Comparative Analysis of Model Performance for Coffee Discrimination Using Hyperspectral Imaging and Machine Learning: Impact of Preprocessing and Variable Selection

AUTHOR
Derick Malavi

Introduction

This document evaluates the performance of various classification models in detecting Arabica coffee adulteration with Robusta. The analysis focuses on key factors affecting model performance, including:

- The type of classification model used.
- Variable or wavelength selection methods, such as Boruta and Genetic Algorithms (GA).
- The impact of spectral preprocessing techniques.
- Comparison of models trained on balanced versus imbalanced datasets.

The models assessed in this study include Linear Discriminant Analysis (LDA), k-Nearest Neighbors (KNN), Support Vector Machines (SVM), Random Forest (RF), Neural Networks, and ensemble methods. The ensemble models were constructed using a stacking approach, where all the individual models served as base learners, and Random Forest acted as the meta-learner. Model performance was evaluated using the Matthews Correlation Coefficient (MCC), a robust metric particularly suitable for imbalanced data sets.

Load Packages

```
suppressWarnings(suppressMessages({
  library(readxl)
  library(readr)
  library(tidyverse)
  library(car)
  library(PMCMRplus)
  library(ggplot2)
  library(tidyr)
  library(dplyr)
  library(skimr)
  library(janitor)
  library(lattice)
  library(dunn.test)
  library(ARTool)
  library(emmeans)
  library(ggpubr)
}))
```

Part 1: Assess Model Performance (External Validation Set): Based on type of model, variable selection & spectral pre-processing

Load and Visualize Data

```
df <- read_excel('summary_model_results_classification_ground _coffee.xlsx',sheet = 'Model')

head(df)
```

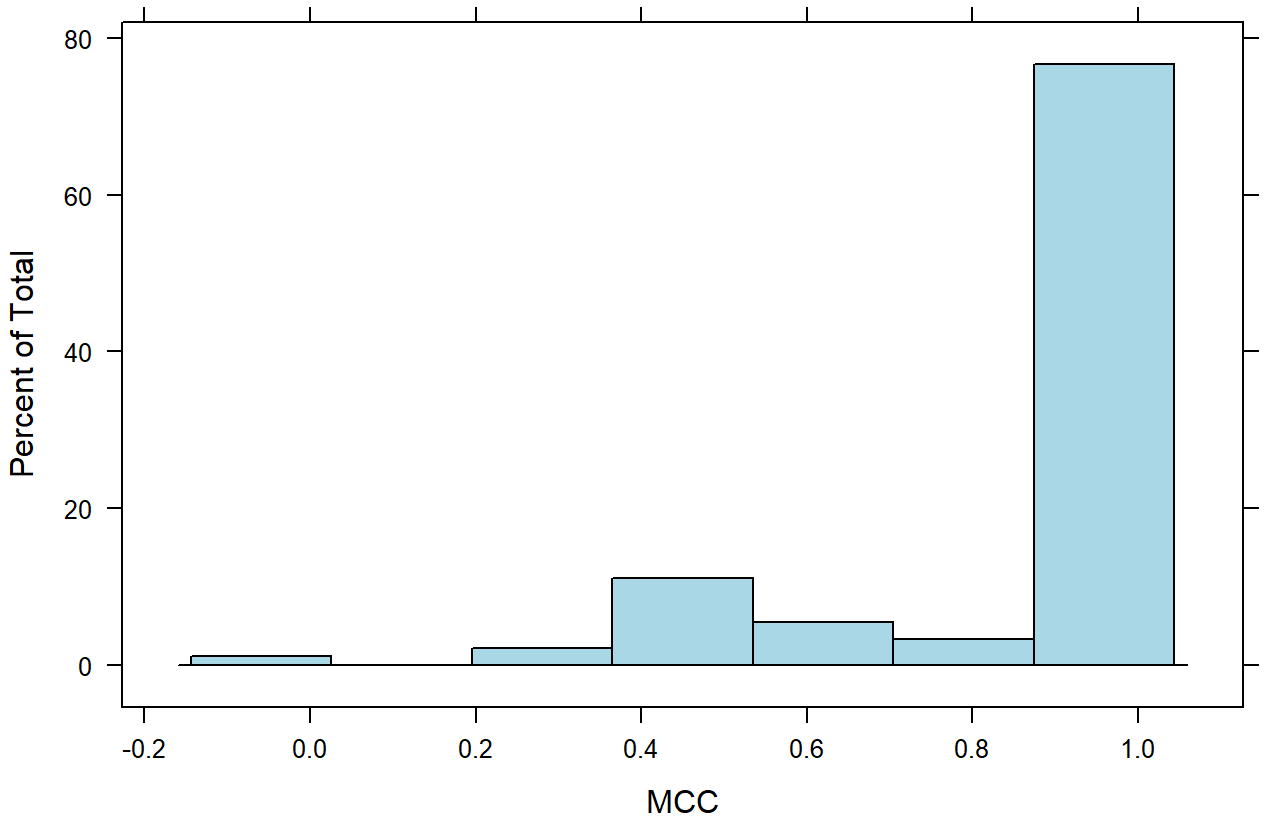
```
# A tibble: 6 × 11
  Model Pre_processing Wav_Selec_Reduction Unbalanced_Data_MCC Balanced_Data_MCC
  <chr> <chr>          <chr>                             <dbl>             <dbl>
1 k-NN  Unprocessed    Full _Spectra_PCA                   0.5               0.1
2 k-NN  SNV+SG+1D      Full _Spectra_PCA                   0.86              0.87
3 k-NN  SNV+SG+2D      Full _Spectra_PCA                   0.94              0.94
4 k-NN  MSC+SG+1D      Full _Spectra_PCA                   0.9               0.89
5 k-NN  MSC+SG+2D      Full _Spectra_PCA                   0.94              0.9
6 LDA   Unprocessed    Full _Spectra_PCA                   0.6               0.53
# ℹ 6 more variables: Imb_Balanced_Accuracy <dbl>, Bal_Balanced_Accuracy <dbl>,
#   Imbal_Specificty <dbl>, Bal_Specificty <dbl>, Imbal_Sensitivity <dbl>,
#   Bal_Sensitivity <dbl>
```

```
# Change the independent variables to factors

df$Model <- as.factor(df$Model)
df$Pre_processing <- as.factor(df$Pre_processing)
df$Wav_Selec_Reduction <- as.factor(df$Wav_Selec_Reduction)
df$Imb_Balanced_Accuracy <- as.numeric(df$Imb_Balanced_Accuracy)
df$Bal_Balanced_Accuracy <- as.numeric(df$Bal_Balanced_Accuracy)

# column names
colnames(df)
```

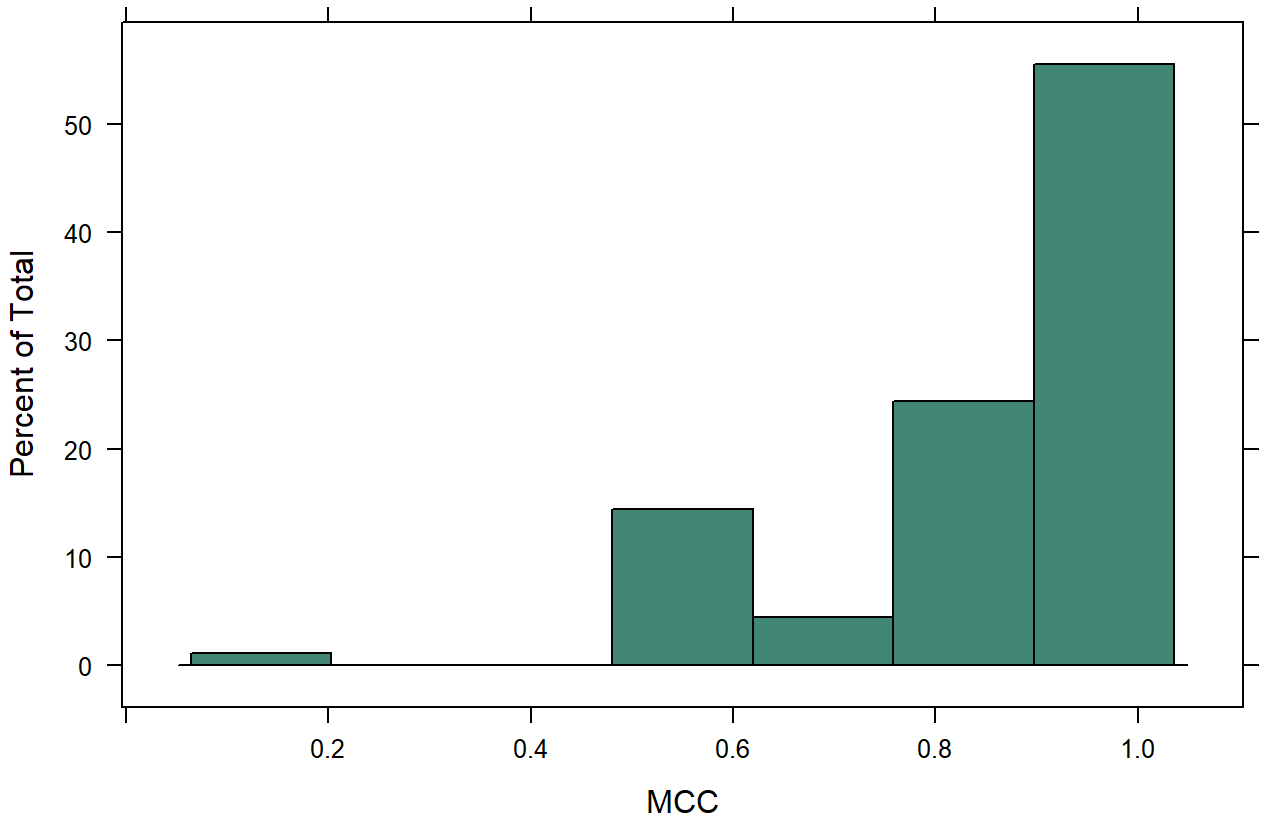
```
 [1] "Model"                "Pre_processing"       "Wav_Selec_Reduction"
 [4] "Unbalanced_Data_MCC"  "Balanced_Data_MCC"    "Imb_Balanced_Accuracy"
 [7] "Bal_Balanced_Accuracy" "Imbal_Specificty"     "Bal_Specificty"
[10] "Imbal_Sensitivity"    "Bal_Sensitivity"
```

Check Distribution Using Histograms

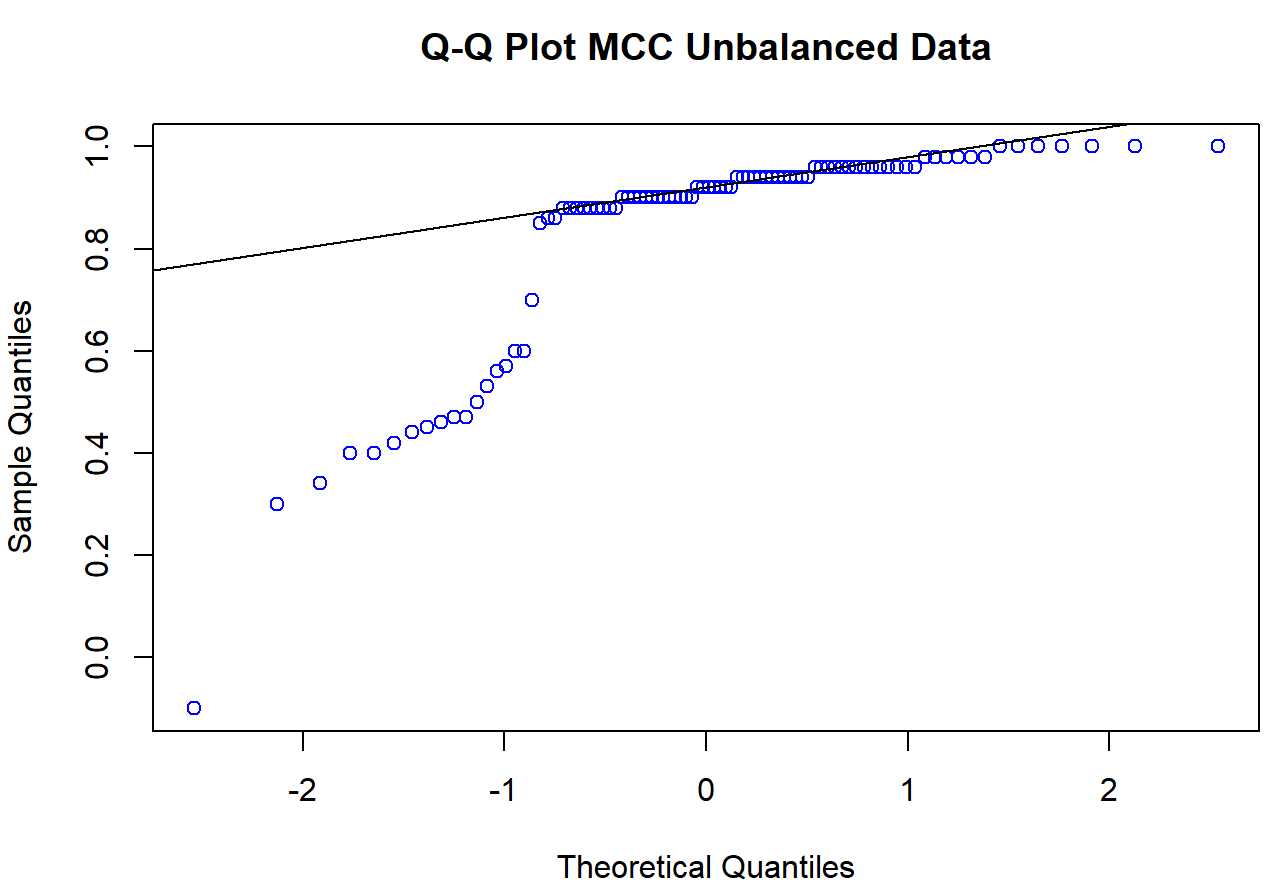
```
# Unbalanced data
histogram(df$Unbalanced_Data_MCC,
          main = 'MCC-Unbalanced Data', xlab = 'MCC',
          cex.main = 1,col = 'lightblue')
```



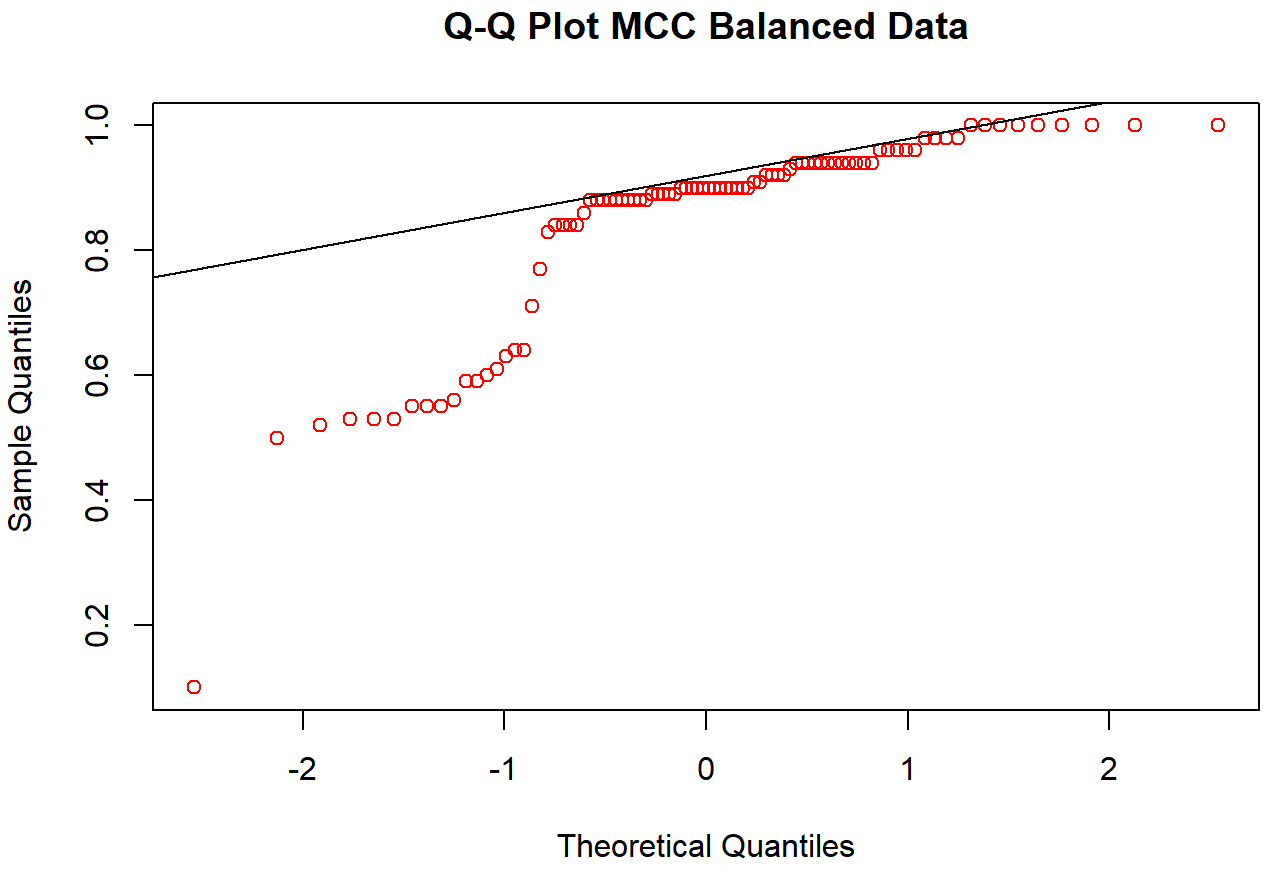

```
histogram(~Balanced_Data_MCC, data = df,
          main = 'MCC-Balanced Data',xlab = 'MCC',
          cex.main = 0.4, col = 'aquamarine4')
```




```
qqnorm(df$Unbalanced_Data_MCC,
       main = 'Q-Q Plot MCC Unbalanced Data', col = "blue") # Unbalanced Data
qqline(df$Unbalanced_Data_MCC)
```




```r
qqnorm(df$Balanced_Data_MCC,
       main = 'Q-Q Plot MCC Balanced Data', col = "red") # Unbalanced Data
qqline(df$Unbalanced_Data_MCC)
```



- Based on the histograms and quantile plots, the data is not normally distributed. This can be verified further by the **Shapiro- Wilke Test**.

