Appendix

Phebe Chen

2024-11-15

```
# Load Libraries
library(readxl)
library(tidyjson)
```

```
##
## Attaching package: 'tidyjson'

## The following object is masked from 'package:stats':
##
##     filter
```

```
library(corrplot)
```

```
## corrplot 0.95 loaded
```

```
library(reshape2)
library(ggplot2)
library(caret)
```

```
## Loading required package: lattice
```

```
library(randomForest)
```

```
## randomForest 4.7-1.2

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(gbm)
```

```
## Loaded gbm 2.2.2

## This version of gbm is no longer under development. Consider transitioning to gbm3, https://github.c
```

```r
library(Metrics)
```

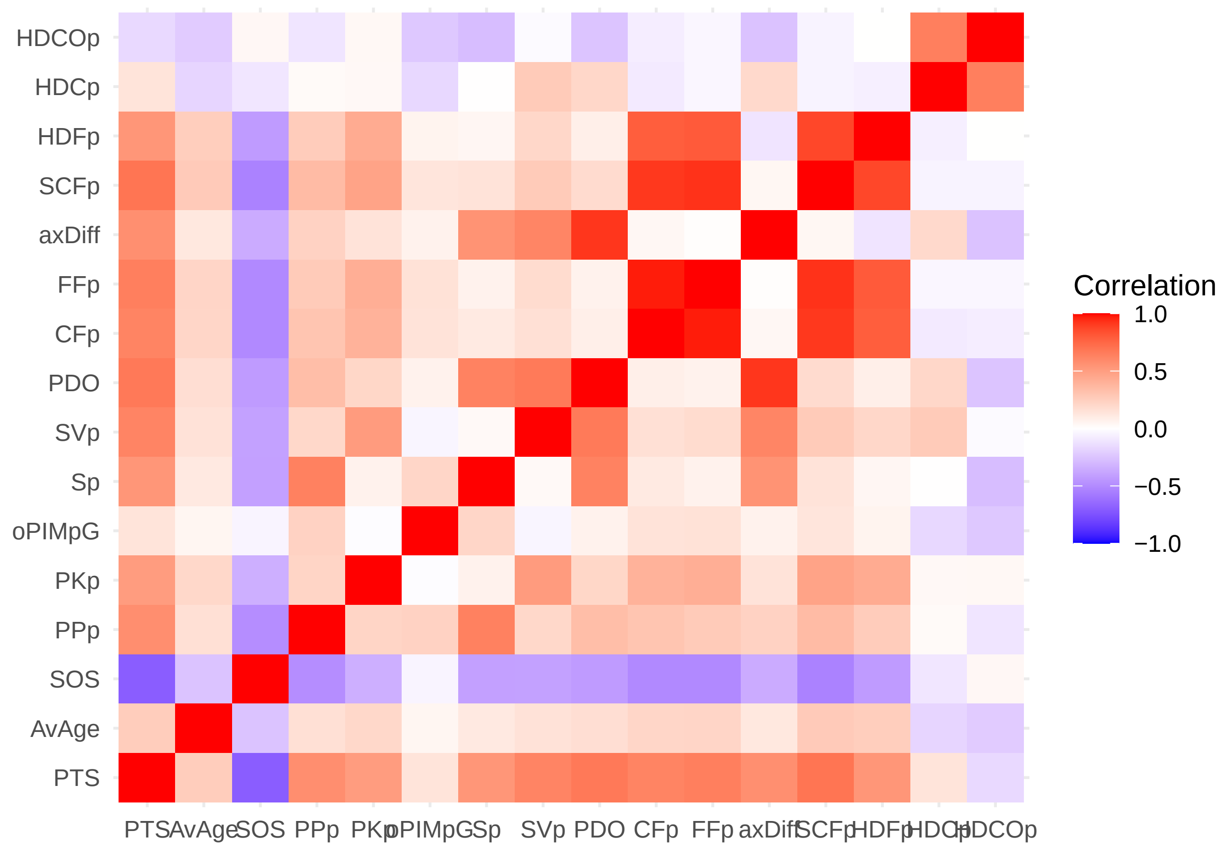
```
##
## Attaching package: 'Metrics'

## The following objects are masked from 'package:caret':
##
##     precision, recall
```

```r
library(tidyr)
```

```
##
## Attaching package: 'tidyr'

## The following object is masked from 'package:reshape2':
##
##     smiths
```

```r
# Section 1: Data Cleaning
file_path <- "seasonDataset.xlsx"
sheets <- excel_sheets(file_path)
data <- lapply(sheets, function(sheet) read_excel(file_path, sheet = sheet))
nhl_data <- bind_rows(data)

# Select only numeric columns and drop "GP"
numeric_data <- nhl_data[, sapply(nhl_data, is.numeric)]
