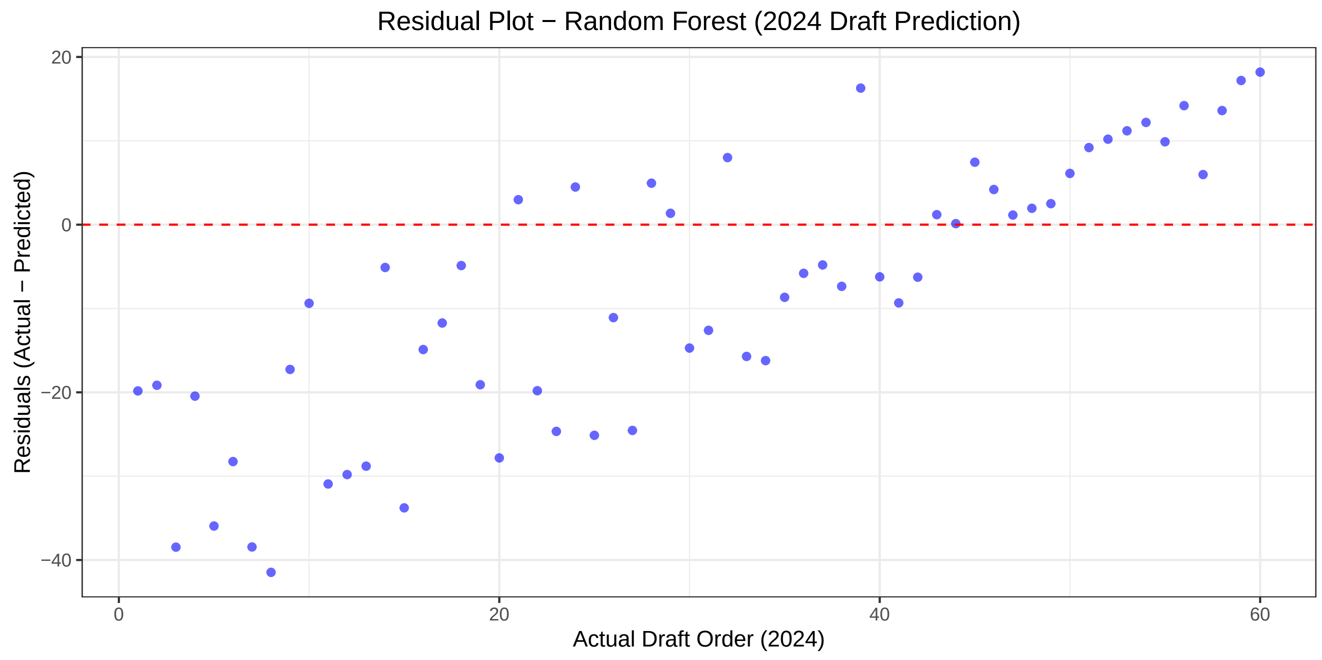
```
## ranger variable importance
##
##                    Overall
## avg_minutes_played  100.000
## three_point_pct       9.948
## free_throw_pct        8.408
## avg_points            7.432
## field_goal_pct        6.262
## win_shares             5.176
## avg_rebounds           4.614
## total_games_played     4.335
## total_assists          3.611
## avg_assists            2.895
## total_points           2.608
## total_rebounds         2.367
## total_minutes_played   1.088
## years_in_draft         0.000
```

```r
# Convert to a data frame for saving to CSV
feature_importance_df <- as.data.frame(feature_importance$importance)

# Save the feature importance data to a CSV file
write.csv(feature_importance_df, "feature_importance.csv", row.names = TRUE)

# Visualization: Create a bar plot for feature importance
# Create data frame for plot
importance_df <- data.frame(Feature = rownames(feature_importance$importance),
                            Importance = feature_importance$importance[, 1])
# Plot feature importance
ggplot(importance_df, aes(x = reorder(Feature, Importance), y = Importance)) +
  geom_bar(stat = "identity", fill = "blue") +
  coord_flip() +  # Flip coordinates for better readability
  labs(
    title = "Feature Importance (Random Forest)",
    x = "Player Attribute",
    y = "Relative Importance"
  )
```

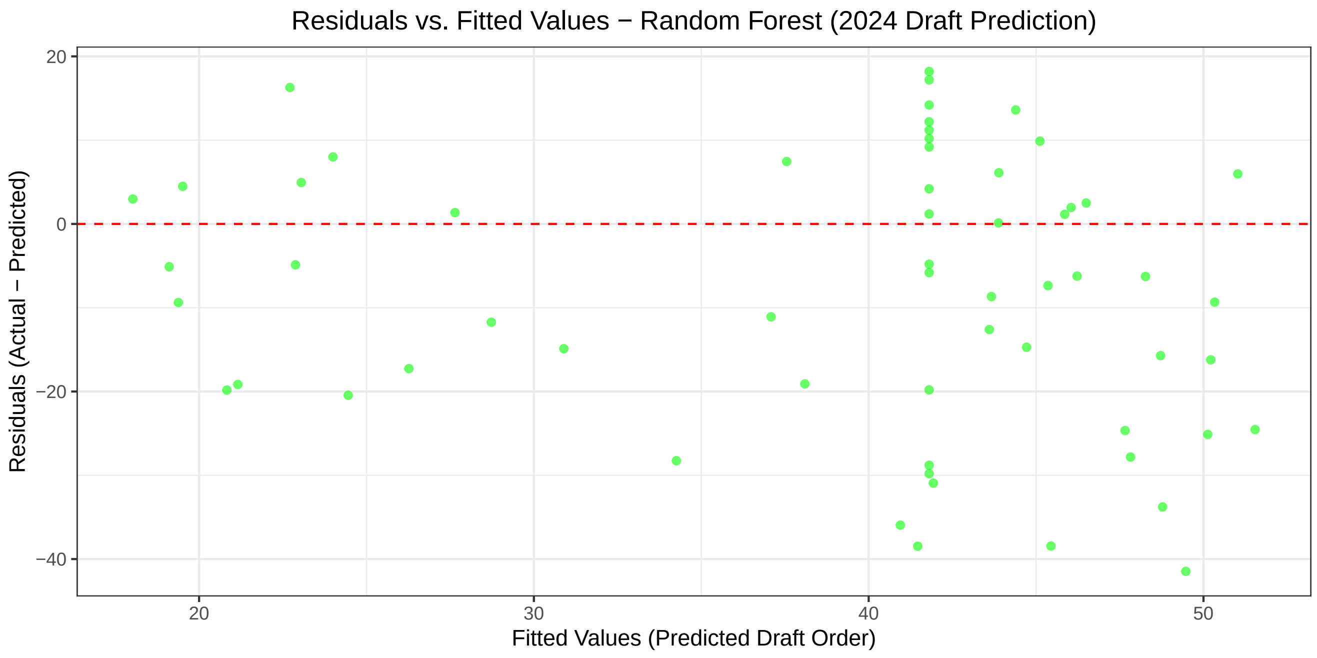


Model Diagnostics

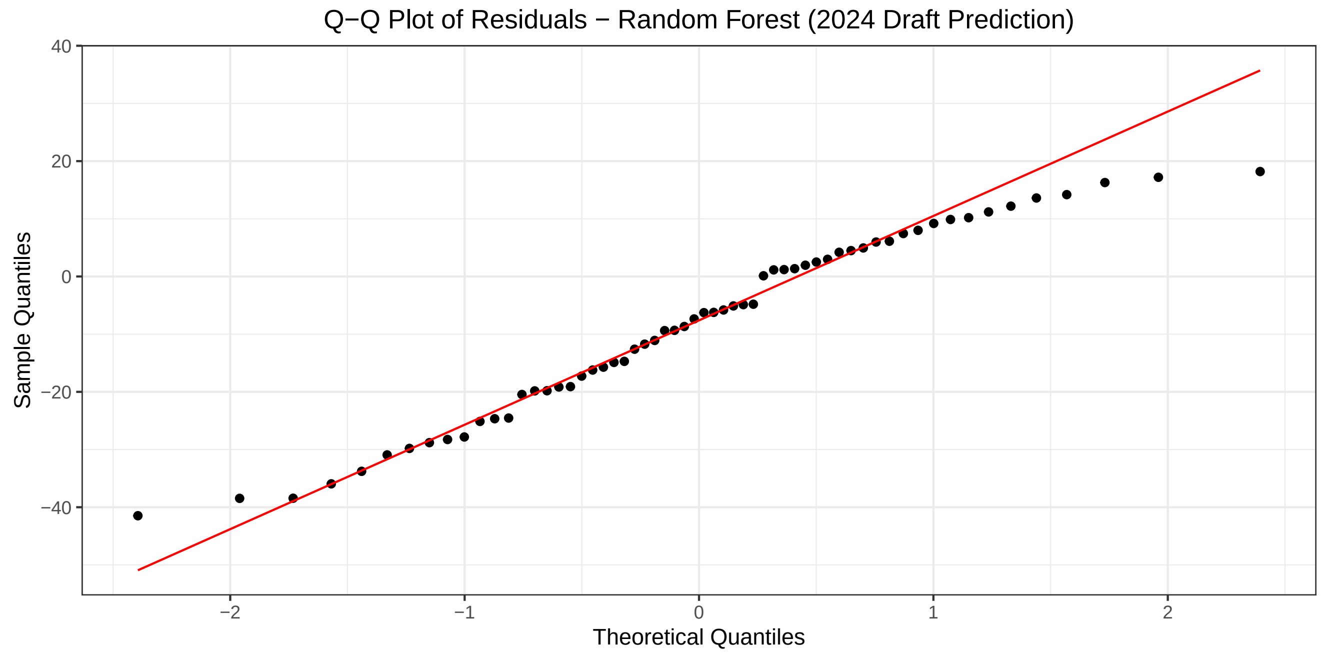
```r
###### Check for Independence, Heteroscedasticity, and Normality ######

# Load necessary library for skewness and kurtosis calculation
library(e1071)

# Residual Diagnostics: Evaluate Model Fit and Bias
# Calculate residuals (difference between actual and predicted values) for 2024 data
residuals_rf <- y_predict - y_2024_pred_rf  # Residuals = Actual - Predicted
fitted_values <- y_2024_pred_rf  # Predicted (fitted) values for 2024

# a. Residual plot (Actual vs. Residuals): Check for independence of residuals
# Plot residuals to observe if there are any patterns (which would suggest non-independence)
residual_plot <- ggplot(data.frame(Actual = y_predict, Residuals = residuals_rf),
                        aes(x = Actual, y = Residuals)) +
  geom_point(color = "blue", alpha = 0.6) +  # Scatter plot of residuals
  # Add reference line at zero residual
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
  labs(
    title = "Residual Plot - Random Forest (2024 Draft Prediction)",
    x = "Actual Draft Order (2024)",
    y = "Residuals (Actual - Predicted)"
  )
# Print the residual plot
print(residual_plot)
```



```r
# b. Residuals vs. Fitted Values Plot: Check for heteroscedasticity (variance of residuals)
# This plot helps detect if the variance of residuals changes with predicted values
residual_vs_fitted_plot <- ggplot(
  data.frame(Fitted = fitted_values, Residuals = residuals_rf),
  aes(x = Fitted, y = Residuals)) +
  geom_point(color = "green", alpha = 0.6) +  # Scatter plot of residuals vs. fitted values
  # Add reference line at zero residual
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
  labs(
    title = "Residuals vs. Fitted Values - Random Forest (2024 Draft Prediction)",
    x = "Fitted Values (Predicted Draft Order)",
    y = "Residuals (Actual - Predicted)"
  )
# Print the residual vs fitted values plot
print(residual_vs_fitted_plot)
```



```r
# c. Q-Q Plot: Check for normality of residuals
# The Q-Q plot compares the distribution of residuals to a normal distribution
qq_plot <- ggplot(data.frame(Residuals = residuals_rf), aes(sample = Residuals)) +
  stat_qq() +  # Create Q-Q plot to compare residuals against normal distribution
  stat_qq_line(color = "red") +  # Add reference line for normality
  labs(
    title = "Q-Q Plot of Residuals - Random Forest (2024 Draft Prediction)",
    x = "Theoretical Quantiles",
    y = "Sample Quantiles"
  )
# Print the Q-Q plot
print(qq_plot)
```



```
# d. Additional statistics to assess the shape of the residual distribution
# Skewness measures asymmetry
# Kurtosis measures the "tailedness" of the distribution
skewness_value <- skewness(residuals_rf)  # Calculate skewness of residuals
kurtosis_value <- kurtosis(residuals_rf)  # Calculate kurtosis of residuals

# Display skewness and kurtosis values for 2024 residuals
cat("\nSkewness of Residuals (2024): ", skewness_value, "\n")
```

```
##
## Skewness of Residuals (2024):  -0.2422466
```

```
cat("Kurtosis of Residuals (2024): ", kurtosis_value, "\n")
```

```
## Kurtosis of Residuals (2024):  -1.008883
```

```
# Summary statistics of residuals for 2024 predictions to understand their distribution
residual_summary <- summary(residuals_rf)
cat("\nSummary of Residuals (2024):\n")
```

```
##
## Summary of Residuals (2024):
```