Test for Normality

```
shapiro.test(df$Unbalanced_Data_MCC) # for imbalanced data
```

```
    Shapiro-Wilk normality test

data:  df$Unbalanced_Data_MCC
W = 0.68233, p-value = 1.176e-12
```

```
shapiro.test(df$Balanced_Data_MCC) # Balanced Data
```

```
    Shapiro-Wilk normality test

data:  df$Balanced_Data_MCC
W = 0.75413, p-value = 5.711e-11
```

```
# Run ANOVA and check the normality of residuals # Unbalanced Data
aov_model_1 <- aov(Unbalanced_Data_MCC~Model, data = df)

# Check assumptions
shapiro.test(residuals(aov_model_1))  # Normality
```

```
    Shapiro-Wilk normality test

data:  residuals(aov_model_1)
W = 0.68617, p-value = 1.423e-12
```

```
leveneTest(Unbalanced_Data_MCC~Model, data = df)  # Variance homogeneity
```

```
Levene's Test for Homogeneity of Variance (center = median)
      Df F value Pr(>F)
group  5  0.4898  0.783
      84
```

```
## Balanced (SMOTE)
aov_model_2 <- aov(Balanced_Data_MCC~Model, data = df)
shapiro.test(residuals(aov_model_2))  # Normality
```

```
    Shapiro-Wilk normality test

data:  residuals(aov_model_2)
W = 0.74333, p-value = 3.043e-11
```

```
leveneTest(Balanced_Data_MCC~Model, data = df)
```

```
Levene's Test for Homogeneity of Variance (center = median)
      Df F value Pr(>F)
group  5  0.1554 0.9778
      84
```

```
colnames(df)
```

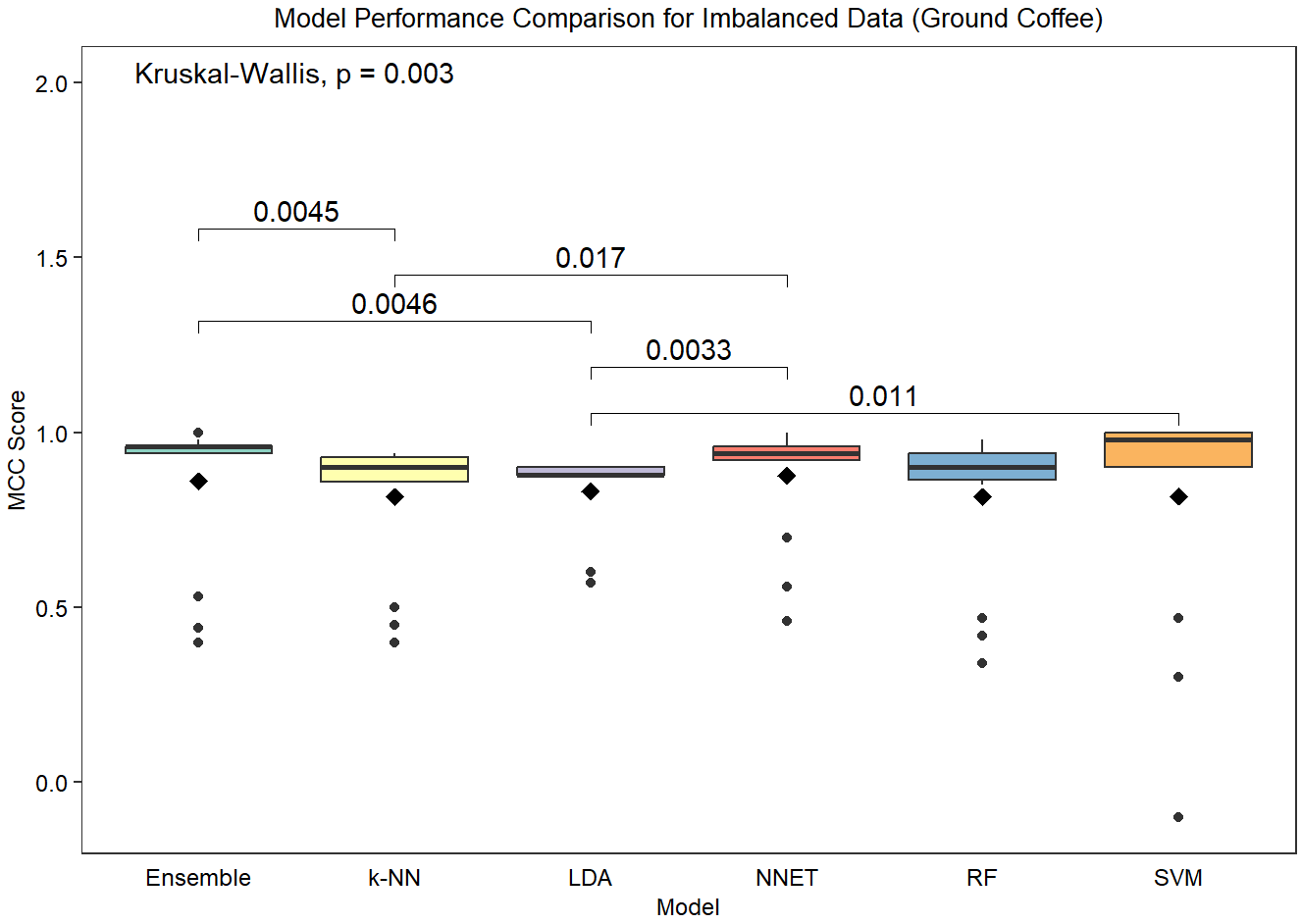
```
[1] "Model"                 "Pre_processing"        "Wav_Selec_Reduction"
[4] "Unbalanced_Data_MCC"   "Balanced_Data_MCC"     "Imb_Balanced_Accuracy"
[7] "Bal_Balanced_Accuracy"
```

- The results of the Shapiro-Wilk test (for the residuals) are statistically significant ($p < 0.05$), providing strong evidence against the assumption of normality. This indicates that the data are not normally distribute. We will therefore consider non-parametric alternatives!

Check Performance based on Type of Model

- This section investigates the models k-NN, LDA, SVM, NNET, Random Forest and stacked models

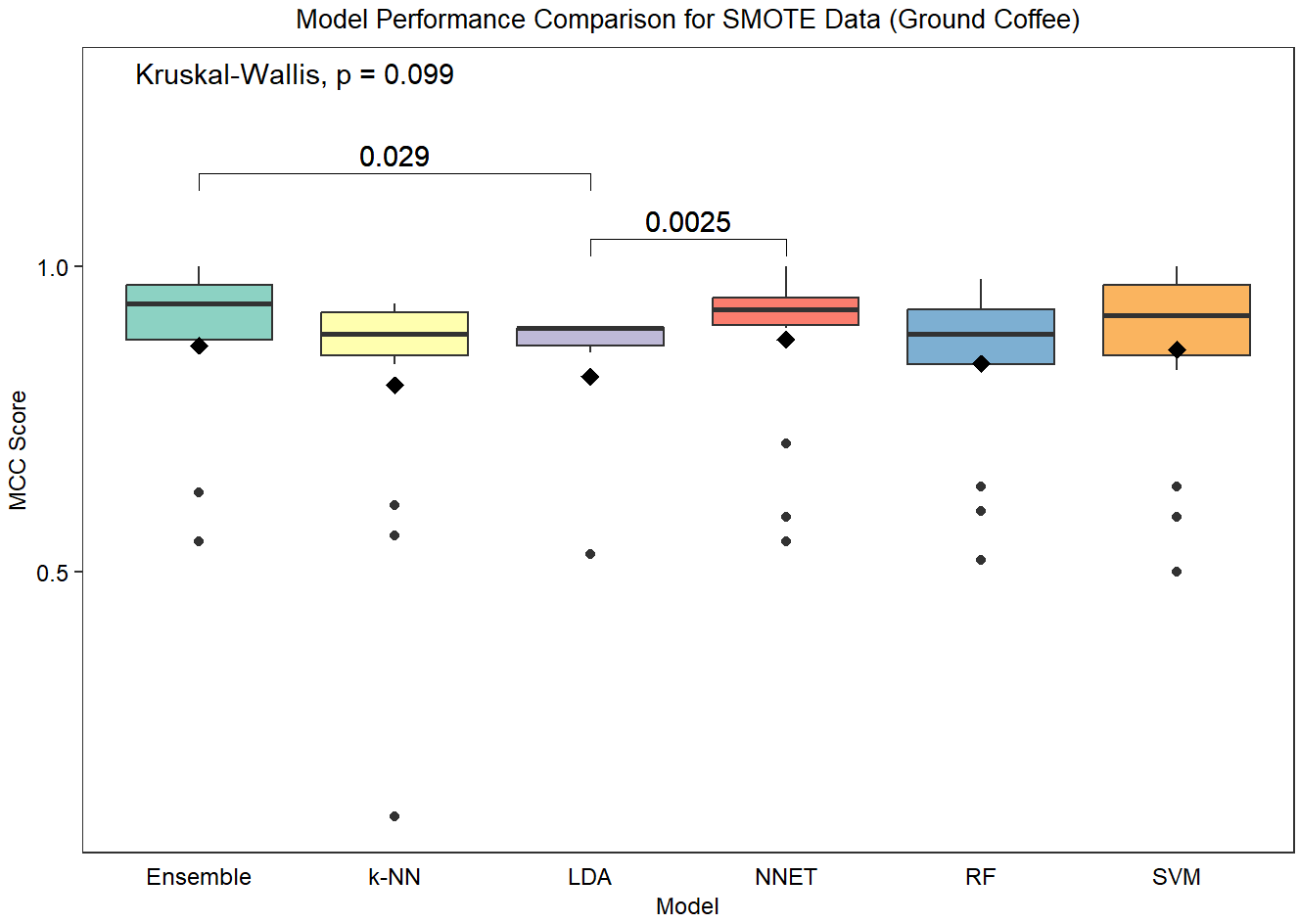
```
# Visualize with box plots
my_comparisons <- list(c('SVM','LDA'),c('NNET','LDA'),c('Ensemble','LDA'),
                       c('NNET','k-NN'),c('Ensemble','k-NN'))


# Imbalanced Data

ggplot(data = df, aes(x = Model, y = Unbalanced_Data_MCC))+
  geom_boxplot(aes(fill = Model))+
  labs(title = 'Model Performance Comparison for Imbalanced Data (Ground Coffee)')+
  ylab('MCC Score')+
  theme_bw()+
  theme(
    axis.title.x = element_text(color = "black", size = 9),
    axis.text.x = element_text(color = "black", size = 9),
    axis.ticks.x = element_blank(),
    axis.text.y = element_text(color = "black", size = 9),
    axis.title.y = element_text(color = "black", size = 9),
    panel.grid = element_blank(),
    plot.title = element_text(hjust = 0.5, color = 'black', size = 10),
    legend.position = 'none'
  )+
  scale_fill_brewer(palette = "Set3", name = 'Model')+
  stat_summary(fun=mean, geom="point", shape=18, size=3, color="black")+
  stat_compare_means(comparisons = my_comparisons)+
  stat_compare_means(label.y = max(df$Unbalanced_Data_MCC) * 2)
```




```
ggsave("Model_No_SMOTE.png", width = 6, height = 4, dpi = 600, bg = "white")
```

```
# Balanced Data

# Visualize with box plots
my_comparisons <- list(c('NNET','LDA'),
                       c('Ensemble', 'LDA')
                       )



# Imbalanced Data

ggplot(data = df, aes(x = Model, y = Balanced_Data_MCC))+
  geom_boxplot(aes(fill = Model))+
  labs(title = 'Model Performance Comparison for SMOTE Data (Ground Coffee)')+
  ylab('MCC Score')+
  theme_bw()+
  theme(
    axis.title.x = element_text(color = "black", size = 9),
    axis.text.x = element_text(color = "black", size = 9),
    axis.ticks.x = element_blank(),
    axis.text.y = element_text(color = "black", size = 9),
```

```
    axis.title.y = element_text(color = "black", size = 9),
    plot.title = element_text(hjust = 0.5, color = 'black', size = 10),
    panel.grid = element_blank(),
    legend.position = 'none'
  )+
  scale_fill_brewer(palette = "Set3", name = 'Model')+
  stat_summary(fun=mean, geom="point", shape=18, size=3, color="black")+
  stat_compare_means(comparisons = my_comparisons)+
  stat_compare_means(comparisons = my_comparisons)+
  stat_compare_means(label.y = max(df$Balanced_Data_MCC) * 1.3)
```




```
ggsave("Model_SMOTE.png", width = 6, height = 4, dpi = 600, bg = "white")
```

```
spec <- df %>% dplyr::select(Model,Bal_Specificty) %>% group_by(Model) %>%
  summarise(Mean_Specificity = mean(Bal_Specificty)) %>% arrange(Mean_Specificity)
spec
```

```
# A tibble: 6 × 2
  Model    Mean_Specificity
  <fct>               <dbl>
1 RF                   88.1
2 k-NN                 91.1
```

```
3 Ensemble             91.9
4 SVM                  91.9
5 NNET                 95.6
6 LDA                  98.5
```

Perform Kruskall-Wallis Test for Model Types

Imbalanced Data Set - Kruskall-Wallis Test

```
kruskal.test(Unbalanced_Data_MCC~Model, data= df)
```

```
    Kruskal-Wallis rank sum test

data:  Unbalanced_Data_MCC by Model
Kruskal-Wallis chi-squared = 17.954, df = 5, p-value = 0.003005
```

Balanced Data Set - Kruskall-Wallis Test

```
kruskal.test(Balanced_Data_MCC~Model, data= df)
```

```
    Kruskal-Wallis rank sum test

data:  Balanced_Data_MCC by Model
Kruskal-Wallis chi-squared = 9.2537, df = 5, p-value = 0.09936
```

- There is statistical difference in performance for models trained with unbalanced data. We will examine the differences using the **Dunn Test** and **Bonferroni correction**, and later by GLM.

Checking for differences in models trained on Unbalanced Data

```
dunn.test(df$Unbalanced_Data_MCC, df$Model, method = 'bonferroni')
```

```
  Kruskal-Wallis rank sum test

data: x and group
Kruskal-Wallis chi-squared = 17.9537, df = 5, p-value = 0

                           Comparison of x by group
                                 (Bonferroni)
Col Mean-|
Row Mean |   Ensemble       k-NN        LDA       NNET         RF
---------+----------------------------------------------------------
    k-NN |   2.429129
         |     0.1135
         |
```

```
     LDA |   3.044321   0.615192
         |    0.0175*     1.0000
         |
    NNET |   0.404269  -2.024860  -2.640052
         |     1.0000     0.3216     0.0622
         |
      RF |   2.017829  -0.411299  -1.026491   1.613560
         |     0.3271     1.0000     1.0000     0.7997
         |
     SVM |   0.014061  -2.415068  -3.030260  -0.390207  -2.003768
         |     1.0000     0.1180    0.0183*     1.0000     0.3382

alpha = 0.05
Reject Ho if p <= alpha/2
```

- These comparisons indicate that the MCC values for LDA models differ significantly from those of the Ensemble and SVM models.

Check Performance based on Variable Selection

- The models were trained using the full spectra (224 variables) and feature selection methods, including Genetic Algorithm, Recursive Feature Elimination, and Boruta. Due to high collinearity among variables, further dimensionality reduction was performed using Principal Component Analysis (PCA).

```r
dunn.test(df$Balanced_Data_MCC, df$Model, method = 'bonferroni')
```

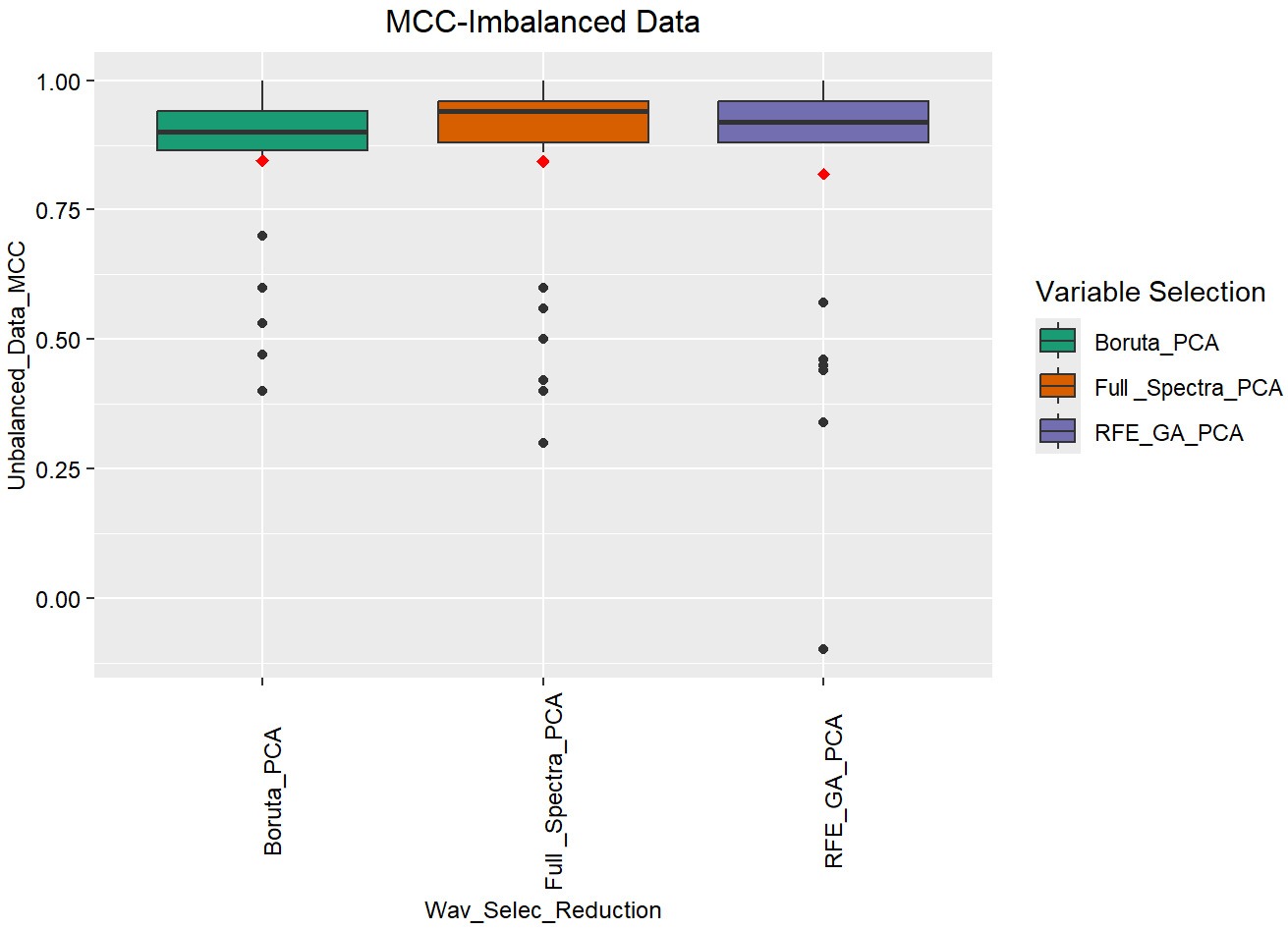
```
  Kruskal-Wallis rank sum test

data: x and group
Kruskal-Wallis chi-squared = 9.2537, df = 5, p-value = 0.1

                           Comparison of x by group
                                 (Bonferroni)
Col Mean-|
Row Mean |   Ensemble       k-NN        LDA       NNET         RF
---------+----------------------------------------------------------
    k-NN |   1.648984
         |     0.7436
         |
     LDA |   2.055967   0.406983
         |     0.2984     1.0000
         |
    NNET |  -0.263135  -1.912120  -2.319103
         |     1.0000     0.4190     0.1529
         |
      RF |   1.413916  -0.235067  -0.642051   1.677052
         |     1.0000     1.0000     1.0000     0.7015
         |
     SVM |   0.343830  -1.305153  -1.712136   0.606966  -1.070085
```

```
        |     1.0000     1.0000     0.6515     1.0000     1.0000

alpha = 0.05
Reject Ho if p <= alpha/2
```