numeric_data <- numeric_data[, setdiff(names(numeric_data), "GP")]
```

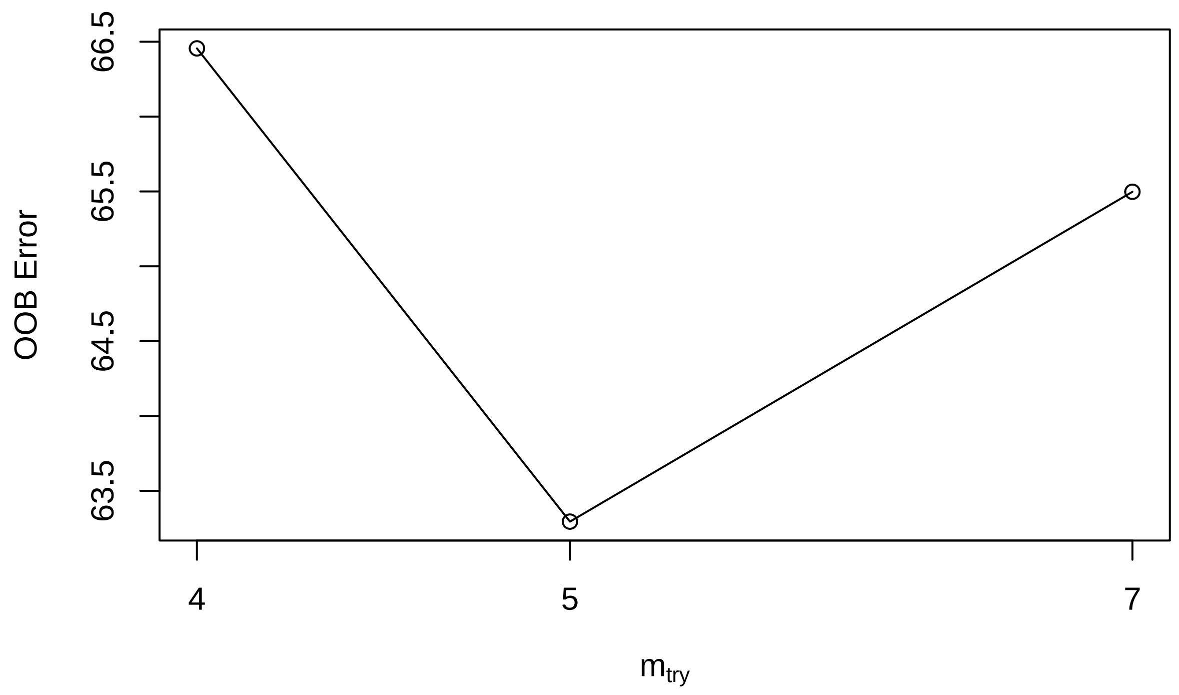
```r
# Section 2: Correlation Analysis
cor_matrix <- cor(numeric_data)
cor_pts <- cor_matrix[, "PTS"]

# Drop redundant variables & filter
columns_to_keep <- c("PTS", "AvAge", "SOS", "PPp", "PKp", "oPIMpG", "Sp", "SVp",
                     "PDO", "CFp", "FFp", "axDiff", "SCFp", "HDFp", "HDCp", "HDCOp")
nhl_data_filtered <- nhl_data[, columns_to_keep]

# Visualize Correlation Matrix
cor_matrix_filtered <- cor(nhl_data_filtered)
melted_cor <- melt(cor_matrix_filtered)
ggplot(data = melted_cor, aes(x = Var1, y = Var2, fill = value)) +
  geom_tile() +
  scale_fill_gradient2(low = "blue", high = "red", limit = c(-1, 1), name = "Correlation") +
  theme_minimal() +
  theme(axis.title.x = element_blank(), axis.title.y = element_blank())
```



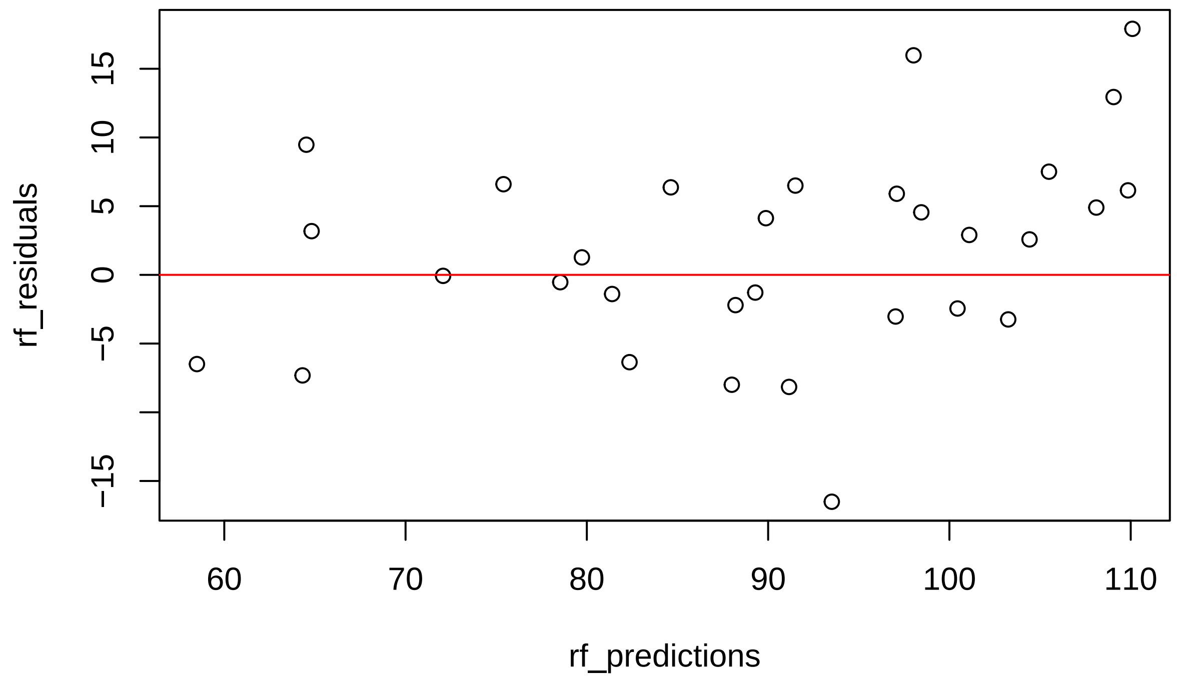
```r
# Section 3: Train-Test Split
set.seed(123)
train_index <- createDataPartition(nhl_data_filtered$PTS, p = 0.8, list = FALSE)
train_data <- nhl_data_filtered[train_index, ]
test_data <- nhl_data_filtered[-train_index, ]
```

```r
# Section 4: Random Forest Model
tune_rf <- tuneRF(train_data[, -which(names(train_data) == "PTS")], train_data$PTS, stepFactor = 1.5, i
```

