```
---
title: "Andrew Harper Assignment 1"
output: pdf_document
date: "2024-11-19"
---

`# Load necessary libraries
library(tidyverse)
library(caret)
library(GGally)
library(car)
library(lmtest)
library(knitr)
library(corrplot)
library(ggplot2)

# Load the dataset
data <- read.csv("Capital Bike Sharing data by hour (1).csv")

# Summary statistics
summary_stats <- data %>%
  summarise(across(
    where(is.numeric),
    list(mean = mean, median = median, sd = sd)
  )) %>%
  pivot_longer(
    everything(),
    names_to = c("variable", "statistic"),
    names_sep = "_"
  ) %>%
  pivot_wider(
    names_from = "statistic",
    values_from = "value"
  )

# Print the summary statistics as a table
kable(summary_stats, caption = "Summary Statistics for Numeric Variables")

# Identify potential issues (missing values)
missing_values <- sum(is.na(data))
cat("Number of missing values:", missing_values, "\n")
if (missing_values > 0) {
  # Handle missing values (imputation or removal)
  data <- na.omit(data) # Drop rows with missing values (or use imputation if needed)
}

# Convert relevant columns to factors
data$season <- factor(data$season, levels = c(1, 2, 3, 4), labels = c("Winter", "Spring",
"Summer", "Fall"))
data$holiday <- factor(data$holiday, levels = c(0, 1), labels = c("No", "Yes"))
data$workingday <- factor(data$workingday, levels = c(0, 1), labels = c("No", "Yes"))
data$weathersit <- factor(data$weathersit, levels = c(1, 2, 3, 4), labels = c("Clear",
"Mist", "Light Rain/Snow", "Heavy Rain/Snow"))
data$yr <- factor(data$yr, levels = c(0, 1), labels = c("2011", "2012"))

# Transform target variable using log transformation
data$log_count <- log1p(data$cnt)

# Scale numerical features for consistency
scaled_data <- data %>%
  mutate(across(c(temp, atemp, hum, windspeed), scale))

# Check correlations between independent variables and dependent variable
cor_matrix <- cor(data %>% select(where(is.numeric)), use = "complete.obs")
```

```
cat("Correlation Matrix:\n")
print(cor_matrix)

# Visualize correlation heatmap
corrplot(cor_matrix, method = "color", type = "lower", diag = FALSE, tl.col = "black",
tl.srt = 45)

# Multicollinearity check (Variance Inflation Factor)
# Fit a temporary model to check for VIF values
temp_model <- lm(log_count ~ season + yr + holiday + workingday + weathersit + temp +
atemp + hum + windspeed, data = data)
vif_values <- vif(temp_model)
cat("VIF Values:\n")
print(vif_values)

# Scatter plot matrix for visualization
ggpairs(data %>% select(temp,atemp,hum,windspeed,cnt),
                       title = "Scatter Plot Matrix of Selected Variables")


# Create a boxplot to compare bike rentals across seasons
ggplot(data, aes(x = season, y = cnt, fill = season)) +
  geom_boxplot() +  # Create boxplots for each season
  labs(title = "Total Rentals by Season",
       x = "Season",
       y = "Rental Count") +
  theme_minimal()
# This visualization highlights seasonal variability in bike rentals, showing summer
demand is highest.

# Create a line plot to show average hourly rentals by season
ggplot(data, aes(x = hr, y = cnt, color = season, group = season)) +
  geom_line(stat = "summary", fun = mean) +  # Plot average rentals (mean) for each hour,
grouped by season
  labs(title = "Average Rentals by Hour and Season",
       x = "Hour",
       y = "Average Rentals") +
  theme_minimal()
# This plot shows commuting patterns with peaks in the morning and evening hours.

# Create a scatterplot to examine the relationship between temperature and bike rentals
ggplot(data, aes(x = temp, y = cnt)) +
  geom_point(alpha = 0.3) +  # Plot individual data points with transparency for better
visualization
  geom_smooth(method = "loess", col = "blue") +  # Add a smoothed trend line using LOESS
regression
  labs(title = "Bike Rentals vs Temperature",
       x = "Temperature (Normalized)",
       y = "Rental Count") +
  theme_minimal()
# This visualization shows a strong positive relationship between temperature and rentals,
# with a peak around 25°C and a decline beyond that.


# Create a histogram to display the distribution of total rentals
ggplot(data, aes(x = cnt)) +
  geom_histogram(bins = 30, fill = "blue", color = "black", alpha = 0.7) +  # Add a
histogram with 30 bins
  labs(title = "Distribution of Total Rentals",
       x = "Rental Count",
       y = "Frequency") +
  theme_minimal()
