Batch: A1	Roll No.: 16010123012
Experiment No. 6	

Title: Data Cleaning, Transformation and Feature Engineering Techniques

Aim: Handle missing values, detect outliers, perform one-hot encoding, label encoding, feature scaling (min-max and z-score normalization), and generate polynomial features.

Course Outcome:

CO3: Learn data cleaning, transformation, and feature engineering techniques.

Books/ Journals/ Websites referred:

1. The Comprehensive R Archive Network
2. Posit

Resources used:
https://www.rdocumentation.org/
https://www.w3schools.com/r/
https://www.geeksforgeeks.org/r-programming-language-introduction/

Theory:

Implementation:

A) Handling Missing Values

In this example we will use the open repository of plants classification Iris.

1) Data Loading

```
> data()
> str(iris_data)
Error: object 'iris_data' not found
> str(iris)
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

2) Introducing missing values in the dataset

```
> iris_copy <- iris
>
> iris_copy$Sepal.Length[c(15, 20, 50, 67, 97, 118)] <- NA
> iris_copy$Sepal.Width[c(4, 80, 97, 106)] <- NA
> iris_copy$Petal.Length[c(5, 17, 35, 49)] <- NA
> summary(iris_copy)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Specie
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa    :5
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:5
 Median :5.800   Median :3.000   Median :4.400   Median :1.300   virginica :5
 Mean   :5.844   Mean   :3.062   Mean   :3.822   Mean   :1.199
 3rd Qu.:6.400   3rd Qu.:3.375   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
 NA's   :6       NA's   :4       NA's   :4
```

3) Find NA's in dataset

The first thing we can do is to ask if there is any missing value in our table

```
> length(which(is.na(iris_copy)))
[1] 14
```

We can check that we introduced 14 missing values in the table

There are several ways to identify rows containing NA's.

First we will use the complete.cases function (check ?complete.cases for information)

This function returns only rows without NA's. Putting ! in front of it we get only rows with NA's

```
> iris_NA <- iris_copy[!complete.cases(iris_copy), ]
> iris_NA
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
4            4.6          NA          1.5         0.2     setosa
5            5.0         3.6           NA         0.2     setosa
15            NA         4.0          1.2         0.2     setosa
17           5.4         3.9           NA         0.4     setosa
20            NA         3.8          1.5         0.3     setosa
35           4.9         3.1           NA         0.2     setosa
49           5.3         3.7           NA         0.2     setosa
50            NA         3.3          1.4         0.2     setosa
```

We see that we have 13 rows with missing values on it

Another way is to search for TRUE values in the is.na function

```
> iris_NA <- iris_copy[rowSums(is.na(iris_copy)) > 0, ]
> iris_NA
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
4            4.6          NA          1.5         0.2     setosa
5            5.0         3.6           NA         0.2     setosa
15            NA         4.0          1.2         0.2     setosa
17           5.4         3.9           NA         0.4     setosa
20            NA         3.8          1.5         0.3     setosa
```

We obtain the same result. However the function complete cases is

4) Removing rows containing NA's

Depending on the case, there are different things we can do with NA's. However there is not a unique and general solution to this issue. If the missing value can be calculated direclty using other columns the problem is solved.

The first choice can be to just remove the rows containing NA's

```
> complete.cases(iris_copy)
  [1]  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [14]  TRUE FALSE  TRUE FALSE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [27]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE
 [40]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE
 [53]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [66]  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [79]  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [92]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[105]  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[118] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[131]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[144]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
```

```
> iris_clean <- iris_copy[complete.cases(iris_copy), ]
> length(which(is.na(iris_clean)))
[1] 0
```

The number of missing values in this table is 0

5) Substituting NA's with numerical values

In other cases we don't want to lose the information that we have in one row with missing values

In this case we will substitute the missing value with a numerical value

The first thing we can do is to introduce the mean of a column in a missing value

However it's more safe to use the median because it's not affected by outliers

However we should be careful as in this case it's more correct to introduce the mean for the proper species

We should do it column by column

```
> iris_copy[is.na(iris_copy$Sepal.Length) & (iris_copy$Species == "setosa"),"Sepal.Leng
th"] <- median(iris_copy$Sepal.Length[which(iris_copy$Species == "setosa")], na.rm = TR
UE)
> iris_NA <- iris_copy[!complete.cases(iris_copy), ]
> iris_NA
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
4            4.6          NA          1.5         0.2     setosa
5            5.0         3.6           NA         0.2     setosa
17           5.4         3.9           NA         0.4     setosa
35           4.9         3.1           NA         0.2     setosa
49           5.3         3.7           NA         0.2     setosa
67            NA         3.0          4.5         1.5 versicolor
```

Now we have removed 3 NA's. Only 11 left

```
> iris_copy[is.na(iris_copy$Sepal.Length) & (iris_copy$Species == "versicolor"),"Sepal.
Length"] <- median(iris_copy$Sepal.Length[which(iris_copy$Species == "versicolor")], n
a.rm = TRUE)
> iris_NA <- iris_copy[!complete.cases(iris_copy), ]
> iris_NA
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
4           4.60          NA          1.5         0.2     setosa
5           5.00         3.6           NA         0.2     setosa
17          5.40         3.9           NA         0.4     setosa
35          4.90         3.1           NA         0.2     setosa
49          5.30         3.7           NA         0.2     setosa
80          5.70          NA          3.5         1.0 versicolor
97          5.95          NA          4.2         1.3 versicolor
106         7.60          NA          6.6         2.1  virginica
```

Now we have removed 1 NA's. Only 8 left