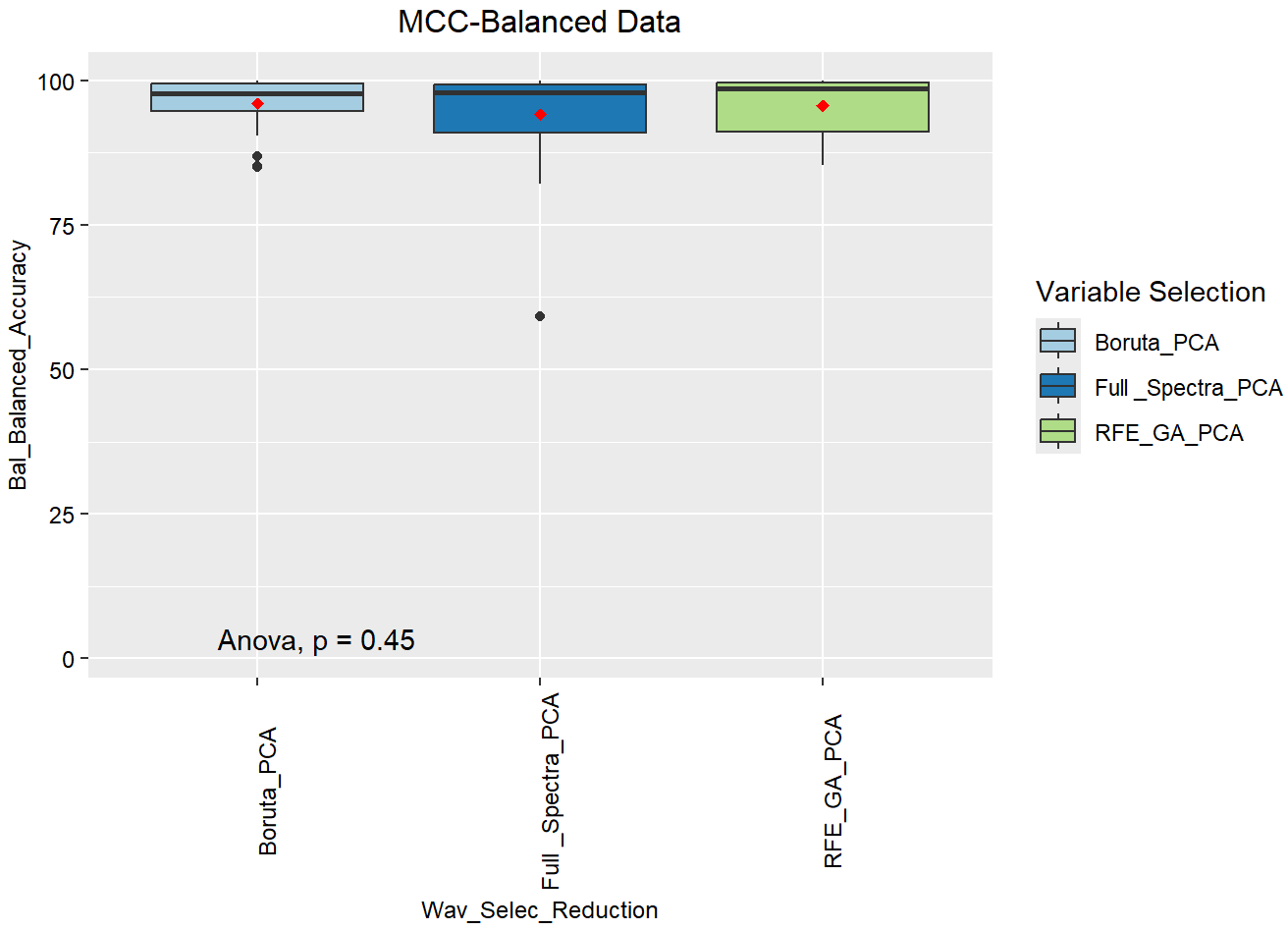
```r
dunn.test(df$Bal_Specificty, df$Model, method = 'bonferroni')
```

```
  Kruskal-Wallis rank sum test

data: x and group
Kruskal-Wallis chi-squared = 16.4697, df = 5, p-value = 0.01

                          Comparison of x by group
                                (Bonferroni)
Col Mean-|
Row Mean |   Ensemble       k-NN        LDA       NNET         RF
---------+------------------------------------------------------
    k-NN |  -0.502557
         |     1.0000
         |
     LDA |  -2.344047  -1.841489
         |     0.1431     0.4916
         |
    NNET |  -1.137174  -0.634616   1.206872
         |     1.0000     1.0000     1.0000
         |
      RF |   1.478327   1.980885   3.822374   2.615502
         |     1.0000     0.3570    0.0010*     0.0668
         |
     SVM |   0.062361   0.564919   2.406408   1.199536  -1.415966
         |     1.0000     1.0000     0.1208     1.0000     1.0000

alpha = 0.05
Reject Ho if p <= alpha/2
```

```r
# Visualize with box plots

# Imbalanced Data

df %>% ggplot(aes(x = Wav_Selec_Reduction, y = Unbalanced_Data_MCC))+
  geom_boxplot(aes(fill = Wav_Selec_Reduction))+
  labs(title = 'MCC-Imbalanced Data')+
  theme(axis.title.x = element_text(color = "black",size= 9),
        axis.text.x = element_text(color = "black", angle = 90, size= 9),
        axis.text.y = element_text(color = "black",size =9),
        axis.title.y = element_text(color = "black",size =9),
        plot.title = element_text(hjust = 0.5, color = 'black', size = 12))+
  scale_fill_brewer(palette = "Dark2", name = 'Variable Selection')+
  stat_summary(fun=mean, geom="point", shape=18, size=2, color="red")
```




```
# Visualize with box plots

# Balanced Data
my_comparisons <- list(c('Boruta_PCA','Full_Spectra_PCA'),
                       c('Full_Spectra_PCA','RFE_GA_PCA'))

df %>% ggplot(aes(x = Wav_Selec_Reduction, y = Bal_Balanced_Accuracy))+
  geom_boxplot(aes(fill = Wav_Selec_Reduction))+
  labs(title = 'MCC-Balanced Data')+
  theme(axis.title.x = element_text(color = "black",size= 9),
        axis.text.x = element_text(color = "black", angle = 90, size= 9),
        axis.text.y = element_text(color = "black",size =9),
        axis.title.y = element_text(color = "black",size =9),
        plot.title = element_text(hjust = 0.5, color = 'black', size = 12))+
  scale_fill_brewer(palette = "Paired", name = 'Variable Selection')+
  stat_summary(fun=mean, geom="point", shape=18, size=2, color="red")+
  stat_compare_means(comparisons = my_comparisons)+
  stat_compare_means(method = 'anova', label.y =   max(df$Balanced_Data_MCC) * 1.5)
```



Perform Kruskall-Wallis Test for Variable Selection

Imbalanced Data Set - Kruskall-Wallis Test

```
kruskal.test(Unbalanced_Data_MCC~Wav_Selec_Reduction, data= df)
```

```
    Kruskal-Wallis rank sum test

data:  Unbalanced_Data_MCC by Wav_Selec_Reduction
Kruskal-Wallis chi-squared = 0.74402, df = 2, p-value = 0.6893
```

```
kruskal.test(Balanced_Data_MCC~Wav_Selec_Reduction, data= df)
```

```
    Kruskal-Wallis rank sum test

data:  Balanced_Data_MCC by Wav_Selec_Reduction
Kruskal-Wallis chi-squared = 0.4891, df = 2, p-value = 0.7831
```

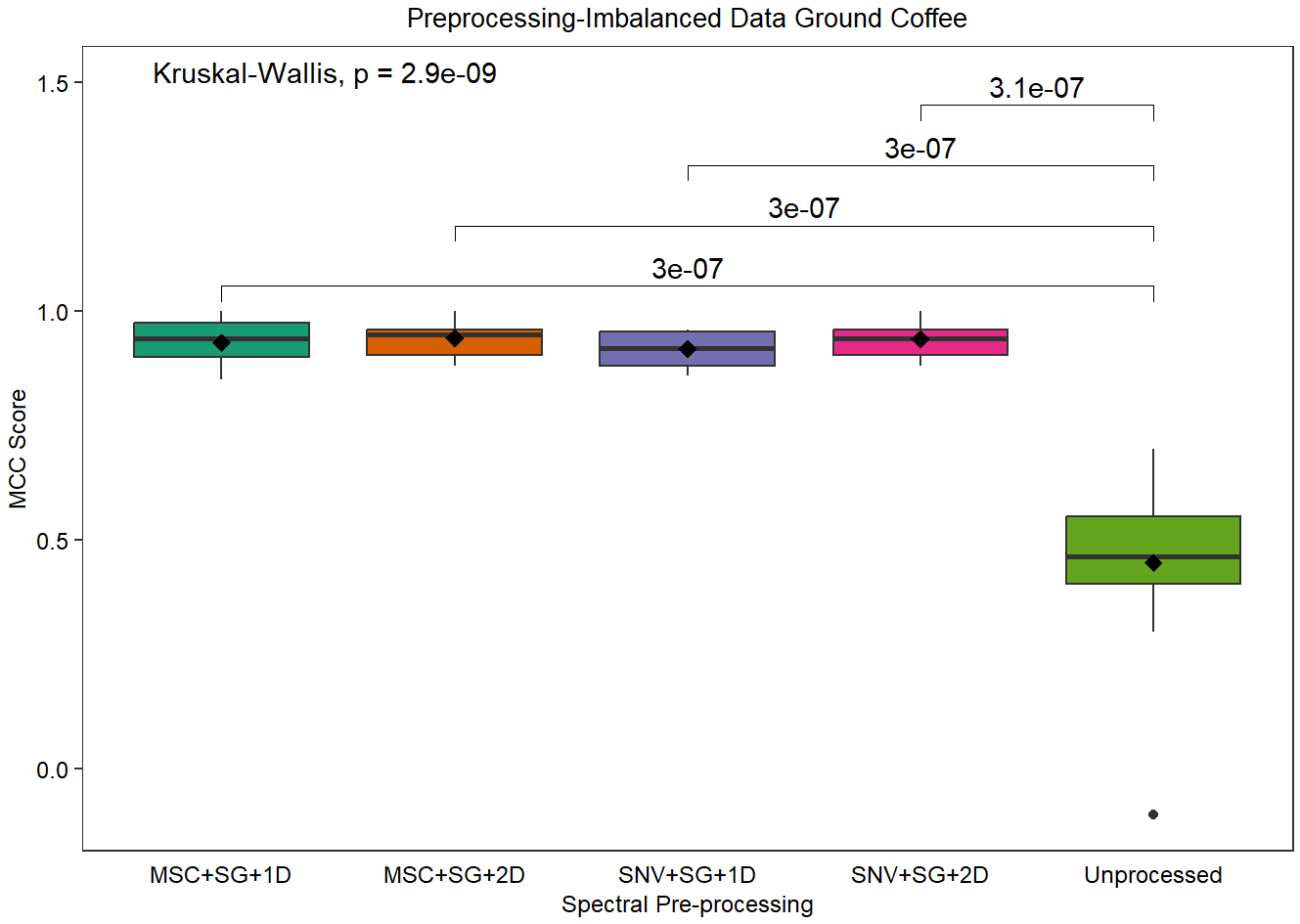
- There is no statistically significant difference in model performance based on variable selection methods ($p > 0.05$). This indicates that these techniques effectively identify variables or regions with useful

information without compromising model performance. This will, however, be investigated further by General Linear Model (GLM).

Check Performance based on spectral preprocessing methods

- The hyperspectral imaging spectra were pre-processed using various techniques, including Standard Normal Variate (SNV) and Multiplicative Scatter Correction (MSC), combined with Savitzky-Golay filtering and derivatives. Unprocessed spectra were used as the baseline for comparison.

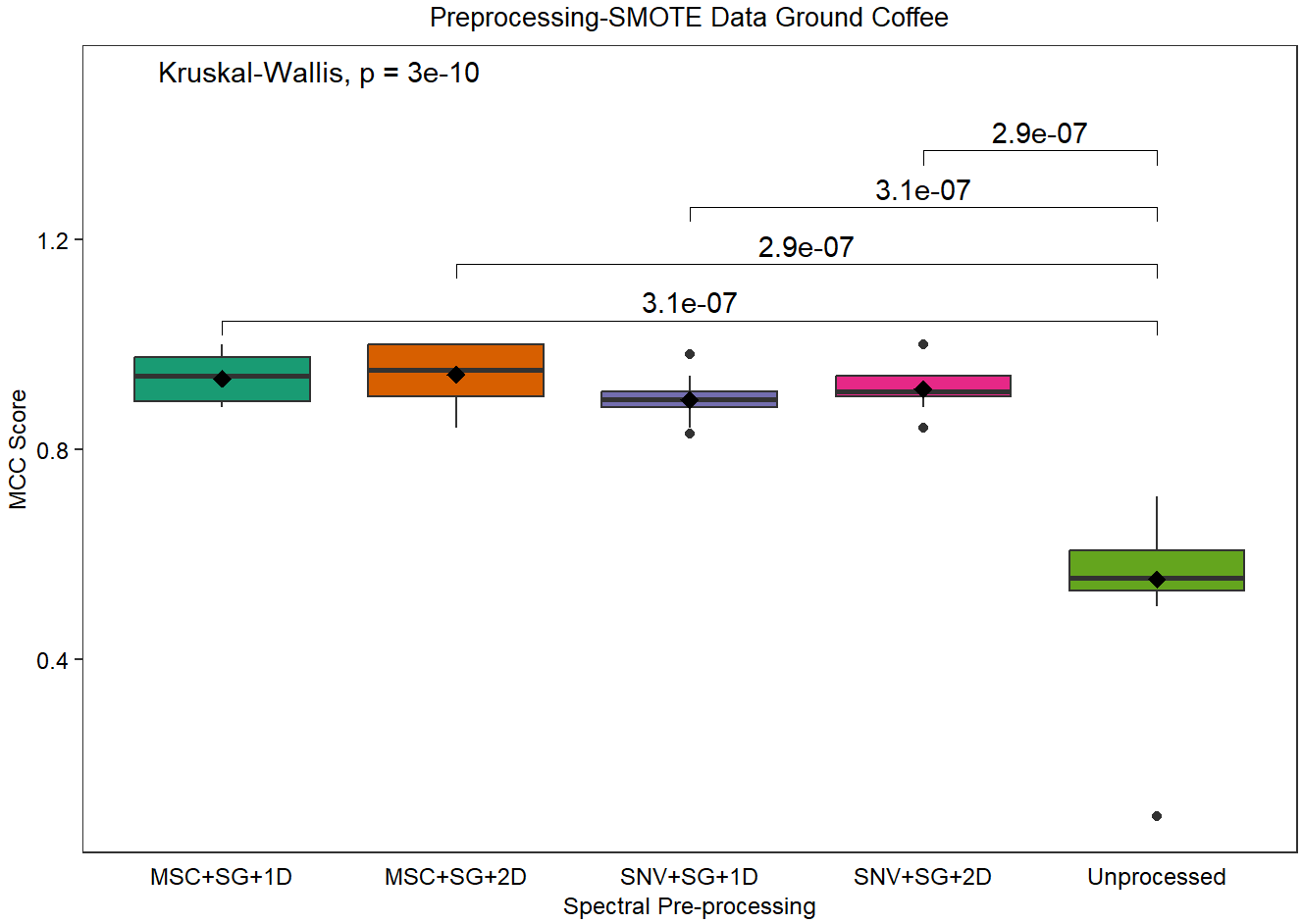
```
# Visualize with box plots

# Imbalanced Data
my_comparisons <- list(
  c('MSC+SG+1D', 'Unprocessed'),
  c('MSC+SG+2D', 'Unprocessed'),
  c('SNV+SG+1D', 'Unprocessed'),
  c('SNV+SG+2D', 'Unprocessed')
)

df %>% ggplot(aes(x = Pre_processing, y = Unbalanced_Data_MCC))+
  geom_boxplot(aes(fill = Pre_processing))+
  labs(title = 'Preprocessing-Imbalanced Data Ground Coffee',x = 'Spectral Pre-processing')+
  ylab('MCC Score')+
  theme_bw()+
  theme(
    axis.title.x = element_text(color = "black", size = 9),
    axis.text.x = element_text(color = "black", size = 9),
    axis.ticks.x = element_blank(),
    axis.text.y = element_text(color = "black", size = 9),
    axis.title.y = element_text(color = "black", size = 9),
    plot.title = element_text(hjust = 0.5, color = 'black', size = 10),
    panel.grid = element_blank(),
    legend.position = 'none')+
  scale_fill_brewer(palette = "Dark2", name = 'Preprocessing')+
  stat_summary(fun=mean, geom="point", shape=18, size=3, color="black")+
  stat_compare_means(comparisons = my_comparisons)+
  stat_compare_means(method = 'kruskal.test', label.y = max(df$Unbalanced_Data_MCC) * 1.5)
```




```
ggsave("MCC_Imbalanced_Data_Preprocessing.png", width = 6, height = 4, dpi = 600, bg = "white")
```

```
# Visualize with box plots
my_comparisons <- list(
  c('MSC+SG+1D', 'Unprocessed'),
  c('MSC+SG+2D', 'Unprocessed'),
  c('SNV+SG+1D', 'Unprocessed'),
  c('SNV+SG+2D', 'Unprocessed')
)

df %>% ggplot(aes(x = Pre_processing, y = Balanced_Data_MCC))+
  geom_boxplot(aes(fill = Pre_processing))+
  labs(title = 'Preprocessing-SMOTE Data Ground Coffee', x = 'Spectral Pre-processing')+
  ylab('MCC Score')+
  theme_bw()+
  theme(
    axis.title.x = element_text(color = "black", size = 9),
    axis.text.x = element_text(color = "black", size = 9),
    axis.ticks.x = element_blank(),
    axis.text.y = element_text(color = "black", size = 9),
    axis.title.y = element_text(color = "black", size = 9),
    plot.title = element_text(hjust = 0.5, color = 'black', size = 10),
    panel.grid = element_blank(),
```

```
  legend.position = 'none')+
scale_fill_brewer(palette = "Dark2", name = 'Preprocessing')+
stat_summary(fun=mean, geom="point", shape=18, size=3, color="black")+
stat_compare_means(comparisons = my_comparisons)+
stat_compare_means(label.y = max(df$Unbalanced_Data_MCC) * 1.5)
```




```
ggsave("MCC_SMOTE_Data_Preprocessing.png", width = 6, height = 4, dpi = 600, bg = "white")
```

Perform Kruskall-Wallis Test for Spectral Preprocessing Techniques

Imbalanced Data Set - Kruskall-Wallis Test

```
kruskal.test(Unbalanced_Data_MCC~Pre_processing, data= df)
```

```
    Kruskal-Wallis rank sum test

data:  Unbalanced_Data_MCC by Pre_processing
Kruskal-Wallis chi-squared = 45.651, df = 4, p-value = 2.911e-09
```

Balanced Data Set - Kruskall-Wallis Test

```
kruskal.test(Balanced_Data_MCC~Pre_processing, data= df)
```

```
    Kruskal-Wallis rank sum test

data:  Balanced_Data_MCC by Pre_processing
Kruskal-Wallis chi-squared = 50.387, df = 4, p-value = 2.997e-10
```

- Spectral preprocessing significantly ($p < 0.05$) influences the performance of models.

Checking for differences in pre-processing techniques for Unbalanced Data

```
dunn.test(df$Unbalanced_Data_MCC, df$Pre_processing, method = 'bonferroni')
```

```
  Kruskal-Wallis rank sum test

data: x and group
Kruskal-Wallis chi-squared = 45.6511, df = 4, p-value = 0

                           Comparison of x by group
                                 (Bonferroni)
Col Mean-|
Row Mean |   MSC+SG+1   MSC+SG+2   SNV+SG+1   SNV+SG+2
---------+--------------------------------------------
MSC+SG+2 |  -0.632190
         |     1.0000
         |
SNV+SG+1 |   0.792645   1.424836
         |     1.0000     0.7710
         |
SNV+SG+2 |  -0.442854   0.189336  -1.235500
         |     1.0000     1.0000     1.0000
         |
Unproces |   5.128127   5.760318   4.335482   5.570982
         |    0.0000*    0.0000*    0.0001*    0.0000*

alpha = 0.05
Reject Ho if p <= alpha/2
```

Checking for differences in pre-processing techniques for Balanced Data

```
dunn.test(df$Balanced_Data_MCC, df$Pre_processing, method = 'bonferroni')
```

```
  Kruskal-Wallis rank sum test

data: x and group
Kruskal-Wallis chi-squared = 50.3874, df = 4, p-value = 0
```

```
                        Comparison of x by group
                              (Bonferroni)
Col Mean-|
Row Mean |   MSC+SG+1   MSC+SG+2   SNV+SG+1   SNV+SG+2
---------+----------------------------------------------
MSC+SG+2 |  -0.602123
         |     1.0000
         |
SNV+SG+1 |   1.956902   2.559026
         |     0.2518     0.0525
         |
SNV+SG+2 |   0.771871   1.373995  -1.185031
         |     1.0000     0.8472     1.0000
         |
Unproces |   5.720177   6.322301   3.763274   4.948305
         |    0.0000*    0.0000*    0.0008*    0.0000*

alpha = 0.05
Reject Ho if p <= alpha/2
```

- Models trained on spectra pre-processed using SNV+SG+1st derivative, SNV+SG+2nd derivative, MSC+SG+1st derivative, and MSC+SG+2nd derivative significantly outperformed those trained on raw, unprocessed spectra.