# The histogram highlights the right-skewed distribution of bike rentals,
# which supports the use of a log transformation for normalization.
```

```r
# Split data into training (70%), validation (15%), and test (15%) sets
set.seed(123)
trainIndex <- createDataPartition(data$cnt, p = 0.7, list = FALSE)
train_data <- data[trainIndex, ]

temp_data <- data[-trainIndex, ]
valIndex <- createDataPartition(temp_data$cnt, p = 0.5, list = FALSE)
validation_data <- temp_data[valIndex, ]
test_data <- temp_data[-valIndex, ]

# Initial linear model
initial_model <- lm(log_count ~ season + yr + holiday + workingday + weathersit + temp +
atemp + hum + windspeed, data = train_data)

# Check model summary
summary(initial_model)

# Diagnostics and Residual Plots
par(mfrow = c(2, 2))
plot(initial_model)

# VIF for multicollinearity
cat("VIF values (Initial Model):\n")
print(vif(initial_model))

# Breusch-Pagan test for heteroscedasticity
bp_test <- bptest(initial_model)
cat("Breusch-Pagan Test p-value (Initial Model):", bp_test$p.value, "\n")

# Define a function to calculate metrics
calculate_metrics <- function(actual, predicted) {
  mse <- mean((actual - predicted)^2)
  rmse <- sqrt(mse)
  mae <- mean(abs(actual - predicted))
  r_squared <- cor(actual, predicted)^2
  return(data.frame(MSE = mse, RMSE = rmse, MAE = mae, R_squared = r_squared))
}

# Training set
train_predictions <- predict(initial_model, train_data)
train_metrics <- calculate_metrics(train_data$log_count, train_predictions)
cat("Training Metrics:\n")
print(train_metrics)

# Validation set
validation_predictions <- predict(initial_model, validation_data)
validation_metrics <- calculate_metrics(validation_data$log_count, validation_predictions)
cat("Validation Metrics:\n")
print(validation_metrics)

# Test set
test_predictions <- predict(initial_model, test_data)
test_metrics <- calculate_metrics(test_data$log_count, test_predictions)
cat("Test Metrics:\n")
print(test_metrics)

# Refined model with interactions and polynomial terms
refined_model <- lm(log_count ~ season * temp + yr + holiday + workingday + weathersit +
temp + I(temp^2) + hum + windspeed, data = train_data)

# Evaluate refined model
summary(refined_model)
```

```r
# Predictions on all datasets
train_predictions_refined <- predict(refined_model, train_data)
validation_predictions_refined <- predict(refined_model, validation_data)
test_predictions_refined <- predict(refined_model, test_data)

# Metrics for the refined model
train_metrics_refined <- calculate_metrics(train_data$log_count,
train_predictions_refined)
validation_metrics_refined <- calculate_metrics(validation_data$log_count,
validation_predictions_refined)
test_metrics_refined <- calculate_metrics(test_data$log_count, test_predictions_refined)

cat("Refined Model Metrics (Training):\n")
print(train_metrics_refined)
cat("Refined Model Metrics (Validation):\n")
print(validation_metrics_refined)
cat("Refined Model Metrics (Test):\n")
print(test_metrics_refined)

# Predictions on the test data using the refined model
final_test_predictions <- predict(refined_model, test_data)

# Back-transform the predictions (log-transformed to original scale)
final_test_predictions_original <- expm1(final_test_predictions)
actual_test_values <- test_data$cnt  # Original scale

# Calculate evaluation metrics
final_test_metrics <- calculate_metrics(actual_test_values,
final_test_predictions_original)

# Display the metrics
cat("Final Evaluation Metrics on Test Data:\n")
print(final_test_metrics)

# Visualize Predictions vs Actuals
ggplot(data.frame(Actual = actual_test_values, Predicted =
final_test_predictions_original), aes(x = Actual, y = Predicted)) +
  geom_point(color = "blue") +
  geom_abline(slope = 1, intercept = 0, color = "red", linetype = "dashed") +
  labs(title = "Actual vs Predicted Rentals (Test Set - Refined Model)", x = "Actual
Rentals", y = "Predicted Rentals") +
  theme_minimal()



## ____________________________________________________________
## Refining the model

# Create time-of-day bins for the `hr` variable
train_data$time_of_day <- cut(
  train_data$hr,
  breaks = c(-1, 6, 12, 18, 23),
  labels = c("Night", "Morning", "Afternoon", "Evening")
)
validation_data$time_of_day <- cut(
  validation_data$hr,