```
> iris_copy[is.na(iris_copy$Sepal.Width) & (iris_copy$Species == "setosa"),"Sepal.Widt
h"] <- median(iris_copy$Sepal.Width[which(iris_copy$Species == "setosa")], na.rm = TRU
E)
> iris_NA <- iris_copy[!complete.cases(iris_copy), ]
> iris_NA
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
5           5.00         3.6           NA         0.2     setosa
17          5.40         3.9           NA         0.4     setosa
35          4.90         3.1           NA         0.2     setosa
49          5.30         3.7           NA         0.2     setosa
80          5.70          NA          3.5         1.0 versicolor
97          5.95          NA          4.2         1.3 versicolor
```

Now we have removed 1 NA's. Only 7 left

```
> iris_copy[is.na(iris_copy$Petal.Length) & (iris_copy$Species == "setosa"),"Petal.Leng
th"] <- median(iris_copy$Petal.Length[which(iris_copy$Species == "setosa")], na.rm = TR
UE)
> iris_NA <- iris_copy[!complete.cases(iris_copy), ]
> iris_NA
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
80          5.70          NA          3.5         1.0 versicolor
97          5.95          NA          4.2         1.3 versicolor
106         7.60          NA          6.6         2.1  virginica
```

 # Now it's your time to finish the logic and remove all the remaining NA's of the table

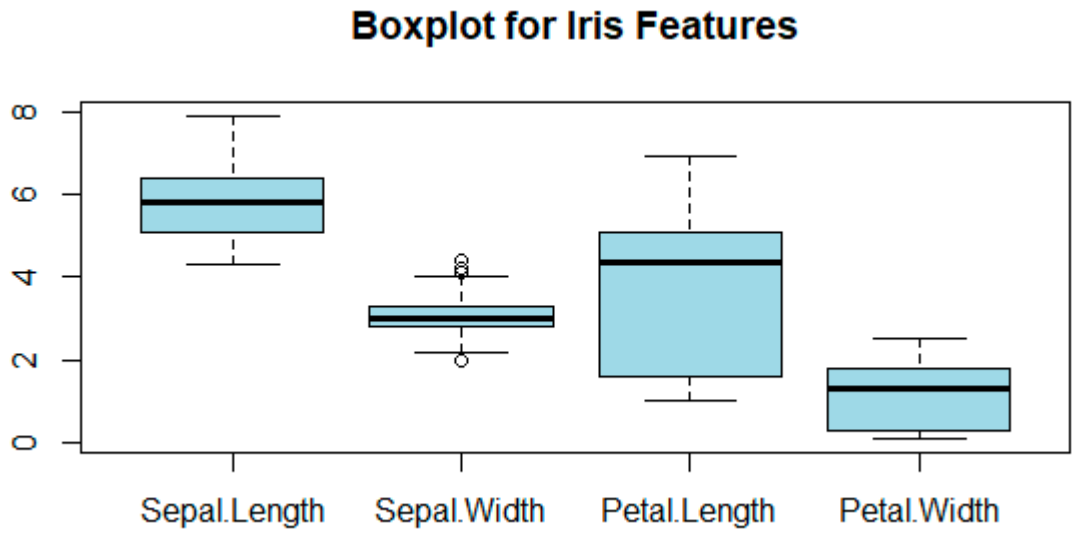
 # We have saved a lot of interesting data. However we must be careful because each case is different and we can affect the result if we introduce the wrong number in the NA position

B) Detecting Outliers

Boxplot Method

Boxplots are useful for identifying outliers. In a boxplot, data points that lie beyond 1.5 times the interquartile range (IQR) are considered outliers.

```
> boxplot(iris[, 1:4], main = "Boxplot for Iris Features", col = "lightblue")
```



- If any points are plotted outside the whiskers, they are considered **outliers**.

To find specific outliers:

```
> detect_outliers <- function(x) {
+     Q1 <- quantile(x, 0.25)  # First quartile (25%)
+     Q3 <- quantile(x, 0.75)  # Third quartile (75%)
+     IQR_value <- Q3 - Q1  # Interquartile range
+
+     lower_bound <- Q1 - 1.5 * IQR_value
+     upper_bound <- Q3 + 1.5 * IQR_value
+
+     return(which(x < lower_bound | x > upper_bound))
+ }
> outliers <- lapply(iris[, 1:4], detect_outliers)
> outliers
```

```
$Sepal.Length
integer(0)

$Sepal.Width
[1] 16 33 34 61

$Petal.Length
integer(0)

$Petal.Width
integer(0)

> |
```

4. Visualization Using Scatterplots

Scatterplots can help visualize potential outliers.

```
r
CopyEdit
# Scatter plot matrix
pairs(iris[, 1:4], col = ifelse(rownames(iris) %in% unlist(outliers),
"red", "black"),
      main = "Scatterplot Matrix with Outliers (Red)")
```

---

5. Outlier Detection Using `outliers` Package

You can also use the `outliers` package:

```
r
CopyEdit
install.packages("outliers")
library(outliers)

# Identify the most extreme value in Sepal.Length
outlier_value <- scores(iris$Sepal.Length, type = "z")
outliers_sepal <- which(abs(outlier_value) > 3)
outliers_sepal
```

C) Perform One-Hot Encoding

One-hot encoding is used to convert categorical variables into numerical format for machine learning models. In the **Iris dataset**, the `Species` column is categorical and can be converted into numerical dummy variables.

Use model.matrix() for One-Hot Encoding

The model.matrix() function in R can be used to perform one-hot encoding:

```
> one_hot_iris <- model.matrix(~Species - 1, data = iris)
> head(one_hot_iris)
  Speciessetosa Speciesversicolor Speciesvirginica
1             1                 0                0
2             1                 0                0
3             1                 0                0
4             1                 0                0
5             1                 0                0
6             1                 0                0
>
```

~Species - 1: The -1 removes the intercept term to ensure we get separate columns for each category.

This will create three new columns: Speciessetosa, Speciesversicolor, and Speciesvirginica, each containing 0 or 1.