Does training on either imbalanced or Balanced Data set affect model performance?

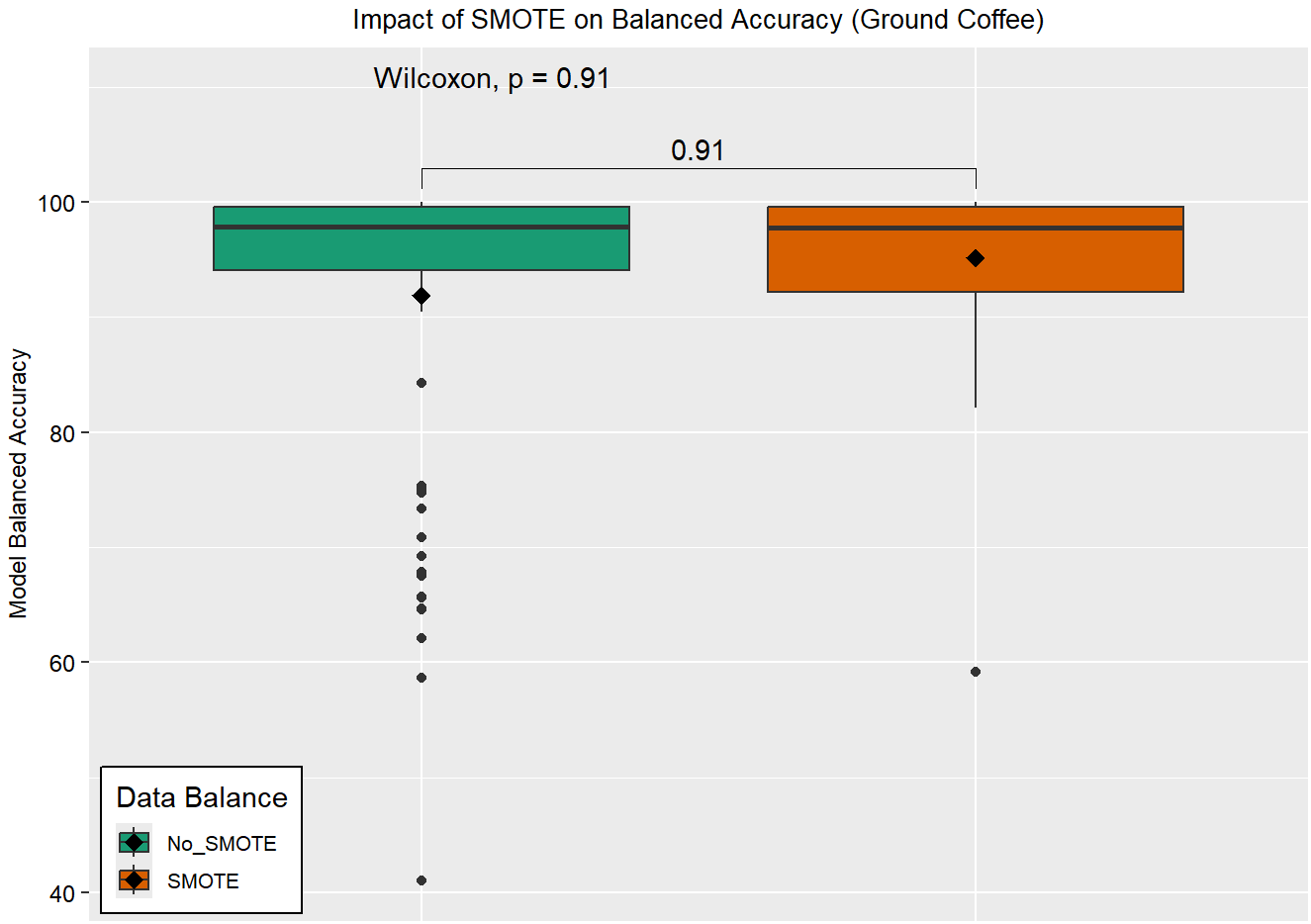
We will use Mann-Whitney U Test to check for differences in performance based on MCC and balanced accuracy.

```r
df_bal <- read_excel('summary_model_results_classification_ground _coffee.xlsx', sheet='data_balar
df_bal$data_balance <- as.factor(df_bal$data_balance)
df_bal$Model <- as.factor(df_bal$Model)
df_bal$Preprocessing<- as.factor(df_bal$Preprocessing)
```

```r
# Define statistical comparisons
my_comparisons <- list(c( "SMOTE","No_SMOTE"))

# Create Boxplot

df_bal %>% ggplot(aes(x = data_balance, y = Balanced_Accuracy, fill = data_balance)) +
  geom_boxplot() +
  labs(title = "Impact of SMOTE on Balanced Accuracy (Ground Coffee)",
       y = "Model Balanced Accuracy")+
 theme(
    axis.title.x = element_blank(),
    axis.text.x = element_blank(),
    axis.ticks.x = element_blank(),
```

```
    axis.text.y = element_text(color = "black", size = 9),
    axis.title.y = element_text(color = "black", size = 9),
    plot.title = element_text(hjust = 0.5, color = 'black', size = 10),
    legend.position = c(0.01, 0.009),
    legend.justification = c(0, 0),
    legend.background = element_rect(fill = "white", color = "black"),
    legend.key.size = unit(0.5, "cm"),
    legend.text = element_text(size = 8)
  )+
  scale_fill_brewer(palette = "Dark2", name = 'Data Balance')+
  stat_summary(fun=mean, geom="point", shape=18, size=3, color="black")+
  stat_compare_means(comparisons = my_comparisons)+
  stat_compare_means(label.y = max(df_bal$Balanced_Accuracy) * 1.1)
```

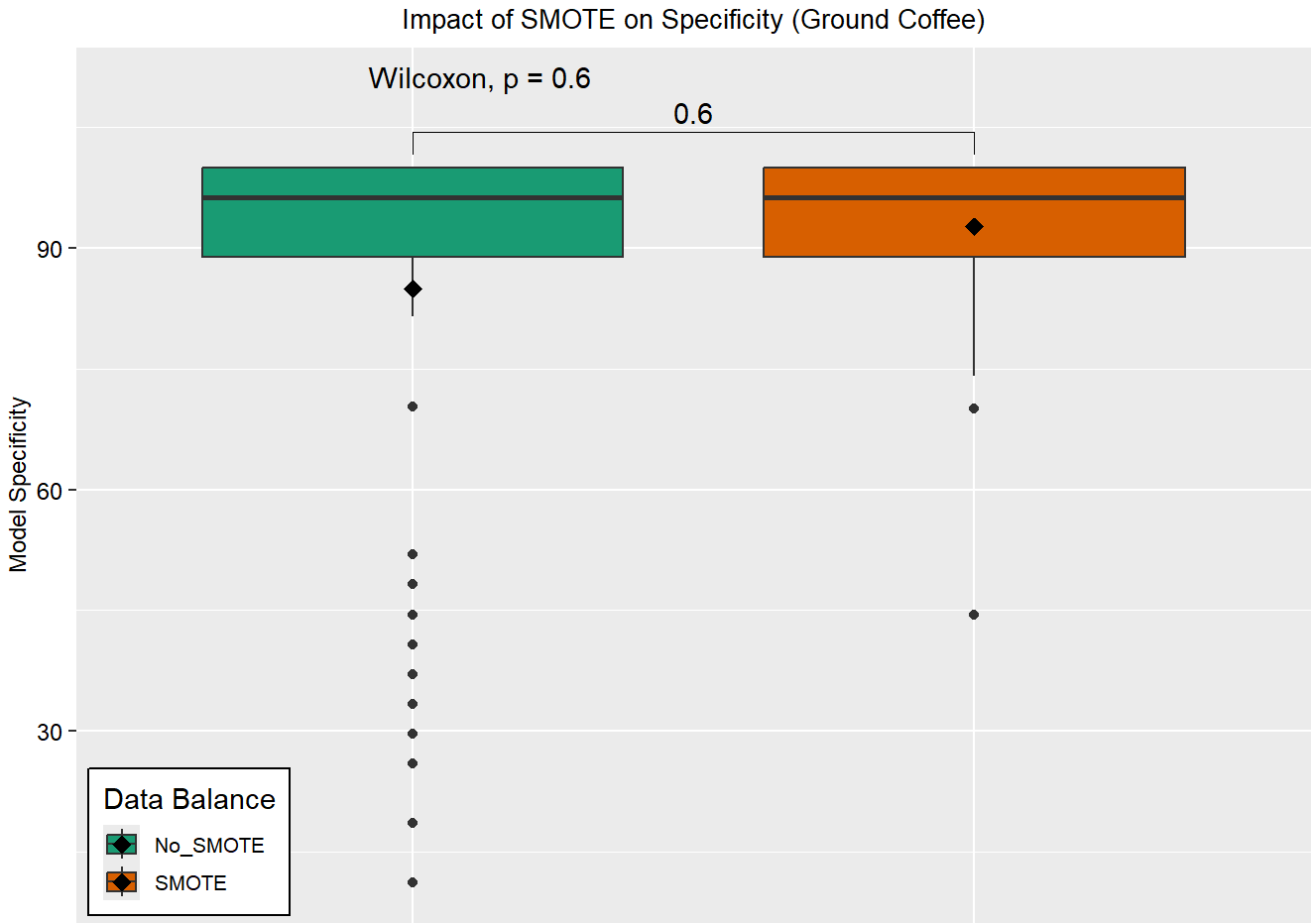
```
Warning: A numeric `legend.position` argument in `theme()` was deprecated in ggplot2
3.5.0.
ℹ Please use the `legend.position.inside` argument of `theme()` instead.
```




```
ggsave("SMOTE_Data_Balance_Ground_Coffee.png", width = 6, height = 4, dpi = 600, bg = "white")
```

```
# Define statistical comparisons
my_comparisons <- list(c("No_SMOTE", "SMOTE"))
```

```r
# Create Boxplot

df_bal %>% ggplot(aes(x = data_balance, y = Specificity, fill = data_balance)) +
  geom_boxplot() +
  labs(title = "Impact of SMOTE on Specificity (Ground Coffee)",
       y = "Model Specificity")+
 theme(
    axis.title.x = element_blank(),
    axis.text.x = element_blank(),
    axis.ticks.x = element_blank(),
    axis.text.y = element_text(color = "black", size = 9),
    axis.title.y = element_text(color = "black", size = 9),
    plot.title = element_text(hjust = 0.5, color = 'black', size = 10),
    legend.position = c(0.01, 0.009),
    legend.justification = c(0, 0),
    legend.background = element_rect(fill = "white", color = "black"),
    legend.key.size = unit(0.5, "cm"),
    legend.text = element_text(size = 8)
  )+
  scale_fill_brewer(palette = "Dark2", name = 'Data Balance')+
  stat_summary(fun=mean, geom="point", shape=18, size=3, color="black")+
  stat_compare_means(comparisons = my_comparisons)+
  stat_compare_means(label.y = max(df_bal$Balanced_Accuracy) * 1.1)
```



```
#ggsave("SMOTE_Data_Balance_Ground_Coffee.png", width = 6, height = 4, dpi = 600, bg = "white")
```

```
# Mean for specifity
spec_smote <- df_bal$Specificity[df_bal$data_balance=='SMOTE']
mean(spec_smote)
```

```
[1] 92.71667
```

```
spec_no_smote <- df_bal$Specificity[df_bal$data_balance=='No_SMOTE']
mean(spec_no_smote)
```

```
[1] 84.98
```

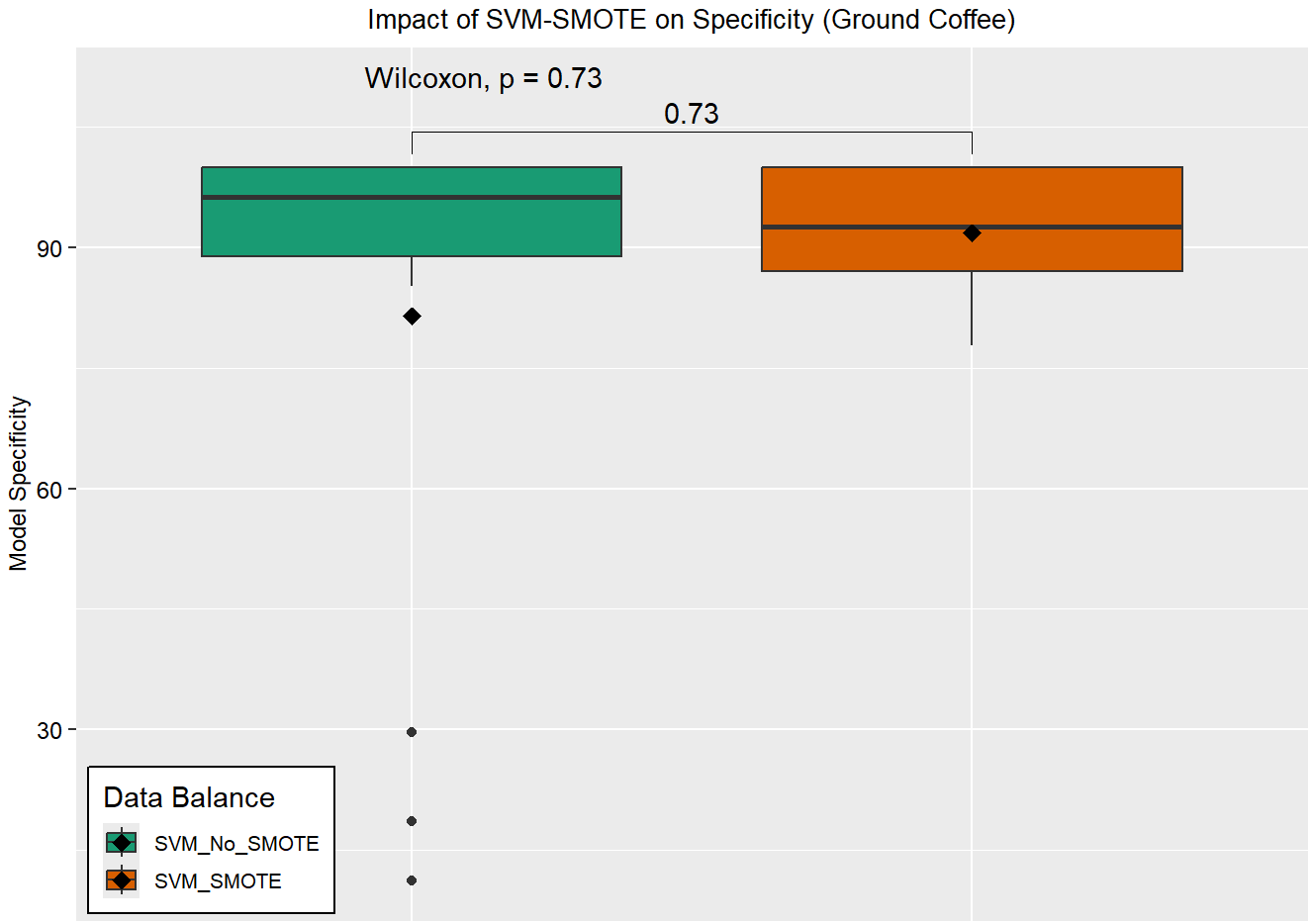
Check model performance with and without SMOTE

```
# SVM Model
svm <- df_bal %>%
  filter(Model %in% c('SVM_No_SMOTE', 'SVM_SMOTE')) %>%
  dplyr::select(Model, MCC, Balanced_Accuracy, Specificity)
head(svm)
```

```
# A tibble: 6 × 4
  Model          MCC Balanced_Accuracy Specificity
  <fct>        <dbl>             <dbl>       <dbl>
1 SVM_No_SMOTE  0.3               58.7        18.5
2 SVM_No_SMOTE  0.94              96.2        92.6
3 SVM_No_SMOTE  1                100         100
4 SVM_No_SMOTE  0.98              98.1        96.3
5 SVM_No_SMOTE  0.98              99.9       100
6 SVM_No_SMOTE -0.1               41          11.1
```

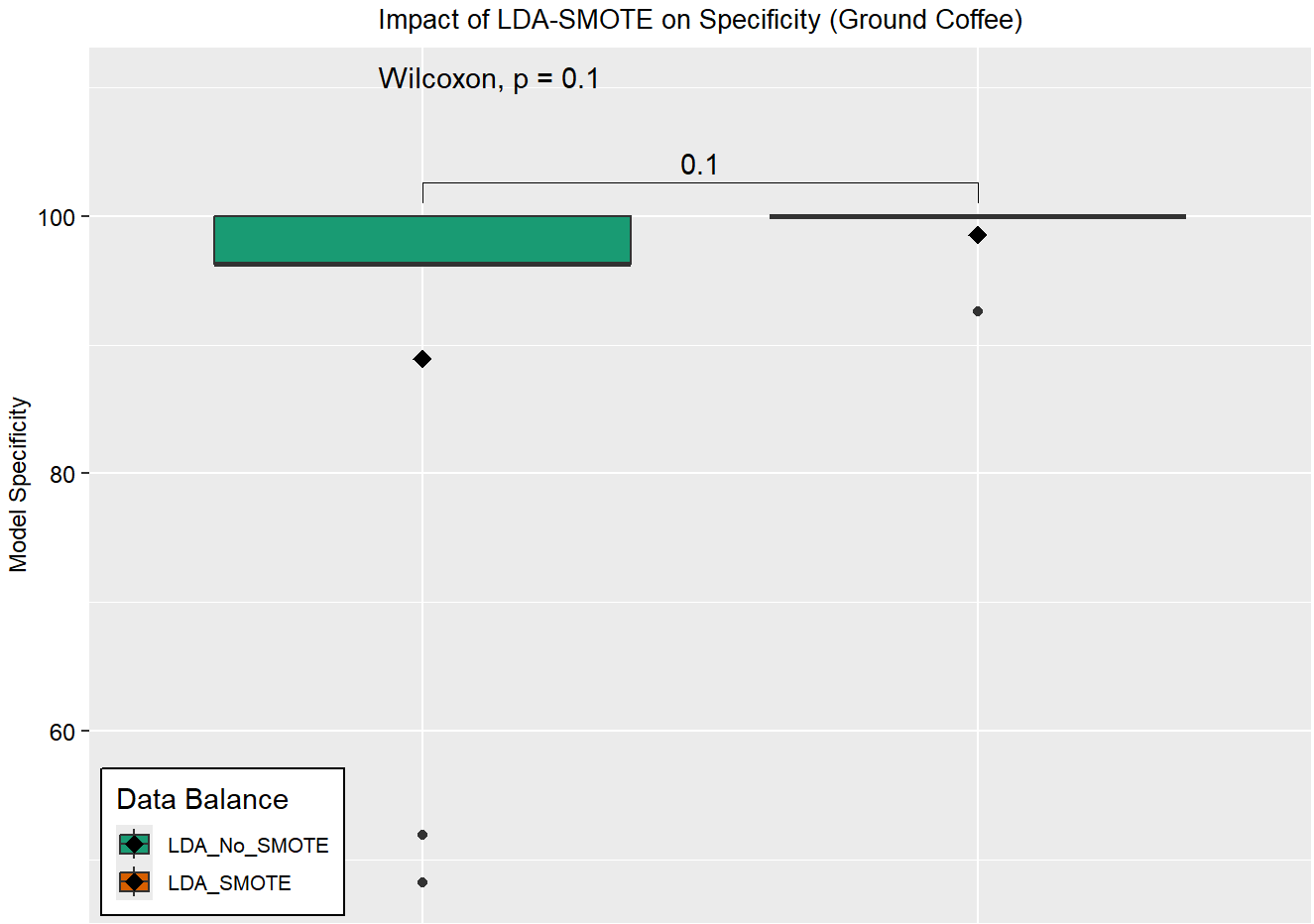
```
# Define statistical comparisons
my_comparisons <- list(c('SVM_No_SMOTE', 'SVM_SMOTE'))

# Create Boxplot

svm %>% ggplot(aes(x = Model, y = Specificity, fill = Model)) +
  geom_boxplot() +
  labs(title = "Impact of SVM-SMOTE on Specificity (Ground Coffee)",
       y = "Model Specificity")+
 theme(
    axis.title.x = element_blank(),
    axis.text.x = element_blank(),
    axis.ticks.x = element_blank(),
    axis.text.y = element_text(color = "black", size = 9),
    axis.title.y = element_text(color = "black", size = 9),
    plot.title = element_text(hjust = 0.5, color = 'black', size = 10),
    legend.position = c(0.01, 0.009),
```

```
    legend.justification = c(0, 0),
    legend.background = element_rect(fill = "white", color = "black"),
    legend.key.size = unit(0.5, "cm"),
    legend.text = element_text(size = 8)
  )+
  scale_fill_brewer(palette = "Dark2", name = 'Data Balance')+
  stat_summary(fun=mean, geom="point", shape=18, size=3, color="black")+
  stat_compare_means(comparisons = my_comparisons)+
  stat_compare_means(label.y = max(df_bal$Specificity) * 1.1)
```



