Batch: E2	Roll No.: 16010123325
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:

CO**3 :** Learn data cleaning, transformation, and feature engineering techniques.

Books/ Journals/ Websites referred:

1. The Comprehensive R Archive Network
2. Posit

Resources used:

(Students should write the datasets used)

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

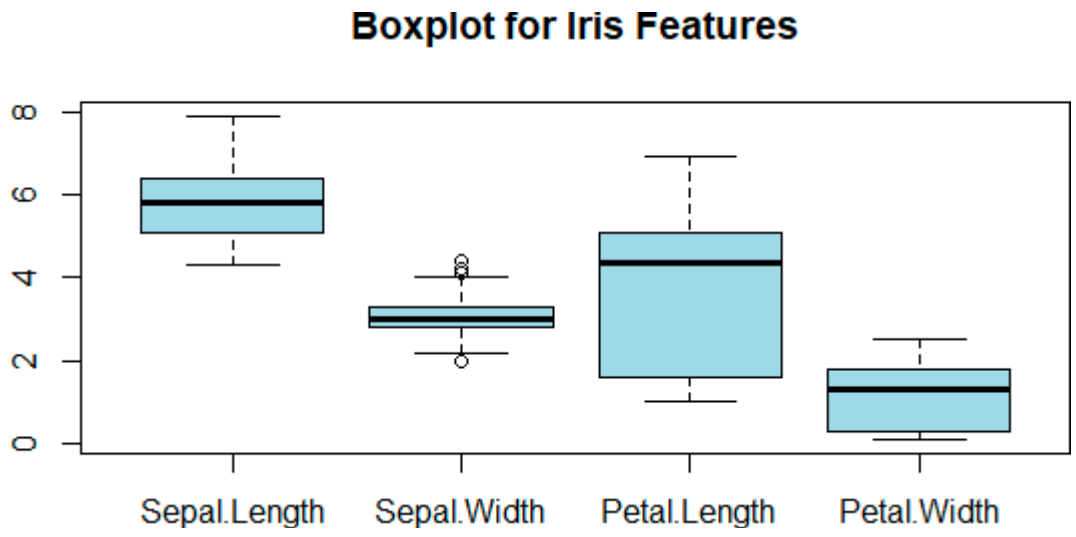
 # We have saved a lot of interesting data. However we must be careful because each case is different and we can affect the result if we introduce the wrong number in the NA position

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)
```

Students have to explore the datasets within R libraries, Kaggle, UCI or any other freely available data repositories.

```
> # Load dataset
> data()
> str(airquality)
'data.frame':   153 obs. of  6 variables:
 $ Ozone  : int  41 36 12 18 NA 28 23 19 8 NA ...
 $ Solar.R: int  190 118 149 313 NA NA 299 99 19 194 ...
 $ Wind   : num  7.4 8 12.6 11.5 14.3 14.9 8.6 13.8 20.1 8.6 ...
 $ Temp   : int  67 72 74 62 56 66 65 59 61 69 ...
 $ Month  : int  5 5 5 5 5 5 5 5 5 5 ...
 $ Day    : int  1 2 3 4 5 6 7 8 9 10 ...
>
> # Create a copy
> airquality_copy <- airquality
>
> # Introduce NA values manually
> airquality_copy$Ozone[c(5, 15, 30, 50, 75)] <- NA
> airquality_copy$Solar.R[c(10, 20, 40, 60, 90)] <- NA
> airquality_copy$Wind[c(7, 25, 55, 85)] <- NA
> airquality_copy$Temp[c(12, 35, 70)] <- NA
>
> # Summary of the dataset
> summary(airquality_copy)
     Ozone           Solar.R           Wind             Temp
 Min.   :  1.00   Min.   :  7.0   Min.   : 1.700   Min.   :56.00
 1st Qu.: 18.00   1st Qu.:118.0   1st Qu.: 7.400   1st Qu.:72.25
 Median : 32.00   Median :207.0   Median : 9.700   Median :79.00
 Mean   : 41.96   Mean   :186.6   Mean   : 9.956   Mean   :77.81
 3rd Qu.: 63.00   3rd Qu.:258.0   3rd Qu.:11.500   3rd Qu.:84.75
```