```
3. Combine One-Hot Encoding with Original Data
To create a new dataframe with encoded values:

r
Copy
Edit
# Convert to dataframe
one_hot_iris <- as.data.frame(one_hot_iris)

# Combine with original numerical features
iris_encoded <- cbind(iris[, 1:4], one_hot_iris)

# View the updated dataset
head(iris_encoded)
4. Using caret Package for One-Hot Encoding
Alternatively, you can use the caret package:

r
Copy
Edit
install.packages("caret")  # Install caret package if not installed
library(caret)
```

```
# Apply one-hot encoding
dummy_vars <- dummyVars(~ Species, data = iris)
iris_transformed <- predict(dummy_vars, newdata = iris)

# Convert to dataframe
iris_encoded <- cbind(iris[, 1:4], as.data.frame(iris_transformed))

# View result
head(iris_encoded)
```

D) Feature scaling (min-max and z-score normalization)

Min-Max Scaling

Min-Max Scaling transforms the data to a fixed range, typically `[0,1]`. The formula is:

$$X' = \frac{X - X_{\min}}{X_{\max} - X_{\min}}$$

```
> min_max_scaling <- function(x) {
+     return((x - min(x)) / (max(x) - min(x)))
+ }
>
> # Apply to numerical columns
> iris_minmax <- as.data.frame(lapply(iris[, 1:4], min_max_scaling))
>
> # Combine with Species column
> iris_minmax$Species <- iris$Species
>
> # View the transformed dataset
> head(iris_minmax)
```

```
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1   0.22222222   0.6250000   0.06779661  0.04166667  setosa
2   0.16666667   0.4166667   0.06779661  0.04166667  setosa
3   0.11111111   0.5000000   0.05084746  0.04166667  setosa
4   0.08333333   0.4583333   0.08474576  0.04166667  setosa
5   0.19444444   0.6666667   0.06779661  0.04166667  setosa
6   0.30555556   0.7916667   0.11864407  0.12500000  setosa
>
```

Z-Score Normalization (Standardization)

Z-Score Normalization transforms data to have **mean = 0** and **standard deviation = 1** using the formula:

$$X' = \frac{X - \mu}{\sigma}$$

where:

- μ = mean of the feature
- σ = standard deviation of the feature

```
> # Function for Z-score Normalization
> z_score_scaling <- function(x) {
+     return((x - mean(x)) / sd(x))
+ }
>
> # Apply to numerical columns
> iris_zscore <- as.data.frame(lapply(iris[, 1:4], z_score_scaling))
>
> # Combine with Species column
> iris_zscore$Species <- iris$Species
>
> # View the transformed dataset
> head(iris_zscore)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1   -0.8976739  1.01560199    -1.335752   -1.311052  setosa
2   -1.1392005 -0.13153881    -1.335752   -1.311052  setosa
```

```
4. Using caret Package for Feature Scaling
You can also use the caret package:
r
CopyEdit
install.packages("caret")  # Install caret package if not installed
library(caret)

# Min-Max Scaling
preprocess_minmax <- preProcess(iris[, 1:4], method = "range")
iris_minmax_caret <- predict(preprocess_minmax, iris[, 1:4])

# Z-Score Normalization
preprocess_zscore <- preProcess(iris[, 1:4], method = "center", "scale")
iris_zscore_caret <- predict(preprocess_zscore, iris[, 1:4])
```

```
# Add Species column back
iris_minmax_caret$Species <- iris$Species
iris_zscore_caret$Species <- iris$Species

# View results
head(iris_minmax_caret)
head(iris_zscore_caret)
```

E) Generate polynomial features.

Using `poly()` Function

The `poly()` function generates orthogonal polynomials of a given degree.

```
> # Generate second-degree polynomial features for Sepal.Length
> iris$Sepal.Length_poly2 <- poly(iris$Sepal.Length, degree = 2, raw = TRUE)[, 2]
>
> # Generate third-degree polynomial features for all numerical columns
> iris_poly <- as.data.frame(lapply(iris[, 1:4], function(x) poly(x, degree = 3, raw =
TRUE)))
>
> # Rename columns for clarity
> colnames(iris_poly) <- c("Sepal.Length_1", "Sepal.Length_2", "Sepal.Length_3",
+                          "Sepal.Width_1", "Sepal.Width_2", "Sepal.Width_3",
+                          "Petal.Length_1", "Petal.Length_2", "Petal.Length_3",
+                          "Petal.Width_1", "Petal.Width_2", "Petal.Width_3")
>
> # Add the Species column back
> iris_poly$Species <- iris$Species
>
> # View transformed dataset
> head(iris_poly)
```

- `poly(x, degree = 3, raw = TRUE)`: Creates polynomial features up to **degree 3**.
- `raw = TRUE`: Ensures regular polynomial terms (without orthogonal transformation).
    - `lapply()`: Applies the function to all numeric columns.

