DS5110 Homework 5

Ameya Santosh Gidh

2024-03-24

Part A

Problem 1

Flash Paper - Param Rajesh Joshi

Data source - https://www.kaggle.com/c/titanic/data

```r
suppressPackageStartupMessages(library(tidyverse))
library(readr)
library(dplyr)
library(ggplot2)
library(tidyverse)

if (!requireNamespace("tidyverse", quietly = TRUE)) {
  install.packages("tidyverse")
}
```

Preprocessing:

The dataset comprises 891 observations and 12 attributes, encompassing details such as passenger class, sex, age, number of siblings/spouses aboard, number of parents/children aboard, fare, and embarkation port. Upon initial examination, missing values were detected in the "Age", "Cabin", and "Embarked" columns. To rectify this, I opted to eliminate the "Cabin" column entirely and filled in missing values in the "Age" column with the median age. For the "Embarked" column, I replaced missing values with the most frequently occurring value. Additionally, I converted categorical variables into factors to facilitate analysis.

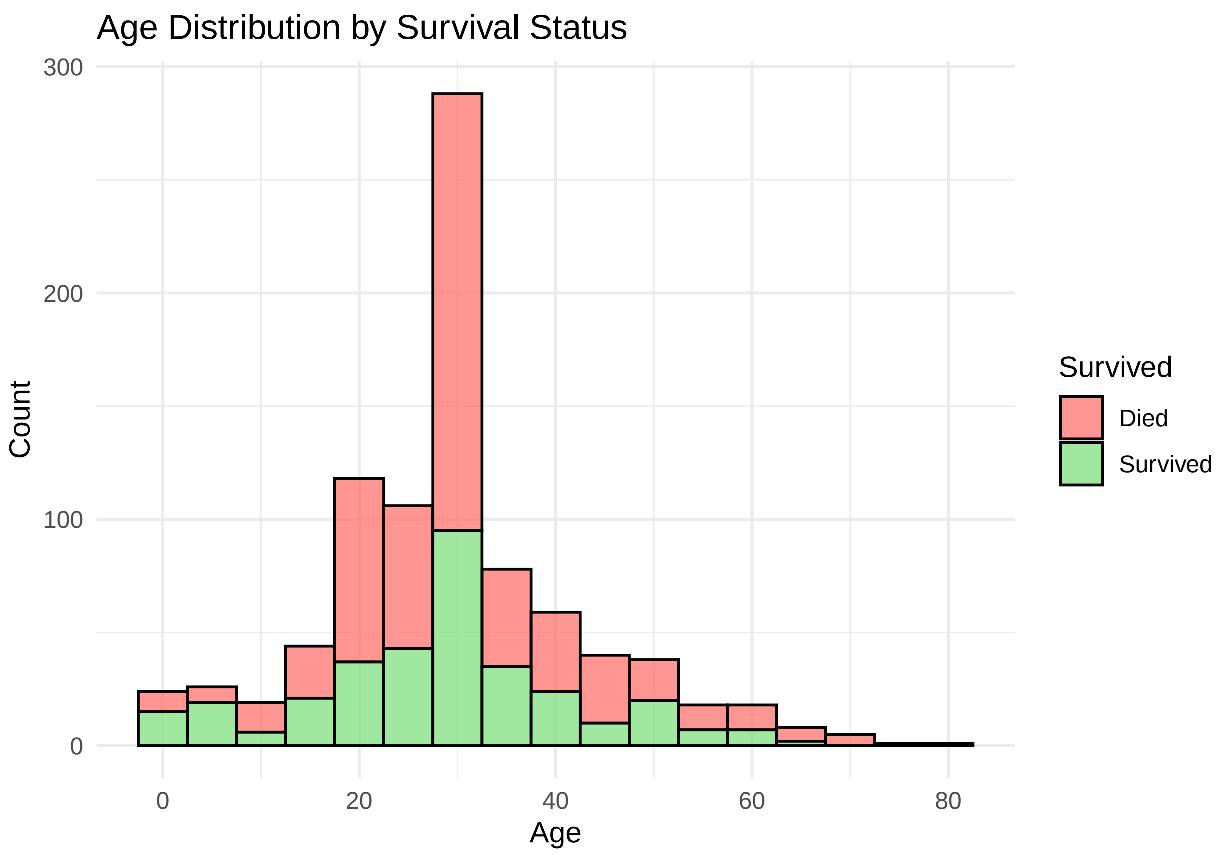
- Begin by importing the dataset.
- Replace any instances of '?' with NA to denote missing values.
- Examine for duplicate rows and remove them if they exist.
- Convert categorical variables into factors as required.
- Investigate the data for outliers or irregularities.
- Let's start by executing these preprocessing tasks and analyzing the dataset.