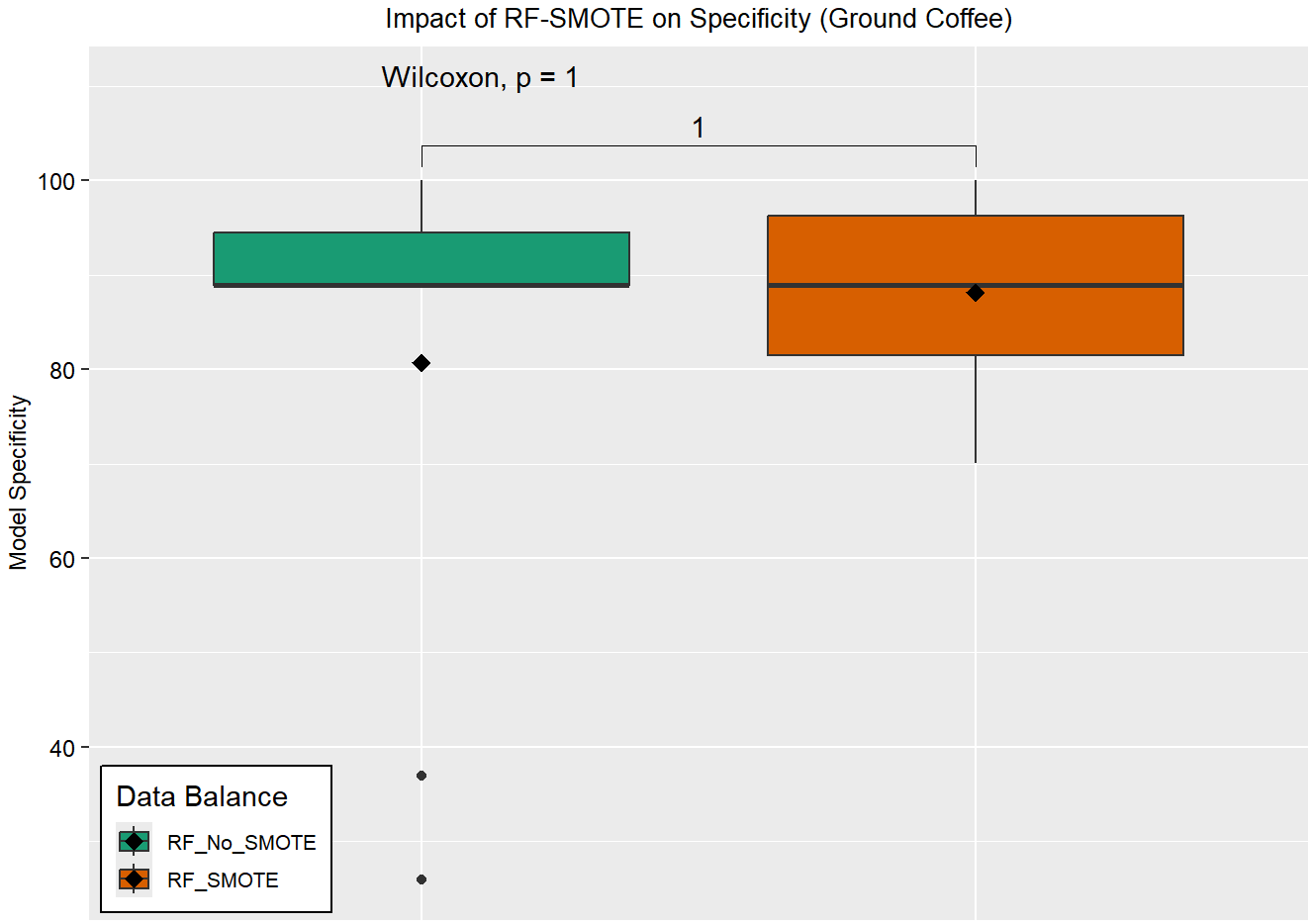

```
# LDA

lda <- df_bal %>%
  filter(Model %in% c('LDA_No_SMOTE', 'LDA_SMOTE')) %>%
  dplyr::select(Model, MCC, Balanced_Accuracy, Specificity)


# Define statistical comparisons
my_comparisons <- list(c('LDA_No_SMOTE', 'LDA_SMOTE'))

# Create Boxplot

lda %>% ggplot(aes(x = Model, y = Specificity, fill = Model)) +
```

```
  geom_boxplot() +
  labs(title = "Impact of LDA-SMOTE on Specificity (Ground Coffee)",
       y = "Model Specificity")+
 theme(
    axis.title.x = element_blank(),
    axis.text.x = element_blank(),
    axis.ticks.x = element_blank(),
    axis.text.y = element_text(color = "black", size = 9),
    axis.title.y = element_text(color = "black", size = 9),
    plot.title = element_text(hjust = 0.5, color = 'black', size = 10),
    legend.position = c(0.01, 0.009),
    legend.justification = c(0, 0),
    legend.background = element_rect(fill = "white", color = "black"),
    legend.key.size = unit(0.5, "cm"),
    legend.text = element_text(size = 8)
  )+
  scale_fill_brewer(palette = "Dark2", name = 'Data Balance')+
  stat_summary(fun=mean, geom="point", shape=18, size=3, color="black")+
  stat_compare_means(comparisons = my_comparisons)+
  stat_compare_means(label.y = max(df_bal$Balanced_Accuracy) * 1.1)
```




```
# NNET
rf <- df_bal %>%
```

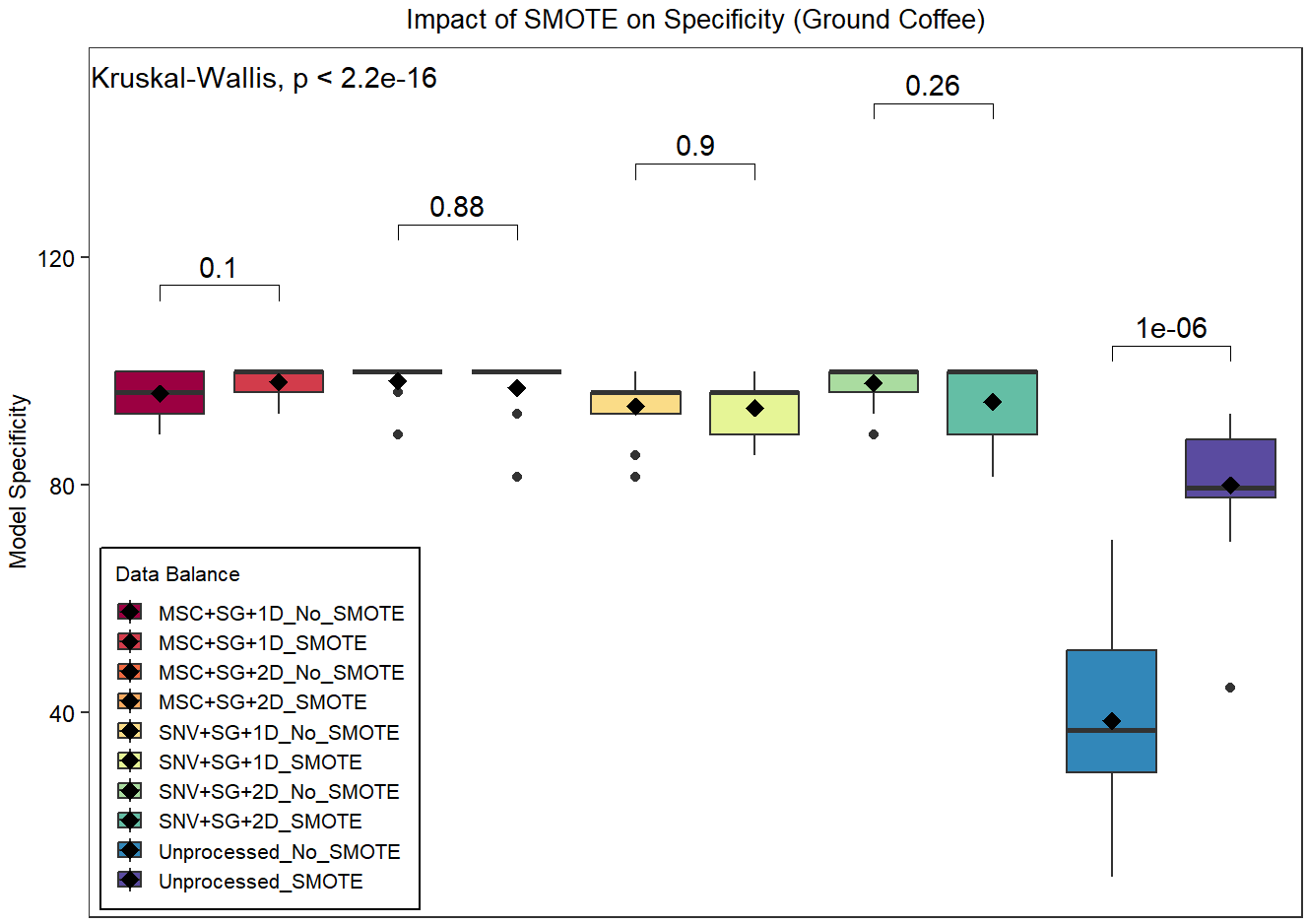
```r
  filter(Model %in% c('RF_No_SMOTE', 'RF_SMOTE')) %>%
  dplyr::select(Model, MCC, Balanced_Accuracy, Specificity)


# Define statistical comparisons
my_comparisons <- list(c('RF_No_SMOTE', 'RF_SMOTE'))

# Create Boxplot

rf %>% ggplot(aes(x = Model, y = Specificity, fill = Model)) +
  geom_boxplot() +
  labs(title = "Impact of RF-SMOTE on Specificity (Ground Coffee)",
       y = "Model Specificity")+
 theme(
    axis.title.x = element_blank(),
    axis.text.x = element_blank(),
    axis.ticks.x = element_blank(),
    axis.text.y = element_text(color = "black", size = 9),
    axis.title.y = element_text(color = "black", size = 9),
    plot.title = element_text(hjust = 0.5, color = 'black', size = 10),
    legend.position = c(0.01, 0.009),
    legend.justification = c(0, 0),
    legend.background = element_rect(fill = "white", color = "black"),
    legend.key.size = unit(0.5, "cm"),
    legend.text = element_text(size = 8)
  )+
  scale_fill_brewer(palette = "Dark2", name = 'Data Balance')+
  stat_summary(fun=mean, geom="point", shape=18, size=3, color="black")+
  stat_compare_means(comparisons = my_comparisons)+
  stat_compare_means(label.y = max(df_bal$Balanced_Accuracy) * 1.1)
```



Check preprocessing and SMOTE

```
# Define statistical comparisons
my_comparisons <- list(c('Unprocessed_No_SMOTE', 'Unprocessed_SMOTE'),
                       c('MSC+SG+1D_No_SMOTE', 'MSC+SG+1D_SMOTE'),
                       c('MSC+SG+2D_No_SMOTE', 'MSC+SG+2D_SMOTE'),
                       c('SNV+SG+1D_No_SMOTE', 'SNV+SG+1D_SMOTE'),
                       c('SNV+SG+2D_No_SMOTE', 'SNV+SG+2D_SMOTE'))
# Create Box plot

df_bal %>% ggplot(aes(x = Preprocessing, y = Specificity, fill = Preprocessing)) +
  geom_boxplot() +
  labs(title = "Impact of SMOTE on Specificity (Ground Coffee)",
       y = "Model Specificity")+
 theme_bw()+
 theme(
    axis.title.x = element_blank(),
    axis.text.x = element_blank(),
    axis.ticks.x = element_blank(),
    axis.text.y = element_text(color = "black", size = 9),
    axis.title.y = element_text(color = "black", size = 9),
    plot.title = element_text(hjust = 0.5, color = 'black', size = 10),
    panel.grid = element_blank(),
```

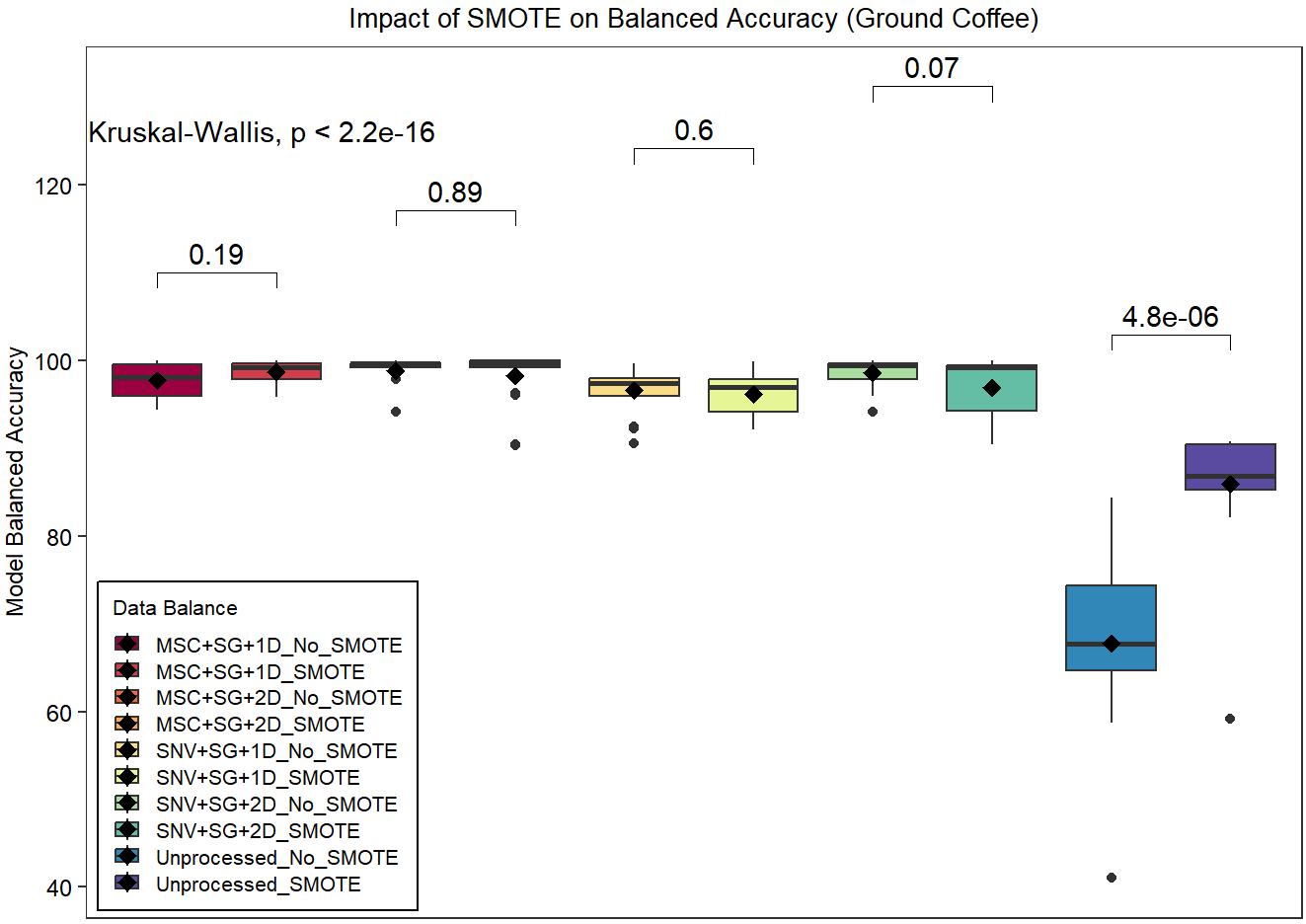
```
    legend.position = c(0.01, 0.009),
    legend.justification = c(0, 0),
    legend.background = element_rect(fill = "white", color = "black"),
    legend.key.size = unit(0.4, "cm"),
    legend.text = element_text(size = 8),
    legend.title = element_text(size = 8)
  )+
  scale_fill_brewer(palette = "Spectral", name = 'Data Balance')+
  stat_summary(fun=mean, geom="point", shape=18, size=3, color="black")+
  stat_compare_means(comparisons = my_comparisons)+
  stat_compare_means(label.y = max(df_bal$Balanced_Accuracy) * 1.5)
```




```
ggsave("SMOTE_Data_Preprocessing_Balance_Ground_Coffee_Specifity.png", width = 6, height = 4, dpi
```

```
# Define statistical comparisons
my_comparisons <- list(c('Unprocessed_No_SMOTE', 'Unprocessed_SMOTE'),
                       c('MSC+SG+1D_No_SMOTE', 'MSC+SG+1D_SMOTE'),
                       c('MSC+SG+2D_No_SMOTE', 'MSC+SG+2D_SMOTE'),
                       c('SNV+SG+1D_No_SMOTE', 'SNV+SG+1D_SMOTE'),
                       c('SNV+SG+2D_No_SMOTE', 'SNV+SG+2D_SMOTE'))
```

```r
# Create Box plot

df_bal %>% ggplot(aes(x = Preprocessing, y = Balanced_Accuracy, fill = Preprocessing)) +
  geom_boxplot() +
  labs(title = "Impact of SMOTE on Balanced Accuracy (Ground Coffee)",
       y = "Model Balanced Accuracy")+
  theme_bw()+
 theme(
    axis.title.x = element_blank(),
    axis.text.x = element_blank(),
    axis.ticks.x = element_blank(),
    axis.text.y = element_text(color = "black", size = 9),
    axis.title.y = element_text(color = "black", size = 9),
    plot.title = element_text(hjust = 0.5, color = 'black', size = 10),
    panel.grid = element_blank(),
    legend.position = c(0.01, 0.009),
    legend.justification = c(0,0),
    legend.background = element_rect(fill = "white", color = "black"),
    legend.key.size = unit(0.4, "cm"),
    legend.text = element_text(size = 8),
    legend.title = element_text(size=8),
    legend.spacing.y = unit(0.1,'lines'),
    legend.key.height = unit(0.7,'lines')
  )+
  scale_fill_brewer(palette = "Spectral",name = 'Data Balance')+
  stat_summary(fun=mean, geom="point", shape=18, size=3, color="black")+
  stat_compare_means(comparisons = my_comparisons)+
  stat_compare_means(label.y = max(df_bal$Balanced_Accuracy) * 1.25)
```