```
Using caret Package
The caret package allows automatic polynomial feature generation.
r
CopyEdit
install.packages("caret")  # Install if not installed
library(caret)

# Generate polynomial features up to degree 3
poly_features <- preProcess(iris[, 1:4], method = "poly", degree = 3, raw =
TRUE)

# Apply transformation
iris_poly_caret <- predict(poly_features, iris[, 1:4])
```

```
# Add Species column back
iris_poly_caret$Species <- iris$Species

# View results
head(iris_poly_caret)
```

```
4. Using polyreg Package
The polyreg package provides another way to generate polynomial features.
r
CopyEdit
install.packages("polyreg")  # Install package
library(polyreg)

# Generate polynomial features up to degree 3
iris_polyreg <- poly_fit(iris[, 1:4], degree = 3)

# View transformed dataset
head(iris_polyreg)
```

Red Colour part (commands) student should try

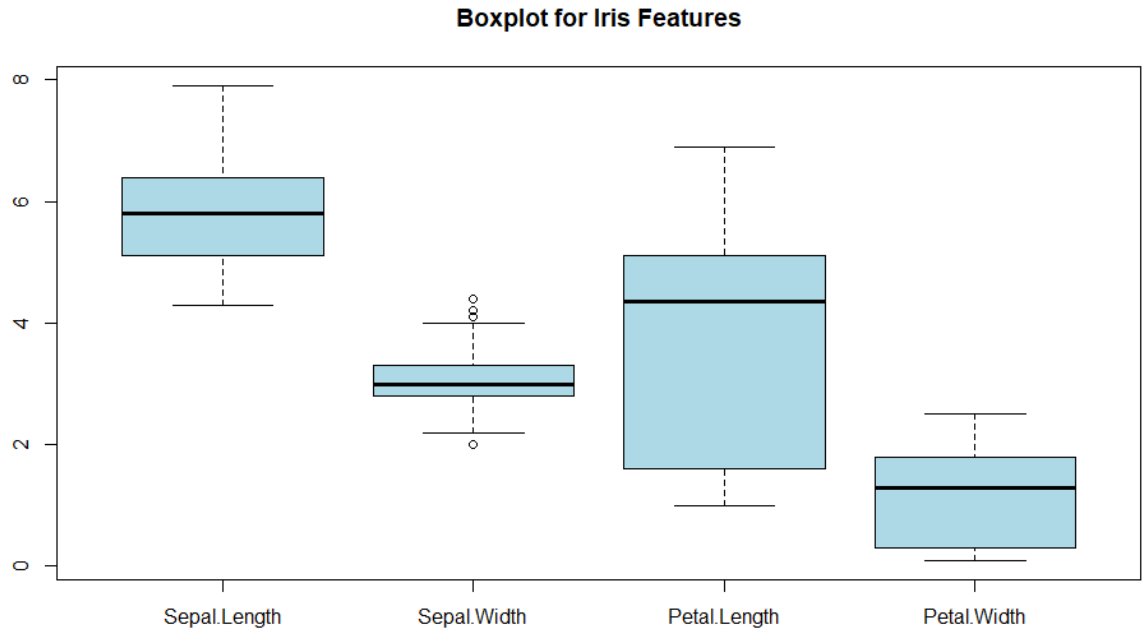
```
> data()
> str(iris)
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
> iris_copy <- iris
> iris_copy$Sepal.Length[c(15, 21, 36, 50, 69, 88, 97, 118)] <- NA
> iris_copy$Sepal.Width[c(9, 54, 80, 97, 106)] <- NA
> iris_copy$Petal.Length[c(4, 17, 35, 49)] <- NA
> summary(iris_copy)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa    :50
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50
 Median :5.800   Median :3.000   Median :4.400   Median :1.300   virginica :50
 Mean   :5.841   Mean   :3.068   Mean   :3.821   Mean   :1.199
 3rd Qu.:6.400   3rd Qu.:3.400   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
 NA's   :8       NA's   :5       NA's   :4
> length(which(is.na(iris_copy)))
[1] 17
```

```
> iris_NA <- iris_copy[!complete.cases(iris_copy), ]
> iris_NA
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
4            4.6         3.1           NA         0.2     setosa
9            4.4          NA          1.4         0.2     setosa
15            NA         4.0          1.2         0.2     setosa
17           5.4         3.9           NA         0.4     setosa
21            NA         3.4          1.7         0.2     setosa
35           4.9         3.1           NA         0.2     setosa
36            NA         3.2          1.2         0.2     setosa
49           5.3         3.7           NA         0.2     setosa
50            NA         3.3          1.4         0.2     setosa
54           5.5          NA          4.0         1.3 versicolor
69            NA         2.2          4.5         1.5 versicolor
80           5.7          NA          3.5         1.0 versicolor
88            NA         2.3          4.4         1.3 versicolor
97            NA          NA          4.2         1.3 versicolor
106          7.6          NA          6.6         2.1  virginica
118           NA         3.8          6.7         2.2  virginica
> iris_NA <- iris_copy[rowSums(is.na(iris_copy)) > 0, ]
> iris_NA
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
4            4.6         3.1           NA         0.2     setosa
9            4.4          NA          1.4         0.2     setosa
15            NA         4.0          1.2         0.2     setosa
17           5.4         3.9           NA         0.4     setosa
21            NA         3.4          1.7         0.2     setosa
35           4.9         3.1           NA         0.2     setosa
36            NA         3.2          1.2         0.2     setosa
49           5.3         3.7           NA         0.2     setosa
50            NA         3.3          1.4         0.2     setosa
54           5.5          NA          4.0         1.3 versicolor
69            NA         2.2          4.5         1.5 versicolor
80           5.7          NA          3.5         1.0 versicolor
88            NA         2.3          4.4         1.3 versicolor
97            NA          NA          4.2         1.3 versicolor
106          7.6          NA          6.6         2.1  virginica
118           NA         3.8          6.7         2.2  virginica
> complete.cases(iris_copy)
  [1]  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
 [15] FALSE  TRUE FALSE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [29]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [43]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE
 [57]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE
 [71]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE
 [85]  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE
 [99]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[113]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[127]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[141]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
```