First, load the dataset. Then, replace any occurrences of '?' with NA to represent missing values. Check for duplicate rows and eliminate them if found. Remove Cabin Column. Convert categorical variables into factors if needed. Next, explore the data to identify any outliers or anomalies. Let's begin by implementing these preprocessing steps and examining the data.

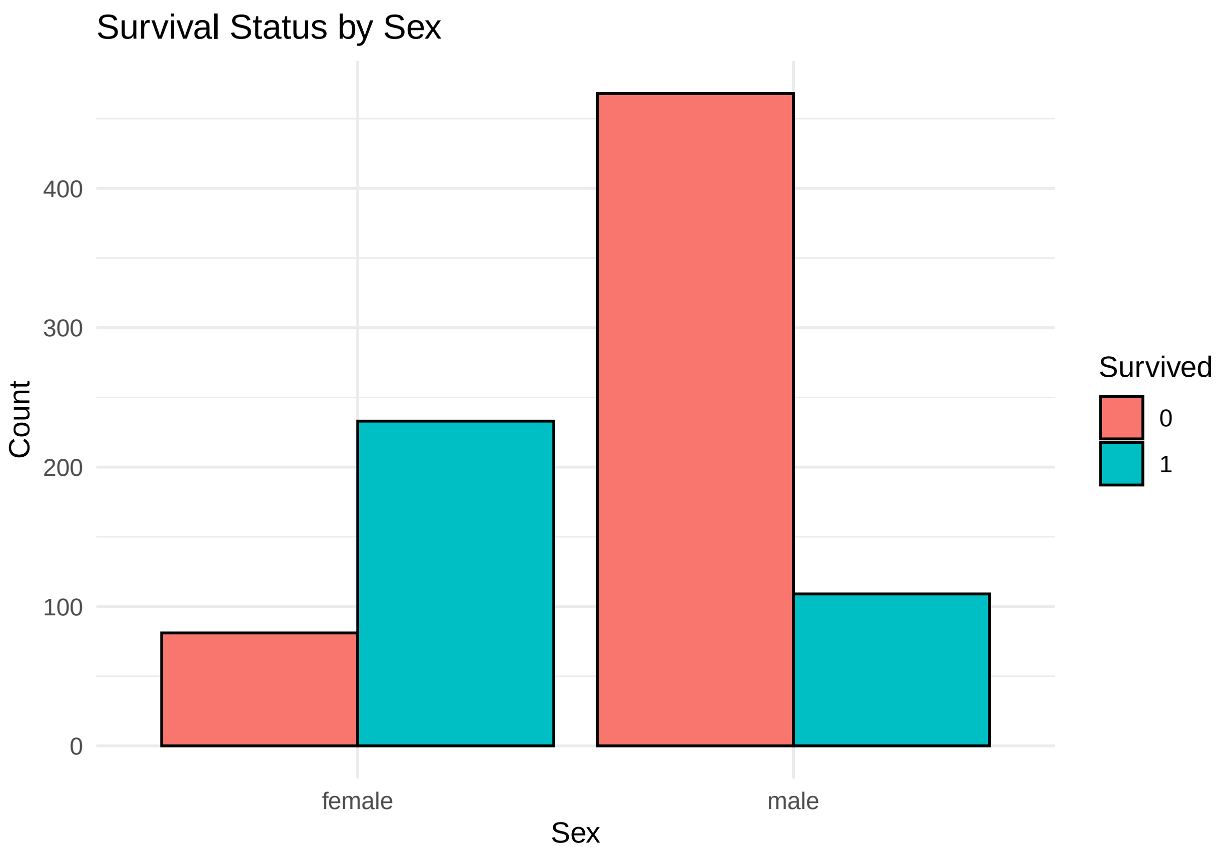
```r
if (!requireNamespace("tidyverse", quietly = TRUE)) {
  install.packages("tidyverse")
}
titanic_data <- read.csv("train.csv", header = TRUE)

# Check for missing values in the Titanic dataset
# Using colSums() to sum up missing values for each column and is.na() to check for missing values
missing_values <- colSums(is.na(titanic_data))

# Filtering out columns with missing values (those with more than 0 missing values)
missing_values <- missing_values[missing_values > 0]

# Remove the "Cabin" column from the Titanic dataset
titanic_data <- titanic_data[, !(names(titanic_data) %in% c("Cabin"))]

# Impute missing values in the "Age" column with the median age
median_age <- median(titanic_data$Age, na.rm = TRUE)  # Calculate the median age
titanic_data$Age[is.na(titanic_data$Age)] <- median_age  # Replace missing values with the median age

# Impute missing values in the "Embarked" column with the most common value
most_common_embarked <- names(sort(table(titanic_data$Embarked), decreasing = TRUE))[1]  # Find the mos
titanic_data$Embarked[is.na(titanic_data$Embarked)] <- most_common_embarked  # Replace missing values w

# Convert categorical variables to factors
titanic_data$Survived <- as.factor(titanic_data$Survived)  # Convert "Survived" to factor
titanic_data$Pclass <- as.factor(titanic_data$Pclass)  # Convert "Pclass" to factor
titanic_data$Sex <- as.factor(titanic_data$Sex)  # Convert "Sex" to factor
```