```
## mtry = 5  OOB error = 63.29442
## Searching left ...
## mtry = 4     OOB error = 66.45564
## -0.04994471 0.01
## Searching right ...
## mtry = 7     OOB error = 65.49779
## -0.03481151 0.01
```



```
#300 trees to reduce overfitting (small dataset)
set.seed(123)
rf_model <- randomForest(PTS ~ ., data = train_data, mtry = 7, ntree = 300, importance = TRUE)
rf_predictions <- predict(rf_model, newdata = test_data)

rf_residuals <- test_data$PTS - rf_predictions
plot(rf_predictions, rf_residuals)
abline(h = 0, col = "red")
```



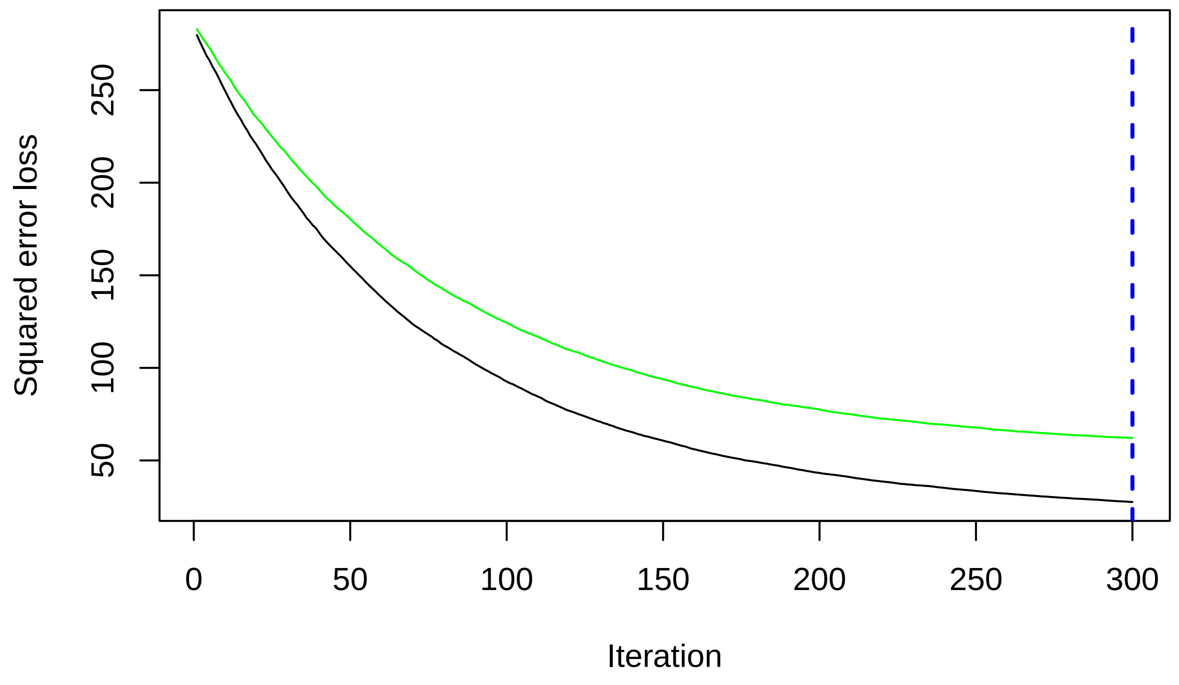
```r
# Evaluate RF Model
rmse_rf <- rmse(test_data$PTS, rf_predictions)
rsq_rf <- cor(test_data$PTS, rf_predictions)^2
cat("Random Forest - RMSE:", rmse_rf, "R-squared:", rsq_rf, "\n")
```

```
## Random Forest - RMSE: 7.514855 R-squared: 0.8568228
```

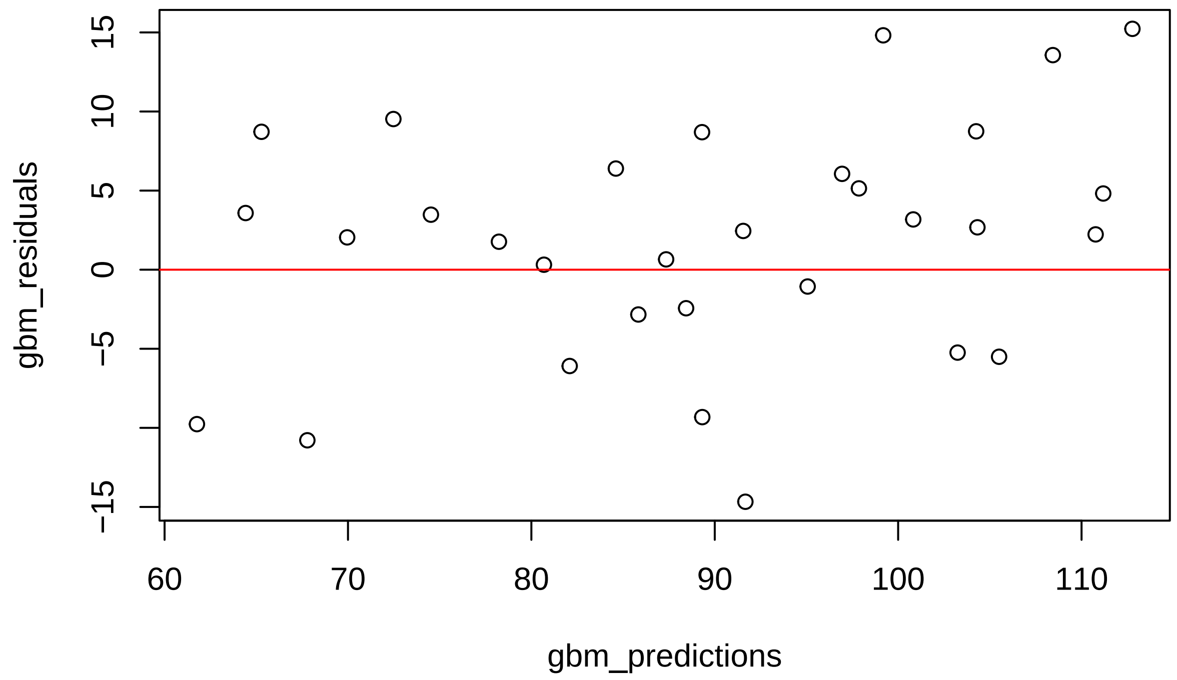
```r
# Cross-validation for Random Forest
set.seed(123)
rf_cv <- train(PTS ~ ., data = train_data,
               method = "rf",
               trControl = trainControl(method = "cv", number = 10),
               tuneGrid = expand.grid(mtry = 7),
               ntree = 300)

rf_predictions_cv <- predict(rf_cv, newdata = test_data)

# Evaluate Random Forest model
rmse_rf_cv <- rmse(test_data$PTS, rf_predictions_cv)
rsq_rf_cv <- cor(test_data$PTS, rf_predictions_cv)^2
cat("Random Forest - RMSE:", rmse_rf_cv, "R-squared:", rsq_rf_cv, "\n")
```

```
## Random Forest - RMSE: 7.602834 R-squared: 0.8545144
```

```r
# Section 5: GBM Model
set.seed(123)
gbm_model <- gbm(PTS ~ ., data = train_data, distribution = "gaussian",
                 n.trees = 300, interaction.depth = 5, shrinkage = 0.01, cv.folds = 5)
best_trees <- gbm.perf(gbm_model, method = "cv")
```



```r
gbm_predictions <- predict(gbm_model, test_data, n.trees = best_trees)