```
> iris_clean <- iris_copy[complete.cases(iris_copy), ]
> length(which(is.na(iris_clean)))
[1] 0
> iris_copy[is.na(iris_copy$Sepal.Length) & (iris_copy$Species == "setosa"), "Sepal.Length"] <-
median(iris_copy$Sepal.Length[which(iris_copy$Species == "setosa")], na.rm = TRUE)
> iris_NA <- iris_copy[!complete.cases(iris_copy), ]
> iris_NA
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
4            4.6         3.1           NA         0.2     setosa
9            4.4          NA          1.4         0.2     setosa
17           5.4         3.9           NA         0.4     setosa
35           4.9         3.1           NA         0.2     setosa
49           5.3         3.7           NA         0.2     setosa
54           5.5          NA          4.0         1.3 versicolor
69            NA         2.2          4.5         1.5 versicolor
80           5.7          NA          3.5         1.0 versicolor
88            NA         2.3          4.4         1.3 versicolor
97            NA          NA          4.2         1.3 versicolor
106          7.6          NA          6.6         2.1  virginica
118           NA         3.8          6.7         2.2  virginica
> iris_copy[is.na(iris_copy$Sepal.Length) & (iris_copy$Species == "versicolor"), "Sepal.Lengt
h"] <- median(iris_copy$Sepal.Length[which(iris_copy$Species == "versicolor")], na.rm = TRUE)
> iris_NA <- iris_copy[!complete.cases(iris_copy), ]
> iris_NA
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
4            4.6         3.1           NA         0.2     setosa
9            4.4          NA          1.4         0.2     setosa
17           5.4         3.9           NA         0.4     setosa
35           4.9         3.1           NA         0.2     setosa
49           5.3         3.7           NA         0.2     setosa
54           5.5          NA          4.0         1.3 versicolor
80           5.7          NA          3.5         1.0 versicolor
97           5.9          NA          4.2         1.3 versicolor
106          7.6          NA          6.6         2.1  virginica
118           NA         3.8          6.7         2.2  virginica
> iris_copy[is.na(iris_copy$Sepal.Length) & (iris_copy$Species == "virginica"), "Sepal.Length"]
<- median(iris_copy$Sepal.Length[which(iris_copy$Species == "virginica")], na.rm = TRUE)
> iris_NA <- iris_copy[!complete.cases(iris_copy), ]
> iris_NA
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
4            4.6         3.1           NA         0.2     setosa
9            4.4          NA          1.4         0.2     setosa
17           5.4         3.9           NA         0.4     setosa
35           4.9         3.1           NA         0.2     setosa
49           5.3         3.7           NA         0.2     setosa
54           5.5          NA          4.0         1.3 versicolor
80           5.7          NA          3.5         1.0 versicolor
97           5.9          NA          4.2         1.3 versicolor
106          7.6          NA          6.6         2.1  virginica
> iris_copy[is.na(iris_copy$Sepal.Width) & (iris_copy$Species == "setosa"), "Sepal.Length"] <-
median(iris_copy$Sepal.Width[which(iris_copy$Species == "setosa")], na.rm = TRUE)
> iris_NA <- iris_copy[!complete.cases(iris_copy), ]
> iris_NA
```