Problem 2

```r
# Visualize age distribution by survival status
# Creating a histogram to visualize the distribution of ages by survival status
# Using ggplot2 library for plotting
# Plotting the histogram
ggplot(titanic_data, aes(x = Age, fill = Survived)) +
  geom_histogram(binwidth = 5, color = "black", alpha = 0.7) +  # Setting bin width, color, and transpa
  labs(title = "Age Distribution by Survival Status",  # Adding title and axis labels
       x = "Age",
       y = "Count") +
  scale_fill_manual(values = c("#FF6961", "#77DD77"),  # Changing color theme for survival status
                    labels = c("Died", "Survived")) +  # Updating legend labels
  theme_minimal()  # Applying a minimal theme for aesthetics
```



```r
# Create a bar plot depicting survival status based on sex
library(ggplot2)

# Assuming titanic_data contains the necessary data

survival_by_sex <- ggplot(titanic_data, aes(x = Sex, fill = Survived)) +  # Define plot aesthetics
  geom_bar(position = "dodge", color = "black") +  # Add bars with dodging for each survival status
  labs(title = "Survival Status by Sex",  # Add plot title
       x = "Sex",  # Label x-axis as "Sex"
       y = "Count",  # Label y-axis as "Count"
       fill = "Survived") +  # Label fill legend as "Survived"
  theme_minimal()  # Apply minimal theme for the plot

# Display the plot
print(survival_by_sex)
```