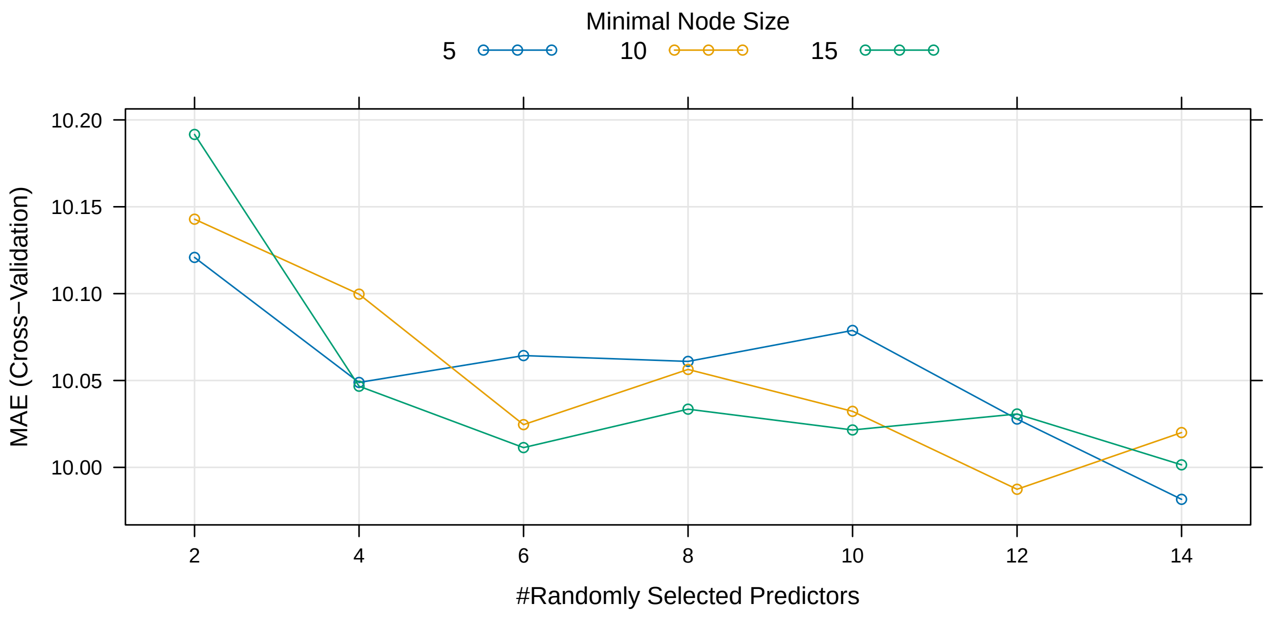
```
print(residual_summary)  # Display summary statistics for residuals
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -41.473 -19.813  -6.813  -8.399   4.603  18.193
```

```
###### Estimate Generalization Performance ######

# Print the summary of the finalized model
print(final_model)
```

```
## Random Forest
##
## 236 samples
##  14 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 212, 212, 213, 212, 213, 212, ...
## Resampling results across tuning parameters:
##
##   mtry  min.node.size  RMSE      Rsquared   MAE
##    2     5             12.55604  0.4600497  10.120855
##    2    10             12.54107  0.4594997  10.142855
##    2    15             12.57437  0.4575969  10.191630
##    4     5             12.52639  0.4597845  10.048883
##    4    10             12.50488  0.4637115  10.099693
##    4    15             12.44380  0.4693681  10.046721
##    6     5             12.54141  0.4583744  10.064348
##    6    10             12.50154  0.4621359  10.024590
##    6    15             12.43577  0.4687089  10.011384
##    8     5             12.54538  0.4575729  10.060996
##    8    10             12.53276  0.4579535  10.056378
##    8    15             12.50551  0.4612261  10.033508
##   10     5             12.59041  0.4538036  10.078828
##   10    10             12.52031  0.4612157  10.032257
##   10    15             12.55249  0.4580409  10.021540
##   12     5             12.54790  0.4577225  10.027810
##   12    10             12.53006  0.4586379   9.987403
##   12    15             12.56483  0.4570108  10.030723
##   14     5             12.55711  0.4558493   9.981604
##   14    10             12.57572  0.4555476  10.020042
##   14    15             12.51340  0.4609477  10.001463
##
## Tuning parameter 'splitrule' was held constant at a value of variance
## MAE was used to select the optimal model using the smallest value.
## The final values used for the model were mtry = 14, splitrule = variance
##  and min.node.size = 5.
```

```r
# Visualize the error summary for the finalized model
plot(final_model)
```



(d) Interpret your results, presenting visuals as needed.

Rationale for Random Forest and `ntree` Selection:

Random Forest is a non-parametric model, meaning it does not assume a specific form (e.g., linear) for the relationship between input features and the target variable. Instead, it builds an ensemble of decision trees, which capture complex, non-linear interactions between features. Each tree in the forest learns patterns independently, and their outputs are aggregated (averaged in regression tasks) to produce robust predictions. This approach makes Random Forest especially suitable for the NBA draft prediction problem, where:

- Relationships between player performance metrics and draft outcomes might likely be non-linear.
- The relative importance of features (e.g., scoring, playing time, efficiency) can vary depending on player and team dynamics.
- Random Forest can automatically handle feature interactions, multicollinearity, and irrelevant features, improving prediction accuracy.

The number of trees (`ntree`) influences Random Forest's stability and accuracy:

- Low number of trees (e.g., `ntree = 100`): Fewer trees may lead to underfitting, as the model might not fully capture the relationships in the data.
- High number of trees (e.g., `ntree = 1000`): Increasing the number of trees improves stability and reduces variance but comes at a higher computational cost.

Optimal Performance: The best-performing model, based on test Mean Absolute Error (MAE), was achieved with `ntree = 500`, yielding:

- Test MAE: **11.10188** (lowest among configurations).
- Test Spearman Correlation: **0.6439236** (indicating strong rank-order alignment).

This configuration effectively balances predictive accuracy and robustness, minimizing errors while maintaining rank consistency.

Model Performance Interpretation:

1. Final Model Based on MAE:

- The model trained with `ntree = 500` and optimal parameters (`mtry = 14`, `min.node.size = 5`) achieved the lowest test MAE of **11.10188**, indicating that, on average, predictions deviated from actual draft positions by approximately 11 picks on the test set, which is a reasonable error, but improvements could still be made.
- A Spearman Correlation of **0.6439236** demonstrates strong rank-order consistency between predicted and actual draft positions on the test set, suggesting the model captures key relationships in the data.

2. Performance on 2024 Data:

- MAE on 2024 Draft Prediction: **14.55034**. This is a higher prediction error on unseen data compared to the test set, suggesting possible differences in the 2024 class compared to historical data. The model is missing the target by about 14.55 units on average for the 2024 draft predictions, reflecting a potential lack of generalizability or variability in the 2024 data that the model did not capture.
- Spearman Correlation: **0.3659773**. This indicates a weak to moderate positive rank-order relationship between the predicted and actual rankings. While the model captures some underlying patterns, this drop compared to the test set implies room for improvement in rank consistency.

3. Feature Importance Analysis:

(1) Feature importance provides insight into the contribution of each predictor variable in the model's decision-making process. It ranks features based on their normalized importance scores, as determined by the `ranger` implementation of Random Forest. These scores reflect how much a feature reduces prediction error across the decision trees in the ensemble.

(2) In analyzing the feature importance for the NBA draft data, we see a clear hierarchy in the contribution of each feature to the model's predictive accuracy. Based on the importance scores, certain factors emerge as critical in shaping draft decisions, while others seem to play a minor or no role at all.

(3) Key Insights:

- Dominance of `avg_minutes_played` (100.00 importance score): The feature with the highest importance is `avg_minutes_played`, which scores a perfect 100. This indicates that teams value a player's consistent playing time above all other metrics. Minutes played serve as a direct reflection of a player's opportunity to perform, their trustworthiness in the eyes of coaches, and their overall role on the team. Players with higher minutes per game often have a greater chance to showcase their skills, leading to higher visibility and, thus, greater draft appeal. The fact that this feature dominates the model with a score of 100 highlights the strong link between playing time and draft positioning.
- Scoring Metrics and Efficiency: Scoring metrics and efficiency are highly valued in draft decisions. `three_point_pct` (9.95) leads, highlighting the importance of a player's ability to score from beyond the arc, which is crucial for modern offensive strategies. `free_throw_pct` (8.41) follows closely, emphasizing the significance of free-throw accuracy for a player's overall offensive efficiency, particularly in close games. While `avg_points` (7.43) remains important, it is ranked lower, suggesting that teams prioritize efficient scoring over raw points. `field_goal_pct` (6.26) is also significant, reflecting the value placed on overall shooting efficiency in high-pressure situations.

- Additional Performance Metrics: `win_shares` (5.18) is an advanced statistic that attempts to quantify a player's contribution to their team's success. Although its importance is notable, it ranks lower than the more direct performance metrics like scoring and minutes played. Teams might not be using advanced metrics like win_shares as heavily when making draft decisions, but it still offers valuable insight into a player's overall impact. `avg_rebounds` (4.61) and `total_assists` (3.61) are additional statistics that give insight into a player's ability to contribute beyond scoring. These features, though still relatively important, show that teams place a higher value on scoring and minutes played than on overall contributions like rebounding or playmaking.
- Cumulative and Durability Metrics: `total_games_played` (4.34) and `total_minutes_played` (1.09) are also significant, but they fall further down the list of priorities. These features reflect a player's overall experience and durability, and while useful in evaluating consistency, they do not weigh as heavily in draft decisions as playing time and scoring ability. For example, players with fewer total games but high averages in minutes and points could still be considered top prospects.
- Least Important: The `years_in_draft` feature has an importance score of 0.00, which essentially means it has no impact on the draft decisions based on this model. This suggests that a player's eligibility history or the number of years they've been available for selection does not play a role in team decisions. Teams appear to be more focused on a player's current ability and potential, rather than how long they've been draft-eligible.

(4) Implications for Draft Strategy:

- Consistency and Reliability on the Court: The overwhelming importance of `avg_minutes_played` underscores that teams prioritize players who can consistently contribute by playing a meaningful number of minutes. This indicates that, for draft prospects, demonstrating reliability in terms of minutes on the court is likely more valuable than simply putting up big scoring numbers. Players who are trusted to contribute regularly, even if not the primary scorers, have an edge in the draft process.
- Efficiency Over Raw Scoring: While scoring ability remains a key factor in evaluations, its importance is secondary to the ability to play consistently and efficiently. Prospects who can efficiently contribute to their team's offense, whether through scoring, shooting efficiency, or overall gameplay, are more likely to impress teams. Players looking to improve their draft position should focus on showcasing their ability to stay on the court and make an impact through efficient play rather than inflating raw scoring numbers.
- Value of Advanced Metrics: Although `win_shares` ranks lower, it still provides valuable insight, especially for teams seeking well-rounded players. These players might not always be the highest scorers but contribute significantly in other areas like defense, playmaking, or overall team success. Teams that value such players might focus on these advanced stats to identify players who excel in less obvious ways.
- Focus on Current Potential, Not Eligibility History: The lack of importance given to `years_in_draft` suggests that the draft process is centered on evaluating a player's present ability and future potential, rather than their eligibility history. Teams are likely to overlook how many years a player has been available for selection and instead focus on their immediate impact and growth potential in the league.

Model Diagnostics Interpretation:

1. Cross-Validation and Evaluation

(1) Cross-Validation for Model Robustness:

- Cross-validation is a crucial diagnostic tool for evaluating the generalization performance of a machine learning model, particularly when dealing with small to medium-sized datasets. In this case, a 10-fold cross-validation strategy was employed. This process involves splitting the dataset into 10 equal parts (folds), where the model is trained on 9 folds (90% of the data) and validated on the remaining 1 fold (10%). This procedure is repeated 10 times, such that each fold serves as the validation set once. The key benefits of this approach are:

  - Reliable Generalization Estimate: Cross-validation helps estimate how well the model will perform on unseen data by using different subsets for training and validation. This provides a robust estimate of how the model is likely to generalize to future data.
  - Mitigation of Overfitting: By using different subsets for training and validation, the process minimizes the risk of overfitting to specific data points, increasing confidence in the model's ability to generalize.

(2) Evaluation of Model Performance:

- The cross-validation results provide insights into the model's behavior:
  - Generalization Performance: The results indicate relatively consistent Mean Absolute Error (MAE) values across different hyperparameter combinations, suggesting that the model generalizes well and is not overly reliant on specific data partitions.
  - Model Stability: The stable MAE across folds reflects a well-balanced bias-variance tradeoff. This indicates the model is neither too simplistic (high bias) nor overly complex (high variance).
  - Optimal Hyperparameters: The best-performing model was selected based on the lowest MAE, with `mtry = 14` and `min.node.size = 5`. This combination ensures the best balance of predictive accuracy and model complexity.

(3) Cross-Validation Summary:

- Methodology: The 10-fold cross-validation approach ensured comprehensive testing of the model across all data subsets, providing a reliable estimate of the generalization error.
- Optimal Model Selection: The hyperparameter tuning process identified `mtry = 14` and `min.node.size = 5` as the optimal values, as they yielded the lowest MAE.
- Error Metrics: The MAE results across folds were consistently low, indicating the model is robust and reliable for predicting NBA draft performance.
- Generalization Insights: The stability in MAE suggests the model can effectively handle unseen data, making it suitable for real-world applications.

(4) Conclusion:

- The finalized model, with `mtry=14` and `min.node.size=5`, was selected based on the smallest Mean Absolute Error (MAE), providing a clear measure of average prediction error. The use of 10-fold cross-validation provided a thorough assessment of the model's generalization ability and performance. The consistent MAE values, coupled with hyperparameter tuning, indicate the model is well-calibrated and unlikely to overfit. These findings support the model's reliability in predicting NBA draft outcomes, making it a robust tool for forecasting in this domain. This methodology ensures the model is both accurate and generalizable, critical for real-world applications where unseen data must be handled effectively.

2. Residual Analysis

(1) Residuals Diagnostics:

- Skewness: **-0.2422466**. The negative skewness indicates that the residuals (errors) are slightly left-skewed, meaning that there are more extreme positive errors (overestimates) than negative errors (underestimates). The skewness is not large, suggesting this asymmetry is not severe, but it's worth noting.
- Kurtosis: **-1.008883**. The negative kurtosis indicates a platykurtic distribution, meaning the residuals are more spread out and have fewer extreme values than a normal distribution. This suggests that the model's residuals exhibit less concentration around the mean and have a flatter distribution, which could imply that the model underestimates variability in the data.

(2) Summary of Residuals (2024):

- Range: Residuals range from a minimum of **-41.473** to a maximum of **18.193**, showing that the model has some large underpredictions (negative residuals).
- Median and Mean: The median residual is **-6.813**, and the mean residual is **-8.399**, both negative. This suggests that, on average, the model tends to underpredict draft positions, assigning players to slightly earlier picks than their actual draft positions.
- Quartiles: The 1st quartile is **-19.813**, and the 3rd quartile is **4.603**, indicating that most residuals are negative, aligning with the overall underprediction bias.

(3) Residual Plot (Actual vs Residuals):

- The discernable trend in the residuals, where they increase as the actual values increase, implies potential violations of independence, meaning the residuals may not be entirely independent of the actual values. This indicates that the model struggles to capture the relationships between features and draft positions for extreme cases or outliers, particularly for players drafted later in the order.

(4) Residuals vs Fitted Value Plot:

- The lack of random scatter suggests possible heteroscedasticity, indicating that the model's prediction errors vary across different ranges of fitted values. This could signal that the model struggles with consistent accuracy across different levels of predicted draft positions, particularly for higher or lower-ranked players.

(5) QQ Plot (Normality of Residuals):

- The QQ plot shows deviations from the 45-degree line, particularly at the tails, suggesting that the residuals deviate from normality. This is consistent with the negative kurtosis, indicating that the model struggles with extreme predictions, resulting in outliers. While strict normality is not required for Random Forest models, the observed heavy tails highlight potential areas where predictions could be improved.

Overall Assessment:

- Strengths: The model demonstrates strong generalizability and is robust across training and validation data, with relatively consistent performance. It appears to capture the key relationships between player features and draft positions, which is crucial for making meaningful predictions in a draft scenario.
- Weaknesses: The model tends to underpredict (a sign that the model has some biases) and struggle with extreme cases (such as very high or very low draft positions). It shows some signs of heteroscedasticity and non-normal residuals, indicating areas where prediction accuracy can be improved and suggesting that there might be underlying patterns the model has not captured yet.