```
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
4            4.6         3.1           NA         0.2     setosa
9            3.4          NA          1.4         0.2     setosa
17           5.4         3.9           NA         0.4     setosa
35           4.9         3.1           NA         0.2     setosa
49           5.3         3.7           NA         0.2     setosa
54           5.5          NA          4.0         1.3 versicolor
80           5.7          NA          3.5         1.0 versicolor
97           5.9          NA          4.2         1.3 versicolor
106          7.6          NA          6.6         2.1  virginica
> iris_copy[is.na(iris_copy$Sepal.Width) & (iris_copy$Species == "setosa"), "Sepal.Width"] <- m
edian(iris_copy$Sepal.Width[which(iris_copy$Species == "setosa")], na.rm = TRUE)
> iris_NA <- iris_copy[!complete.cases(iris_copy), ]
> iris_NA
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
4            4.6         3.1           NA         0.2     setosa
17           5.4         3.9           NA         0.4     setosa
35           4.9         3.1           NA         0.2     setosa
49           5.3         3.7           NA         0.2     setosa
54           5.5          NA          4.0         1.3 versicolor
80           5.7          NA          3.5         1.0 versicolor
97           5.9          NA          4.2         1.3 versicolor
106          7.6          NA          6.6         2.1  virginica
> iris_copy[is.na(iris_copy$Sepal.Width) & (iris_copy$Species == "versicolor"), "Sepal.Width"]
<- median(iris_copy$Sepal.Width[which(iris_copy$Species == "versicolor")], na.rm = TRUE)
> iris_NA <- iris_copy[!complete.cases(iris_copy), ]
> iris_NA
    Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
4            4.6         3.1           NA         0.2    setosa
17           5.4         3.9           NA         0.4    setosa
35           4.9         3.1           NA         0.2    setosa
49           5.3         3.7           NA         0.2    setosa
106          7.6          NA          6.6         2.1 virginica
> iris_copy[is.na(iris_copy$Sepal.Width) & (iris_copy$Species == "virginica"), "Sepal.Width"] <
- median(iris_copy$Sepal.Width[which(iris_copy$Species == "virginica")], na.rm = TRUE)
> iris_NA <- iris_copy[!complete.cases(iris_copy), ]
> iris_NA
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
4           4.6         3.1           NA         0.2  setosa
17          5.4         3.9           NA         0.4  setosa
35          4.9         3.1           NA         0.2  setosa
49          5.3         3.7           NA         0.2  setosa
> iris_copy[is.na(iris_copy$Petal.Length) & (iris_copy$Species == "setosa"), "Petal.Length"] <-
median(iris_copy$Petal.Length[which(iris_copy$Species == "setosa")], na.rm = TRUE)
> iris_NA <- iris_copy[!complete.cases(iris_copy), ]
> iris_NA
[1] Sepal.Length Sepal.Width  Petal.Length Petal.Width  Species
<0 rows> (or 0-length row.names)
```

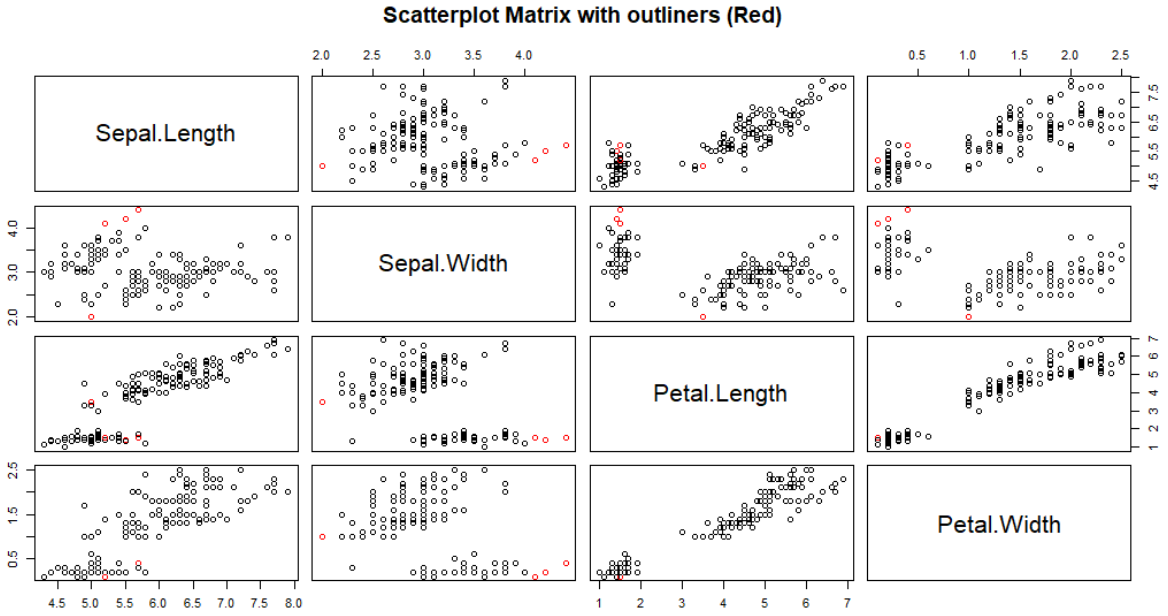
```
> boxplot(iris[, 1:4], main = "Boxplot for Iris Features", col = "lightblue")
```




```
> detect_outliners <- function(x) {}
> detect_outliners <- function(x) {
+     Q1 <- quantile(x, 0.25)
+     Q3 <- quantile(x, 0.75)
+     IQR_value <- Q3 - Q1
+
+     lower_bound <- Q1 - 1.5 * IQR_value
+     upper_bound <- Q3 + 1.5 * IQR_value
+
+     return(which(x < lower_bound | x > upper_bound))
+ }
> outliners <- lapply(iris[, 1:4], detect_outliners)
> outliners
$Sepal.Length
integer(0)

$Sepal.Width
[1] 16 33 34 61

$Petal.Length
integer(0)

$Petal.Width
integer(0)
```

```
> pairs(iris[, 1:4], col = ifelse(rownames(iris) %in% unlist(outliners), "red", "black"),
Warning messages:
1: In doTryCatch(return(expr), name, parentenv, handler) :
  display list redraw incomplete
2: In doTryCatch(return(expr), name, parentenv, handler) :
  display list redraw incomplete
3: In doTryCatch(return(expr), name, parentenv, handler) :
  display list redraw incomplete
+       main = "Scatterplot Matrix with outliners (Red)")
```

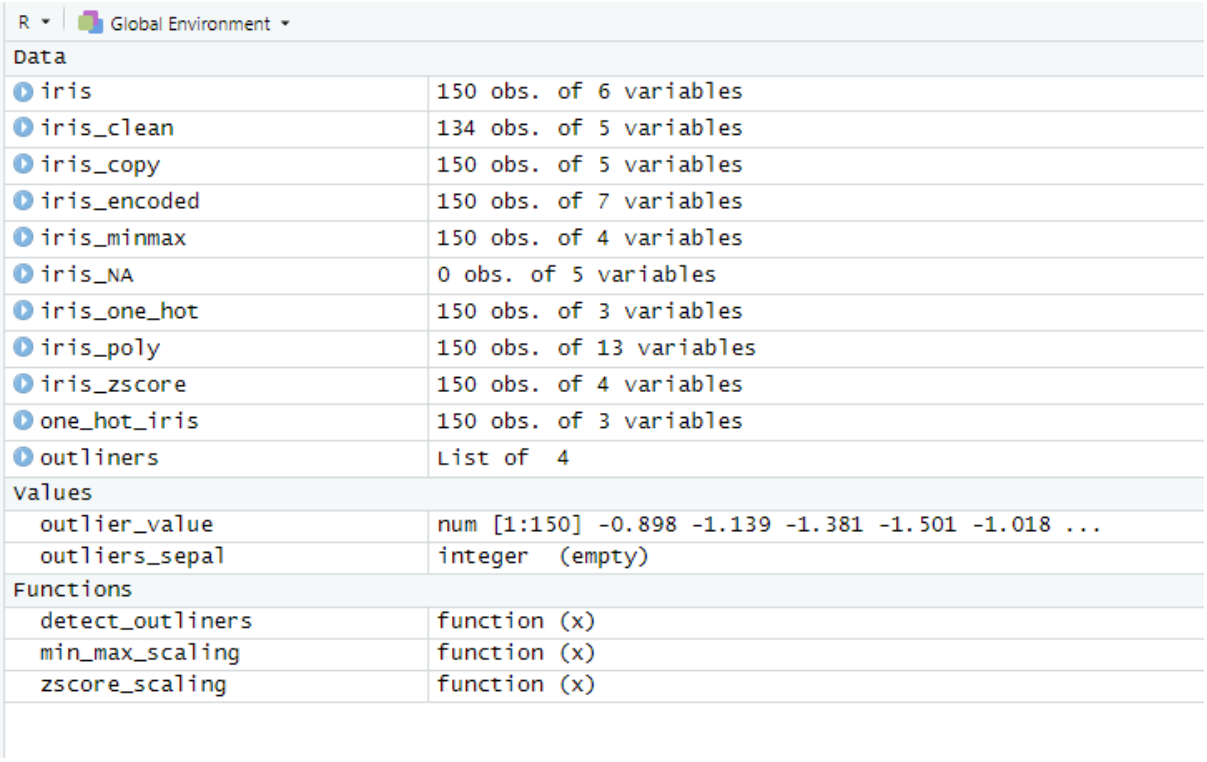
```
> install.packages("outliers")
WARNING: Rtools is required to build R packages but is not currently installed. Please download and install the a
ppropriate version of Rtools before proceeding:

https://cran.rstudio.com/bin/windows/Rtools/
Installing package into 'C:/Users/KJSCE/AppData/Local/R/win-library/4.2'
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.2/outliers_0.15.zip'
Content type 'application/zip' length 84751 bytes (82 KB)
downloaded 82 KB

package 'outliers' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\KJSCE\AppData\Local\Temp\RtmpSOFo7k\downloaded_packages
Warning messages:
1: In doTryCatch(return(expr), name, parentenv, handler) :
  display list redraw incomplete
2: In doTryCatch(return(expr), name, parentenv, handler) :
  invalid graphics state
3: In doTryCatch(return(expr), name, parentenv, handler) :
  invalid graphics state
> library(outliers)
> outlier_value <- scores(iris$Sepal.Length, type = "z")
> outliers_sepal <- which(abs(outlier_value) > 3)
> outliers_sepal
integer(0)
>
> one_hot_iris <- model.matrix(~Species - 1, data = iris)
> head(one_hot_iris)
  Speciessetosa Speciesversicolor Speciesvirginica
1             1                 0                0
2             1                 0                0
3             1                 0                0
4             1                 0                0
5             1                 0                0
6             1                 0                0
> one_hot_iris <- as.data.frame(one_hot_iris)
> iris_encoded <- cbind(iris[, 1:4], one_hot_iris)
> head(iris_encoded)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Speciessetosa Speciesversicolor Speciesvirginica
1          5.1         3.5          1.4         0.2             1                 0                0
2          4.9         3.0          1.4         0.2             1                 0                0
3          4.7         3.2          1.3         0.2             1                 0                0
4          4.6         3.1          1.5         0.2             1                 0                0
5          5.0         3.6          1.4         0.2             1                 0                0
6          5.4         3.9          1.7         0.4             1                 0                0
>
> install.packages("caret")
WARNING: Rtools is required to build R packages but is not currently installed. Please download and install the a
ppropriate version of Rtools before proceeding:

https://cran.rstudio.com/bin/windows/Rtools/
```




```
> iris_one_hot <- as.data.frame(model.matrix(~ Species - 1, data = iris))
> iris_encoded <- cbind(iris[, 1:4], iris_one_hot)
> head(iris_encoded)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Speciessetosa Speciesversicolor
1          5.1         3.5          1.4         0.2             1                 0
2          4.9         3.0          1.4         0.2             1                 0
3          4.7         3.2          1.3         0.2             1                 0
4          4.6         3.1          1.5         0.2             1                 0
5          5.0         3.6          1.4         0.2             1                 0
6          5.4         3.9          1.7         0.4             1                 0
  Speciesvirginica
1                0
2                0
3                0
4                0
5                0
6                0
>
```