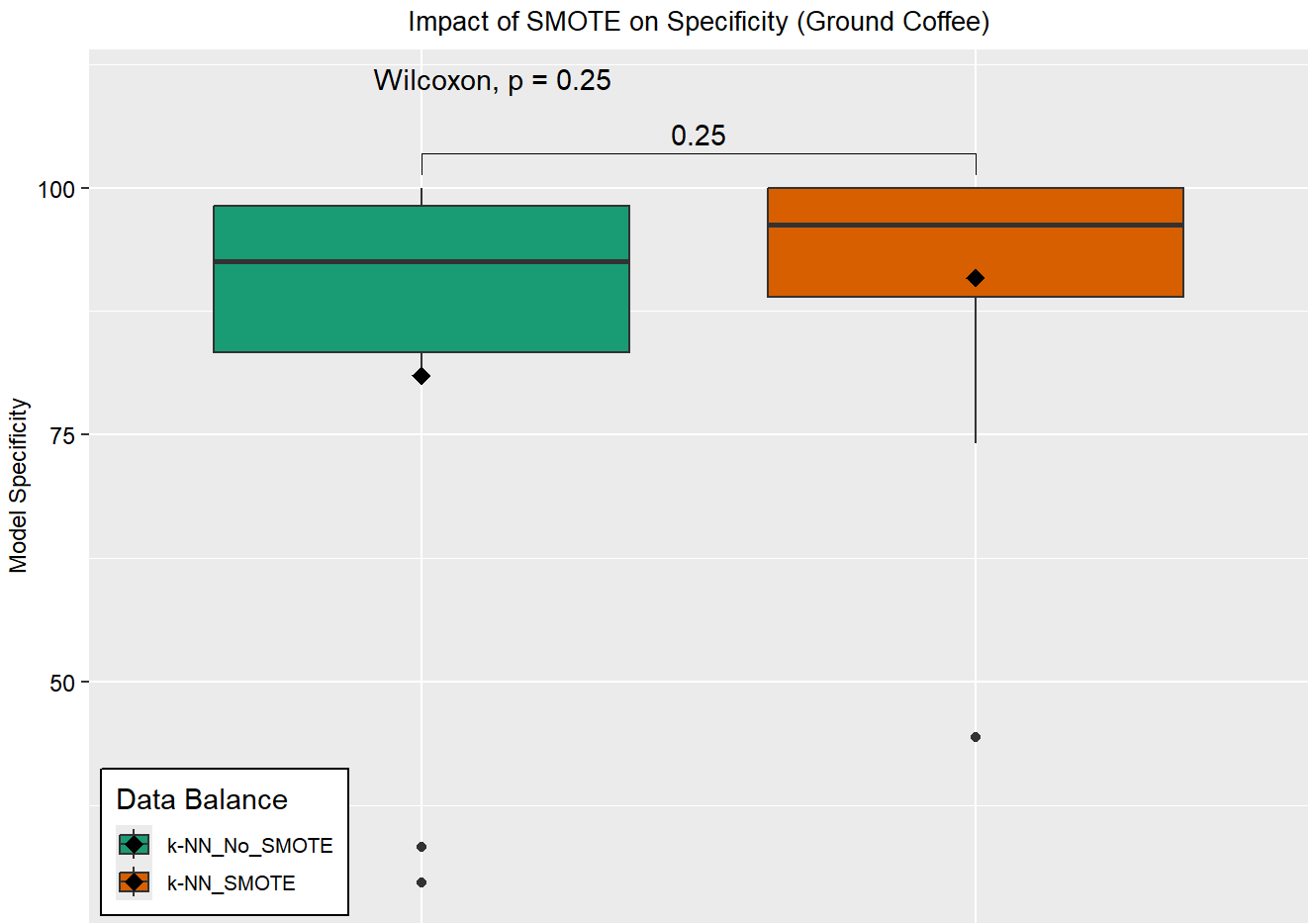

```
ggsave("SMOTE_Data_Preprocessing_Balance_Ground_Coffee_Accuracy.png", width = 6, height = 4, dpi =
```

```
knn <- df_bal %>%
  filter(Model %in% c('k-NN_No_SMOTE', 'k-NN_SMOTE')) %>%
  dplyr::select(Model, MCC, Balanced_Accuracy, Specificity)
head(knn)
```

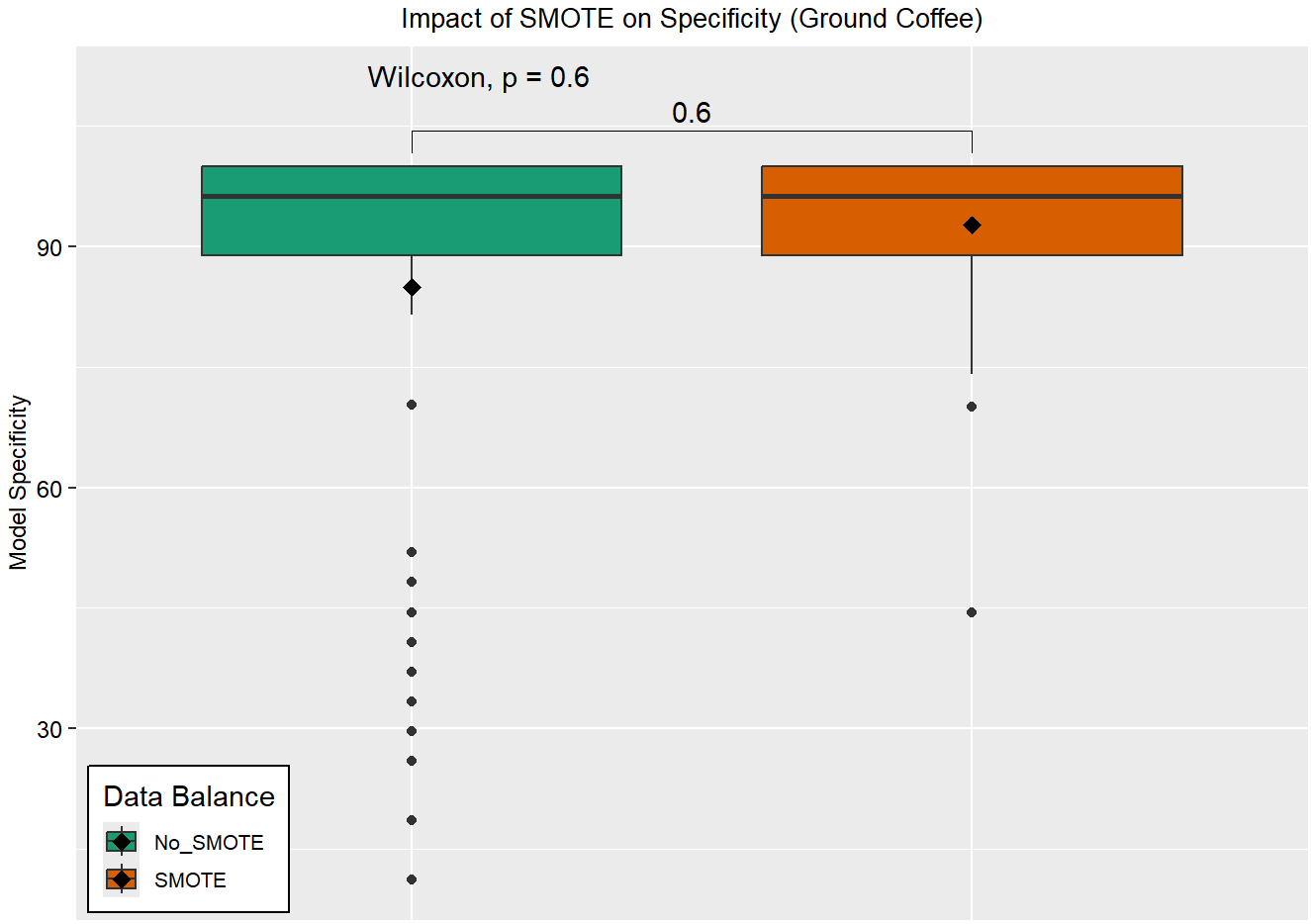
```
# A tibble: 6 × 4
  Model           MCC Balanced_Accuracy Specificity
  <fct>         <dbl>             <dbl>       <dbl>
1 k-NN_No_SMOTE  0.5               64.7        29.6
2 k-NN_No_SMOTE  0.86              90.5        81.5
3 k-NN_No_SMOTE  0.94              99.6       100
4 k-NN_No_SMOTE  0.9               95.9        92.6
5 k-NN_No_SMOTE  0.94              99.6       100
6 k-NN_No_SMOTE  0.45              64.7        29.6
```

```
# Define statistical comparisons
my_comparisons <- list(c('k-NN_No_SMOTE', 'k-NN_SMOTE'))
```

```r
# Create Boxplot

knn %>% ggplot(aes(x = Model, y = Specificity, fill = Model)) +
  geom_boxplot() +
  labs(title = "Impact of SMOTE on Specificity (Ground Coffee)",
       y = "Model Specificity")+
 theme(
    axis.title.x = element_blank(),
    axis.text.x = element_blank(),
    axis.ticks.x = element_blank(),
    axis.text.y = element_text(color = "black", size = 9),
    axis.title.y = element_text(color = "black", size = 9),
    plot.title = element_text(hjust = 0.5, color = 'black', size = 10),
    legend.position = c(0.01, 0.009),
    legend.justification = c(0, 0),
    legend.background = element_rect(fill = "white", color = "black"),
    legend.key.size = unit(0.5, "cm"),
    legend.text = element_text(size = 8)
  )+
  scale_fill_brewer(palette = "Dark2", name = 'Data Balance')+
  stat_summary(fun=mean, geom="point", shape=18, size=3, color="black")+
  stat_compare_means(comparisons = my_comparisons)+
  stat_compare_means(label.y = max(df_bal$Balanced_Accuracy) * 1.1)
```



```r
# Define statistical comparisons
my_comparisons <- list(c("No_SMOTE", "SMOTE"))

# Create Boxplot

df_bal %>% ggplot(aes(x = data_balance, y = Specificity, fill = data_balance)) +
  geom_boxplot() +
  labs(title = "Impact of SMOTE on Specificity (Ground Coffee)",
       y = "Model Specificity")+
 theme(
    axis.title.x = element_blank(),
    axis.text.x = element_blank(),
    axis.ticks.x = element_blank(),
    axis.text.y = element_text(color = "black", size = 9),
    axis.title.y = element_text(color = "black", size = 9),
    plot.title = element_text(hjust = 0.5, color = 'black', size = 10),
    legend.position = c(0.01, 0.009),
    legend.justification = c(0, 0),
    legend.background = element_rect(fill = "white", color = "black"),
    legend.key.size = unit(0.5, "cm"),
    legend.text = element_text(size = 8)
  )+
  scale_fill_brewer(palette = "Dark2", name = 'Data Balance')+
  stat_summary(fun=mean, geom="point", shape=18, size=3, color="black")+
  stat_compare_means(comparisons = my_comparisons)+
  stat_compare_means(label.y = max(df_bal$Balanced_Accuracy) * 1.1)
```




```
#ggsave("SMOTE_Data_Balance_Ground_Coffee.png", width = 6, height = 4, dpi = 600, bg = "white")
```

```
# Perform the Wilcoxon rank-sum test
# Based on MCC
overall_test <- wilcox.test(df$Unbalanced_Data_MCC, df$Balanced_Data_MCC)

print(overall_test)
```

```
    Wilcoxon rank sum test with continuity correction

data:  df$Unbalanced_Data_MCC and df$Balanced_Data_MCC
W = 4373.5, p-value = 0.3532
alternative hypothesis: true location shift is not equal to 0
```

```
# Based on Balanced Accuracy
overall_bal_acc <- wilcox.test(df$Imb_Balanced_Accuracy, df$Bal_Balanced_Accuracy)

# Print the results
print(overall_bal_acc)
```

```
    Wilcoxon rank sum test with continuity correction

data:  df$Imb_Balanced_Accuracy and df$Bal_Balanced_Accuracy
W = 4053.5, p-value = 0.9931
alternative hypothesis: true location shift is not equal to 0
```

- There is no statistical difference in balanced accuracy or MCC between the two groups although models trained with balanced data show improved balanced accuracy when tested on an external data set.

```
mean_imbalanced <- mean(df$Imb_Balanced_Accuracy)
mean_balanced <- mean(df$Bal_Balanced_Accuracy)

print(paste('The mean balanced accuracy for imbalanced dataset is', round(mean_imbalanced, 2), 'wh
```

```
[1] "The mean balanced accuracy for imbalanced dataset is 91.9 while for the balanced dataset is
95.25"
```

Check SMOTE performance based on pre_processing

Verify with General Linear Model (GLM)

Fit GLM-Unbalanced Data

```
glm_model <- glm(Unbalanced_Data_MCC ~ Model + Pre_processing + Wav_Selec_Reduction,
                 data = df,
                 family = gaussian())
summary(glm_model)
```

```
Call:
glm(formula = Unbalanced_Data_MCC ~ Model + Pre_processing +
    Wav_Selec_Reduction, family = gaussian(), data = df)

Coefficients:
                                   Estimate Std. Error t value Pr(>|t|)
(Intercept)                        0.964778   0.030153  31.997   <2e-16 ***
Modelk-NN                         -0.044000   0.030153  -1.459    0.149
ModelLDA                          -0.029333   0.030153  -0.973    0.334
ModelNNET                          0.015333   0.030153   0.509    0.613
ModelRF                           -0.044667   0.030153  -1.481    0.143
ModelSVM                          -0.042667   0.030153  -1.415    0.161
Pre_processingMSC+SG+2D            0.010556   0.027525   0.383    0.702
Pre_processingSNV+SG+1D           -0.013889   0.027525  -0.505    0.615
Pre_processingSNV+SG+2D            0.008333   0.027525   0.303    0.763
Pre_processingUnprocessed         -0.481111   0.027525 -17.479   <2e-16 ***
Wav_Selec_ReductionFull _Spectra_PCA -0.001333   0.021321  -0.063    0.950
```

```
Wav_Selec_ReductionRFE_GA_PCA        -0.025333   0.021321  -1.188    0.238
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 0.006818832)

    Null deviance: 3.95106  on 89  degrees of freedom
Residual deviance: 0.53187  on 78  degrees of freedom
AIC: -180.4

Number of Fisher Scoring iterations: 2
```

Pairwise comparisons-Unbalanced data

```r
# Pairwise comparisons for Model
emmeans_model <- emmeans(glm_model, ~ Model)
pairs(emmeans_model, adjust = "bonferroni")  # Bonferroni correction
```

```
 contrast           estimate     SE df t.ratio p.value
 Ensemble - (k-NN)  0.044000 0.0302 78   1.459  1.0000
 Ensemble - LDA     0.029333 0.0302 78   0.973  1.0000
 Ensemble - NNET   -0.015333 0.0302 78  -0.509  1.0000
 Ensemble - RF      0.044667 0.0302 78   1.481  1.0000
 Ensemble - SVM     0.042667 0.0302 78   1.415  1.0000
 (k-NN) - LDA      -0.014667 0.0302 78  -0.486  1.0000
 (k-NN) - NNET     -0.059333 0.0302 78  -1.968  0.7897
 (k-NN) - RF        0.000667 0.0302 78   0.022  1.0000
 (k-NN) - SVM      -0.001333 0.0302 78  -0.044  1.0000
 LDA - NNET        -0.044667 0.0302 78  -1.481  1.0000
 LDA - RF           0.015333 0.0302 78   0.509  1.0000
 LDA - SVM          0.013333 0.0302 78   0.442  1.0000
 NNET - RF          0.060000 0.0302 78   1.990  0.7516
 NNET - SVM         0.058000 0.0302 78   1.924  0.8709
 RF - SVM          -0.002000 0.0302 78  -0.066  1.0000

Results are averaged over the levels of: Pre_processing, Wav_Selec_Reduction
P value adjustment: bonferroni method for 15 tests
```

```r
# Pairwise comparisons for Pre_processing
emmeans_preprocessing <- emmeans(glm_model, ~ Pre_processing)
pairs(emmeans_preprocessing, adjust = "bonferroni")
```

```
 contrast                  estimate     SE df t.ratio p.value
 (MSC+SG+1D) - (MSC+SG+2D) -0.01056 0.0275 78  -0.383  1.0000
 (MSC+SG+1D) - (SNV+SG+1D)  0.01389 0.0275 78   0.505  1.0000
 (MSC+SG+1D) - (SNV+SG+2D) -0.00833 0.0275 78  -0.303  1.0000
 (MSC+SG+1D) - Unprocessed  0.48111 0.0275 78  17.479  <.0001
 (MSC+SG+2D) - (SNV+SG+1D)  0.02444 0.0275 78   0.888  1.0000
 (MSC+SG+2D) - (SNV+SG+2D)  0.00222 0.0275 78   0.081  1.0000
```

```
 (MSC+SG+2D) - Unprocessed  0.49167 0.0275 78  17.862  <.0001
 (SNV+SG+1D) - (SNV+SG+2D) -0.02222 0.0275 78  -0.807  1.0000
 (SNV+SG+1D) - Unprocessed  0.46722 0.0275 78  16.974  <.0001
 (SNV+SG+2D) - Unprocessed  0.48944 0.0275 78  17.782  <.0001

Results are averaged over the levels of: Model, Wav_Selec_Reduction
P value adjustment: bonferroni method for 10 tests
```

```r
# Pairwise comparisons for Wav_Selec_Reduction
emmeans_wave <- emmeans(glm_model, ~ Wav_Selec_Reduction)
pairs(emmeans_wave, adjust = "bonferroni")
```

```
 contrast                       estimate     SE df t.ratio p.value
 Boruta_PCA - Full _Spectra_PCA  0.00133 0.0213 78   0.063  1.0000
 Boruta_PCA - RFE_GA_PCA         0.02533 0.0213 78   1.188  0.7151
 Full _Spectra_PCA - RFE_GA_PCA  0.02400 0.0213 78   1.126  0.7913

Results are averaged over the levels of: Model, Pre_processing
P value adjustment: bonferroni method for 3 tests
```