Potential Improvement of Model Performance:

1. Address Underprediction Bias:

- The model tends to underpredict draft positions, as indicated by the negative residuals. To address this bias, consider adding additional features that may better capture the complexity of the draft process. For example, incorporating team-specific data (e.g., historical team performance, team needs, or injury reports) could help the model better understand factors influencing draft decisions.
- Alternatively, transforming features could capture non-linear relationships that the model might not currently be handling effectively. For instance, using logarithmic transformations for highly skewed features or creating interaction terms between key features (like points and minutes played) may help the model better capture the relationships between these features and draft outcomes.

2. Handle Heteroscedasticity:

   - The residuals vs. fitted value plot suggests possible heteroscedasticity, where the variance of residuals changes at different levels of the predicted values. This could indicate that the model is less accurate for extreme predictions (e.g., players drafted at the beginning or end of the draft).
   - One way to address heteroscedasticity is through weighted regression, where observations with larger residuals could be given less weight during training, allowing the model to focus more on predictions with smaller errors. Another method is to apply feature transformations to stabilize variance. For example, log-transforming highly skewed features (e.g., points, minutes played, etc.) could help mitigate the impact of extreme values and lead to more consistent residuals.

3. Incorporate Advanced Modeling Techniques:

   - Random Forest is a robust model but may still struggle with capturing complex patterns in the data, especially when it comes to outliers or extreme draft cases. To improve the model's ability to handle such cases, consider incorporating more advanced techniques like Gradient Boosting Machines (GBMs).
   - Gradient Boosting Machines (GBMs) are powerful because they focus on correcting errors made by previous models. By iterating and adjusting based on residuals from the previous iteration, GBMs can more accurately capture non-linear patterns and complex interactions between features that Random Forests might miss. Techniques like XGBoost or LightGBM could be particularly useful for improving the prediction of extreme cases in the draft.

(e) How well did your algorithm do, in comparison to the true order of the 2024 NBA draft?

Evaluation of 2024 Predictions:

The Random Forest model shows a solid performance on historical data but faces challenges when applied to the 2024 draft class, reflecting a need for further refinement to handle unseen data.

1. Prediction Accuracy:

   - The Mean Absolute Error (MAE) of **14.55034** for the 2024 data indicates that the model's predictions deviate by an average of approximately 14.55 picks from actual draft positions. This error is higher than the MAE of **11.10188** on the test set, suggesting that the model struggles to generalize to the 2024 class, which may have different evaluation criteria or characteristics.

2. Rank-Order Consistency:

   - The Spearman Correlation of **0.3659773** reflects a weak to moderate alignment between predicted and actual draft rankings. While the model captures some general trends in draft order, it struggles with precise ranking, particularly for players drafted in the lower positions or outliers. The correlation highlights that the model is better at capturing broad trends than detailed rank-order predictions.

3. Bias in Predictions:

   - The negative mean residual of **-8.399** indicates a consistent underprediction bias, where the model tends to assign players to earlier draft positions than their actual ranks. This may indicate that the model has difficulty capturing the nuances of the 2024 draft class, possibly due to unmeasured factors or evolving draft strategies.

4. Strengths and Limitations:

- Strengths: The model successfully identifies key player attributes, such as playing time and scoring, that are likely to influence draft decisions. It performs reasonably well for players drafted in the middle of the order, where the relationships between features and draft positions are clearer.
- Limitations: The model struggles with outliers and late-draft players, as seen in the platykurtic residual distribution and the discernable trend in residuals. This suggests the model faces difficulties with players who exhibit atypical performance metrics or are evaluated based on factors not fully captured by the features.

5. Opportunities for Improvement:

- Additional Features: Incorporating factors like team-specific preferences, player injury history, or scouting reports could help improve prediction accuracy, especially for outliers or players in the lower draft positions.
- Advanced Models: Exploring alternative models such as Gradient Boosting Machines (GBMs) that could capture more complex, non-linear relationships between features and draft positions, potentially improving predictions for extreme cases.

In conclusion, while the model provides a strong baseline, its performance on the 2024 data highlights the need for further refinement, especially for late-draft players and outliers. Additional features and more sophisticated modeling techniques could help enhance its predictive accuracy and rank-order consistency.

2. Team Draft Behaviors

The draft is extremely important for team strategy not just for the upcoming season, but also for the years ahead. Construct an approach by which team behaviors can be described, in terms of the type of players they tend to recruit in the draft based on all data provided to you (2019-2024), according to the following schematic:

(a) Define the problem

In this task, we aim to analyze team draft behaviors by exploring the types of players they tend to recruit based on historical data from 2019–2024. Since the goal is to explore the patterns in how teams select players, unsupervised learning techniques (e.g., k-means clustering) are appropriate to uncover groupings and trends in the data.

Problem Class:

This is an unsupervised learning problem aimed at identifying patterns in team draft behaviors by analyzing the types of players they select during the NBA draft from 2019 to 2024. The objective is to group teams based on similarities in player attributes, such as performance metrics and characteristics, to reveal insights into their drafting strategies.

1. Data Provided to the Algorithm:

- The dataset consists of "unlabeled" data, including player performance metrics, efficiency statistics, and advanced analytics from the 2019–2024 NBA drafts. These include attributes such as points, assists, rebounds, field goal percentage (FG%), three-point percentage (3P%), Win Shares, Box Plus-Minus (BPM), and other relevant statistics. Each player is linked to a team that drafted them, providing implicit information about the preferences and strategies of those teams.

2. Desired Conclusions:

- The algorithm identifies groups of teams with similar drafting behaviors based on the attributes of the players they select. This algorithm will uncover patterns in team preferences, such as whether they tend to select high-scoring players, players with strong defensive capabilities, or all-around contributors, providing insights into team drafting strategies. The identified clusters can also help predict team behaviors in future drafts based on their historical preferences by highlighting consistent drafting tendencies.

3. Behavior of Algorithm:

    - The algorithm will analyze the provided player data and group teams according to similarities in their selection patterns. It will process the features of drafted players, determining which teams tend to prioritize similar player characteristics. Through this analysis, the algorithm will identify clusters of teams that share common drafting preferences. The resulting groupings can be further interpreted to assess team strategies and uncover insights into long-term drafting trends.

Assumptions:

1. Nature of Data:

    - The dataset consists of "unlabeled" data, including player performance metrics, efficiency statistics, and advanced analytics which are assumed to be reliable and representative of what teams value in draft decisions. Features such as points, assists, and shooting efficiency are numerical, normalized for comparability, and considered relevant in capturing the key characteristics teams evaluate during drafts.

2. Assumptions about the Problem:

    - Team draft behaviors can be inferred from the measurable attributes of the players they select, with similarities in player traits reflecting consistent drafting strategies over time.
    - Teams with similar drafting behaviors will be grouped together based on the prioritization of key player traits (e.g., points, assists, rebounds, shooting percentages, etc.), revealing patterns in team strategies.
    - The data from 2019–2024 is representative of team draft behaviors and provides a reliable foundation for identifying current trends in drafting strategies, despite potential changes in strategies over time.
    - Missing or unmeasured factors (e.g., player personality, team culture) are assumed to have minimal impact on clustering results, while data preprocessing will address any missing data to ensure the dataset is clean and reliable.
    - Clustering will be based on quantitative player performance metrics, assuming these measurable traits are the primary factors in teams' draft decisions.
    - The number of clusters will be determined using statistical methods (e.g., Elbow Method) to ensure that the clusters are stable, interpretable, and meaningful for understanding team draft strategies.

Evaluation Criteria:

1. Goal of the Model:

    - The objective of this clustering model is to identify distinct behavioral patterns in NBA team draft strategies. By grouping teams with similar drafting preferences based on player performance metrics, the model aims to uncover actionable insights into team decision-making tendencies, such as prioritizing offensive, defensive, or balanced player profiles.

2. Query Evaluation:

    - Cluster Meaningfulness: The algorithm's output will be evaluated by interpreting the feature centers of each cluster to assess whether they meaningfully represent different drafting behaviors. Features such as shooting efficiency, per-game performance, and advanced statistics will be analyzed and compared against known team strategies (e.g., teams prioritizing scorers vs. defenders) to ensure alignment with realistic drafting behaviors.

- Query-Driven Analysis: For specific questions, such as identifying teams favoring high scorers or versatile defenders, the model will evaluate whether the clustering results provide clear, interpretable answers based on dominant features within each group.

3. Overall Model Performance:

- Cohesion and Separation: The overall performance will be measured based on both the cohesion within clusters and the separation between clusters. Tools like the Elbow method and Silhouette analysis can help assess how well teams are grouped within clusters and how distinct the clusters are from each other, with higher scores indicating better-defined and more meaningful clusters.
  - Elbow Method: Total within-cluster sum of squares (WSS) will be analyzed using the Elbow Method to determine the optimal number of clusters, ensuring a balance between model simplicity and data representation quality.
  - Silhouette Scores: Average silhouette scores across all clusters will measure how well data points fit within their clusters and how distinct the clusters are. Higher scores indicate better-defined groupings.
  - Cluster Variance: Intra-cluster variance will be evaluated to ensure tight and cohesive clusters. Smaller variances indicate that data points within clusters are highly similar.
- Cluster Interpretability: The system's success will also depend on whether the clusters are interpretable and actionable. This will be determined by the clarity of differences in feature importance across clusters and the alignment of findings with real-world team strategies.
- Practical Relevance: The overall performance will be validated by how effectively the clusters aid in understanding team behaviors, as judged by their ability to inform strategic decisions or validate hypotheses about drafting tendencies.

(b) Define a solution

Model Type:

1. Model Selection:

K-Means clustering is particularly well-suited for analyzing team draft behaviors for several key reasons:

- Unsupervised Learning for Pattern Discovery: K-Means, an unsupervised learning algorithm that groups data points based on inherent similarities without requiring predefined labels or outcomes, is ideal for this task as it identifies clusters of teams with similar drafting strategies by analyzing the relationships between player attributes (e.g., points, minutes played, college performance), uncovering broader team strategies without relying on a specific target variable.
- Handling Multivariate Data: The draft data involves multiple player attributes (e.g., scoring, assists, defensive stats, physical traits). K-Means works effectively with multivariate draft data, making it suitable for analyzing teams' drafting decisions across various factors simultaneously and uncovering how teams prioritize specific player traits.
- Grouping Based on Similarity: K-Means groups teams based on similarities in their player selections, making it a natural fit for understanding drafting behavior. Teams that select players with similar traits will be grouped into the same cluster, making it easy to identify common drafting strategies.
- Simplicity and Interpretability: K-Means is relatively simple to implement and understand, which is important for interpreting and communicating the results. Once the clustering is done, it's straightforward to analyze the centers of each cluster (i.e., the "average" behavior of teams in each cluster) and interpret these centers in terms of player traits and drafting strategy.

- Scalability: K-Means is computationally efficient and can handle large datasets, which is important when working with multiple years of draft data (2019-2024). Teams might have varied strategies over time, and K-Means can effectively group teams based on patterns across the entire dataset, while handling changes in drafting strategies over the years.
- Evaluation and Fine-tuning: K-Means offers methods for evaluating cluster quality, such as the Elbow method and Silhouette analysis, which help determine the optimal number of clusters. These tools allow us to assess how well the clustering represents distinct drafting behaviors and adjust the model accordingly for better accuracy.

2. Aspects of the Data to Be Modeled:

- The K-Means algorithm models the relationship between player performance metrics (e.g., points, assists, shooting efficiency, FG%, Win Shares) and the teams that select them, aiming to group teams based on the similarities in the types of players they prioritize in drafts. This approach highlights the traits teams value in draft picks, uncovering distinct drafting strategies.

3. Use of Model:

- The K-Means model identifies clusters that summarize drafting behaviors, offering insights into the types of players different teams prioritize. While unsupervised learning does not make traditional predictions, the model can infer the draft behavior of new or unseen teams by assigning them to one of the established clusters based on their player selection patterns.

Model Class:

1. Parametric Class: K-Means, a non-parametric clustering algorithm, is the chosen model class for this analysis. It does not assume any specific data distribution, making it a flexible and suitable choice. By minimizing within-cluster variance, K-Means effectively groups teams based on similar patterns in their player selections.

2. Model Selection Criteria: The optimal number of clusters (`k`) will be determined using the Elbow method, which identifies the point where adding more clusters no longer significantly improves the sum of squared distances within clusters. Additionally, Silhouette scores will be used to evaluate the coherence and separation of the clusters, ensuring a balance between interpretability and the meaningfulness of the groupings.

(1) Elbow Method: The Elbow Method is a commonly used approach to determine the optimal number of clusters (k) in a clustering algorithm. It involves analyzing how the Within-Cluster Sum of Squares (WSS) or inertia changes as the number of clusters increases.

- WSS measures the compactness of the clusters, calculated as the sum of squared distances between each data point and its corresponding cluster centroid. A lower WSS indicates more compact clusters. As the number of clusters (k) increases, WSS decreases because more clusters allow for tighter groupings. However, this decrease in WSS becomes less significant after a certain point, indicating diminishing returns.
- The "elbow point" is the value of k where the decrease in WSS slows down dramatically, forming an angle or "elbow" in the plot of WSS versus k. This point represents the optimal number of clusters, as adding more clusters beyond this does not significantly improve the clustering performance.
- Goal: The Elbow Method aims to minimize intra-cluster variance while avoiding overfitting by selecting the number of clusters at the "elbow" point. This ensures a balance between compactness and simplicity of the clustering model.
- The rationale behind this method is that the optimal k occurs at the point where the decrease in WSS slows down dramatically, indicating that additional clusters do not meaningfully improve the clustering.

(2) Silhouette Analysis: Silhouette Analysis is a method used to evaluate the quality of clustering by assessing how well each data point fits within its assigned cluster compared to other clusters. It measures the balance between cohesion (how close points are to their own cluster) and separation (how far points are from other clusters).

- Silhouette Score ranges from -1 to $+1$:
  - "+1": The point is well-clustered, with strong cohesion and good separation from other clusters.
  - "0": The point lies near the boundary between two clusters, indicating overlap.
  - "-1": The point is poorly clustered and likely belongs to a different cluster.
- The average silhouette score across all points provides an overall measure of clustering performance.
- Goal: Maximize the silhouette score by achieving high intra-cluster cohesion and inter-cluster separation. The optimal number of clusters (k) is the one with the highest average silhouette score, indicating well-defined and distinct clusters.
- The rationale behind this method is that the optimal k is the one where the average silhouette score across all points is maximized. Higher silhouette scores indicate well-separated and cohesive clusters.

Algorithm:

The computational process for implementing the K-means Clustering model involves the following steps:

Step 1: Data Preparation

- Collect the dataset containing player performance attributes (e.g., points, assists, shooting efficiency, win shares) for players drafted by each team.
- Normalize the data to ensure all features are on a similar scale, using techniques such as Min-Max scaling or Z-score standardization. This step addresses differences in ranges between attributes (e.g., points may range in the hundreds, while win shares may range between 0 and 1).
- Handle missing or inconsistent data by imputing missing values using the median for numerical features, ensuring a clean, reliable, and consistent dataset for analysis.