gbm_residuals <- test_data$PTS - gbm_predictions
plot(gbm_predictions, gbm_residuals)
abline(h = 0, col = "red")
```



```r
# Evaluate GBM Model
rmse_gbm <- rmse(test_data$PTS, gbm_predictions)
rsq_gbm <- cor(test_data$PTS, gbm_predictions)^2
cat("GBM - RMSE:", rmse_gbm, "R-squared:", rsq_gbm, "\n")
```

```
## GBM - RMSE: 7.547409 R-squared: 0.8520441
```

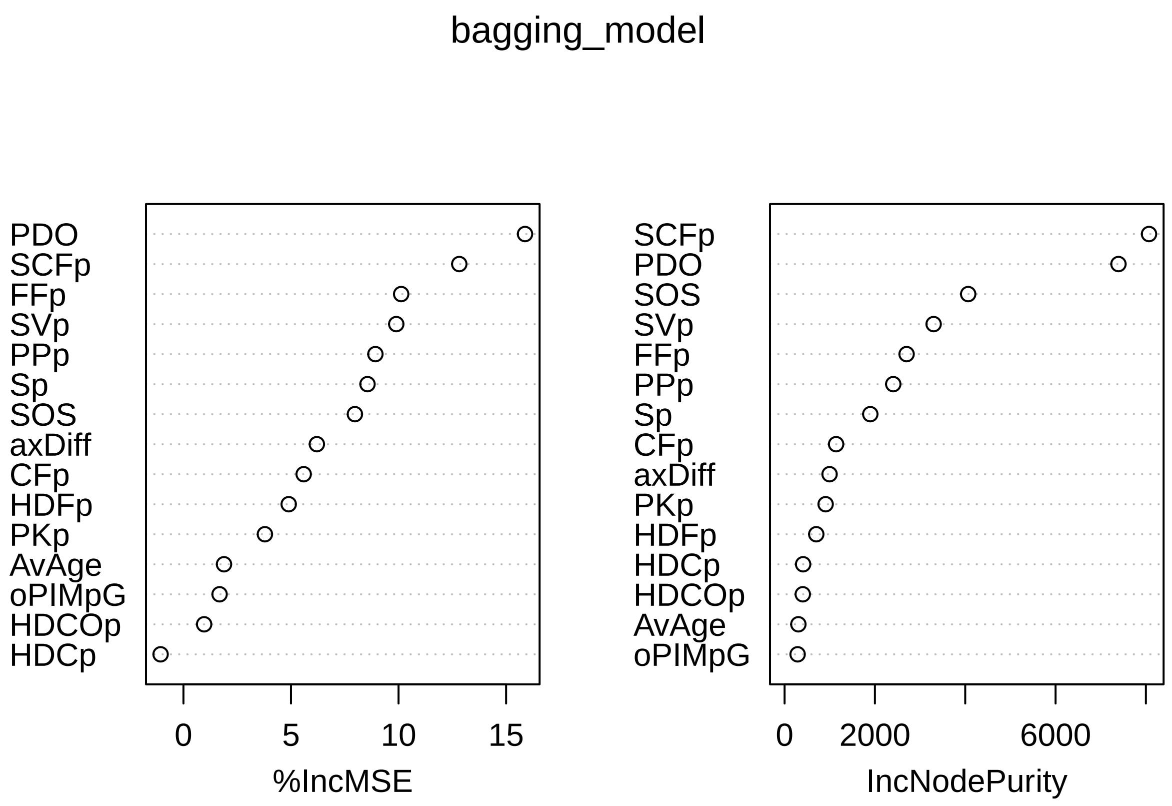
```r
# Cross-validation for GBM
set.seed(123)

# Define tuning grid with correct parameters for GBM
gbm_grid <- expand.grid(
  n.trees = c(100, 200, 300),
  interaction.depth = c(3, 5, 7),
  shrinkage = c(0.01, 0.1),
  n.minobsinnode = c(10, 20)
)

# Train the GBM model using caret with 10-fold cross-validation
gbm_cv <- train(
  PTS ~ .,
  data = train_data,
  method = "gbm",
  trControl = trainControl(method = "cv", number = 10),
  tuneGrid = gbm_grid,
  verbose = FALSE
)

# Get the best model and make predictions
gbm_predictions_cv <- predict(gbm_cv, newdata = test_data)

# Evaluate the GBM model
rmse_gbm_cv <- rmse(test_data$PTS, gbm_predictions_cv)
rsq_gbm_cv <- cor(test_data$PTS, gbm_predictions_cv)^2
cat("GBM - RMSE:", rmse_gbm_cv, "R-squared:", rsq_gbm_cv, "\n")
```

```
## GBM - RMSE: 7.324712 R-squared: 0.8574602
```

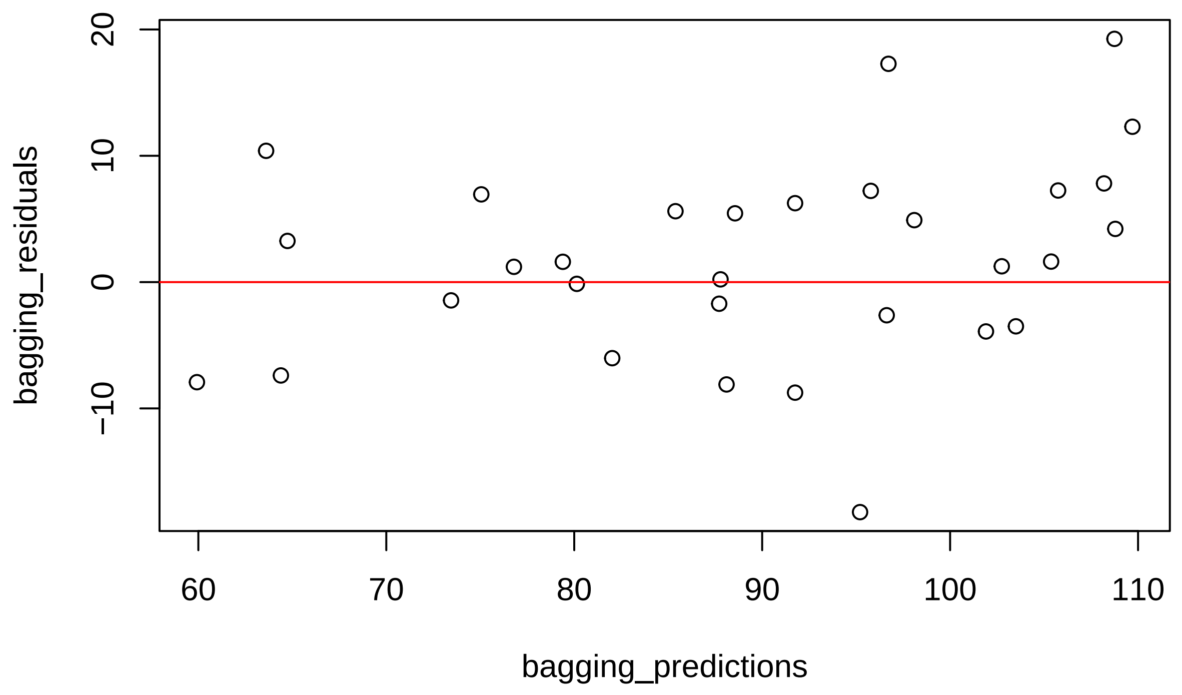
```r
# Section 6: Bagging
set.seed(123)
bagging_model <- randomForest(PTS ~ ., data = train_data, mtry = ncol(train_data) - 1,
                              importance = TRUE, ntree = 200)
importance(bagging_model)
```

```
##           %IncMSE IncNodePurity
## AvAge   1.8857131      305.3603
## SOS     7.9708661     4065.2239
## PPp     8.9236187     2405.9426
## PKp     3.7873342      909.1050
## oPIMpG  1.6795372      289.1515
## Sp      8.5562809     1897.8392
## SVp     9.8943423     3300.1974
