test

# Load necessary libraries  
library(tidyverse)

Warning: package 'ggplot2' was built under R version 4.3.3

Warning: package 'tidyr' was built under R version 4.3.3

Warning: package 'dplyr' was built under R version 4.3.2

Warning: package 'stringr' was built under R version 4.3.2

Warning: package 'lubridate' was built under R version 4.3.2

── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
✔ dplyr 1.1.4 ✔ readr 2.1.4  
✔ forcats 1.0.0 ✔ stringr 1.5.1  
✔ ggplot2 3.5.1 ✔ tibble 3.2.1  
✔ lubridate 1.9.3 ✔ tidyr 1.3.1  
✔ purrr 1.0.2   
── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
✖ dplyr::filter() masks stats::filter()  
✖ dplyr::lag() masks stats::lag()  
ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

library(ggplot2)  
library(knitr)  
library(dplyr)  
library(here)

here() starts at C:/Users/Leonel/Desktop/MSDA/MS Data Analytics/Current Class/DA 6833/Practicum 2 Github/leonelsalazar-P2-portfolio

library(caret) # For machine learning models

Warning: package 'caret' was built under R version 4.3.3

Loading required package: lattice

Warning: package 'lattice' was built under R version 4.3.3

Attaching package: 'caret'  
  
The following object is masked from 'package:purrr':  
  
 lift

library(rpart) # For decision trees  
library(randomForest) # For random forest

Warning: package 'randomForest' was built under R version 4.3.3

randomForest 4.7-1.1  
Type rfNews() to see new features/changes/bug fixes.  
  
Attaching package: 'randomForest'  
  
The following object is masked from 'package:dplyr':  
  
 combine  
  
The following object is masked from 'package:ggplot2':  
  
 margin

library(gbm) # For gradient boosting

Warning: package 'gbm' was built under R version 4.3.3

Loaded gbm 2.2.2  
This version of gbm is no longer under development. Consider transitioning to gbm3, https://github.com/gbm-developers/gbm3

library(e1071) # For SVM

Warning: package 'e1071' was built under R version 4.3.3

# Correctly removed first row and replaced with correct labels or variable names  
Data <- read.csv("C:/Users/Leonel/Desktop/MSDA/MS Data Analytics/Current Class/DA 6833/Practicum 2 Github/leonelsalazar-P2-portfolio/tidytuesday-exercise/finalists.csv", header = TRUE, na.strings = "?",  
stringsAsFactors = TRUE)

# Display the structure of the dataset  
str(Data)

'data.frame': 190 obs. of 6 variables:  
 $ Contestant : Factor w/ 190 levels "Jos\x8e \"Sway\" Penala",..: 107 105 147 179 158 45 164 6 96 67 ...  
 $ Birthday : Factor w/ 187 levels "1-Aug-93","1-Feb-94",..: 104 132 133 116 54 89 91 159 176 36 ...  
 $ Birthplace : Factor w/ 164 levels "Agoura Hills, California",..: 49 28 56 150 116 18 147 132 23 82 ...  
 $ Hometown : Factor w/ 97 levels "Amory, Mississippi",..: 19 29 65 4 26 65 65 89 65 65 ...  
 $ Description: Factor w/ 179 levels "As a child, Uch\x8e visited family in Nigeria and learned how to dance from his aunts and uncles. He had an epi"| \_\_truncated\_\_,..: 159 68 143 141 101 10 91 110 58 40 ...  
 $ Season : int 1 1 1 1 1 1 1 1 1 1 ...

# View the data  
view(Data)

# Select all columns except 4 and 5  
contestants\_data <- dplyr::select(Data, -c(4,5))

# Convert Birthday to date format and reformat  
contestants\_data <- contestants\_data %>%  
 mutate(Birthday = as.Date(Birthday, format = "%d-%b-%y")) %>%  
 mutate(Birthday = format(Birthday, "%d-%m-%y"))

# View the transformed data  
view(contestants\_data)

# Display the structure of the transformed data  
str(contestants\_data)

'data.frame': 190 obs. of 4 variables:  
 $ Contestant: Factor w/ 190 levels "Jos\x8e \"Sway\" Penala",..: 107 105 147 179 158 45 164 6 96 67 ...  
 $ Birthday : chr "24-04-82" "28-10-78" "28-09-78" "26-07-79" ...  
 $ Birthplace: Factor w/ 164 levels "Agoura Hills, California",..: 49 28 56 150 116 18 147 132 23 82 ...  
 $ Season : int 1 1 1 1 1 1 1 1 1 1 ...