Step 2: Clustering

- Use R's `stats` package to implement the K-Means algorithm, assigning players to clusters based on their performance metrics. Apply the K-Means algorithm to the normalized data, grouping players based on similarities in their drafted performance attributes. The algorithm initializes with random cluster centroids and iterates to minimize within-cluster variance by assigning players to the nearest cluster. This process continues until convergence, ensuring that the clusters stabilize.
- Determine the optimal number of clusters (k):
  - Use the Elbow Method to identify a reasonable value for k. The Elbow Method involves plotting the sum of squared distances (inertia) or within-cluster sum of squares (WSS) for a range of k values (typically from 1 to 10). The plot will show a sharp drop in WSS at first, followed by a less dramatic decline. The "elbow" point on this curve is where the rate of improvement in WSS slows down, suggesting that adding more clusters does not significantly improve the clustering. This point provides a starting estimate for k.
  - Supplement with Silhouette Analysis to further evaluate the quality of the clustering. Silhouette analysis calculates the silhouette score for each data point, which reflects how similar a point is to its own cluster compared to other clusters. The average silhouette score for each k is computed, and the k that maximizes this score indicates the most well-separated and coherent clusters. A high silhouette score implies that the clusters are distinct and well-defined, with minimal overlap between them.

- Decision Rule for Selecting the Optimal k:
  - If both methods suggest the same value of k, that value should be chosen as the optimal k. This consistency between methods strengthens the confidence in the selected k, as both approaches converge on the same conclusion.
  - If the methods suggest different values of k, prioritize Silhouette Analysis as it directly measures clustering quality by evaluating both cohesion within clusters and separation between them. A higher silhouette score indicates well-separated, internally consistent clusters. While the Elbow Method helps balance WSS and model complexity, it doesn't focus on cluster quality. Thus, Silhouette Analysis is more reliable for selecting the optimal number of clusters, ensuring the clusters are meaningful and distinct.
- Repeat the clustering process with different initializations (e.g., using `nstart = 25` to try multiple starting points) to confirm the stability and consistency of the results. This ensures that the clusters are not an artifact of a particular random initialization and that the results are robust across different runs.

Step 3: Model Evaluation

(1) Evaluate Cluster Characteristics:

- The first task is to analyze the characteristics of each cluster by inspecting the cluster centers, which helps us to understand the defining features of each group based on the average feature values for each cluster. By analyzing the cluster centers, we understand the key features that define each group. This is essential for interpreting the behavior or characteristics of each cluster (e.g., which cluster represents high-performing players).

(2) Visualize Clusters using Principal Component Analysis (PCA):

- PCA is used for dimensionality reduction. Since there are many features, reducing the data to two principal components helps in visualizing the clustering results in a 2D space, which allows us to see how well the clusters are separated and identify any overlap. The visualization can be done in two ways:
  - Using the `fviz_cluster` function from the `factoextra` package, which automatically handles PCA and cluster visualization.
  - Using `ggplot2` for a more manual and customizable visualization by first performing PCA.

(3) Calculate and Interpret Silhouette Scores:

- Silhouette scores are used to evaluate how well each data point fits into its assigned cluster. A high silhouette score indicates that the points are well-clustered and distant from other clusters. This is useful for determining the quality of the clustering and whether any clusters are overlapping or poorly defined. The silhouette score can be visualized using the `fviz_silhouette` function to assess cluster cohesion and separation.

Step 4: Implementation

- Use R programming for implementation, leveraging the following packages:
  - `stats` package for running the K-Means algorithm (`kmeans()`).
  - `factoextra` package for visualizing clusters and evaluating clustering quality using tools like the Elbow method and Silhouette analysis.
  - `cluster` package for additional clustering metrics and validation.

- Cluster Assignment and Summary: After running the K-Means algorithm, assign the resulting cluster labels to each player in the dataset, linking them to specific groups based on their performance metrics. Then, summarize key metrics such as total points, assists, and rebounds by calculating their average values for each cluster. This provides insights into the general characteristics of the players in each cluster, helping to identify patterns in their performance.
- Feature Contribution Analysis: Feature Contribution Analysis involves calculating the mean value of each feature for each cluster to identify which performance metrics are most dominant within each group. This helps pinpoint key features that differentiate clusters, such as players with high field goal percentages or total points. To visualize this, use bar plots or similar visualizations to display the mean values of features for each cluster, highlighting the features that contribute most to defining each group.
- Cluster Separation Analysis: Cluster Separation Analysis involves analyzing the differences in feature values across clusters by calculating the maximum and minimum values for each feature. This reveals which features most effectively distinguish one cluster from another. To assess the relative importance of each feature, evaluate the variation between the maximum and minimum values across clusters. Features with the greatest differences, such as "total points," are likely key in separating player groups, as they highlight significant distinctions between clusters.

Step 5: Generalization

- Identify Dominant Features: Reshape the cluster summary into a long format, excluding the `cluster` column from pivoting. For each cluster, select the top 5 features with the highest average values, representing the dominant metrics, and arrange them for clarity.
- Summarize Insights by Cluster: Loop through each cluster, extract its dominant features, and print a summary of key metrics that differentiate the cluster. These summaries provide generalizations about the characteristics of each group.
- Visualize and Save Results: Create a bar chart to visualize the dominant metrics for each cluster. Save the dominant features and their values to a CSV file for reporting and further analysis.

(c) Implement the selected algorithm and evaluate model performance diagnostics, presenting visuals as needed.

Model Application

```r
# Load necessary libraries
library(stats)       # For K-means clustering and statistical methods
library(factoextra)  # For cluster visualization and diagnostics
library(cluster)     # For silhouette analysis
library(tidyverse)   # For data manipulation and visualization
library(dplyr)       # For data manipulation functions

# Step 1: Data Preparation -----------------------------------------------

# Load the dataset containing NBA draft data
nba_draft_data <- read_excel("nba_draft.xlsx")

# Ensure all clustering features (e.g., performance metrics) are numeric after selection
player_data <- nba_draft_data %>%
  dplyr::select(where(is.numeric)) %>%  # Only numeric features
  dplyr::select(-pick, -year)  # Remove identifiers

###### Handle missing (NA), NaN, and Inf values ######

# Check for problematic values
cat("Checking for missing or problematic values...\n")
```

```
## Checking for missing or problematic values...
```

```r
cat("NA values:", any(is.na(player_data)), "\n")
```

```
## NA values: TRUE
```

```r
cat("NaN values:", any(sapply(player_data, function(x) any(is.nan(x)))), "\n")
```

```
## NaN values: FALSE
```

```r
cat("Infinite values:", any(sapply(player_data, function(x) any(is.infinite(x)))), "\n")
```

```
## Infinite values: FALSE
```

```r
# Apply is.infinite to each column and replace infinite values with NA
player_data[] <- lapply(player_data, function(x) {
  x[is.infinite(x)] <- NA  # Replace infinite values with NA
  return(x)
})

# Impute missing and NA values with the median of each column
player_data_imputed <- player_data %>%
  mutate(across(everything(), ~ ifelse(is.na(.), median(., na.rm = TRUE), .)))

# Verify that there are no more problematic values
cat("\nChecking again after cleaning...\n")
```

```
##
## Checking again after cleaning...
```

```r
cat("NA values:", any(is.na(player_data_imputed)), "\n")
```

```
## NA values: FALSE
```

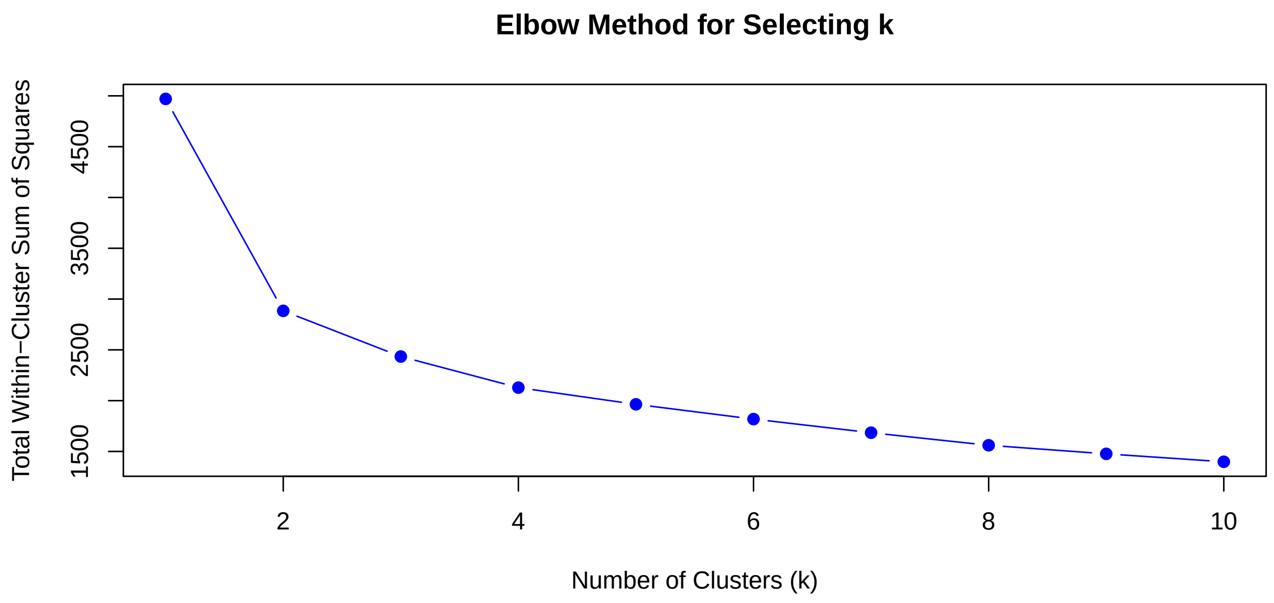
```r
cat("NaN values:", any(is.nan(as.matrix(player_data_imputed))), "\n")
```

```
## NaN values: FALSE
```

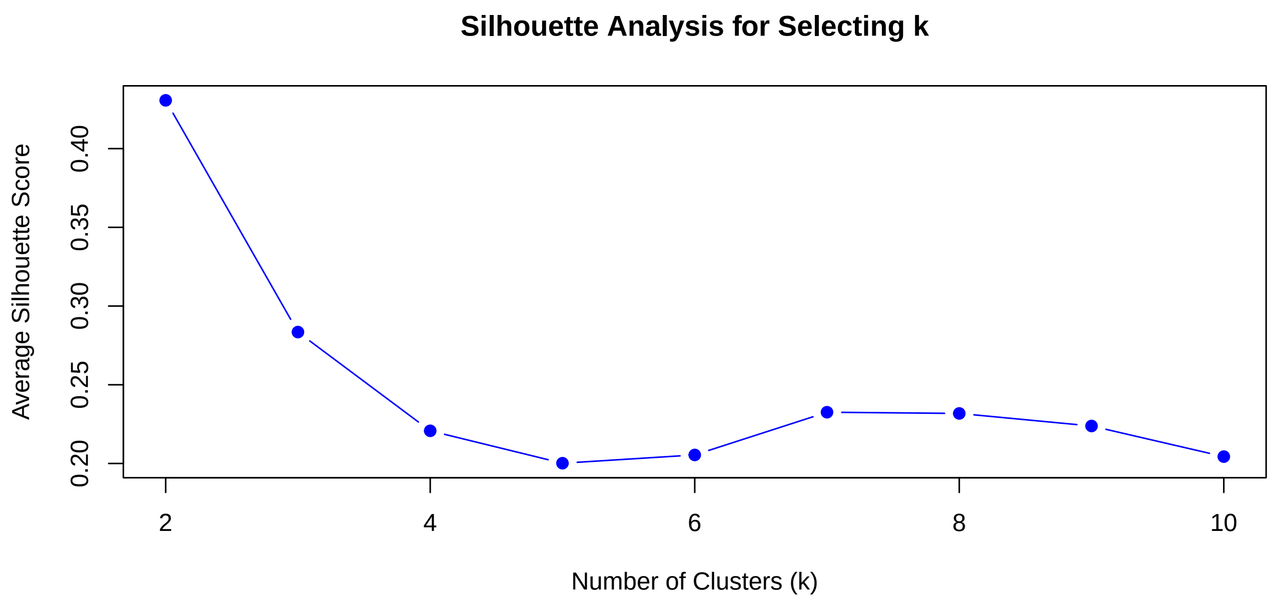
```r
cat("Infinite values:", any(is.infinite(as.matrix(player_data_imputed))), "\n")
```

```
## Infinite values: FALSE
```

```r
# Normalize the data to ensure all features are on the same scale
scaled_data_imputed <- scale(player_data_imputed)  # Z-score standardization

# Step 2: Clustering -------------------------------------------

# 2.1 Determine the optimal number of clusters (k)

# Elbow Method
# fviz_nbclust(scaled_data_imputed, kmeans, method = "wss") +
#  labs(title = "Elbow Method for Selecting k",) +  # Sum of squared distances plot
#  theme(plot.title = element_text(hjust = 0.5))  # Center the title

# Silhouette analysis
# fviz_nbclust(scaled_data_imputed, kmeans, method = "silhouette") +
#  labs(title = "Silhouette Analysis for Selecting k") +  # Silhouette analysis plot
#  theme(plot.title = element_text(hjust = 0.5))  # Center the title

# Based on the results of the above analyses, choose the optimal k = 2
# optimal_k <- 2
# cat("\nOptimal k determined from analyses:", optimal_k, "\n")

# Set seed for reproducibility
set.seed(1234)

# Elbow Method: Calculate WSS for different values of k
wss_values <- sapply(1:10, function(k) {
  kmeans_model <- kmeans(scaled_data_imputed, centers = k, nstart = 25)
  return(kmeans_model$tot.withinss)
})

# Plot WSS values (Elbow Method)
plot(1:10, wss_values, type = "b", pch = 19, col = "blue",
     xlab = "Number of Clusters (k)",
     ylab = "Total Within-Cluster Sum of Squares",
     main = "Elbow Method for Selecting k")
```



```r
elbow_k <- which.min(diff(diff(wss_values))) + 1  # Finding the "elbow"

# Silhouette Analysis: Calculate silhouette scores for different values of k
sil_scores <- sapply(2:10, function(k) {
  kmeans_model <- kmeans(scaled_data_imputed, centers = k, nstart = 25)
  sil <- silhouette(kmeans_model$cluster, dist(scaled_data_imputed))
  return(mean(sil[, 3]))  # average silhouette score
})

# Plot Silhouette scores
plot(2:10, sil_scores, type = "b", pch = 19, col = "blue",
     xlab = "Number of Clusters (k)",
     ylab = "Average Silhouette Score",
     main = "Silhouette Analysis for Selecting k")
```



```r
silhouette_k <- which.max(sil_scores) + 1  # highest silhouette score

# Print the optimal k calculated from the Elbow Method
cat("\nOptimal k from Elbow Method:", elbow_k, "\n")
```

```
##
## Optimal k from Elbow Method: 9
```

```r
# Print the optimal k calculated from the Silhouette Analysis
cat("Optimal k from Silhouette Analysis:", silhouette_k, "\n")
```

```
## Optimal k from Silhouette Analysis: 2
```

```r
# Decision rule: Choose k based on both methods
# Use the k suggested by both methods, or the one with the highest silhouette score
if (elbow_k == silhouette_k) {
  # If the two methods agree, use the value from the elbow method
  optimal_k <- elbow_k
} else {
  # If the methods suggest different values, prioritize the silhouette score
  optimal_k <- silhouette_k
}

# Display the optimal k chosen based on both methods
cat("\nOptimal k based on both methods:", optimal_k, "\n")
```

```
##
## Optimal k based on both methods: 2
```