Part B

Problem 3

```r
# Suppress package startup messages
suppressPackageStartupMessages(library(dplyr))
suppressPackageStartupMessages(library(tidyverse))

# Load necessary libraries
library(modelr)
library(ggplot2)

# Load data
load("37938-0001-Data.rda")
my_data <- da37938.0001

# Select relevant variables
my_data <- my_data %>%
  select(LIFESAT_I, SOCIALWB_I, NONAFFIRM_I, NONDISCLOSURE_I, HCTHREAT_I, KESSLER6_I, EVERYDAY_I)

# Remove missing values
my_data <- na.omit(my_data)

# Set seed for reproducibility
set.seed(2)
```

```
# Split data into training and testing sets
my_data_part <- resample_partition(my_data, p = c(train = 0.5, test = 0.5))
trainSet <- my_data[my_data_part$train$idx,]
testSet <- my_data[my_data_part$test$idx,]
```

```
# Extract predictors from the training set
predictors <- colnames(trainSet)[-1]

# Initialize an empty vector to store RMSE results
rmse_results <- c()

# Iterate over each predictor
for (predictor in predictors) {
  # Create a formula for linear regression
  formula <- as.formula(paste("LIFESAT_I ~", predictor))

  # Fit linear regression model
  model <- lm(formula, data = trainSet)

  # Make predictions on the test set
  predictions <- predict(model, newdata = testSet)

  # Calculate residuals
  residuals <- testSet$LIFESAT_I - predictions

  # Calculate root mean squared error (RMSE)
  rmse_value <- sqrt(mean(residuals^2))

  # Append RMSE value to results vector
  rmse_results <- append(rmse_results, rmse_value)
}

# Create a data frame to store predictors and corresponding RMSE values
rmse_df <- data.frame(Predictor = predictors, RMSE = rmse_results)

# Display the data frame
rmse_df
```

```
##         Predictor     RMSE
## 1      SOCIALWB_I 1.557210
## 2     NONAFFIRM_I 1.609596
## 3 NONDISCLOSURE_I 1.667608
## 4      HCTHREAT_I 1.603713
## 5      KESSLER6_I 1.339974
## 6      EVERYDAY_I 1.518621
```

The model employing the Mental Distress/Disorder predictor (KESSLER6_I) exhibits the lowest Root Mean Square Error (RMSE) value. Consequently, within these scales, it stands out as the most efficient individual predictor for gauging life satisfaction, as evidenced by the model's superior performance on the test set.

```
# Extract RMSE value for a single predictor
single_predictor <- rmse_df$RMSE[rmse_df$Predictor == "KESSLER6_I"]
```

Problem 4

Step 1:

```r
# Fit linear regression model using KESSLER6_I as predictor
model <- lm(LIFESAT_I ~ KESSLER6_I, data=trainSet)
```

```r
# Function to perform steps for model building
steps <- function(response, predictors, candidates, train, test) {
  # Create right-hand side of formulas
  rhs <- paste0(paste0(predictors, collapse="+"), "+", candidates)

  # Create formulas
  formulas <- lapply(paste0(response, "~", rhs), as.formula)

  # Calculate RMSE for each formula
  rmses <- sapply(formulas,
                  function(fm) rmse(lm(fm, data=train),
                                    data=test))

  # Assign names to RMSE values
  names(rmses) <- candidates

  # Identify the best RMSE
  attr(rmses, "best") <- rmses[which.min(rmses)]

  # Return RMSE values
  rmses
}
```

```r
# Define predictors and candidates
preds <- "KESSLER6_I"
cands <- c("SOCIALWB_I", "NONAFFIRM_I", "NONDISCLOSURE_I", "HCTHREAT_I",
           "EVERYDAY_I")

# Perform steps for model building
s1 <- steps("LIFESAT_I", preds, cands, trainSet, testSet)

# Update model with the best candidate
model <- c(model, attr(s1, "best"))

# Display RMSE values for each candidate
s1
```

```
##      SOCIALWB_I     NONAFFIRM_I NONDISCLOSURE_I      HCTHREAT_I      EVERYDAY_I
##        1.298163        1.339775        1.340157        1.334188        1.320025
## attr(,"best")
## SOCIALWB_I
##   1.298163
```