# Clean the Contestant column  
contestants\_data$Contestant <- as.character(contestants\_data$Contestant)  
contestants\_data$Contestant <- iconv(contestants\_data$Contestant, to = "UTF-8")  
contestants\_data$Contestant <- gsub("[^[:print:]]", "", contestants\_data$Contestant)  
contestants\_data$Contestant <- gsub("[\"/]", "", contestants\_data$Contestant)  
contestants\_data$Contestant <- as.factor(contestants\_data$Contestant)  
contestants\_data$Birthday <- as.factor(contestants\_data$Birthday)  
  
# Remove rows with NA values in Contestant, Birthday, and Season columns  
contestants\_data\_clean <- contestants\_data %>%  
 drop\_na(Contestant, Birthday, Season)

# Display the structure of the cleaned data  
str(contestants\_data\_clean)

'data.frame': 187 obs. of 4 variables:  
 $ Contestant: Factor w/ 188 levels "Aaron Kelly",..: 106 104 146 178 157 44 163 5 95 66 ...  
 $ Birthday : Factor w/ 186 levels "01-02-94","01-05-89",..: 144 171 170 154 102 127 132 27 42 84 ...  
 $ Birthplace: Factor w/ 164 levels "Agoura Hills, California",..: 49 28 56 150 116 18 147 132 23 82 ...  
 $ Season : int 1 1 1 1 1 1 1 1 1 1 ...

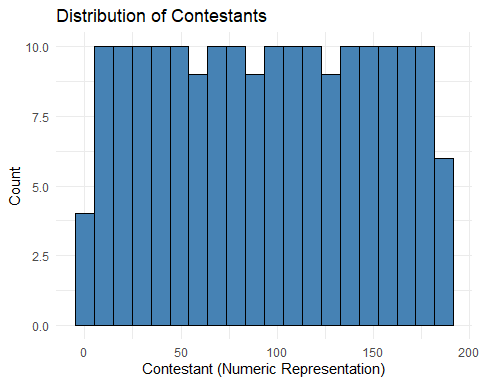
# View the cleaned data  
view(contestants\_data\_clean)

# Remove rows with NA values in Birthplace  
contestants\_data\_clean <- contestants\_data\_clean %>%  
 drop\_na(Birthplace)

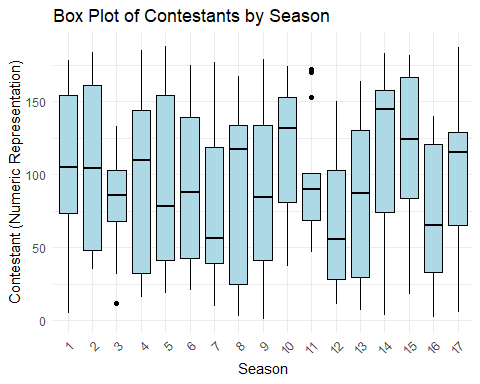
# View the cleaned data  
view(contestants\_data\_clean)

# Save the cleaned data to a CSV file  
write.csv(contestants\_data\_clean, "C:/Users/Leonel/Desktop/MSDA/MS Data Analytics/Current Class/DA 6833/Practicum 2 Github/leonelsalazar-P2-portfolio/tidytuesday-exercise/contestants\_data\_clean.csv", row.names = FALSE)

# Load ggplot2 for plotting  
library(ggplot2)  
  
# Convert Contestant to numeric for plotting  
contestants\_data\_clean$Contestant <- as.numeric(contestants\_data\_clean$Contestant)  
  
# Create histogram of Contestant numbers  
ggplot(contestants\_data\_clean, aes(x = Contestant)) +  
 geom\_histogram(bins = 20, fill = "steelblue", color = "black") +  
 labs(title = "Distribution of Contestants",  
 x = "Contestant (Numeric Representation)",  
 y = "Count") +  
 theme\_minimal()



# Add numeric representation of Contestant  
contestants\_data\_clean$Contestant\_Num <- as.numeric(contestants\_data\_clean$Contestant)  
  
# Create box plot of Contestants by Season  
ggplot(contestants\_data\_clean, aes(x = factor(Season), y = Contestant\_Num)) +  
 geom\_boxplot(fill = "lightblue", color = "black") +  
 labs(title = "Box Plot of Contestants by Season",  
 x = "Season",  
 y = "Contestant (Numeric Representation)") +  
 theme\_minimal() +  
 theme(axis.text.x = element\_text(angle = 45, hjust = 1))



# Split the data into training and testing sets  
set.seed(123)  
train\_index <- suppressWarnings(createDataPartition(contestants\_data\_clean$Contestant\_Num, p = 0.8, list = FALSE))  
train\_data <- contestants\_data\_clean[train\_index, ]  
test\_data <- contestants\_data\_clean[-train\_index, ]  
  