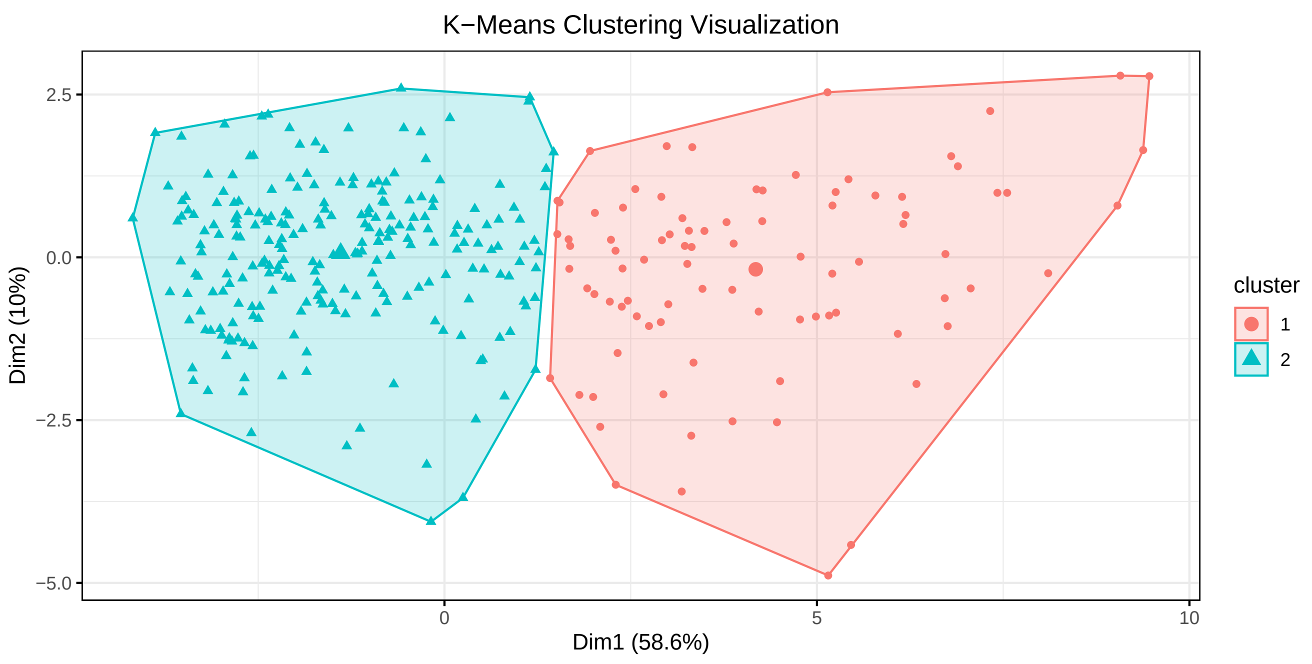
```
# 2.2 Apply the K-means clustering algorithm

# Set seed for reproducibility
set.seed(1234)
# Run K-means clustering with the optimal number of clusters
kmeans_model <- kmeans(scaled_data_imputed, centers = optimal_k, nstart = 25)

# Step 3: Model Evaluation ------------------------------------------------

# Set seed for reproducibility
set.seed(1234)

# 3.1 Analyze cluster centers to interpret team drafting behaviors
# Convert to a dataframe
cluster_centers <- as.data.frame(kmeans_model$centers)

# Print the cluster centers (average feature values for each cluster)
print("Cluster Centers (Average Feature Values):")
```

```
## [1] "Cluster Centers (Average Feature Values):"
```

```
print(cluster_centers)
```

```
##   years_in_draft total_games_played total_minutes_played total_points
## 1      1.1791698          1.3999694            1.4848180    1.4216891
## 2     -0.3930566         -0.4666565           -0.4949393   -0.4738964
##   total_rebounds total_assists field_goal_pct three_point_pct free_throw_pct
## 1      1.4295205     1.2349863      0.3325753      0.28676480     0.23773526
## 2     -0.4765068    -0.4116621     -0.1108584     -0.09558827    -0.07924509
##   avg_minutes_played avg_points avg_rebounds avg_assists win_shares
## 1          1.1207029  1.1502071    0.9193897   0.9120440  1.3890141
## 2         -0.3735676 -0.3834024   -0.3064632  -0.3040147 -0.4630047
```

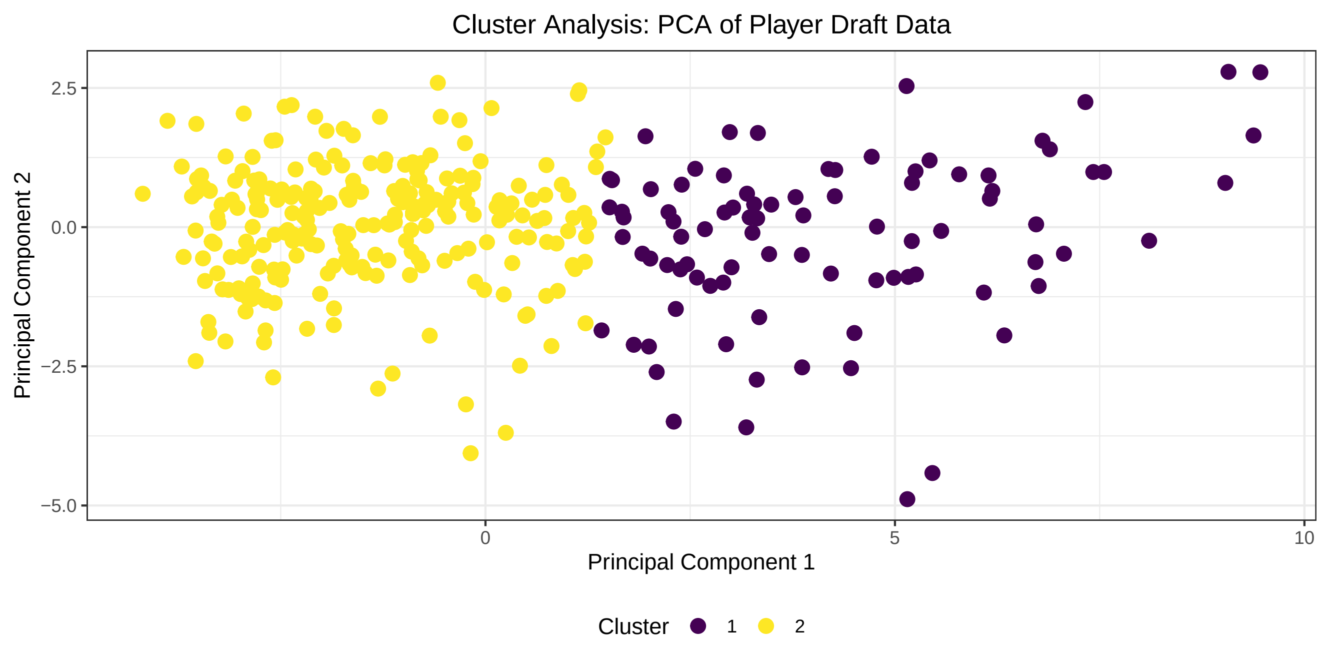
```
# Save the cluster centers to a CSV file
write.csv(as.data.frame(cluster_centers), "cluster_centers.csv", row.names = TRUE)

# 3.2 Visualize the clusters using PCA for dimensionality reduction
# Option 1: Using fviz_cluster from factoextra (automatically handles PCA)
fviz_cluster(kmeans_model, data = scaled_data_imputed,
             geom = "point", ellipse.type = "convex",
             ggtheme = theme_minimal()) +
  labs(title = "K-Means Clustering Visualization") +
  theme(plot.title = element_text(hjust = 0.5),  # Center the title
        panel.border = element_rect(colour = "black", fill = NA, size = 0.5)) # Add border
```



```r
# Option 2: Using PCA and ggplot2 (manual PCA for visualization)
pca_result <- prcomp(player_data_imputed, scale. = TRUE)  # Scale the data to standardize

# Add PCA components to the dataset
player_data_imputed$pca1 <- pca_result$x[, 1]  # First principal component
player_data_imputed$pca2 <- pca_result$x[, 2]  # Second principal component

# Add the cluster assignments to the original dataset
player_data_imputed$cluster <- kmeans_model$cluster  # Add cluster labels from K-means

# Visualize the clusters in the PCA space (2D representation)
cluster_viz <- player_data_imputed %>%
  ggplot(aes(x = pca1, y = pca2, color = as.factor(cluster))) +
  geom_point(size = 3) +
  labs(
    title = "Cluster Analysis: PCA of Player Draft Data",
    x = "Principal Component 1",
    y = "Principal Component 2",
    color = "Cluster"
  ) +
  theme(plot.title = element_text(hjust = 0.5))  # Center the title

# Print the PCA visualization
print("Saving cluster visualization...")
```

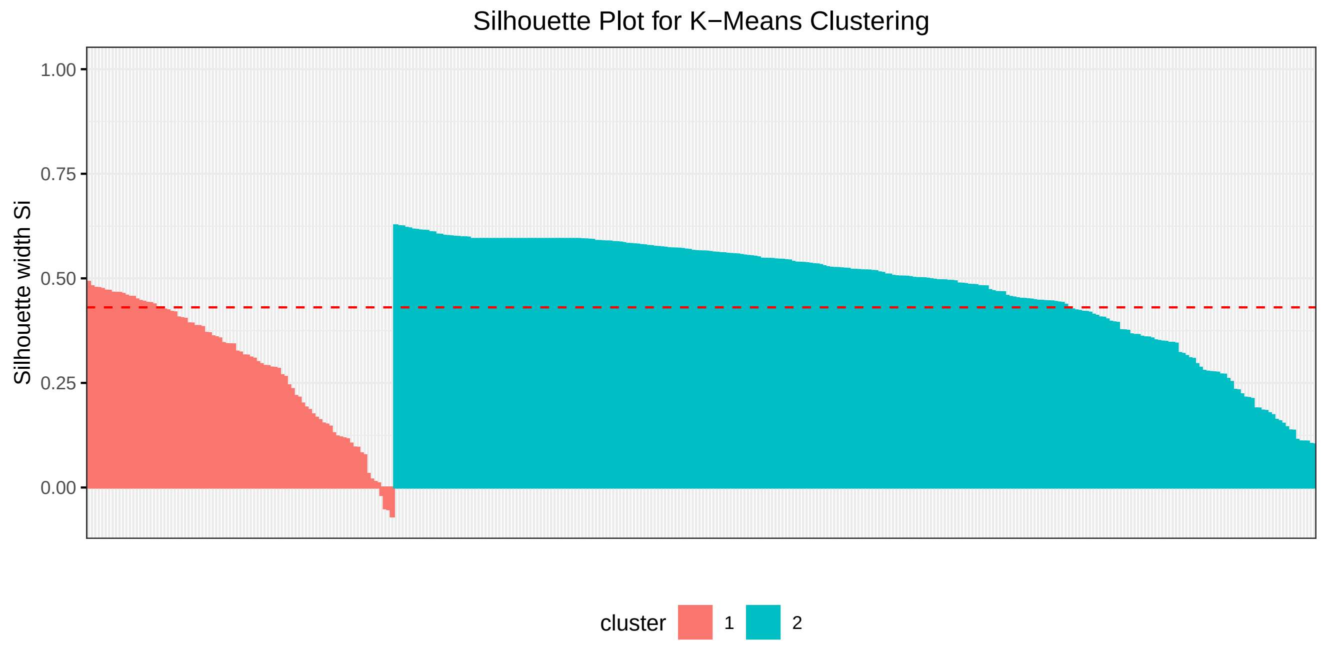
```
## [1] "Saving cluster visualization..."
```

```r
print(cluster_viz)
```



```r
# ggsave("cluster_visualization.png", plot = cluster_viz)

# 3.3 Calculate silhouette scores to evaluate cluster cohesion and separation
silhouette_scores <- silhouette(kmeans_model$cluster, dist(scaled_data_imputed))
# Visualize silhouette plot for cluster evaluation
fviz_silhouette(silhouette_scores) +
  labs(title = "Silhouette Plot for K-Means Clustering") +
  theme(plot.title = element_text(hjust = 0.5))  # Center the title
```

```
##   cluster size ave.sil.width
## 1       1   89          0.29
## 2       2  267          0.48
```



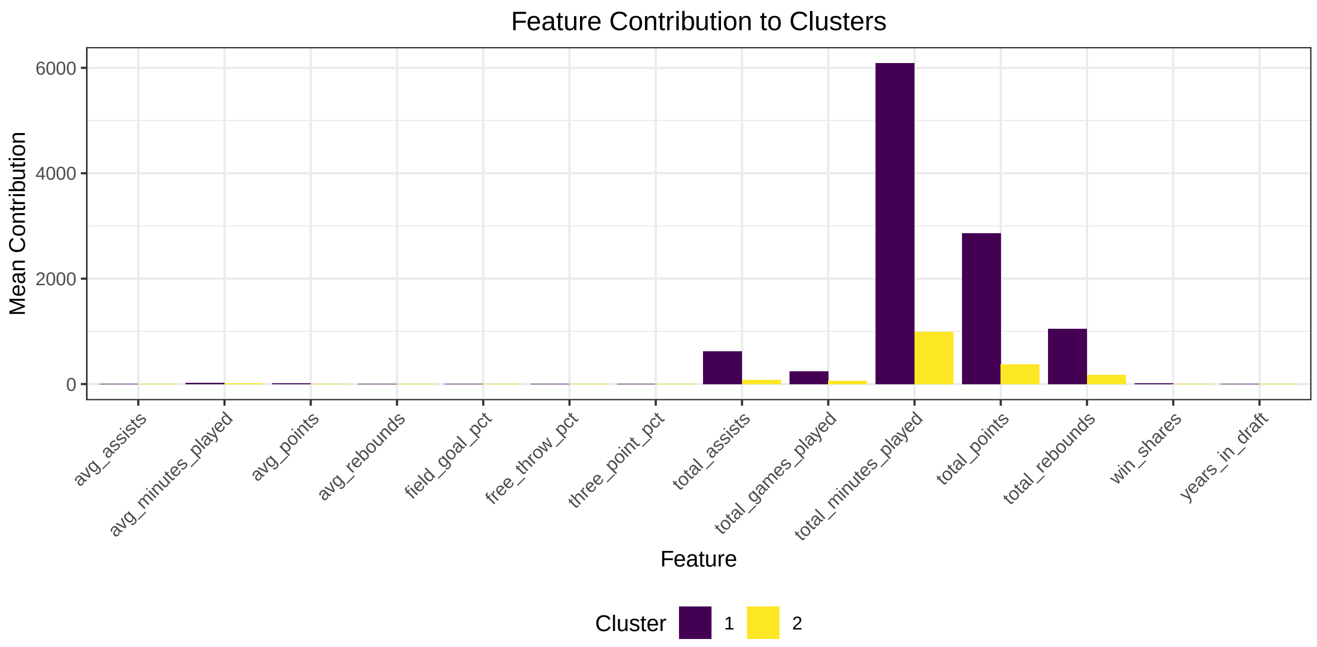
```r
# Step 4: Implementation and Insights ------------------------------------------------

# Add the cluster assignments to the original dataset
# Assign cluster labels from the K-means model to the imputed player data
# player_data_imputed$cluster <- kmeans_model$cluster

# Assign cluster labels from the K-means model to the NBA draft dataset
nba_draft_data$cluster <- kmeans_model$cluster

# Summarize clusters for insights
cluster_summary <- player_data_imputed %>%
  group_by(cluster) %>%
  summarize(across(
    where(is.numeric),
    ~mean(.x, na.rm = TRUE)))  # Use na.rm = TRUE to ignore NAs in the calculation

# Print the cluster summary
print("Cluster Summary (Average Performance Metrics by Cluster):")
```

```
## [1] "Cluster Summary (Average Performance Metrics by Cluster):"
```