## PDO    15.8799504     7392.7431
## CFp     5.5904913     1141.0801
## FFp    10.1227632     2701.4835
## axDiff  6.2022986      996.1368
## SCFp   12.8233852     8069.3328
## HDFp    4.8950224      701.9289
## HDCp   -1.0627580      410.4966
## HDCOp   0.9619299      402.9820
```

```r
varImpPlot(bagging_model)
```



```r
bagging_predictions <- predict(bagging_model, test_data)
bagging_residuals <- test_data$PTS - bagging_predictions
plot(bagging_predictions, bagging_residuals)
abline(h = 0, col = "red")
```



```
#Evaluate Bagging Model
rmse_bagging <- rmse(test_data$PTS, bagging_predictions)
rsq_bagging <- cor(test_data$PTS, bagging_predictions)^2
cat("Bagging - RMSE:", rmse_bagging, "R-squared:", rsq_bagging, "\n")
```

```
## Bagging - RMSE: 7.972839 R-squared: 0.8327265
```

```
# Cross-validation for Bagging (Random Forest)
set.seed(123)  # Set seed for reproducibility
bagging_cv <- train(PTS ~ ., data = train_data,
                    method = "rf",
                    trControl = trainControl(method = "cv", number = 10),
                    tuneGrid = expand.grid(mtry = ncol(train_data) - 1),
                    ntree = 200)  # Set number of trees

bagging_predictions_cv <- predict(bagging_cv, newdata = test_data)

# Evaluate Bagging model
rmse_bagging_cv <- rmse(test_data$PTS, bagging_predictions_cv)
rsq_bagging_cv <- cor(test_data$PTS, bagging_predictions_cv)^2
cat("Bagging - RMSE:", rmse_bagging_cv, "R-squared:", rsq_bagging_cv, "\n")
```

```
## Bagging - RMSE: 7.845305 R-squared: 0.8403278
```

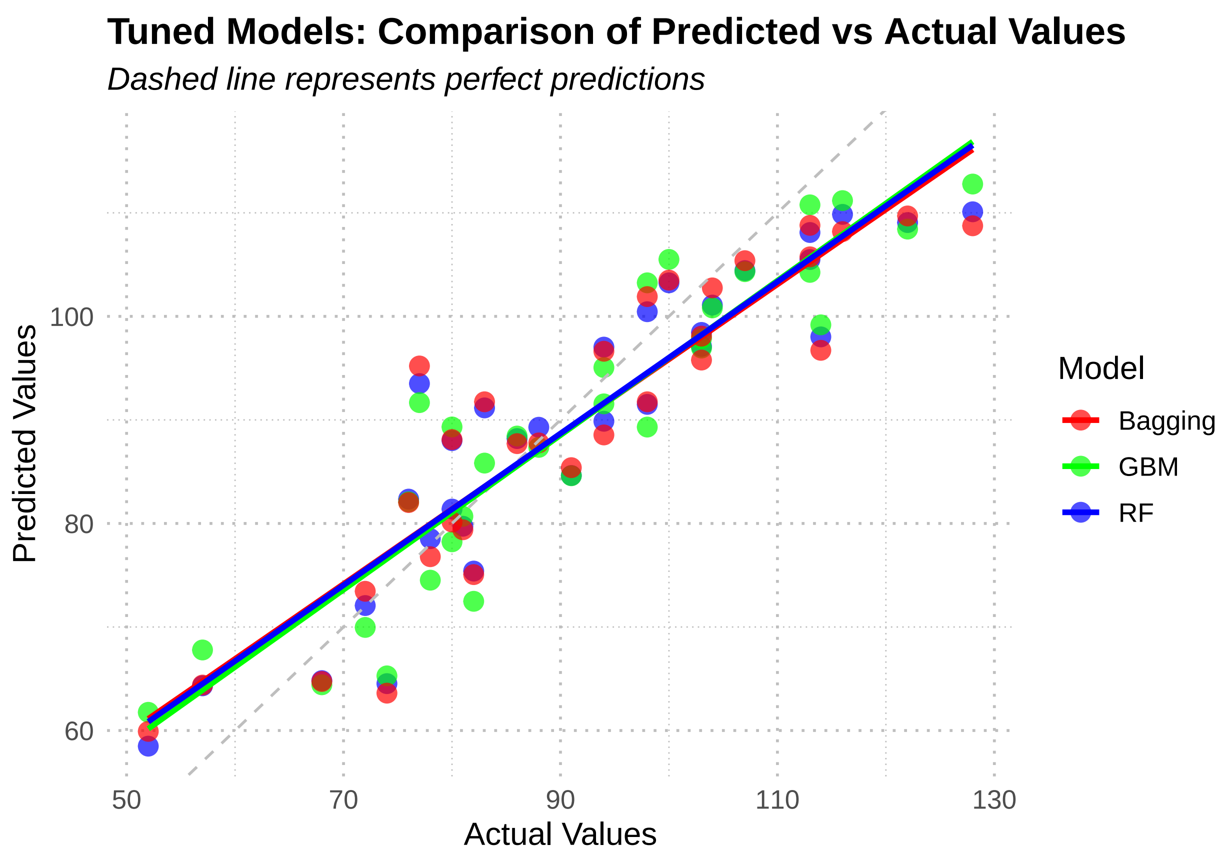
```r
# Section 7: Model Comparison

# Create a data frame with predictions from the original models
predictions <- data.frame(
  Actual = test_data$PTS,
  RF = round(rf_predictions, 2),
  GBM = round(gbm_predictions, 2),