I start by utilizing the Mental Distress/Disorder (KESSLER6_I) predictor in the model, as identified in Problem 3. Subsequently, upon further analysis, I find that including Social Well-being (SOCIALWB_I) as an additional predictor results in the lowest RMSE value. Consequently, I incorporate Social Well-being into the model.

```r
# Fit linear regression model using KESSLER6_I and SOCIALWB_I as predictors
fit1 <- lm(LIFESAT_I ~ KESSLER6_I + SOCIALWB_I, data=trainSet)
```

```r
# Calculate RMSE for the model
double_predictor <- rmse(fit1, testSet)
```

Step 2:

```r
# Define predictors and candidates
preds <- c("KESSLER6_I", "SOCIALWB_I")
cands <- c("NONAFFIRM_I", "NONDISCLOSURE_I", "HCTHREAT_I", "EVERYDAY_I")

# Perform steps for model building
s1 <- steps("LIFESAT_I", preds, cands, trainSet, testSet)

# Update model with the best candidate
model <- c(model, attr(s1, "best"))

# Display RMSE values for each candidate
s1
```

```
##     NONAFFIRM_I NONDISCLOSURE_I      HCTHREAT_I      EVERYDAY_I
##        1.306177        1.291490        1.294142        1.280575
## attr(,"best")
## EVERYDAY_I
##   1.280575
```

Upon completing step 1, I found that including Everyday Discrimination (EVERYDAY_I) as the third predictor resulted in the lowest RMSE value. Consequently, I opted to integrate Everyday Discrimination into the model as the third predictor.

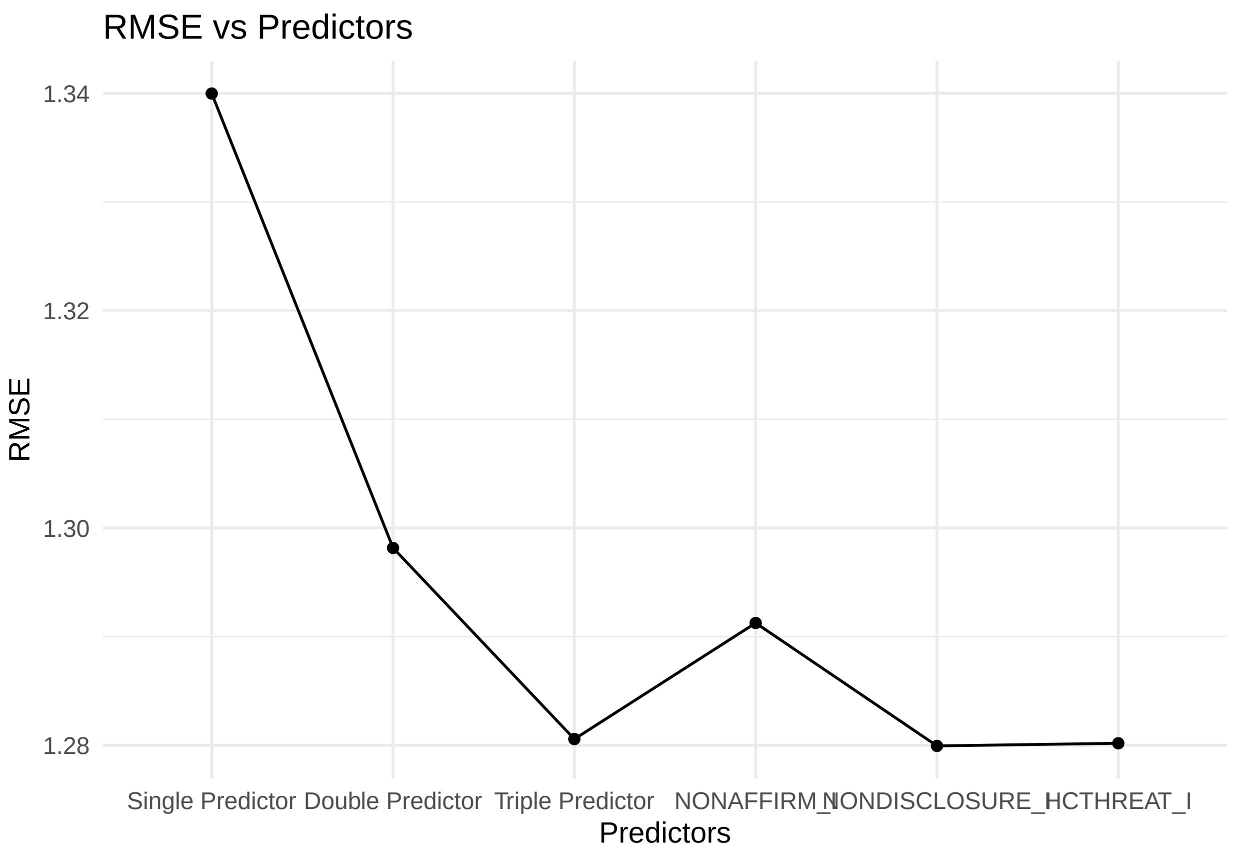
```r
# Fit linear regression model using KESSLER6_I, SOCIALWB_I, and EVERYDAY_I as predictors
fit2 <- lm(LIFESAT_I ~ KESSLER6_I + SOCIALWB_I + EVERYDAY_I, data=trainSet)

# Calculate RMSE for the model
triple_predictor <- rmse(fit2, testSet)
```