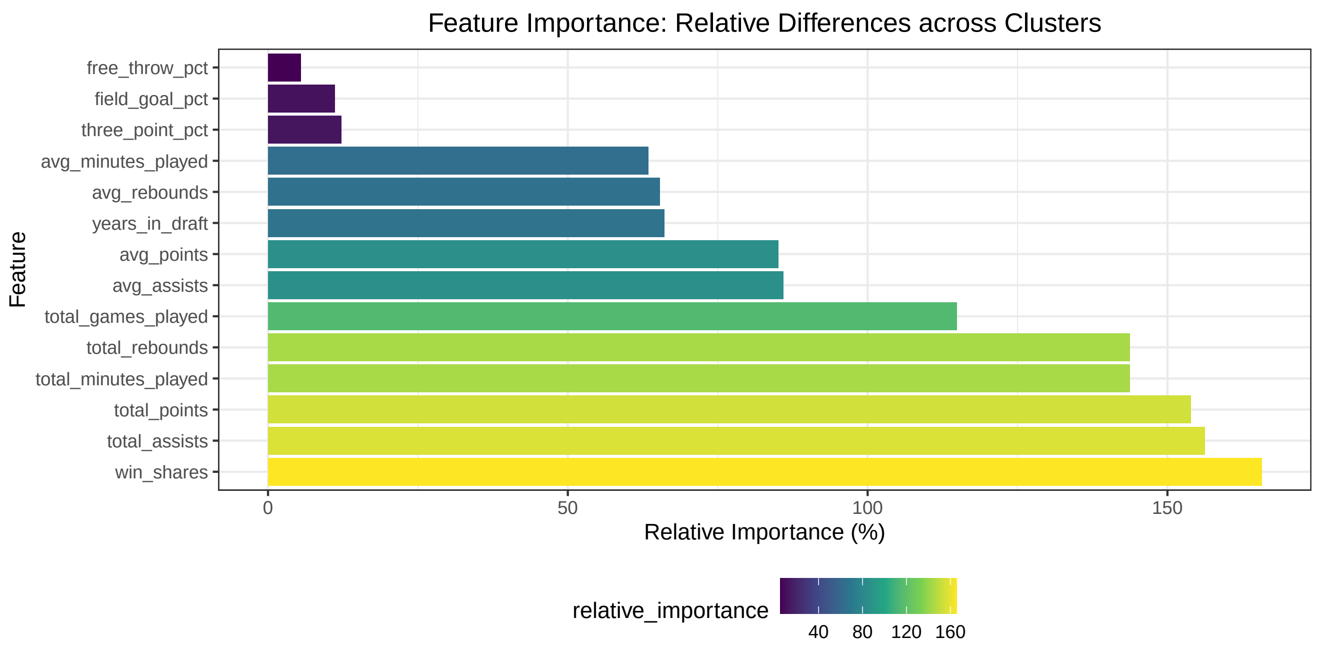
```r
print(cluster_summary)
```

```
## # A tibble: 2 x 17
##   cluster years_in_draft total_games_played total_minutes_played total_points
##     <int>          <dbl>              <dbl>                <dbl>        <dbl>
## 1       1           4.72              240.                 6086.        2864.
## 2       2           2.37               65.0                 996.         373.
## # i 12 more variables: total_rebounds <dbl>, total_assists <dbl>,
## #   field_goal_pct <dbl>, three_point_pct <dbl>, free_throw_pct <dbl>,
## #   avg_minutes_played <dbl>, avg_points <dbl>, avg_rebounds <dbl>,
## #   avg_assists <dbl>, win_shares <dbl>, pca1 <dbl>, pca2 <dbl>
```

```r
# Save cluster assignments and summary for reporting
# Clustered data with assignments
write.csv(nba_draft_data, "draft_clusters.csv", row.names = FALSE)
# Summary of each cluster
write.csv(cluster_summary, "cluster_summary.csv", row.names = FALSE)

###### Feature Contribution Analysis ######
# Emphasize mean values of features at each cluster center
# Give insights into which features dominate within each cluster
# Compare Average Performance Metrics across Clusters

# Reshape cluster_summary for contribution analysis
feature_contributions <- cluster_summary %>%
  pivot_longer(cols = -c(cluster, pca1, pca2), names_to = "feature", values_to = "mean_value")

# Visualize contributions
feature_contributions %>%
  ggplot(aes(x = feature, y = mean_value, fill = as.factor(cluster))) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(
    title = "Feature Contribution to Clusters",
    x = "Feature",
    y = "Mean Contribution",
    fill = "Cluster"
  ) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),  # Rotate x-axis labels
        plot.title = element_text(hjust = 0.5))  # Center the title
```



```
###### Cluster Separation Analysis ######
# Focus on how much each feature contributes to the separation of clusters
# Highlight the separation power of each feature

# Calculate relative differences between cluster averages
cluster_differences <- cluster_summary %>%
  pivot_longer(cols = -c(cluster, pca1, pca2), names_to = "metric",
               values_to = "mean_value") %>%
  group_by(metric) %>%
  summarize(
    # Largest difference across clusters
    max_difference = max(mean_value) - min(mean_value),
    # Relative % difference
    relative_importance = (max(mean_value) - min(mean_value)) / mean(mean_value) * 100
  ) %>%
  arrange(desc(relative_importance))  # Sort by relative importance

# Visualize feature importance based on differences
cluster_differences %>%
  ggplot(aes(x = reorder(metric, -relative_importance), y = relative_importance,
             fill = relative_importance)) +
  geom_bar(stat = "identity") +
  coord_flip() +  # Flip for better readability
  labs(
    title = "Feature Importance: Relative Differences across Clusters",
    x = "Feature",
    y = "Relative Importance (%)"
  ) +
  theme(plot.title = element_text(hjust = 0.5))  # Center the title
```



```
# Step 5: Generalizations from Findings ---------------------------------------------

# Analyze cluster characteristics
print("Analyzing generalizations from clusters...")
```

```
## [1] "Analyzing generalizations from clusters..."
```

```
# Identify dominant features for each cluster
dominant_features <- cluster_summary %>%
  # Exclude 'cluster' from pivoting
  pivot_longer(cols = -cluster, names_to = "metric", values_to = "value") %>%
  group_by(cluster) %>%  # Group by cluster
  slice_max(value, n = 5, with_ties = FALSE) %>%  # Top 5 metrics per cluster
  arrange(cluster, desc(value))  # Arrange for clarity

# Print the dominant features in each cluster
print("Dominant features in each cluster:")
```

```
## [1] "Dominant features in each cluster:"
```

```
print(dominant_features)
```

```
## # A tibble: 10 x 3
## # Groups:   cluster [2]
##    cluster metric                value
##      <int> <chr>                 <dbl>
##  1       1 total_minutes_played 6086.
##  2       1 total_points         2864.
##  3       1 total_rebounds       1045.
##  4       1 total_assists         625.
##  5       1 total_games_played    240.
##  6       2 total_minutes_played  996.
##  7       2 total_points          373.
##  8       2 total_rebounds        171.
##  9       2 total_assists          76.8
## 10       2 total_games_played     65.0
```

```
# Generalization: What differentiates clusters
cat("\nKey insights by cluster:\n")
```

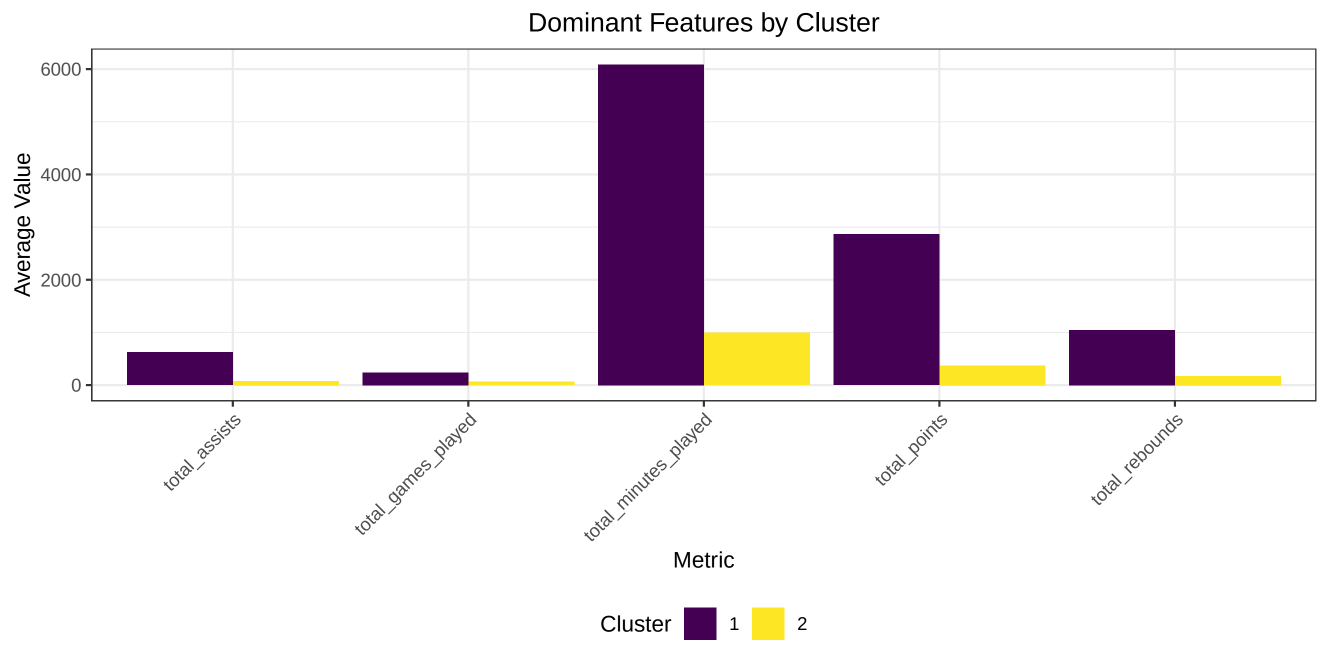
```
##
## Key insights by cluster:
```

```
for (i in unique(dominant_features$cluster)) {
  cat("\nCluster", i, "Insights:\n")
  cluster_data <- dominant_features %>% filter(cluster == i)
  print(cluster_data)
  cat("\nThis cluster shows high values in:\n")
  cat(paste(cluster_data$metric, collapse = "\n"), "\n")
}
```

```
##
## Cluster 1 Insights:
## # A tibble: 5 x 3
## # Groups:   cluster [1]
##   cluster metric               value
##     <int> <chr>                <dbl>
## 1       1 total_minutes_played 6086.
## 2       1 total_points         2864.
## 3       1 total_rebounds       1045.
## 4       1 total_assists         625.
## 5       1 total_games_played    240.
##
## This cluster shows high values in:
## total_minutes_played
## total_points
## total_rebounds
## total_assists
## total_games_played
##
## Cluster 2 Insights:
## # A tibble: 5 x 3
## # Groups:   cluster [1]
##   cluster metric               value
##     <int> <chr>                <dbl>
## 1       2 total_minutes_played 996.
## 2       2 total_points         373.
## 3       2 total_rebounds       171.
## 4       2 total_assists         76.8
## 5       2 total_games_played    65.0
##
```

```
## This cluster shows high values in:
## total_minutes_played
## total_points
## total_rebounds
## total_assists
## total_games_played
```

```r
# Visualize dominant features in each cluster
dominant_features %>%
  ggplot(aes(x = metric, y = value, fill = as.factor(cluster))) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(
    title = "Dominant Features by Cluster",
    x = "Metric",
    y = "Average Value",
    fill = "Cluster"
  ) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        plot.title = element_text(hjust = 0.5))  # Center the title
```



```r
# Save key findings for reporting
write.csv(dominant_features, "dominant_features_by_cluster.csv", row.names = FALSE)
```