```
 3rd Qu.: 63.00   3rd Qu.:258.0   3rd Qu.:11.500   3rd Qu.:84.75
 Max.   :168.00   Max.   :334.0   Max.   :20.700   Max.   :97.00
 NA's   :40       NA's   :12      NA's   :4        NA's   :3
     Month            Day
 Min.   :5.000   Min.   : 1.0
 1st Qu.:6.000   1st Qu.: 8.0
 Median :7.000   Median :16.0
 Mean   :6.993   Mean   :15.8
 3rd Qu.:8.000   3rd Qu.:23.0
 Max.   :9.000   Max.   :31.0

>
> # Count NA values
> length(which(is.na(airquality_copy)))
[1] 59
>
> # Extract rows with NA values
> airquality_NA <- airquality_copy[!complete.cases(airquality_copy),]
> airquality_NA
    Ozone Solar.R Wind Temp Month Day
5      NA      NA 14.3   56     5   5
6      28      NA 14.9   66     5   6
7      23     299   NA   65     5   7
10     NA      NA  8.6   69     5  10
11      7      NA  6.9   74     5  11
12     16     256  9.7   NA     5  12
15     NA      65 13.2   58     5  15
20     11      NA  8.7   67     5  20
34     NA     242 16.1   67     6   3
35     NA     186  9.2   NA     6   4
36     NA     220  8.6   85     6   5
37     NA     264 14.3   79     6   6
39     NA     273  6.9   87     6   8
40     71      NA 13.8   90     6   9
42     NA     259 10.9   93     6  11
43     NA     250  9.2   92     6  12
45     NA     332 13.8   80     6  14
46     NA     322 11.5   79     6  15
50     NA     120 11.5   73     6  19
52     NA     150  6.3   77     6  21
53     NA      59  1.7   76     6  22
54     NA      91  4.6   76     6  23
55     NA     250   NA   76     6  24
56     NA     135  8.0   75     6  25
57     NA     127  8.0   78     6  26
58     NA      47 10.3   73     6  27
59     NA      98 11.5   80     6  28
60     NA      NA 14.9   77     6  29
61     NA     138  8.0   83     6  30
65     NA     101 10.9   84     7   4
70     97     272  5.7   NA     7   9
72     NA     139  8.6   82     7  11
75     NA     291 14.9   91     7  14
83     NA     258  9.7   81     7  22
84     NA     295 11.5   82     7  23
85     80     294   NA   86     7  24
```