  breaks = c(-1, 6, 12, 18, 23),
  labels = c("Night", "Morning", "Afternoon", "Evening")
)
test_data$time_of_day <- cut(
  test_data$hr,
  breaks = c(-1, 6, 12, 18, 23),
  labels = c("Night", "Morning", "Afternoon", "Evening")
)
```

```
# Enhanced model with additional features and interactions
final_enhanced_model <- lm(
  log_count ~ season * temp + season * hum + workingday * time_of_day + yr +
    holiday + weathersit + temp + I(temp^2) + hum + I(hum^2) + windspeed + I(windspeed^2),
  data = train_data
)

# Summary of the enhanced model
summary(final_enhanced_model)


# Calculate Cook's distance for the final enhanced model
cooks_dist <- cooks.distance(final_enhanced_model)

# Plot Cook's distance to visualize influential observations
plot(cooks_dist, type = "h", main = "Cook's Distance for Final Enhanced Model", ylab =
"Cook's Distance")
abline(h = 4 / nrow(train_data), col = "red", lty = 2)  # Threshold line

# Identify and remove influential points
threshold <- 4 / nrow(train_data)
influential_points <- which(cooks_dist > threshold)
cat("Number of influential points in final enhanced model:", length(influential_points),
"\n")
cat("Indices of influential points in final enhanced model:", influential_points, "\n")

# Remove influential points from training data
train_data_filtered <- train_data[-influential_points, ]

# Refit the model without influential points
final_enhanced_model_filtered <- lm(
  log_count ~ season * temp + season * hum + workingday * time_of_day + yr +
    holiday + weathersit + temp + I(temp^2) + hum + I(hum^2) + windspeed + I(windspeed^2),
  data = train_data_filtered
)

# Summary of the refitted model
summary(final_enhanced_model_filtered)

# Perform Durbin-Watson test on residuals of the refitted model
dw_test <- dwtest(final_enhanced_model_filtered)
cat("Durbin-Watson Test Statistic (Filtered Model):", dw_test$statistic, "\n")
cat("p-value for Durbin-Watson Test (Filtered Model):", dw_test$p.value, "\n")

# Interpret Durbin-Watson test results
if (dw_test$p.value < 0.05) {
  cat("Autocorrelation detected in residuals of the filtered model.\n")
} else {
  cat("No significant autocorrelation in residuals of the filtered model.\n")
}

# Diagnostics and Residual Plots for the final filtered model
par(mfrow = c(2, 2))
plot(final_enhanced_model_filtered)

# Re-calculate evaluation metrics with filtered data
train_predictions_final_filtered <- predict(final_enhanced_model_filtered,
train_data_filtered)
validation_predictions_final <- predict(final_enhanced_model_filtered, validation_data)  #
Use original validation set
test_predictions_final <- predict(final_enhanced_model_filtered, test_data)  # Use
original test set

# Metrics for the final enhanced filtered model
```

```
train_metrics_final_filtered <- calculate_metrics(train_data_filtered$log_count,
train_predictions_final_filtered)
validation_metrics_final <- calculate_metrics(validation_data$log_count,
validation_predictions_final)
test_metrics_final <- calculate_metrics(test_data$log_count, test_predictions_final)

filtered_model_metrics <- bind_rows(
  Training = train_metrics_final_filtered,
  Validation = validation_metrics_final,
  Test = test_metrics_final,
  .id = "Dataset"
)

# Print the metrics as a table
kable(filtered_model_metrics, caption = "Filtered Model Metrics for Training, Validation,
and Test Sets")

# Back-transform predictions for the test set to original scale
final_test_predictions_original <- expm1(test_predictions_final)
actual_test_values <- test_data$cnt

# Calculate final metrics for test data on the original scale
final_test_metrics_original <- calculate_metrics(actual_test_values,
final_test_predictions_original)

cat("Final Evaluation Metrics for Filtered Enhanced Model (Test Data):\n")
print(final_test_metrics_original)

# Visualization of Actual vs Predicted for the filtered model
ggplot(data.frame(Actual = actual_test_values, Predicted =
final_test_predictions_original),
       aes(x = Actual, y = Predicted)) +
  geom_point(color = "blue", alpha = 0.5) +
  geom_abline(slope = 1, intercept = 0, color = "red", linetype = "dashed") +
  labs(title = "Filtered Enhanced Model: Actual vs Predicted Rentals (Test Set)",
       x = "Actual Rentals",
       y = "Predicted Rentals") +
  theme_minimal()
```