Model Diagnostics

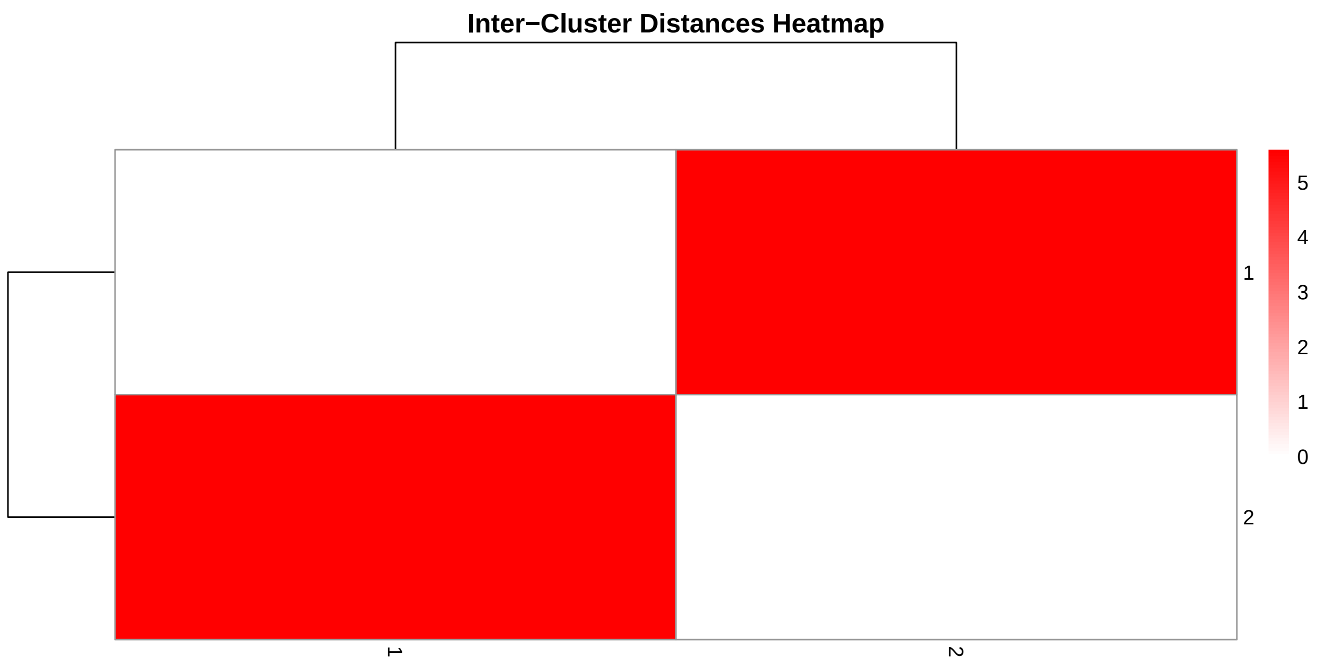
```r
###### Inter-Cluster and Intra-Cluster Distances ######

library(pheatmap)  # For creating heatmaps

# Compute distances between clusters
cluster_distances <- dist(kmeans_model$centers)

# Print pairwise cluster distances
cat("\nPairwise Cluster Distances:\n")
```

```
##
## Pairwise Cluster Distances:
```

```r
print(as.matrix(cluster_distances))
```

```
##          1        2
## 1 0.000000 5.590425
## 2 5.590425 0.000000
```

```r
# Heatmap of inter-cluster distances
pheatmap(as.matrix(cluster_distances),
         color = colorRampPalette(c("white", "red"))(100),
         main = "Inter-Cluster Distances Heatmap")
```



```r
# Analyze intra-cluster variance (tightness of clusters)
intra_cluster_variance <- aggregate(
  scaled_data_imputed,
  by = list(Cluster = kmeans_model$cluster),
  FUN = function(x) var(x, na.rm = TRUE))
```

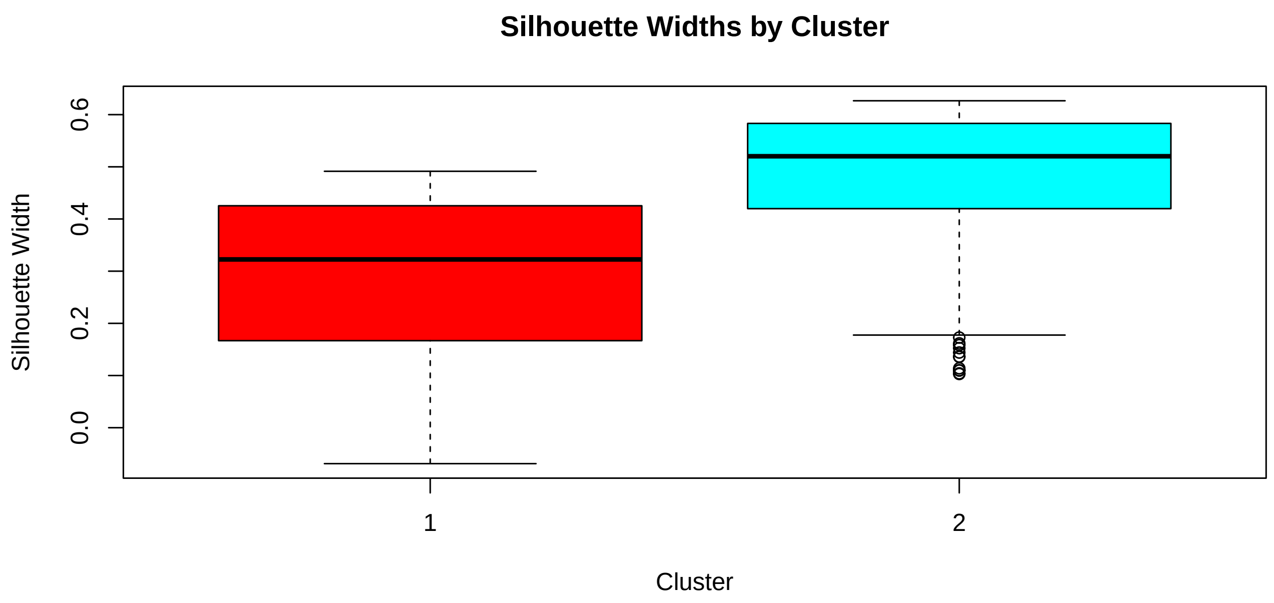
```
# Print intra-cluster variance
cat("\nIntra-Cluster Variance:\n")
```

```
##
## Intra-Cluster Variance:
```

```
print(intra_cluster_variance)
```

```
##   Cluster years_in_draft total_games_played total_minutes_played total_points
## 1       1      0.5006657          0.4282143            0.6430843   1.04728819
## 2       2      0.5486555          0.3185739            0.1382933   0.08642707
##   total_rebounds total_assists field_goal_pct three_point_pct free_throw_pct
## 1      0.7978020    1.75522541      0.3522582       0.4476701      0.3370862
## 2      0.1590025    0.07349934      1.1687068       1.1497992      1.1978557
##   avg_minutes_played avg_points avg_rebounds avg_assists win_shares
## 1          0.4534199  1.0236156    1.0870535   1.7136357 1.18495864
## 2          0.6242733  0.4057467    0.5978686   0.3965798 0.08185299
```

```
# Save the results as a CSV file
write.csv(as.data.frame(intra_cluster_variance), "intra_cluster_variance.csv",
          row.names = FALSE)

###### Analyze Cluster Silhouette Widths ######

# Extract silhouette widths
sil_widths <- silhouette_scores[, 3]

# Boxplot of silhouette widths by cluster
boxplot(sil_widths ~ kmeans_model$cluster,
        col = rainbow(optimal_k),
        main = "Silhouette Widths by Cluster",
        xlab = "Cluster", ylab = "Silhouette Width")
```



```r
# Overall average silhouette score
avg_silhouette_score <- mean(sil_widths)

# Print average silhouette score
cat("\nAverage Silhouette Score:", avg_silhouette_score, "\n")
```

```
##
## Average Silhouette Score: 0.4306632
```

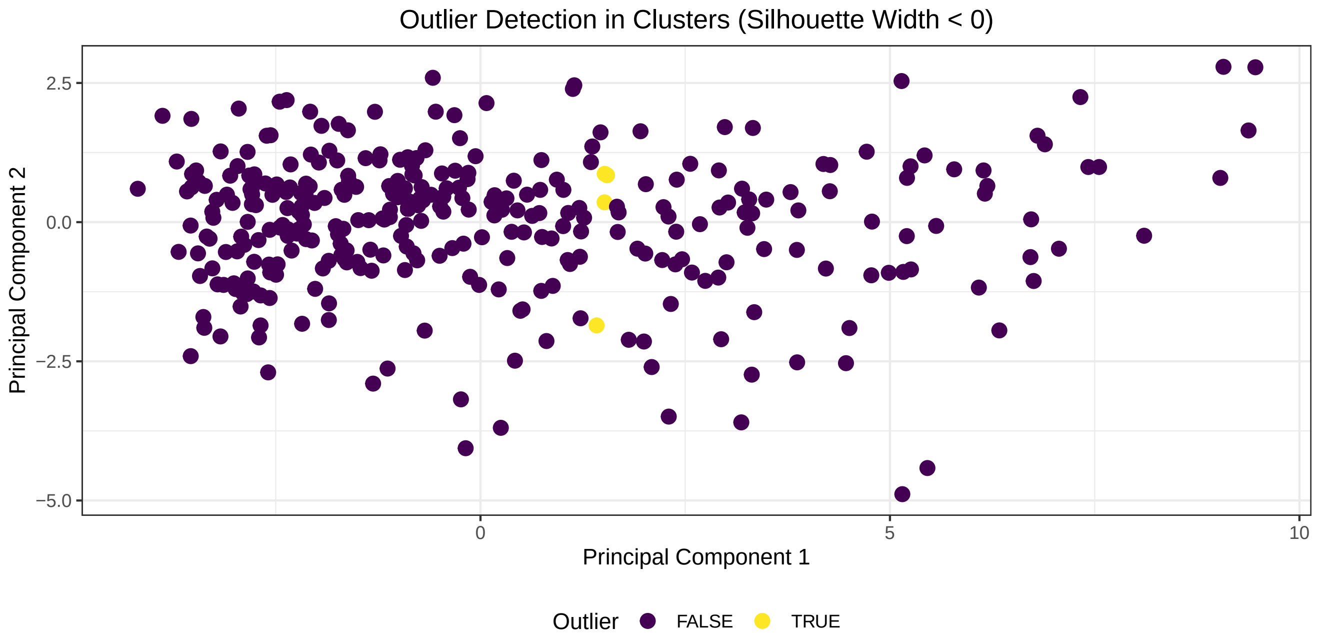
```r
###### Optional: Outlier Detection ######

# Identify points with low silhouette scores (potential outliers)
low_silhouette_points <- which(sil_widths < 0)

# Print points with low silhouette widths (potential outliers)
cat("\nPoints with Low Silhouette Widths:\n")
```

```
##
## Points with Low Silhouette Widths:
```

```r
print(low_silhouette_points)
```

```
## [1] 194 201 294 320
```

```r
# Mark points with low silhouette scores as outliers
player_data_imputed$outlier <- sil_widths < 0

# Visualize outliers in the PCA space
outliers_plot <- ggplot(player_data_imputed, aes(x = pca1, y = pca2, color = outlier)) +
  geom_point(size = 3) +
  labs(
    title = "Outlier Detection in Clusters (Silhouette Width < 0)",
    x = "Principal Component 1",
    y = "Principal Component 2",
    color = "Outlier"
  ) +
  theme(plot.title = element_text(hjust = 0.5))  # Center the title
print(outliers_plot)
```



(d) Interpret your results, presenting visuals as needed.

Choose the optimal k:

When selecting the optimal number of clusters (k) for a clustering model like K-Means, two commonly used methods are the Elbow Method and Silhouette Analysis.

- Elbow Plot: The Elbow Method identifies k=9 as the optimal number of clusters, suggesting that beyond k=9, further increases in the number of clusters do not substantially reduce the Within-Cluster Sum of Squares (WSS). While this result highlights k=9 as a possible choice, the method alone does not evaluate the quality of the cluster separation or cohesion.
- Silhouette Plot: The Silhouette Analysis shows the highest average silhouette score at k=2. This indicates that for k=2, the clusters are well-separated, and the data points within each cluster are relatively cohesive. A high silhouette score suggests that k=2 provides meaningful and well-defined clusters, making it a reliable choice for clustering quality.
- Final Choice: Despite the Elbow Method suggesting k=9, the Silhouette Analysis indicates that k=2 provides better-defined and higher-quality clusters. Since the Silhouette Analysis directly evaluates the clustering quality, it is prioritized over the Elbow Method in this case. Thus, k=2 is selected as the optimal number of clusters for the K-Means clustering model. This decision ensures that the chosen clusters are both meaningful and well-separated, aligning with the goal of creating distinct groupings in the data.

Model Performance Interpretation:

1. Cluster Center Analysis

The cluster centers provide insights into the defining characteristics of each group based on the standardized feature values. The clear distinction between the two clusters aligns with the goal of identifying meaningful groupings based on player performance.

(1) Cluster 1: Players in this group exhibit positive values for all performance metrics, indicating above-average performance.

- The smallest value is for free throw percentage (`free_throw_pct`: **0.23773526**), while the largest is for total minutes played (`total_minutes_played`: **1.4848180**).
- These players consistently show higher values across metrics such as total games played, total minutes, total points, average points, average rebounds, and win shares.
- This suggests that Cluster 1 represents high-performing and experienced players who excel across multiple dimensions of on-court performance.

(2) Cluster 2: Players in this group exhibit negative values for all performance metrics, indicating below-average performance.

- The smallest value is for total minutes played (`total_minutes_played`: **-0.4949393**), while the largest is for free throw percentage (`free_throw_pct`: **-0.07924509**).
- Metrics such as total games played, minutes, points, and win shares are all lower compared to Cluster 1.
- This group likely represents players with lower overall impact and less experience, performing below average in all measured areas.

2. Visualization of Clusters (PCA Analysis)

- The PCA visualization provides a two-dimensional representation of the clusters. The convex ellipses help illustrate the separation between clusters. If the clusters overlap significantly, it may indicate a need to refine the features or reevaluate the choice of k. From the results, clusters appear reasonably separated, reinforcing the meaningfulness of the identified groups.

3. Silhouette Analysis

- The silhouette scores assess the cohesion (similarity within clusters) and separation (distinctiveness from other clusters):
  - Cluster 1: Average silhouette width = **0.29**. A relatively low score indicates that some data points in this cluster may be close to the boundary of another cluster. This suggests moderate cohesion and separation.
  - Cluster 2: Average silhouette width = **0.48**. A higher score reflects better-defined clusters with stronger internal cohesion and clear separation from the other cluster.
- Overall, the global silhouette score is not exceptionally high. However, the distinct characteristics of the clusters based on their centers suggest meaningful separation in terms of player performance.
- Based on the meaningful insights from the cluster centers and the reasonable silhouette score for Cluster 2, the clustering model effectively captures key patterns in the data. While there is potential for further refinement, such as revisiting the features or experimenting with different values of k, the current model already provides valuable insights and performs well in identifying the distinct groupings.

Cluster Summary:

1. Cluster 1: Established Players

(1) Key Characteristics:

- Years in Draft: 4.72 years on average, indicating that players in this cluster are more experienced and likely well-established in the league.

- Total Games Played: 240.36 games on average, indicating that these players have spent more time on the court, likely as regular starters or key rotation players.
- Total Minutes Played: 6086.38 minutes on average, reflecting that these players are more heavily involved in their teams' games, contributing substantial playing time.
- Total Points: 2863.74 points on average, highlighting that the players in Cluster 1 are significant contributors on offense, either by scoring or assisting.

(2) Performance Metrics:

- Field Goal Percentage (FG%): 47.95% on average, suggesting that players in Cluster 1 are efficient in scoring, which is typical of established players with more experience.
- Win Shares: 11.28 on average, showing the total contribution of a player to their team's wins. A high number (11.28) indicates that these players are effective contributors to their teams' success.

(3) Per-Game Metrics:

- Average Minutes Played: 25.44 minutes, reinforcing the idea that Cluster 1 players are regular contributors, likely playing significant roles in games.
- Average Points: 12.08 points, consistent with the profile of established players who often take on scoring responsibilities.
- Average Rebounds: 4.52 rebounds, showing a well-rounded contribution to other areas of the game beyond scoring.
- Average Assists: 2.64 assists, suggesting that these players not only score but also help in facilitating the offense.