```
90      50       NA  7.4   86      7  29
96      78       NA  6.9   86      8   4
97      35       NA  7.4   85      8   5
98      66       NA  4.6   87      8   6
102     NA      222  8.6   92      8  10
103     NA      137 11.5   86      8  11
107     NA       64 11.5   79      8  15
115     NA      255 12.6   75      8  23
119     NA      153  5.7   88      8  27
150     NA      145 13.2   77      9  27
>
> airquality_NA <- airquality_copy[rowSums(is.na(airquality_copy)) > 0,]
> airquality_NA
    Ozone Solar.R Wind Temp Month Day
5      NA      NA 14.3   56      5   5
6      28      NA 14.9   66      5   6
7      23     299   NA   65      5   7
10     NA      NA  8.6   69      5  10
11      7      NA  6.9   74      5  11
12     16     256  9.7   NA      5  12
15     NA      65 13.2   58      5  15
20     11      NA  9.7   62      5  20
25     NA      66   NA   57      5  25
26     NA     266 14.9   58      5  26
27     NA      NA  8.0   57      5  27
30     NA     223  5.7   79      5  30
32     NA     286  8.6   78      6   1
```

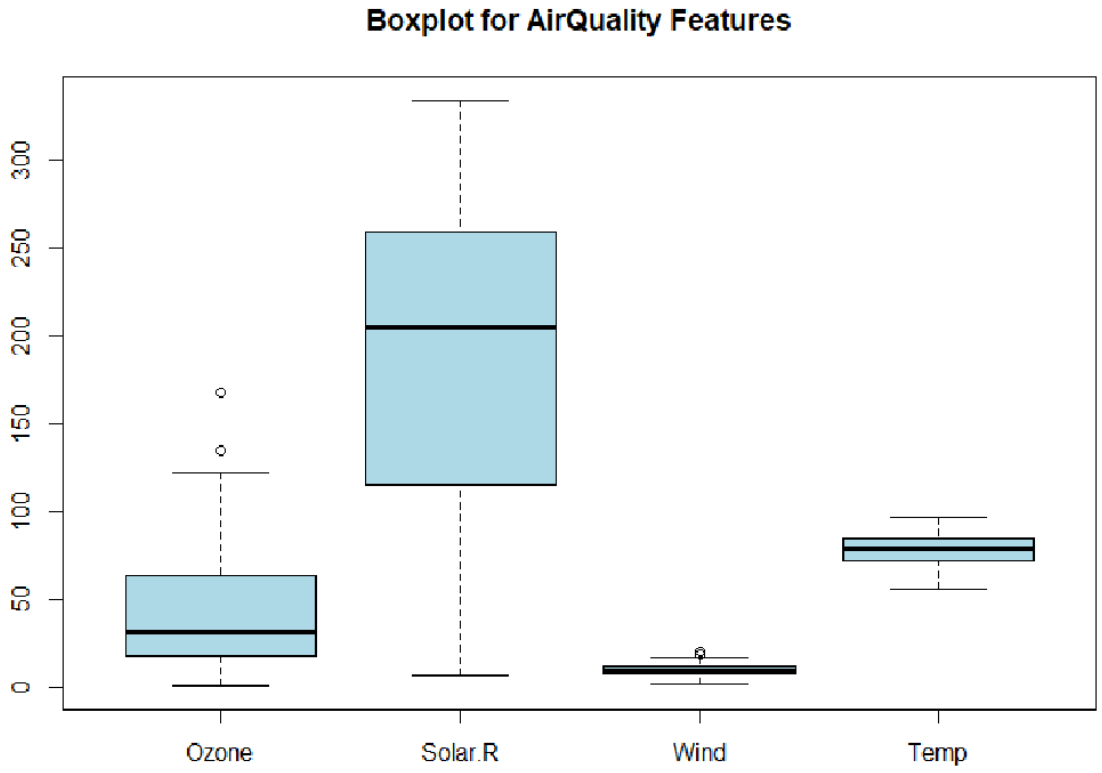
```
36     NA     220  8.6   85      6   5
37     NA     264 14.3   79      6   6
39     NA     273  6.9   87      6   8
40     71      NA 13.8   90      6   9
42     NA     259 10.9   93      6  11
43     NA     250  9.2   92      6  12
45     NA     332 13.8   80      6  14
46     NA     322 11.5   79      6  15
50     NA     120 11.5   73      6  19
52     NA     150  6.3   77      6  21
53     NA      59  1.7   76      6  22
54     NA      91  4.6   76      6  23
55     NA     250   NA   76      6  24
56     NA     135  8.0   75      6  25
57     NA     127  8.0   78      6  26
58     NA      47 10.3   73      6  27
59     NA      98 11.5   80      6  28
60     NA      NA 14.9   77      6  29
61     NA     138  8.0   83      6  30
65     NA     101 10.9   84      7   4
70     97     272  5.7   NA      7   9
72     NA     139  8.6   82      7  11
75     NA     291 14.9   91      7  14
83     NA     258  9.7   81      7  22
84     NA     295 11.5   82      7  23
85     80     294   NA   86      7  24
90     50      NA  7.4   86      7  29
```

```
119    NA     153  5.7   88     8  27
150    NA     145 13.2   77     9  27
>
> # Identify complete cases
> complete.cases(airquality_copy)
  [1]  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE
 [13]  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE
 [25] FALSE FALSE FALSE  TRUE  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE
 [37] FALSE  TRUE FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE FALSE  TRUE  TRUE
 [49]  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [61] FALSE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE
 [73]  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE
 [85] FALSE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE
 [97] FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE FALSE  TRUE
[109]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE FALSE  TRUE
[121]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[133]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[145]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE
>
> # Create a clean dataset without NA values
> airquality_clean <- airquality_copy[complete.cases(airquality_copy),]
> length(which(is.na(airquality_clean)))
[1] 0
>
> # Impute missing values with median based on Month
> airquality_copy[is.na(airquality_copy$Ozone) & (airquality_copy$Month == 5), "Ozon
e"] <- median(airquality_copy$Ozone[which(airquality_copy$Month == 5)], na.rm = TRUE)
```