In conclusion, I found that Mental Distress/Disorder (KESSLER6_I), Social Well-being (SOCIALWB_I), and Everyday Discrimination (EVERYDAY_I) are the top three predictors among these scales for life satisfaction among transgender individuals.

Problem 5

```r
library(ggplot2)

# Create an empty data frame to store results
s1_df <- data.frame(Variable = character(), RMSE = numeric())

# Add RMSE values for single, double, and triple predictors
s1_df <- rbind(s1_df, data.frame(Variable = "Single Predictor", RMSE = single_predictor))
s1_df <- rbind(s1_df, data.frame(Variable = "Double Predictor", RMSE = double_predictor))
s1_df <- rbind(s1_df, data.frame(Variable = "Triple Predictor", RMSE = triple_predictor))

# Define predictors and candidates
preds <- c("KESSLER6_I", "SOCIALWB_I", "EVERYDAY_I")
cands <- c("NONAFFIRM_I", "NONDISCLOSURE_I", "HCTHREAT_I")

# Define function to perform steps for model building
```

```r
steps <- function(response, predictors, candidates, train, test) {
  rhs <- paste0(paste0(predictors, collapse="+"), "+", candidates)
  formulas <- lapply(paste0(response, "~", rhs), as.formula)
  rmses <- sapply(formulas,
                  function(fm) rmse(lm(fm, data=train),
                                    data=test))
  names(rmses) <- candidates
  attr(rmses, "best") <- rmses[which.min(rmses)]
  rmses
}

# Perform steps for model building
s1 <- steps("LIFESAT_I", preds, cands, trainSet, testSet)

# Add RMSE values for additional predictors
s1_df <- rbind(s1_df, data.frame(Variable = names(s1), RMSE = as.numeric(s1)))

# Convert Variable to factor with specified levels
s1_df$Variable <- factor(s1_df$Variable, levels = s1_df$Variable)

# Plot RMSE vs Predictors using ggplot2
ggplot(s1_df, aes(x = Variable, y = RMSE)) +
  geom_point() +
  geom_line(group = 1) +
  ggtitle("RMSE vs Predictors") +
  xlab("Predictors") +
  ylab("RMSE") +
  theme_minimal()
```



The plot indicates a marginal decrease in RMSE upon adding HCTHREAT_I or NONDISCLOSURE_I, implying that including more than three predictors may not be advisable for predicting life satisfaction using these scales. Introducing additional predictors can elevate model complexity without substantially enhancing performance, complicating interpretation and heightening the likelihood of overfitting. Therefore, opting for a simpler model with fewer predictors is generally preferred, as it offers improved interpretability and mitigates the risk of overfitting.