2. Cluster 2: Developing or Peripheral Players

(1) Key Characteristics:

- Years in Draft: 2.37 years on average, indicating that these players are relatively new to the league and still developing their skills.
- Total Games Played: 64.99 games on average, significantly lower than Cluster 1, indicating that these players have spent less time on the court, possibly due to being bench players, injured, or less frequently used in games.
- Total Minutes Played: 996.42 minutes on average, further supporting the idea that these players are peripheral to their teams' core rotations.
- Total Points: 372.83 points on average, reflecting that these players are not as heavily involved in scoring, likely due to their role as developing or secondary players.

(2) Performance Metrics:

- Field Goal Percentage (FG%): 42.89% on average, lower than Cluster 1, indicating less efficiency in scoring, possibly because these players are still refining their skills or taking lower-quality shots.
- Win Shares: 1.06 on average, much lower than Cluster 1, suggesting that these players have a smaller impact on their teams' success, either because they are not key contributors or are on teams with lower performance.

(3) Per-Game Metrics:

- Average Minutes Played: 13.19 minutes, consistent with the idea that these players play fewer minutes, perhaps due to limited roles or development-focused teams.
- Average Points: 4.87 points, showing that these players may not be primary offensive options.
- Average Rebounds: 2.29 rebounds, indicating that these players may not be as active on the boards.
- Average Assists: 1.05 assists, further suggesting that these players are likely more focused on developing rather than facilitating team plays.

3. Key Differences Between Clusters

- Years in Draft: Cluster 1 players are more experienced, with nearly double the average years in the draft compared to Cluster 2. This contributes to their higher performance levels and more consistent contributions.
- Games Played & Minutes Played: Cluster 1 players, averaging 240.36 games and 6086.38 minutes, are central figures in their teams, often as starters or regular rotation players. In contrast, Cluster 2 players, with just 64.99 games and 996.42 minutes, typically play peripheral roles with less court time.
- Field Goal Percentage: Cluster 1 players shoot more efficiently (47.95% FG%) compared to Cluster 2 (42.89% FG%). This suggests that experience plays a role in shooting efficiency, with established players being more proficient in their shot selection and execution.
- Total Points: Cluster 1 players score significantly more (2863.74 total points) compared to Cluster 2 (372.83 total points), highlighting their greater offensive roles.
- Win Shares: Cluster 1's higher Win Shares (11.28) point to their greater contribution to their team's success. Cluster 2's low Win Shares (1.06) suggest that they are less impactful on their team's overall performance.

Insights on Feature Importance:

To analyze feature importance, we examine which metrics most effectively distinguish between the two clusters:

- Win Shares: This feature is highly important in separating the two clusters. Cluster 1 players are significantly more impactful, contributing to their teams' success, while Cluster 2 players have minimal impact.
- Total Points, Total Assists, Total Rebounds, and Minutes Played: These metrics distinguish the clusters clearly. Cluster 1 players contribute much more in terms of total points and assists, reflecting their established roles and experience in the league. In contrast, Cluster 2 players contribute less across these metrics, confirming their status as peripheral or developing players.
- Field Goal Percentage (and other shooting percentages): Although these features show some differences between clusters, they are not as impactful as total points or win shares. This suggests that while shooting efficiency plays a role in a player's effectiveness, it is less important than their overall involvement in games (i.e., minutes played, points scored, etc.).

Feature Importance Chart Interpretation:

- The chart visualizes the relative importance of features by showing the largest differences between cluster averages. Features like Win Shares, Total Points, and Minutes Played have the largest differences between the two clusters. These metrics dominate the distinction between Cluster 1 (established players) and Cluster 2 (developing players), highlighting that experience and contributions in these areas are crucial for distinguishing between more seasoned players and those still developing their roles.

Final Thoughts:

- Cluster 1 represents experienced, high-contribution players who are central to their teams' success. Their higher win shares, total points, and field goal percentage indicate that they are more efficient and valuable on the court.
- Cluster 2 represents newer or peripheral players with fewer minutes, lower performance metrics, and a smaller impact on their teams' success.
- The feature importance analysis confirms that the most significant differentiators between clusters are win shares (team contribution), total points, and total minutes played, with other features like shooting efficiency showing less contrast between the groups.

Model Diagnostics Interpretation:

1. Pairwise Cluster Distances

   - The distance between cluster 1 and cluster 2 is **5.590425**, which indicates the separation between these two clusters in the feature space (on a scale where larger numbers suggest more distinct groups).
   - A distance of **5.590425** is fairly large, suggesting that the two clusters are well-separated. This is a positive indicator of the clustering, meaning that the members of the two clusters differ significantly from each other based on the chosen features. If the distance were smaller, it would imply that the clusters overlap, which would lower the quality of the clustering.

2. Intra-Cluster Variance

   - Purpose: Intra-cluster variance measures the tightness of the clusters, i.e., how similar players are within each cluster. Lower variance means players are more alike, while higher variance suggests greater diversity.
   - Interpretation of Intra-Cluster Variance:
     - Cluster 1 exhibits more variability, particularly in features like `total_assists` (1.755), `avg_assists` (1.714), and `win_shares` (1.185), where the maximum values (e.g., `total_assists` = 1.755) are notably higher than the minimum (e.g., `free_throw_pct` = 0.337), showing greater diversity in player performance within the cluster. This indicates that Cluster 1 contains players with a wide range of abilities and performance levels, with some players showing high performance while others are more consistent.
     - Cluster 2 shows more consistency in player performance. The lower variability between features, such as the minimum value for `total_assists` (0.073) and the maximum values like `free_throw_pct` (1.198), `field_goal_pct` (1.169), and `three_point_pct` (1.150), suggests that the players in this cluster have more homogeneous performance levels. This narrower range of variability contributes to the overall consistency of Cluster 2.
   - Conclusion: Cluster 1 has more variance, with players displaying a wide range of performances across different metrics. Cluster 2 is more consistent, with players showing more uniform performances across key metrics.

3. Average Silhouette Score

   - Silhouette Score: An average silhouette score of **0.4306632** indicates that the clustering is moderately good, with reasonably distinct clusters but room for improvement. This score suggests that while the clusters are somewhat separated, there may be overlap or ambiguity between them, meaning some players might not fit perfectly into their assigned clusters. Although the score is not poor, refining the model could enhance the separation and clarity of the cluster boundaries.

4. Outlier Detection

   - Low Silhouette Widths: Points with low silhouette widths (typically < 0) are considered outliers or misclassified points. The points 194, 201, 294, 320 have negative silhouette scores, meaning they likely don't belong well to their assigned clusters.

- Interpretation: These points might be misclassified due to unusual feature values or represent edge cases that require additional preprocessing or feature engineering. However, the PCA plot shows that these outliers are mixed with other points, suggesting they do not deviate significantly from the overall data distribution. This could mean they are valid but rare player profiles rather than noise.
- Action: Instead of removal, further investigation is recommended to understand their unique characteristics. These points might provide valuable insights into edge cases that could refine the clustering approach.

5. Insights and Recommendations

(1) Distinctiveness of Clusters:

- The two clusters are well-separated, as indicated by the large inter-cluster distance (**5.590425**). However, the moderate average silhouette score (**0.4306632**) suggests there may still be some overlap or ambiguity in assigning players to clusters.
- Cluster 1 exhibits higher intra-cluster variance, indicating a mix of players with diverse performance levels. This cluster likely includes both inconsistent contributors and high-potential outliers. Cluster 2 shows lower variance and greater consistency in player performance, representing more homogeneous and dependable profiles.

(2) Outliers:

- Players with negative silhouette scores (e.g., 194, 201, 294, 320) require further analysis. These outliers might represent unique player profiles rather than noise or misclassifications. Investigating their specific characteristics could provide insights into specialized roles or rare skill sets.

(3) Actionable Insights:

- Cluster Utilization:
  - Cluster 1: Players in this cluster show greater variability in performance metrics. This makes them ideal candidates for targeted development programs to maximize their potential or for teams willing to invest in players with long-term growth prospects. Managing this variability effectively could unlock significant value for the team.
  - Cluster 2: These players demonstrate more consistent and predictable performance across key metrics. Teams can rely on them for steady contributions, making them valuable assets for maintaining consistent team dynamics and output.
- Draft Strategy:
  - Cluster 1: Best suited for teams prioritizing long-term development and growth. These players may not make an immediate impact but offer high potential for future success with proper coaching and training.
  - Cluster 2: Ideal for teams looking for immediate, reliable contributions. Players in this cluster are dependable and can quickly adapt to professional-level demands, providing a solid foundation for team performance.

(4) Future Steps:

- Model Refinement: Test additional cluster numbers (e.g., k = 3 or 4) to capture more nuanced player profiles. Explore advanced algorithms like Gaussian Mixture Models or hierarchical clustering to validate cluster quality.
- Feature Engineering: Refine or add features that better capture key differences in player performance, potentially improving silhouette scores and reducing overlap.
- Outlier Analysis: Retain outliers for further investigation to uncover edge cases or rare player archetypes that might have strategic value.
- Validation: Cross-validate findings with external benchmarks such as historical player success or team performance to ensure practical relevance.

(e) Are there any generalizations that can be made from your findings?

Yes, generalizations can be drawn from the findings by interpreting the dominant features of each cluster and linking them to meaningful patterns. Below is a potential generalization of the results:

1. Cluster Profiles

   (1) Cluster 1: Players in Cluster 1 are characterized by significantly higher values in total minutes played (6086.38), total points (2863.74), total rebounds (1044.69), total assists (625.44), and total games played (240.36).

   - These metrics indicate that Cluster 1 players are key contributors for their teams, taking on significant roles as starters or regular rotation players.
   - The high number of games and minutes suggests they have consistent involvement on the court, likely due to their skill, reliability, and experience.
   - Their strong performance in scoring (points) and playmaking (assists) further reflects their versatility and centrality to team strategies.

   (2) Cluster 2: Players in Cluster 2 show much lower values in the same metrics: total minutes played (996.42), total points (372.83), total rebounds (171.07), total assists (76.84), and total games played (64.99).

   - These players are more likely to be peripheral contributors or developmental players, as evidenced by their limited involvement in games and shorter playing time.
   - Their lower stats suggest they are either still growing in their roles, recovering from injury, or less integral to team performance.

2. Comparison Between Clusters

   (1) Court Time and Contribution: Cluster 1 players average over 6 times the total minutes played and games compared to Cluster 2 players, demonstrating their greater court presence and reliability as core team members.

   - Cluster 1: 6086.38 minutes, 240.36 games
   - Cluster 2: 996.42 minutes, 64.99 games

   (2) Performance Metrics: Cluster 1 significantly outperforms Cluster 2 in all key metrics, including points (2863.74 vs. 372.83), rebounds (1044.69 vs. 171.07), and assists (625.44 vs. 76.84). This suggests that Cluster 1 players not only play more but also contribute more actively across multiple aspects of the game.

   (3) Player Roles: Cluster 1 players likely represent veterans or high-performing starters, while Cluster 2 players might include rookies, role players, or reserves. This aligns with their overall performance profiles and playing time.

3. Common Patterns or Trends

   - Both clusters share a focus on total contributions, as seen in the dominance of metrics like total minutes played, total points, and total rebounds. However, the magnitude of these contributions significantly distinguishes the clusters.
   - The metrics suggest a progression from Cluster 2 to Cluster 1: players in Cluster 2 might aspire to reach the performance levels seen in Cluster 1, depending on their development, opportunity, and team roles.

4. Contextual Insights

In the context of NBA draft data:

- Cluster 1 likely represents players who are high-value draft picks or established players aimed at making an immediate impact. These players could be sought by teams looking for proven performance and versatility.
- Cluster 2 may reflect developmental or situational players, either younger prospects who need time to grow into their roles or players who fit specific niche roles on their teams. Teams drafting these players might prioritize potential or specific skillsets over immediate contribution.

5. Generalized Insights

- Cluster 1 players are experienced, versatile contributors who are central to their teams' success, as reflected in their high involvement and well-rounded performance metrics.
- Cluster 2 players are less established, with more limited contributions, likely representing developmental players or those with specific, less prominent roles.
- The differences between the clusters highlight the stratification within player roles, from core team members (Cluster 1) to peripheral contributors or players in development (Cluster 2).
- These insights can guide drafting strategies, resource allocation, and player development decisions, tailoring approaches based on the expected contribution levels from each group.

Drafting Strategies Based on Cluster Analysis

1. Cluster 1: Immediate Contributors and All-Around Performers

Cluster 1 players are defined by high values in metrics like total minutes played, points, rebounds, and assists. These characteristics suggest they are established, versatile contributors. Drafting strategies for this group include:

- Target for Immediate Impact: Cluster 1 players, ready to contribute significantly, are ideal for playoff-contending teams seeking proven performers to bolster their starting lineup or rotation in high-stakes situations.
- Invest in All-Around Talent: Their well-rounded performance across scoring, rebounding, and playmaking makes them adaptable to various roles, whether leading the offense or supporting teammates defensively.
- Prioritize Durability and Reliability: High totals in games and minutes played indicate consistency and the ability to handle demanding workloads, making them dependable assets for a full season.
- Secure Star-Level Potential: These players often align with star profiles and leadership roles, justifying the use of high draft picks or trade assets for long-term value and immediate impact.

2. Cluster 2: Developmental Prospects and Role Specialists

Cluster 2 players have significantly lower metrics, indicating limited current contributions but potential for growth. Teams can adopt the following strategies:

- Draft for Development: These younger players with untapped potential are best suited for teams with strong development programs and a long-term focus, making them ideal candidates for later-round or second-round picks.

- Seek Specialized Skillsets: Despite their lower overall metrics, Cluster 2 players may excel in niche areas like shooting, defense, or playmaking, making them valuable draft picks for teams seeking to fill specific roster gaps or add specialized role players.
- Embrace Risk-Reward Opportunities: Cluster 2 players could blossom into valuable contributors under the right coaching and system fit. Teams can take a calculated risk on these "project players" with lower draft picks.
- Align with Rebuilding Timelines: These players suit teams in rebuilding phases, offering the chance to grow alongside other young talents and reach their peak as the team develops.

3. General Drafting Insights

- High Draft Picks: Should prioritize Cluster 1 players for immediate impact, leadership, and overall reliability, particularly for teams in win-now situations.
- Later Draft Picks: Cluster 2 players are better suited for late first-round or second-round picks, providing developmental upside or specialized skillsets for a lower investment.
- Role-Specific Recruitment: Cluster 1 players fit leadership and high-minute starter roles. Cluster 2 players are ideal for bench depth or specific tasks like defense or three-point shooting.
- Mitigating Draft Risks: By providing clear profiles, cluster analysis helps teams reduce uncertainty, aligning player selection with team needs, whether aiming for short-term success or long-term growth.

4. Conclusion

- Cluster 1 players are ideal for teams looking to strengthen their core with immediate starters or key rotation players, while Cluster 2 players represent developmental projects or role specialists for the future. Combining these insights allows teams to strategically balance short-term impact and long-term potential, optimizing drafting decisions.