```
    Ozone Solar.R Wind Temp Month Day
5      17      NA 14.3   56     5   5
6      28      NA 14.9   66     5   6
7      23     299   NA   65     5   7
10     17      NA  8.6   69     5  10
11      7      NA  6.9   74     5  11
12     16     256  9.7   NA     5  12
20     11      NA  9.7   62     5  20
25     17      66   NA   57     5  25
27     17      NA  8.0   57     5  27
32     NA     286  8.6   78     6   1
33     NA     287  9.7   74     6   2
34     NA     242 16.1   67     6   3
35     NA     186  9.2   NA     6   4
36     NA     220  8.6   85     6   5
37     NA     264 14.3   79     6   6
39     NA     273  6.9   87     6   8
40     71      NA 13.8   90     6   9
42     NA     259 10.9   93     6  11
43     NA     250  9.2   92     6  12
45     NA     332 13.8   80     6  14
46     NA     322 11.5   79     6  15
50     NA     120 11.5   73     6  19
52     NA     150  6.3   77     6  21
53     NA      59  1.7   76     6  22
54     NA      91  4.6   76     6  23
55     NA     250   NA   76     6  24
```

```
> airquality_copy[is.na(airquality_copy$Solar.R) & (airquality_copy$Month == 6), "Sol
ar.R"] <- median(airquality_copy$Solar.R[which(airquality_copy$Month == 6)], na.rm =
TRUE)
> airquality_NA <- airquality_copy[!complete.cases(airquality_copy),]
> airquality_NA
    Ozone Solar.R Wind Temp Month Day
5      17      NA 14.3   56     5   5
6      28      NA 14.9   66     5   6
7      23   299.0   NA   65     5   7
10     17      NA  8.6   69     5  10
11      7      NA  6.9   74     5  11
12     16   256.0  9.7   NA     5  12
20     11      NA  9.7   62     5  20
25     17    66.0   NA   57     5  25
27     17      NA  8.0   57     5  27
32     NA   286.0  8.6   78     6   1
33     NA   287.0  9.7   74     6   2
34     NA   242.0 16.1   67     6   3
35     NA   186.0  9.2   NA     6   4
36     NA   220.0  8.6   85     6   5
37     NA   264.0 14.3   79     6   6
39     NA   273.0  6.9   87     6   8
42     NA   259.0 10.9   93     6  11
43     NA   250.0  9.2   92     6  12
45     NA   332.0 13.8   80     6  14
46     NA   322.0 11.5   79     6  15
50     NA   120.0 11.5   73     6  19
52     NA   150.0  6.3   77     6  21
> airquality_copy[is.na(airquality_copy$Wind) & (airquality_copy$Month == 7), "Wind"]
<- median(airquality_copy$Wind[which(airquality_copy$Month == 7)], na.rm = TRUE)
> airquality_NA <- airquality_copy[!complete.cases(airquality_copy),]
> airquality_NA
    Ozone Solar.R Wind Temp Month Day
5      17      NA 14.3   56     5   5
6      28      NA 14.9   66     5   6
7      23   299.0   NA   65     5   7
10     17      NA  8.6   69     5  10
11      7      NA  6.9   74     5  11
12     16   256.0  9.7   NA     5  12
20     11      NA  9.7   62     5  20
25     17    66.0   NA   57     5  25
27     17      NA  8.0   57     5  27
32     NA   286.0  8.6   78     6   1
33     NA   287.0  9.7   74     6   2
34     NA   242.0 16.1   67     6   3
35     NA   186.0  9.2   NA     6   4
36     NA   220.0  8.6   85     6   5
37     NA   264.0 14.3   79     6   6
39     NA   273.0  6.9   87     6   8
42     NA   259.0 10.9   93     6  11
43     NA   250.0  9.2   92     6  12
45     NA   332.0 13.8   80     6  14
46     NA   322.0 11.5   79     6  15
50     NA   120.0 11.5   73     6  19
52     NA   150.0  6.3   77     6  21
```