```
> min_max_scaling <- function(x) {
+     return((x - min(x))) / (max(x) - min(x))
+ }
>
> iris_minmax <- as.data.frame(lapply(iris[, 1:4], min_max_scaling))
> iris_minmax$Species <- iris$Species
> head(iris_minmax)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          0.8         1.5          0.4         0.1  setosa
2          0.6         1.0          0.4         0.1  setosa
3          0.4         1.2          0.3         0.1  setosa
4          0.3         1.1          0.5         0.1  setosa
5          0.7         1.6          0.4         0.1  setosa
6          1.1         1.9          0.7         0.3  setosa
>
> zscore_scaling <- function(x) {
+     return((x - mean(x)) / sd(x))
+ }
> iris_zscore <- as.data.frame(lapply(iris[, 1:4], zscore_scaling))
> iris_zscore$Species <- iris$Species
> head(iris_zscore)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1   -0.8976739  1.01560199    -1.335752   -1.311052  setosa
2   -1.1392005 -0.13153881    -1.335752   -1.311052  setosa
3   -1.3807271  0.32731751    -1.392399   -1.311052  setosa
4   -1.5014904  0.09788935    -1.279104   -1.311052  setosa
5   -1.0184372  1.24503015    -1.335752   -1.311052  setosa
6   -0.5353840  1.93331463    -1.165809   -1.048667  setosa
>
> iris_minmax <- as.data.frame(apply(iris[, 1:4], 2, function(x) (x - min(x)) / (max(x) - min
(x))))
> head(iris_minmax)
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1   0.22222222   0.6250000   0.06779661  0.04166667
2   0.16666667   0.4166667   0.06779661  0.04166667
3   0.11111111   0.5000000   0.05084746  0.04166667
4   0.08333333   0.4583333   0.08474576  0.04166667
5   0.19444444   0.6666667   0.06779661  0.04166667
6   0.30555556   0.7916667   0.11864407  0.12500000
>
> iris_zscore <- as.data.frame(scale(iris[, 1:4]))
> head(iris_zscore)
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1   -0.8976739  1.01560199    -1.335752   -1.311052
2   -1.1392005 -0.13153881    -1.335752   -1.311052
3   -1.3807271  0.32731751    -1.392399   -1.311052
4   -1.5014904  0.09788935    -1.279104   -1.311052
5   -1.0184372  1.24503015    -1.335752   -1.311052
6   -0.5353840  1.93331463    -1.165809   -1.048667
>
```

```
> iris$Sepal.Length_poly2 <- poly(iris$Sepal.Length, degree = 2, raw = TRUE)[, 2]
> iris_poly <- as.data.frame(lapply(iris[, 1:4], function(x) poly(x, degree = 3, raw = TRUE)))
>
> colnames(iris_poly) <- c("Sepal.Length_1", "Sepal.Length_2", "Sepal.Length_3",
+                          "Sepal.Width_1", "Sepal.Width_2", "Sepal.Width_3",
+                          "Petal.Length_1", "Petal.Length_2", "Petal.Length_3",
+                          "Petal.Width_1", "Petal.Width_2", "Petal.Width_3")
> iris_poly$Species <- iris$Species
> head(iris_poly)
  Sepal.Length_1 Sepal.Length_2 Sepal.Length_3 Sepal.Width_1 Sepal.Width_2 Sepal.Width_3
1            5.1          26.01        132.651           3.5         12.25        42.875
2            4.9          24.01        117.649           3.0          9.00        27.000
3            4.7          22.09        103.823           3.2         10.24        32.768
4            4.6          21.16         97.336           3.1          9.61        29.791
5            5.0          25.00        125.000           3.6         12.96        46.656
6            5.4          29.16        157.464           3.9         15.21        59.319
  Petal.Length_1 Petal.Length_2 Petal.Length_3 Petal.Width_1 Petal.Width_2 Petal.Width_3
1            1.4           1.96          2.744           0.2          0.04          0.008
2            1.4           1.96          2.744           0.2          0.04          0.008
3            1.3           1.69          2.197           0.2          0.04          0.008
4            1.5           2.25          3.375           0.2          0.04          0.008
5            1.4           1.96          2.744           0.2          0.04          0.008
6            1.7           2.89          4.913           0.4          0.16          0.064
  Species
1  setosa
2  setosa
3  setosa
4  setosa
5  setosa
6  setosa
```



R ▾ | Global Environment ▾

Data	
iris	150 obs. of 6 variables
iris_clean	134 obs. of 5 variables
iris_copy	150 obs. of 5 variables
iris_encoded	150 obs. of 7 variables
iris_minmax	150 obs. of 4 variables
iris_NA	0 obs. of 5 variables
iris_one_hot	150 obs. of 3 variables
iris_poly	150 obs. of 13 variables
iris_zscore	150 obs. of 4 variables
one_hot_iris	150 obs. of 3 variables
outliners	List of 4
Values	
outlier_value	num [1:150] -0.898 -1.139 -1.381 -1.501 -1.018 ...
outliers_sepal	integer (empty)
Functions	
detect_outliners	function (x)
min_max_scaling	function (x)
zscore_scaling	function (x)

Conclusion:
I successfully applied data cleaning, transformation, and feature engineering techniques in R. I handled missing values using different strategies such as removal and imputation, detected outliers using statistical methods like the IQR rule and Z-score, and performed one-hot encoding and label encoding to convert categorical data into numerical form. Additionally, I implemented feature scaling using Min-Max normalization and Z-score standardization to standardize data, making it suitable for machine learning models. Lastly, I explored polynomial feature generation to capture complex relationships within the dataset. This experiment enhanced my understanding of data preprocessing, which is crucial for improving model performance and ensuring meaningful analysis.

Post Lab Question

1) Discuss an alternative approach to capturing non-linear relationships in a dataset instead of polynomial feature generation.

An alternative approach is using kernel methods such as the Radial Basis Function (RBF) kernel in Support Vector Machines (SVM) or Gaussian Processes. Neural networks, especially deep learning models, can also capture complex non-linear relationships without manually generating polynomial features.

2) What is the difference between univariate and multivariate outlier detection?

Univariate outlier detection analyzes a single variable, identifying extreme values using methods like Z-score, IQR, and box plots. For example, detecting unusually high or low heights in a student dataset is a univariate approach. In contrast, multivariate outlier detection considers relationships between multiple variables to identify anomalies. Techniques like Mahalanobis Distance, PCA, and machine learning models detect outliers that may not be extreme in individual variables but are unusual when analyzed together. For instance, a student with an average height but extremely low weight could be a multivariate outlier. While univariate methods are simpler, multivariate techniques offer deeper insights, making them essential for high-dimensional data analysis.