# Linear Regression  
lm\_model <- suppressWarnings(train(Contestant\_Num ~ ., data = train\_data, method = "lm"))  
lm\_pred <- suppressWarnings(predict(lm\_model, test\_data))  
lm\_rmse <- RMSE(lm\_pred, test\_data$Contestant\_Num)  
  
# Decision Tree  
tree\_model <- suppressWarnings(train(Contestant\_Num ~ ., data = train\_data, method = "rpart"))  
tree\_pred <- suppressWarnings(predict(tree\_model, test\_data))  
tree\_rmse <- RMSE(tree\_pred, test\_data$Contestant\_Num)  
  
# Random Forest  
rf\_model <- suppressWarnings(train(Contestant\_Num ~ ., data = train\_data, method = "rf"))  
rf\_pred <- suppressWarnings(predict(rf\_model, test\_data))  
rf\_rmse <- RMSE(rf\_pred, test\_data$Contestant\_Num)  
  
# Gradient Boosting  
gbm\_model <- suppressWarnings(train(Contestant\_Num ~ ., data = train\_data, method = "gbm", verbose = FALSE))  
gbm\_pred <- suppressWarnings(predict(gbm\_model, test\_data))  
gbm\_rmse <- RMSE(gbm\_pred, test\_data$Contestant\_Num)  
  
# Support Vector Machine  
svm\_model <- suppressWarnings(train(Contestant\_Num ~ ., data = train\_data, method = "svmRadial"))  
svm\_pred <- suppressWarnings(predict(svm\_model, test\_data))  
svm\_rmse <- RMSE(svm\_pred, test\_data$Contestant\_Num)  
  
# Combine RMSE results into a dataframe for comparison  
results <- suppressWarnings(data.frame(  
 Model = c("Linear Regression", "Decision Tree", "Random Forest", "Gradient Boosting", "Support Vector Machine"),  
 RMSE = c(lm\_rmse, tree\_rmse, rf\_rmse, gbm\_rmse, svm\_rmse)  
))  
  
print(results)

Model RMSE  
1 Linear Regression 0.0000000  
2 Decision Tree 23.2707546  
3 Random Forest 0.8732771  
4 Gradient Boosting 3.2988005  
5 Support Vector Machine 42.7373149

| Model | RMSE |
| --- | --- |
| Linear Regression | 0.000000 |
| Decision Tree | 23.2707546 |
| Random Forest | 0.8732771 |
| Gradient Boosting | 3.2988005 |
| Support Vector Machine | 42.7373149 |

### Analysis

1. **Linear Regression**:
   * **RMSE: 0.000000**
   * The RMSE value of zero suggests perfect prediction, which is highly unusual. This might indicate an issue with the model or the way it was applied. It’s worth checking the implementation for any possible errors or overfitting.
2. **Decision Tree**:
   * **RMSE: 23.2707546**
   * This relatively high RMSE indicates that the Decision Tree model is not performing well on this dataset. Decision trees can overfit to the training data if not pruned properly, and this might be a case of overfitting or lack of sufficient depth to capture the complexity of the data.
3. **Random Forest**:
   * **RMSE: 0.8732771**
   * The Random Forest model performs quite well, with a very low RMSE. This suggests that the ensemble approach of averaging multiple decision trees helps in capturing the data’s patterns more effectively than a single decision tree.
4. **Gradient Boosting**:
   * **RMSE: 3.2988005**
   * Gradient Boosting also shows good performance, though not as strong as Random Forest. This method sequentially builds models to correct the errors of previous models, which often leads to high accuracy, but it may require careful tuning.
5. **Support Vector Machine (SVM)**:
   * **RMSE: 42.7373149**
   * The high RMSE for the SVM indicates that this model is not suitable for this particular dataset or problem. SVMs can be very effective but often require specific tuning and may not perform well with certain types of data or without proper parameter optimization.

### Conclusion

* **Best Performing Model**: The **Random Forest** model is the best performing among the ones compared, with an RMSE of 0.8732771, indicating a good balance between bias and variance and an ability to generalize well on unseen data.
* **Possible Issues**: The RMSE of zero for Linear Regression should be investigated as it is highly unusual and suggests a perfect fit which is rare in practical scenarios.
* **Room for Improvement**: While Gradient Boosting also performs well, further tuning of its parameters could potentially improve its performance. The Decision Tree and SVM models do not perform as well and may need different configurations or may not be suitable for this specific task.

The next steps could involve: 1. Investigating the Linear Regression model for any anomalies. 2. Fine-tuning the Random Forest and Gradient Boosting models further. 3. Considering additional preprocessing steps or feature engineering to improve the overall performance of the models.