```
> airquality_copy[is.na(airquality_copy$Temp) & (airquality_copy$Month == 8), "Temp"]
<- median(airquality_copy$Temp[which(airquality_copy$Month == 8)], na.rm = TRUE)
> airquality_NA <- airquality_copy[!complete.cases(airquality_copy),]
> airquality_NA
    Ozone Solar.R Wind Temp Month Day
5      17      NA 14.3   56     5   5
6      28      NA 14.9   66     5   6
7      23   299.0   NA   65     5   7
10     17      NA  8.6   69     5  10
11      7      NA  6.9   74     5  11
12     16   256.0  9.7   NA     5  12
20     11      NA  9.7   62     5  20
25     17    66.0   NA   57     5  25
27     17      NA  8.0   57     5  27
32     NA   286.0  8.6   78     6   1
33     NA   287.0  9.7   74     6   2
34     NA   242.0 16.1   67     6   3
35     NA   186.0  9.2   NA     6   4
36     NA   220.0  8.6   85     6   5
37     NA   264.0 14.3   79     6   6
39     NA   273.0  6.9   87     6   8
42     NA   259.0 10.9   93     6  11
43     NA   250.0  9.2   92     6  12
45     NA   332.0 13.8   80     6  14
46     NA   322.0 11.5   79     6  15
50     NA   120.0 11.5   73     6  19
```

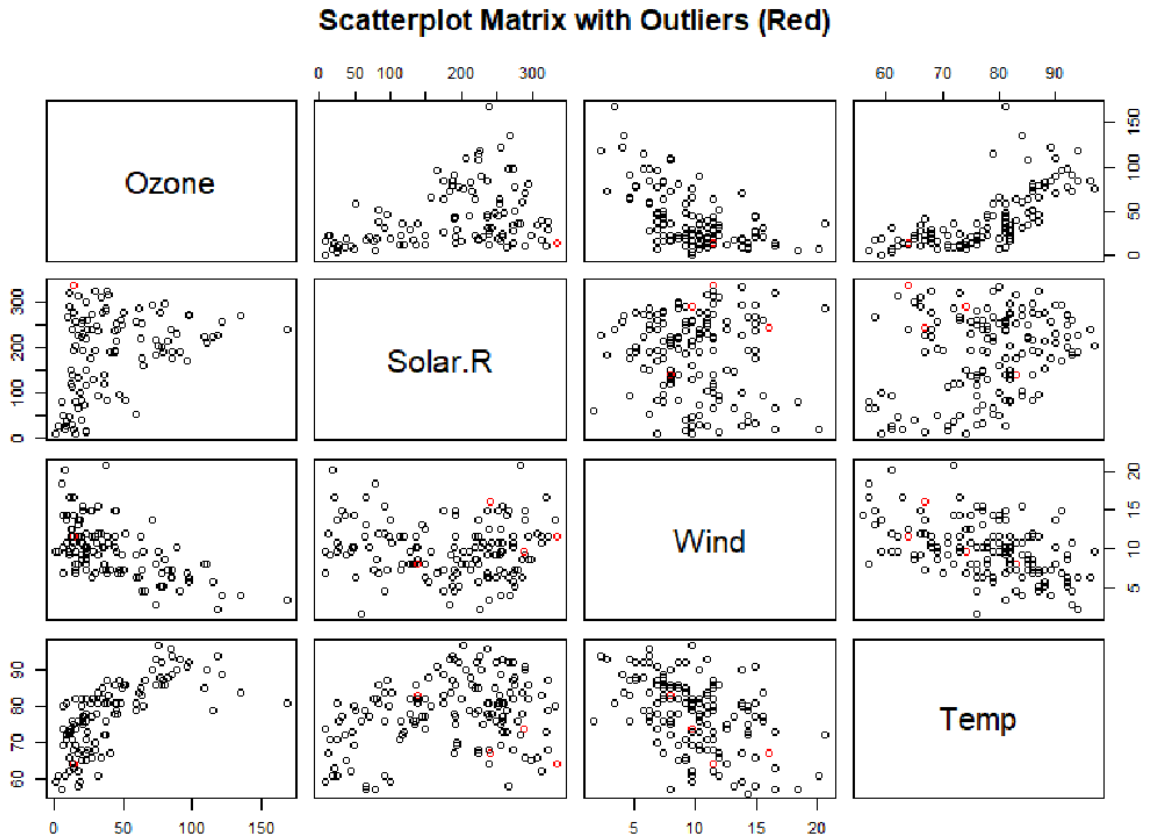