  Bagging = round(bagging_predictions, 2),
  RF_CV = round(rf_predictions_cv, 2),
  GBM_CV = round(gbm_predictions_cv, 2),
  Bagging_CV = round(bagging_predictions_cv, 2)
)

# Print the predictions for comparison
print(predictions)
```

```
##    Actual     RF    GBM Bagging  RF_CV GBM_CV Bagging_CV
## 1     114  98.02  99.18   96.72  98.21 106.82      98.11
## 2     113 108.10 110.77  108.79 107.97 114.25     108.07
## 3     107 104.42 104.32  105.38 104.02 102.36     104.96
## 4      91  84.63  84.61   85.39  84.68  81.85      83.39
## 5      88  89.29  87.35   87.78  89.89  85.78      89.32
## 6      78  78.53  74.52   76.79  78.69  75.77      80.00
## 7      77  93.51  91.67   95.21  93.61  88.62      94.12
## 8      52  58.49  61.76   59.92  58.89  58.38      58.59
## 9     103  97.10  97.86   95.78  96.77  98.34      96.37
## 10     98 100.45 103.24  101.91 101.43 101.06     102.17
## 11     86  88.20  88.44   87.71  88.69  89.45      88.42
## 12     81  79.73  80.69   79.39  80.78  75.24      78.96
## 13    122 109.06 108.43  109.70 108.63 109.05     109.18
## 14    116 109.85 111.18  108.19 110.06 115.45     108.64
## 15    113 105.49 104.25  105.75 105.10 106.99     105.74
## 16     94  97.03  95.06   96.63  95.95  98.64      96.46
## 17     76  82.35  82.09   82.02  81.73  85.25      83.26
## 18     74  64.53  65.28   63.61  64.85  59.72      64.55
## 19     68  64.82  64.42   64.74  64.86  58.05      65.62
## 20     57  64.32  67.78   64.39  63.93  62.39      64.49
## 21    128 110.10 112.78  108.75 109.77 116.38     109.45
## 22    104 101.09 100.82  102.75 101.40 101.59     103.11
## 23    103  98.45  96.94   98.09  97.67  96.50      96.53
## 24    100 103.24 105.50  103.50 103.03 102.20     103.00
## 25     82  75.40  72.48   75.06  75.25  68.66      76.15