Fit GLM-Balanced Data

```r
glm_model_bal <- glm(Balanced_Data_MCC ~ Model + Pre_processing + Wav_Selec_Reduction,
                 data = df,
                 family = gaussian())
summary(glm_model_bal)
```

```
Call:
glm(formula = Balanced_Data_MCC ~ Model + Pre_processing + Wav_Selec_Reduction,
    family = gaussian(), data = df)

Coefficients:
                                    Estimate Std. Error t value Pr(>|t|)
(Intercept)                         0.974667   0.022645  43.042  < 2e-16 ***
Modelk-NN                          -0.065333   0.022645  -2.885  0.00506 **
ModelLDA                           -0.050667   0.022645  -2.237  0.02811 *
ModelNNET                           0.009333   0.022645   0.412  0.68135
ModelRF                            -0.030000   0.022645  -1.325  0.18910
ModelSVM                           -0.006667   0.022645  -0.294  0.76923
Pre_processingMSC+SG+2D             0.008333   0.020672   0.403  0.68796
Pre_processingSNV+SG+1D            -0.039444   0.020672  -1.908  0.06005 .
Pre_processingSNV+SG+2D            -0.018889   0.020672  -0.914  0.36366
Pre_processingUnprocessed          -0.382222   0.020672 -18.490  < 2e-16 ***
Wav_Selec_ReductionFull _Spectra_PCA -0.038000   0.016012  -2.373  0.02010 *
Wav_Selec_ReductionRFE_GA_PCA      -0.012667   0.016012  -0.791  0.43130
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
(Dispersion parameter for gaussian family taken to be 0.00384584)

    Null deviance: 2.38191  on 89  degrees of freedom
Residual deviance: 0.29998  on 78  degrees of freedom
AIC: -231.94

Number of Fisher Scoring iterations: 2
```

```r
# Pairwise comparisons for Model
emmeans_model_bal <- emmeans(glm_model_bal, ~ Model)
pairs(emmeans_model_bal, adjust = "bonferroni")  # Bonferroni correction
```

```
 contrast          estimate     SE df t.ratio p.value
 Ensemble - (k-NN)  0.06533 0.0226 78   2.885  0.0758
 Ensemble - LDA     0.05067 0.0226 78   2.237  0.4216
 Ensemble - NNET   -0.00933 0.0226 78  -0.412  1.0000
 Ensemble - RF      0.03000 0.0226 78   1.325  1.0000
 Ensemble - SVM     0.00667 0.0226 78   0.294  1.0000
 (k-NN) - LDA      -0.01467 0.0226 78  -0.648  1.0000
 (k-NN) - NNET     -0.07467 0.0226 78  -3.297  0.0221
 (k-NN) - RF       -0.03533 0.0226 78  -1.560  1.0000
 (k-NN) - SVM      -0.05867 0.0226 78  -2.591  0.1714
 LDA - NNET        -0.06000 0.0226 78  -2.650  0.1463
 LDA - RF          -0.02067 0.0226 78  -0.913  1.0000
 LDA - SVM         -0.04400 0.0226 78  -1.943  0.8342
 NNET - RF          0.03933 0.0226 78   1.737  1.0000
 NNET - SVM         0.01600 0.0226 78   0.707  1.0000
 RF - SVM          -0.02333 0.0226 78  -1.030  1.0000

Results are averaged over the levels of: Pre_processing, Wav_Selec_Reduction
P value adjustment: bonferroni method for 15 tests
```

```r
# Pairwise comparisons for Pre_processing
emmeans_preprocessing_bal <- emmeans(glm_model_bal, ~ Pre_processing)
pairs(emmeans_preprocessing_bal, adjust = "bonferroni")
```

```
 contrast                  estimate     SE df t.ratio p.value
 (MSC+SG+1D) - (MSC+SG+2D) -0.00833 0.0207 78  -0.403  1.0000
 (MSC+SG+1D) - (SNV+SG+1D)  0.03944 0.0207 78   1.908  0.6005
 (MSC+SG+1D) - (SNV+SG+2D)  0.01889 0.0207 78   0.914  1.0000
 (MSC+SG+1D) - Unprocessed  0.38222 0.0207 78  18.490  <.0001
 (MSC+SG+2D) - (SNV+SG+1D)  0.04778 0.0207 78   2.311  0.2346
 (MSC+SG+2D) - (SNV+SG+2D)  0.02722 0.0207 78   1.317  1.0000
 (MSC+SG+2D) - Unprocessed  0.39056 0.0207 78  18.893  <.0001
 (SNV+SG+1D) - (SNV+SG+2D) -0.02056 0.0207 78  -0.994  1.0000
 (SNV+SG+1D) - Unprocessed  0.34278 0.0207 78  16.582  <.0001
 (SNV+SG+2D) - Unprocessed  0.36333 0.0207 78  17.576  <.0001
```

```
Results are averaged over the levels of: Model, Wav_Selec_Reduction
P value adjustment: bonferroni method for 10 tests
```

```
# Pairwise comparisons for Wav_Selec_Reduction
emmeans_wave_bal <- emmeans(glm_model_bal, ~ Wav_Selec_Reduction)
pairs(emmeans_wave_bal, adjust = "bonferroni")
```

```
 contrast                       estimate    SE df t.ratio p.value
 Boruta_PCA - Full _Spectra_PCA   0.0380 0.016 78   2.373  0.0603
 Boruta_PCA - RFE_GA_PCA          0.0127 0.016 78   0.791  1.0000
 Full _Spectra_PCA - RFE_GA_PCA  -0.0253 0.016 78  -1.582  0.3530

Results are averaged over the levels of: Model, Pre_processing
P value adjustment: bonferroni method for 3 tests
```

```
# Group by some models

Model_Means <- df %>%
  dplyr::select(Model, Unbalanced_Data_MCC, Balanced_Data_MCC) %>%
  group_by(Model) %>%
  summarise(
    Mean_Unbalanced_Data_MCC = mean(Unbalanced_Data_MCC, na.rm = TRUE),
    Mean_Balanced_Data_MCC = mean(Balanced_Data_MCC, na.rm = TRUE)
  ) %>% arrange(desc(Mean_Balanced_Data_MCC))

# View the result
print(Model_Means)
```

```
# A tibble: 6 × 3
  Model    Mean_Unbalanced_Data_MCC Mean_Balanced_Data_MCC
  <fct>                       <dbl>                  <dbl>
1 NNET                        0.876                  0.881
2 Ensemble                    0.861                  0.871
3 SVM                         0.818                  0.865
4 RF                          0.816                  0.841
5 LDA                         0.831                  0.821
6 k-NN                        0.817                  0.806
```

```
# Group Variable Selection MCC Averages
model_means_var_selec <- df %>%
  dplyr::select(Wav_Selec_Reduction, Unbalanced_Data_MCC, Balanced_Data_MCC,Bal_Balanced_Accuracy
  group_by(Wav_Selec_Reduction) %>%
  summarise(
    Mean_Unbalanced_Data_MCC = mean(Unbalanced_Data_MCC, na.rm = TRUE),
    Mean_Balanced_Data_MCC = mean(Balanced_Data_MCC, na.rm = TRUE),
    Mean_Balanced_Data_Acc = mean(Bal_Balanced_Accuracy, na.rm = TRUE)
  ) %>% arrange(desc(Mean_Balanced_Data_MCC))
```

```
print(model_means_var_selec)
```

```
# A tibble: 3 × 4
  Wav_Selec_Reduction Mean_Unbalanced_Data_MCC Mean_Balanced_Data_MCC
  <fct>                                 <dbl>                  <dbl>
1 Boruta_PCA                            0.845                  0.864
2 RFE_GA_PCA                            0.82                   0.852
3 Full _Spectra_PCA                     0.844                  0.826
# ℹ 1 more variable: Mean_Balanced_Data_Acc <dbl>
```

```
mcc_models<-df %>%
  dplyr::select(Model, Unbalanced_Data_MCC, Balanced_Data_MCC,
         Imb_Balanced_Accuracy, Bal_Balanced_Accuracy,
         Imbal_Specificty, Bal_Specificty) %>%
  group_by(Model) %>%
  summarise(across(where(is.numeric), \(x) mean(x, na.rm = TRUE)))
mcc_models
```

```
# A tibble: 6 × 7
  Model    Unbalanced_Data_MCC Balanced_Data_MCC Imb_Balanced_Accuracy
  <fct>                  <dbl>             <dbl>                 <dbl>
1 Ensemble               0.861             0.871                  93.8
2 k-NN                   0.817             0.806                  90.1
3 LDA                    0.831             0.821                  93.7
4 NNET                   0.876             0.881                  94.3
5 RF                     0.816             0.841                  90.0
6 SVM                    0.818             0.865                  89.6
# ℹ 3 more variables: Bal_Balanced_Accuracy <dbl>, Imbal_Specificty <dbl>,
#   Bal_Specificty <dbl>
```

The GLM analysis on data balanced using SMOTE reveals intriguing insights. Stacked models and NNET demonstrate significantly superior performance compared to k-NN. Furthermore, spectral pre-processing has been validated as an effective approach to enhance model performance. Notably, feature selection using Boruta significantly outperforms models trained on full spectra data.

Part 1: Key Messages/Conclusions

Superior Model Performance: Stacked models and NNET consistently outperform k-NN in both balanced and unbalanced data sets, highlighting their effectiveness for coffee discrimination tasks.

Impact of Spectral Pre-Processing: Spectral pre-processing techniques, particularly SNV+SG (1st and 2nd derivatives) and MSC+SG (1st and 2nd derivatives), significantly enhance model performance compared to raw, unprocessed spectra.

Effectiveness of Feature Selection: Models trained with feature selection using Boruta outperform those trained on the full spectra, underscoring the importance of variable selection in improving classification accuracy.

Balanced Data Insights: While there is no statistical difference in balanced accuracy or MCC between balanced and unbalanced data, models trained on balanced data sets demonstrate improved balanced accuracy when tested on external data sets, emphasizing the value of data balancing through SMOTE.

Validation of Pre-Processing and Data Balancing: Both pre-processing and data balancing are confirmed as critical steps to achieve superior model performance, providing a roadmap for optimizing machine learning applications in hyperspectral data analysis.

Part 2: Assess Time Differences for Training the Models

```
df_time <- read_excel('model_training_time.xlsx')
head(df_time)
```

```
# A tibble: 6 × 4
  Model Wav_Selec_Reduction Imbalanced_Training_Time Balanced_Training_Time
  <chr> <chr>                                  <dbl>                   <dbl>
1 k-NN  Full _Spectra_PCA                       13                      33.4
2 k-NN  Full _Spectra_PCA                       16                      40
3 k-NN  Full _Spectra_PCA                       21.4                    48.4
4 k-NN  Full _Spectra_PCA                       20                      44.3
5 k-NN  Full _Spectra_PCA                       18.9                    43.5
6 LDA   Full _Spectra_PCA                        1.7                     2
```

```
df_time$Model <- as.factor(df_time$Model) # convert to a factor
df_time$Wav_Selec_Reduction <- as.factor(df_time$Wav_Selec_Reduction)
```

```
# Get the average/mean training time grouped by model type
df_time %>% dplyr::select(Model,Imbalanced_Training_Time,Balanced_Training_Time) %>%
  group_by(Model) %>% summarise(
    Mean_Training_Time_Imbalanced = mean(Imbalanced_Training_Time),
    Mean_Training_Time_Balanced = mean(Balanced_Training_Time)) %>%
  arrange(desc(Mean_Training_Time_Balanced))
```

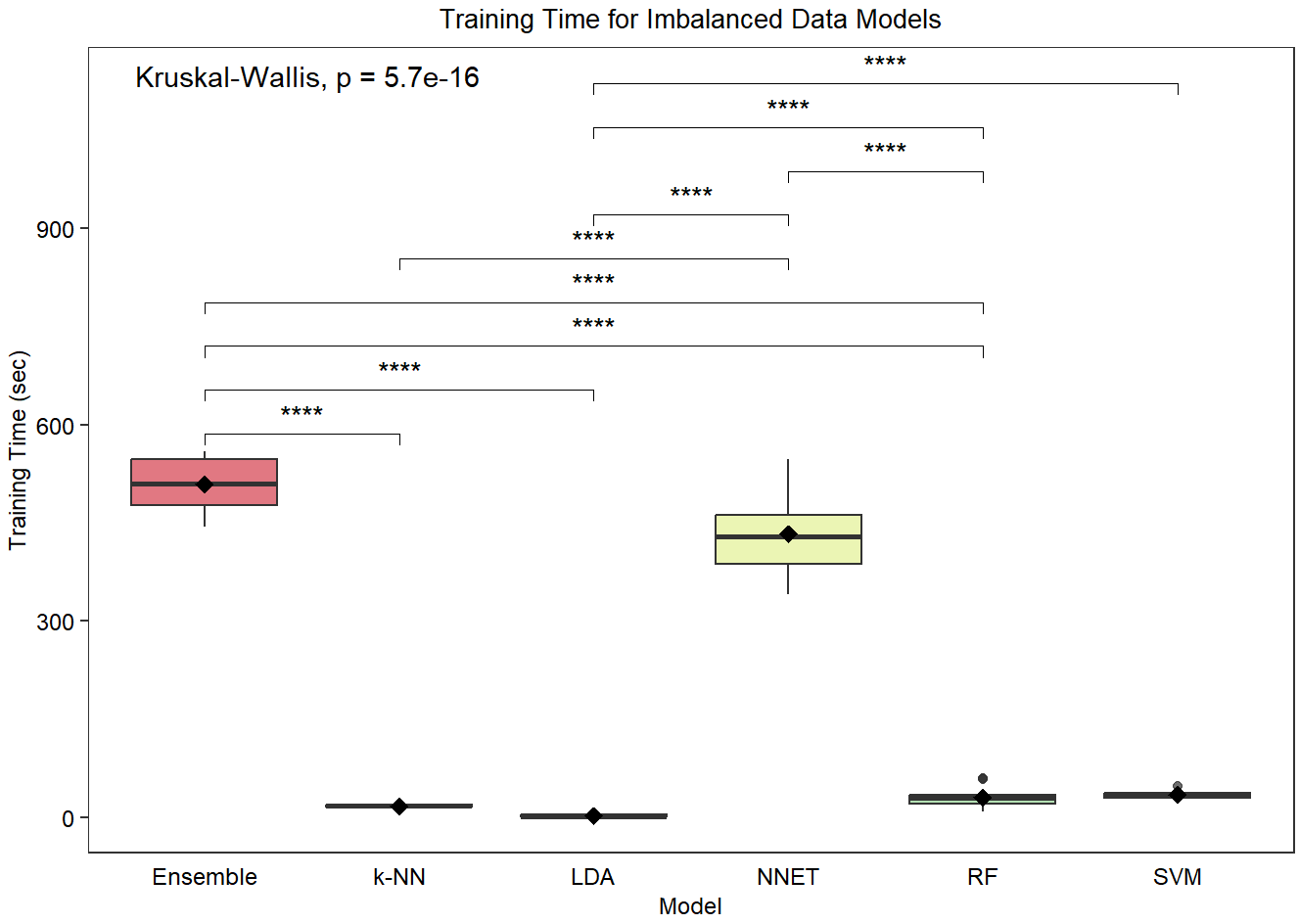
```
# A tibble: 6 × 3
  Model    Mean_Training_Time_Imbalanced Mean_Training_Time_Balanced
  <fct>                            <dbl>                       <dbl>
1 NNET                             433.                        875.
2 Ensemble                         510.                        691.
3 RF                                31.1                        71.1
4 SVM                               34.6                        64.8
5 k-NN                              17.9                        29.5
6 LDA                                3.01                        2.97
```

```r
# Get the average/mean training time grouped by variable selection
df_time %>% dplyr::select(Wav_Selec_Reduction,Imbalanced_Training_Time,Balanced_Training_Time) %>%
  group_by(Wav_Selec_Reduction) %>% summarise(
    Mean_Training_Time_Imbalanced = mean(Imbalanced_Training_Time),
    Mean_Training_Time_Balanced = mean(Balanced_Training_Time)) %>%
  arrange(desc(Mean_Training_Time_Balanced))
```

```
# A tibble: 3 × 3
  Wav_Selec_Reduction Mean_Training_Time_Imbalanced Mean_Training_Time_Balanced
  <fct>                                      <dbl>                       <dbl>
1 Full _Spectra_PCA                           171.                        329.
2 RFE_GA_PCA                                  179.                        326.
3 Boruta_PCA                                  164.                        212.
```

```r
# Visualizations

# Training Time for Imbalanced Data based on type of model

# Pairwise comparisons based on Bonferroni-corrected results
my_comparisons <- list(
  c("Ensemble","k-NN"), c("Ensemble","LDA"), c("Ensemble","RF"),
  c("Ensemble","RF"),c("NNET","k-NN"),c("NNET","LDA"),c("NNET","RF"),
  c("RF", "LDA"),c("SVM", "LDA"))

# Boxplot with statistical comparisons
ggplot(df_time, aes(x = Model, y = Imbalanced_Training_Time, fill = Model)) +
geom_boxplot(alpha = 0.7) +
ylab('Training Time (sec)')+
labs(title = 'Training Time for Imbalanced Data Models')+
theme_bw()+
theme(
    axis.title.x = element_text(color = "black", size = 9),
    axis.text.x = element_text(color = "black", size = 9),
    axis.ticks.x = element_blank(),
    axis.text.y = element_text(color = "black", size = 9),
    axis.title.y = element_text(color = "black", size = 9),
    plot.title = element_text(hjust = 0.5, color = 'black', size = 10),
    panel.grid = element_blank(),
    legend.position = 'none',
    legend.justification = c(0,0),
    legend.background = element_rect(fill = "white", color = "black"),
    legend.key.size = unit(0.4, "cm"),
    legend.text = element_text(size = 8),
    legend.title = element_text(size=8),
    legend.spacing.y = unit(0.1,'lines'),
    legend.key.height = unit(0.7,'lines')
  )+
```

```
  scale_fill_brewer(palette = "Spectral",name = 'Data Balance')+
  stat_summary(fun=mean, geom="point", shape=18, size=3, color="black")+
  stat_compare_means(comparisons = my_comparisons, label = 'p.signif')+
  stat_compare_means(label.y = max(df_time$Imbalanced_Training_Time) * 2)
```




```
ggsave("No_SMOTE_Data_Model_Training_Time.png", width = 6, height = 4, dpi = 600, bg = "white")
```