```
> # Detecting Outliers using Boxplot Method
> boxplot(airquality[, 1:4], main = "Boxplot for AirQuality Features", col = "lightbl
ue")
>
> detect_outliers <- function(x) {
+   Q1 <- quantile(x, 0.25, na.rm = TRUE) # First quartile (25%)
+   Q3 <- quantile(x, 0.75, na.rm = TRUE) # Third quartile (75%)
+   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(airquality[, 1:4], detect_outliers)
> outliers
$Ozone
[1]  62 117

$Solar.R
integer(0)

$Wind
[1]  9 18 48

$Temp
integer(0)
```



```
> # Scatterplot Matrix with Outliers Highlighted
> pairs(airquality[, 1:4], col = ifelse(rownames(airquality) %in% unlist(outliers),
"red", "black"), main = "Scatterplot Matrix with Outliers (Red)")
>
> # Install and Load Outliers Package
> library(outliers)
>
> outlier_value <- scores(airquality$Ozone, type = "z")
> outliers_ozone <- which(abs(outlier_value) > 3)
> outliers_ozone
integer(0)
>
> # Perform One-Hot Encoding
> one_hot_airquality <- model.matrix(~Month - 1, data = airquality)
> head(one_hot_airquality)
  Month
1     5
2     5
3     5
4     5
5     5
6     5
>
> # Convert to dataframe
> one_hot_airquality <- as.data.frame(one_hot_airquality)
>
> # Combine with original numerical features
> airquality_encoded <- cbind(airquality[, 1:4], one_hot_airquality)
>
> # View the updated dataset
> head(airquality_encoded)
  Ozone Solar.R Wind Temp Month
1    41     190  7.4   67     5
2    36     118  8.0   72     5
3    12     149 12.6   74     5
4    18     313 11.5   62     5
5    NA      NA 14.3   56     5
6    28      NA 14.9   66     5
>
> # Install and Load caret Package
> library(caret)
>
```

Red Colour part (commands) student should try

Conclusion:

Learnt various techniques for data cleaning, transformation, and feature engineering. These included handling missing values, detecting outliers, performing one-hot encoding, label encoding, feature scaling (min-max and z-score normalization), and generating polynomial features. By applying these techniques to the Iris dataset, students learned how to prepare data for machine learning models, ensuring better model performance and insights.

Post Lab Question

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

• Kernel Methods: Use SVMs with non-linear kernels (e.g., RBF) to implicitly map data to higher dimensions.
• Neural Networks: Use layers with non-linear activation functions (e.g., ReLU) for complex patterns.
• Local Regression (LOESS): Fit simple models to localized data subsets.
• Bayesian Models: Use Gaussian Processes for non-linear relationships with uncertainty estimates.
• Rule-Based Systems: Build interpretable non-linear models with decision rules.

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

Univariate Outlier Detection:
• Focuses on a single feature at a time.
• Uses statistical methods like Z-scores, IQR (Interquartile Range), or boxplots to identify extreme values.
• Ignores relationships between features, so it may miss outliers that are only apparent when considering multiple features together.

Multivariate Outlier Detection:
• Considers multiple features simultaneously.
• Captures relationships and interactions between features.
• Better suited for detecting complex outliers that depend on the interaction of multiple variables.
Key Difference: Univariate is simpler and feature-specific, while multivariate is more comprehensive and accounts for feature relationships.