## 26     80  81.39  78.23   80.14  81.01  80.98      80.14
## 27     72  72.07  69.96   73.45  72.27  67.90      73.06
## 28     98  91.50  89.31   91.75  90.24  85.56      91.99
## 29     94  89.87  91.55   88.55  90.05  89.88      86.69
## 30     83  91.16  85.83   91.75  91.88  85.77      93.05
## 31     80  88.00  89.32   88.10  87.11  86.09      87.26
```

```r
# Save the rounded predictions to a CSV file
write.csv(predictions, "predictions.csv", row.names = FALSE)
```

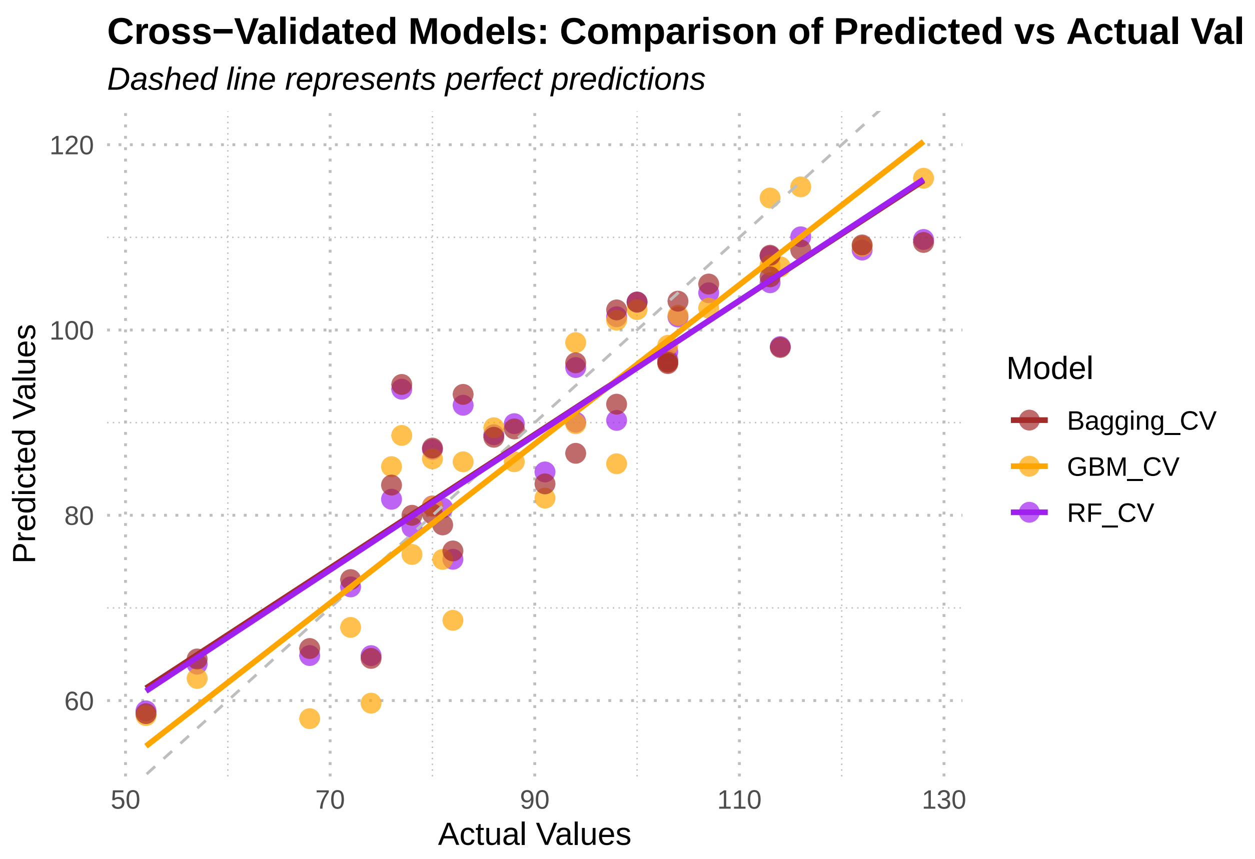
```r
# Separate the predictions into two datasets: Tuned Models and Cross-Validated Models
predictions_long <- predictions %>%
  gather(key = "Model", value = "Prediction", -Actual)

predictions_tuned <- predictions_long %>%
  filter(Model %in% c("RF", "GBM", "Bagging"))

predictions_cv <- predictions_long %>%
  filter(Model %in% c("RF_CV", "GBM_CV", "Bagging_CV"))

# Create the plot for Tuned Models
ggplot(predictions_tuned, aes(x = Actual, y = Prediction, color = Model)) +
  geom_point(alpha = 0.7, size = 3) +
  geom_abline(slope = 1, intercept = 0, linetype = "dashed", color = "gray") +
  geom_smooth(method = "lm", se = FALSE, aes(color = Model), linetype = "solid") +
  labs(
    title = "Tuned Models: Comparison of Predicted vs Actual Values",
    subtitle = "Dashed line represents perfect predictions",
    x = "Actual Values",
    y = "Predicted Values"
  ) +
  scale_color_manual(values = c("RF" = "blue", "GBM" = "green", "Bagging" = "red")) +
  theme_minimal() +
  theme(
    plot.title = element_text(size = 14, face = "bold"),
    plot.subtitle = element_text(size = 12, face = "italic"),
    axis.text = element_text(size = 10),
    axis.title = element_text(size = 12),
    legend.title = element_text(size = 12),
    legend.text = element_text(size = 10),