Check the statistics

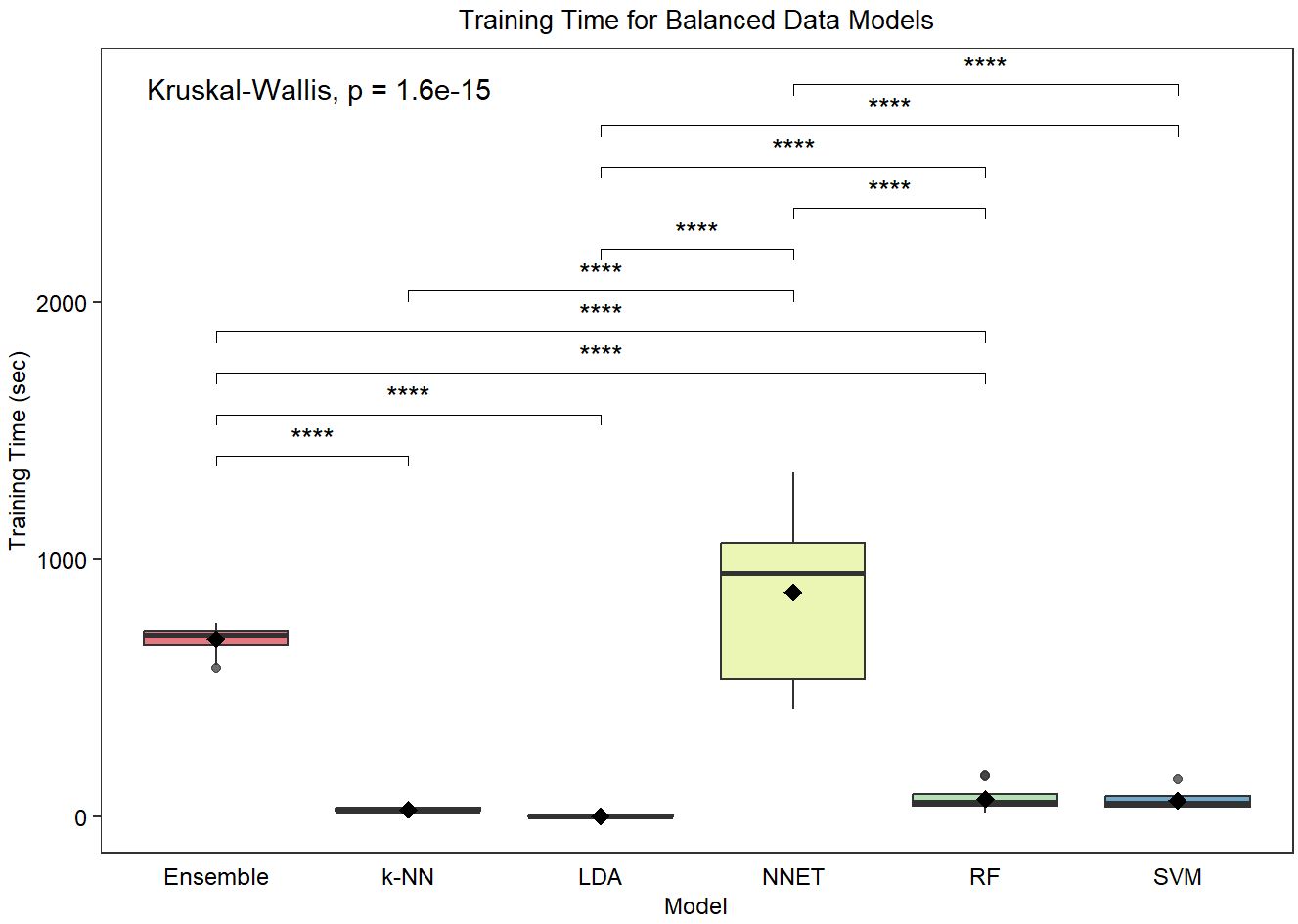
```
dunn.test(df_time$Imbalanced_Training_Time, df_time$Model, method = 'bonferroni')
```

```
  Kruskal-Wallis rank sum test

data: x and group
Kruskal-Wallis chi-squared = 80.8109, df = 5, p-value = 0

                           Comparison of x by group
                                 (Bonferroni)
Col Mean-|
Row Mean |    Ensemble       k-NN        LDA       NNET         RF
---------+----------------------------------------------------------
    k-NN |    5.727365
```

```
          |    0.0000*
          |
      LDA |   7.635322   1.907956
          |    0.0000*     0.4230
          |
     NNET |   1.118216  -4.609148  -6.517105
          |     1.0000    0.0000*    0.0000*
          |
       RF |   4.315616  -1.411748  -3.319705   3.197400
          |    0.0001*     1.0000    0.0068*    0.0104*
          |
      SVM |   3.428032  -2.299332  -4.207289   2.309815  -0.887584
          |    0.0046*     0.1611    0.0002*     0.1567     1.0000

alpha = 0.05
Reject Ho if p <= alpha/2
```

```r
library(ggthemes)
# Pairwise comparisons based on Bonferroni-corrected results
my_comparisons <- list(
  c("Ensemble","k-NN"), c("Ensemble","LDA"), c("Ensemble","RF"),
  c("Ensemble","RF"),c("NNET","k-NN"),c("NNET","LDA"),c("NNET","RF"),
  c("RF", "LDA"),c("SVM", "LDA"),c('NNET','SVM'))

# Boxplot with statistical comparisons
ggplot(df_time, aes(x = Model, y = Balanced_Training_Time, fill = Model)) +
geom_boxplot(alpha = 0.7) +
ylab('Training Time (sec)')+
labs(title = 'Training Time for Balanced Data Models')+
theme_bw()+
theme(
    axis.title.x = element_text(color = "black", size = 9),
    axis.text.x = element_text(color = "black", size = 9),
    axis.ticks.x = element_blank(),
    axis.text.y = element_text(color = "black", size = 9),
    axis.title.y = element_text(color = "black", size = 9),
    plot.title = element_text(hjust = 0.5, color = 'black', size = 10),
    legend.position = 'none',
    legend.justification = c(0,0),
    legend.background = element_rect(fill = "white", color = "black"),
    legend.key.size = unit(0.4, "cm"),
    legend.text = element_text(size = 8),
    legend.title = element_text(size=8),
    legend.spacing.y = unit(0.1,'lines'),
    legend.key.height = unit(0.7,'lines'),
    panel.grid = element_blank(),
  )+
  scale_fill_brewer(palette = "Spectral",name = 'Model')+
  stat_summary(fun=mean, geom="point", shape=18, size=3, color="black")+
```

```
  stat_compare_means(comparisons = my_comparisons, label = 'p.signif')+
  stat_compare_means(label.y = max(df_time$Imbalanced_Training_Time) * 5)
```




```
ggsave("SMOTE_Data_Model_Training_Time.png", width = 6, height = 4, dpi = 600, bg = "white")
```

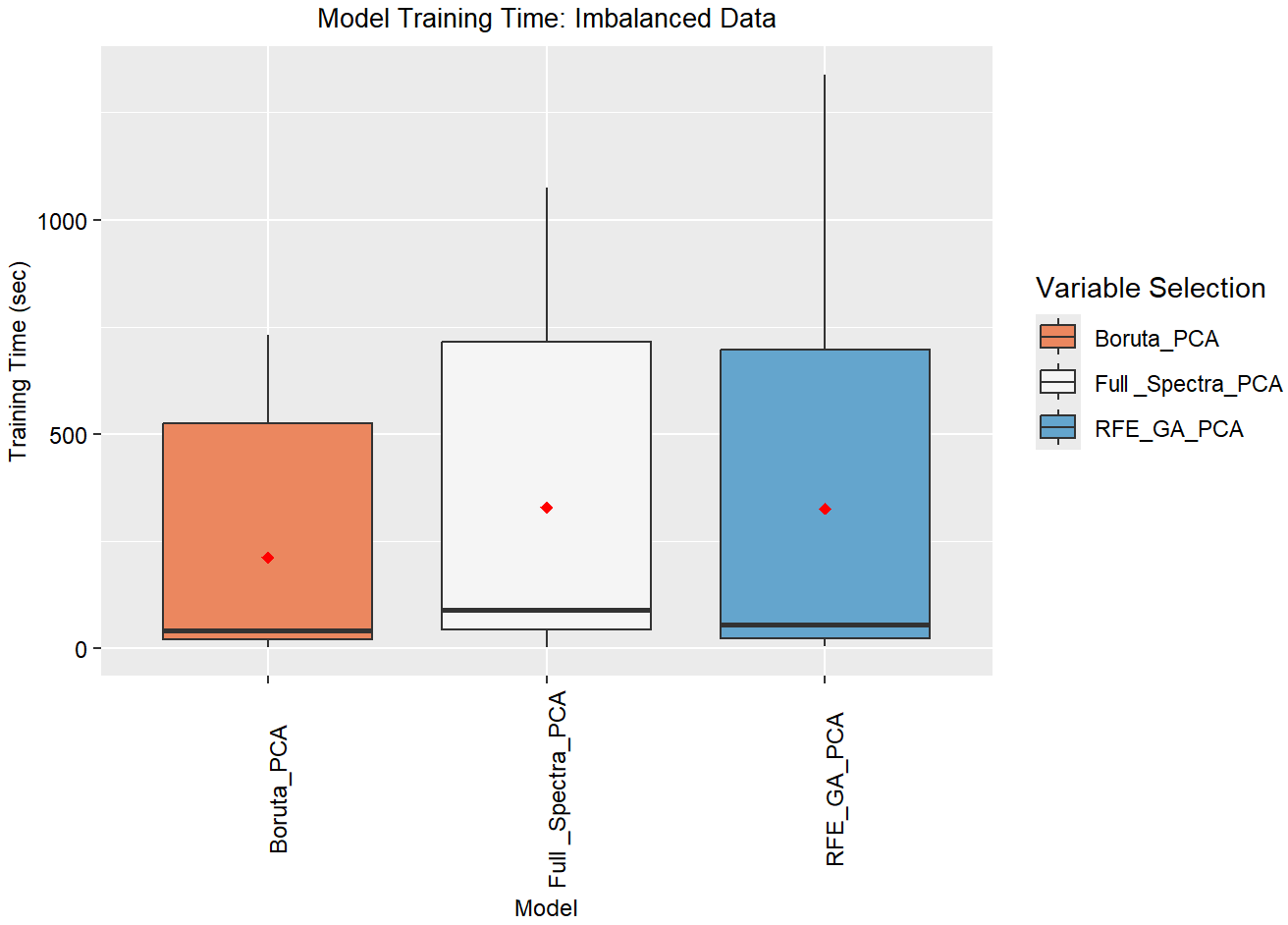
```
dunn.test(df_time$Balanced_Training_Time, df_time$Model, method = 'bonferroni')
```

```
  Kruskal-Wallis rank sum test

data: x and group
Kruskal-Wallis chi-squared = 78.6426, df = 5, p-value = 0

                          Comparison of x by group
                                (Bonferroni)
Col Mean-|
Row Mean |   Ensemble       k-NN        LDA       NNET         RF
---------+----------------------------------------------------------
    k-NN |   4.850264
         |    0.0000*
         |
     LDA |   6.814131   1.963867
         |    0.0000*     0.3716
```

```
        |
   NNET |  -0.524163  -5.374428  -7.338295
        |    1.0000    0.0000*    0.0000*
        |
     RF |   3.064612  -1.785651  -3.749519   3.588776
        |    0.0163*    0.5562    0.0013*    0.0025*
        |
    SVM |   3.092567  -1.757696  -3.721564   3.616731   0.027955
        |    0.0149*    0.5910    0.0015*    0.0022*     1.0000

alpha = 0.05
Reject Ho if p <= alpha/2
```

```r
# Training Time for Imbalanced Data based on wavelength selection method

df_time %>% ggplot(aes(x = Wav_Selec_Reduction, y = Imbalanced_Training_Time))+
  geom_boxplot(aes(fill = Wav_Selec_Reduction))+
  labs(title = 'Model Training Time: Imbalanced Data',
       y = 'Training Time (sec)',
       x = 'Model')+
  theme(axis.title.x = element_text(color = "black",size= 9),
        axis.text.x = element_text(color = "black", angle = 90, size= 9),
        axis.text.y = element_text(color = "black",size =9),
        axis.title.y = element_text(color = "black",size =9),
        plot.title = element_text(hjust = 0.5, color = 'black', size = 10))+
  scale_fill_brewer(palette = "BrBG", name = 'Variable Selection')+
  stat_summary(fun=mean, geom="point", shape=18, size=2, color="red")
```


```
# Training Time for Balanced Data based on wavelength selection method

df_time %>% ggplot(aes(x = Wav_Selec_Reduction, y = Balanced_Training_Time))+
  geom_boxplot(aes(fill = Wav_Selec_Reduction))+
  labs(title = 'Model Training Time: Imbalanced Data',
       y = 'Training Time (sec)',
       x = 'Model')+
  theme(axis.title.x = element_text(color = "black",size= 9),
        axis.text.x = element_text(color = "black", angle = 90, size= 9),
        axis.text.y = element_text(color = "black",size =9),
        axis.title.y = element_text(color = "black",size =9),
        plot.title = element_text(hjust = 0.5, color = 'black', size = 10))+
  scale_fill_brewer(palette = "RdBu", name = 'Variable Selection')+
  stat_summary(fun=mean, geom="point", shape=18, size=2, color="red")
```



Check differences with GLM Model

```
colnames(df_time)
```

```
[1] "Model"                    "Wav_Selec_Reduction"
[3] "Imbalanced_Training_Time" "Balanced_Training_Time"
```

```
# Unbalanced data
glm_time <- glm(Imbalanced_Training_Time~Model+Wav_Selec_Reduction,
                family =gaussian(), data = df_time )
print(summary(glm_time))
```

```
Call:
glm(formula = Imbalanced_Training_Time ~ Model + Wav_Selec_Reduction,
    family = gaussian(), data = df_time)

Coefficients:
                                Estimate Std. Error t value Pr(>|t|)
(Intercept)                      502.101      9.359  53.651  < 2e-16 ***
Modelk-NN                       -491.600     11.462 -42.889  < 2e-16 ***
ModelLDA                        -506.500     11.462 -44.189  < 2e-16 ***
```

```
ModelNNET                              -76.127     11.462  -6.642 3.15e-09 ***
ModelRF                                -478.373     11.462 -41.735  < 2e-16 ***
ModelSVM                               -474.907     11.462 -41.433  < 2e-16 ***
Wav_Selec_ReductionFull _Spectra_PCA     7.140      8.105   0.881   0.3809
Wav_Selec_ReductionRFE_GA_PCA           15.097      8.105   1.863   0.0661 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 985.3422)

    Null deviance: 4183044  on 89  degrees of freedom
Residual deviance:   80798  on 82  degrees of freedom
AIC: 885.4

Number of Fisher Scoring iterations: 2
```

```r
# Check pairwise comparisons

# Pairwise comparisons for Model
emmeans_model_imbal_time <- emmeans(glm_time, ~ Model)
pairs(emmeans_model_imbal_time, adjust = "bonferroni")  # Bonferroni correction
```

```
 contrast          estimate   SE df t.ratio p.value
 Ensemble - (k-NN)   491.60 11.5 82  42.889  <.0001
 Ensemble - LDA      506.50 11.5 82  44.189  <.0001
 Ensemble - NNET      76.13 11.5 82   6.642  <.0001
 Ensemble - RF       478.37 11.5 82  41.735  <.0001
 Ensemble - SVM      474.91 11.5 82  41.433  <.0001
 (k-NN) - LDA         14.90 11.5 82   1.300  1.0000
 (k-NN) - NNET      -415.47 11.5 82 -36.248  <.0001
 (k-NN) - RF         -13.23 11.5 82  -1.154  1.0000
 (k-NN) - SVM        -16.69 11.5 82  -1.456  1.0000
 LDA - NNET         -430.37 11.5 82 -37.548  <.0001
 LDA - RF            -28.13 11.5 82  -2.454  0.2437
 LDA - SVM           -31.59 11.5 82  -2.756  0.1080
 NNET - RF           402.25 11.5 82  35.094  <.0001
 NNET - SVM          398.78 11.5 82  34.791  <.0001
 RF - SVM             -3.47 11.5 82  -0.302  1.0000

Results are averaged over the levels of: Wav_Selec_Reduction
P value adjustment: bonferroni method for 15 tests
```

```r
# Pairwise comparisons for Wav_Selec_Reduction
emmeans_wave_imbal_time <- emmeans(glm_time, ~ Wav_Selec_Reduction)
pairs(emmeans_wave_imbal_time, adjust = "bonferroni")
```

```
 contrast                      estimate  SE df t.ratio p.value
 Boruta_PCA - Full _Spectra_PCA    -7.14 8.1 82  -0.881  1.0000
 Boruta_PCA - RFE_GA_PCA          -15.10 8.1 82  -1.863  0.1983
```

```
 Full _Spectra_PCA - RFE_GA_PCA    -7.96 8.1 82  -0.982  0.9874

Results are averaged over the levels of: Model
P value adjustment: bonferroni method for 3 tests
```

```r
# Balanced data
glm_time_bal <- glm(Balanced_Training_Time~Model+Wav_Selec_Reduction,
                family =gaussian(), data = df_time )
print(summary(glm_time_bal))
```

```
Call:
glm(formula = Balanced_Training_Time ~ Model + Wav_Selec_Reduction,
    family = gaussian(), data = df_time)

Coefficients:
                                    Estimate Std. Error t value Pr(>|t|)
(Intercept)                           614.14      33.99  18.069  < 2e-16 ***
Modelk-NN                            -661.46      41.63 -15.890  < 2e-16 ***
ModelLDA                             -687.99      41.63 -16.527  < 2e-16 ***
ModelNNET                             184.41      41.63   4.430 2.89e-05 ***
ModelRF                              -619.86      41.63 -14.890  < 2e-16 ***
ModelSVM                             -626.14      41.63 -15.041  < 2e-16 ***
Wav_Selec_ReductionFull _Spectra_PCA  116.72      29.44   3.965 0.000156 ***
Wav_Selec_ReductionRFE_GA_PCA         113.75      29.44   3.865 0.000222 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 12996.63)

    Null deviance: 12615731  on 89  degrees of freedom
Residual deviance:  1065724  on 82  degrees of freedom
AIC: 1117.6

Number of Fisher Scoring iterations: 2
```

```r
# Check pairwise comparisons for the SMOTE model

# Pairwise comparisons for Model
emmeans_model_bal_time <- emmeans(glm_time_bal, ~ Model)
pairs(emmeans_model_bal_time, adjust = "bonferroni")  # Bonferroni correction
```

```
 contrast          estimate   SE df t.ratio p.value
 Ensemble - (k-NN)   661.46 41.6 82  15.890  <.0001
 Ensemble - LDA      687.99 41.6 82  16.527  <.0001
 Ensemble - NNET    -184.41 41.6 82  -4.430  0.0004
 Ensemble - RF       619.86 41.6 82  14.890  <.0001
 Ensemble - SVM      626.14 41.6 82  15.041  <.0001
 (k-NN) - LDA         26.53 41.6 82   0.637  1.0000
```

```
 (k-NN) - NNET      -845.87 41.6 82 -20.320  <.0001
 (k-NN) - RF         -41.60 41.6 82  -0.999  1.0000
 (k-NN) - SVM        -35.32 41.6 82  -0.848  1.0000
 LDA - NNET         -872.39 41.6 82 -20.957  <.0001
 LDA - RF            -68.13 41.6 82  -1.637  1.0000
 LDA - SVM           -61.85 41.6 82  -1.486  1.0000
 NNET - RF           804.27 41.6 82  19.320  <.0001
 NNET - SVM          810.55 41.6 82  19.471  <.0001
 RF - SVM              6.28 41.6 82   0.151  1.0000

Results are averaged over the levels of: Wav_Selec_Reduction
P value adjustment: bonferroni method for 15 tests
```

```r
# Pairwise comparisons for Wav_Selec_Reduction
emmeans_wave_bal_time <- emmeans(glm_time_bal, ~ Wav_Selec_Reduction)
pairs(emmeans_wave_bal_time, adjust = "bonferroni")
```

```
 contrast                       estimate   SE df t.ratio p.value
 Boruta_PCA - Full _Spectra_PCA  -116.72 29.4 82  -3.965  0.0005
 Boruta_PCA - RFE_GA_PCA         -113.75 29.4 82  -3.865  0.0007
 Full _Spectra_PCA - RFE_GA_PCA     2.96 29.4 82   0.101  1.0000

Results are averaged over the levels of: Model
P value adjustment: bonferroni method for 3 tests
```

```r
df_time %>% select(Wav_Selec_Reduction,Imbalanced_Training_Time,Balanced_Training_Time) %>%
  group_by(Wav_Selec_Reduction) %>% summarise(
    Mean_Training_Time_Imbalanced = mean(Imbalanced_Training_Time),
    Mean_Training_Time_Balanced = mean(Balanced_Training_Time)) %>%
  arrange(desc(Mean_Training_Time_Balanced))
```

```
# A tibble: 3 × 3
  Wav_Selec_Reduction Mean_Training_Time_Imbalanced Mean_Training_Time_Balanced
  <fct>                                     <dbl>                       <dbl>
1 Full _Spectra_PCA                          171.                        329.
2 RFE_GA_PCA                                 179.                        326.
3 Boruta_PCA                                 164.                        212.
```

Part 2: Key Takeaways

Model Training Time:

Ensemble Models: Have the longest training times for both unbalanced and balanced data, with further increases observed under balanced data due to the larger data set size.

NNET: Significantly slower than simpler models (e.g., k-NN, LDA, RF, SVM) but faster than Ensemble models in both data sets.

Simpler Models (k-NN, LDA, RF, SVM): Maintain comparable and efficient training times in both data sets, making them ideal for time-sensitive applications

Variable Selection:

Boruta_PCA: Shows the most significant reduction in training time, especially for balanced data, making it the preferred method for large-scale data sets.

Full _Spectra_PCA and RFE_GA_PCA: Perform similarly in training time but are less efficient compared to Boruta_PCA when data sets are balanced.

Practical Recommendations

Use Boruta_PCA for Variable Selection:

- Variable selection with Boruta and subsequent reduction dimension by PCA is appropriate as significantly reduces training time without compromising performance.

Balance Between Performance and Training Time:

- Ensemble models and NNET can be utilized in discrimination tasks where time is not a constraint.
- Time sensitive analysis should opt for k-NN, LDA, LDA, RF or SVM as they offer shorter training times while maintaining competitive performance.

SMOTE

- While SMOTE improves model performance (as observed from the balanced accuracy), it increases training time for complex models. There should be a consideration between accuracy and computational efficiency.