    panel.grid.major = element_line(color = "gray", size = 0.5, linetype = "dotted"),
    panel.grid.minor = element_line(color = "gray", size = 0.25, linetype = "dotted")
  )
```

```
## Warning: The `size` argument of `element_line()` is deprecated as of ggplot2 3.4.0.
## i Please use the `linewidth` argument instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.

## `geom_smooth()` using formula = 'y ~ x'
```



```r
# Create the plot for Cross-Validated Models
ggplot(predictions_cv, aes(x = Actual, y = Prediction, color = Model)) +
  geom_point(alpha = 0.7, size = 3) +
  geom_abline(slope = 1, intercept = 0, linetype = "dashed", color = "gray") +
  geom_smooth(method = "lm", se = FALSE, aes(color = Model), linetype = "solid") +
  labs(
    title = "Cross-Validated Models: Comparison of Predicted vs Actual Values",
    subtitle = "Dashed line represents perfect predictions",
    x = "Actual Values",
    y = "Predicted Values"
  ) +
  scale_color_manual(values = c("RF_CV" = "purple", "GBM_CV" = "orange", "Bagging_CV" = "brown")) +
  theme_minimal() +
  theme(
    plot.title = element_text(size = 14, face = "bold"),
    plot.subtitle = element_text(size = 12, face = "italic"),
    axis.text = element_text(size = 10),
    axis.title = element_text(size = 12),
    legend.title = element_text(size = 12),
    legend.text = element_text(size = 10),
    panel.grid.major = element_line(color = "gray", size = 0.5, linetype = "dotted"),
    panel.grid.minor = element_line(color = "gray", size = 0.25, linetype = "dotted")
  )
```

```
## `geom_smooth()` using formula = 'y ~ x'
```


Cross−Validated Models: Comparison of Predicted vs Actual Val
Dashed line represents perfect predictions
Predicted Values
Actual Values
120
100
80
60
50
70
90
110
130
Model
Bagging_CV
GBM_CV
RF_CV