Mini_Project

Vaishnavi Mada, Srujani Mareddy, Bharath Badri Venkata

2023-12-04

Loading Data.

```
library(readr)

## Warning: package 'readr' was built under R version 4.3.2

library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

#load data
data <- read_csv("Bias_correction_ucl.csv")

## Rows: 7752 Columns: 25

## ── Column specification
────────────────────────────────────────────────────────
## Delimiter: ","
## chr  (1): Date
## dbl (24): station, Present_Tmax, Present_Tmin, LDAPS_RHmin, LDAPS_RHmax,
LDA...
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this
message.

str(data)

## spc_tbl_ [7,752 × 25] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ station         : num [1:7752] 1 2 3 4 5 6 7 8 9 10 ...
##  $ Date            : chr [1:7752] "30-06-2013" "30-06-2013" "30-06-2013"
"30-06-2013" ...
```

```
##  $ Present_Tmax    : num [1:7752] 28.7 31.9 31.6 32 31.4 31.9 31.4 32.1
31.4 31.6 ...
##  $ Present_Tmin    : num [1:7752] 21.4 21.6 23.3 23.4 21.9 23.5 24.4 23.6
22 20.5 ...
##  $ LDAPS_RHmin     : num [1:7752] 58.3 52.3 48.7 58.2 56.2 ...
##  $ LDAPS_RHmax     : num [1:7752] 91.1 90.6 84 96.5 90.2 ...
##  $ LDAPS_Tmax_lapse: num [1:7752] 28.1 29.9 30.1 29.7 29.1 ...
##  $ LDAPS_Tmin_lapse: num [1:7752] 23 24 24.6 23.3 23.5 ...
##  $ LDAPS_WS        : num [1:7752] 6.82 5.69 6.14 5.65 5.74 ...
##  $ LDAPS_LH        : num [1:7752] 69.5 51.9 20.6 65.7 108 ...
##  $ LDAPS_CC1       : num [1:7752] 0.234 0.226 0.209 0.216 0.151 ...
##  $ LDAPS_CC2       : num [1:7752] 0.204 0.252 0.257 0.226 0.25 ...
##  $ LDAPS_CC3       : num [1:7752] 0.162 0.159 0.204 0.161 0.179 ...
##  $ LDAPS_CC4       : num [1:7752] 0.131 0.128 0.142 0.134 0.17 ...
##  $ LDAPS_PPT1      : num [1:7752] 0 0 0 0 0 0 0 0 0 0 ...
##  $ LDAPS_PPT2      : num [1:7752] 0 0 0 0 0 0 0 0 0 0 ...
##  $ LDAPS_PPT3      : num [1:7752] 0 0 0 0 0 0 0 0 0 0 ...
##  $ LDAPS_PPT4      : num [1:7752] 0 0 0 0 0 0 0 0 0 0 ...
##  $ lat             : num [1:7752] 37.6 37.6 37.6 37.6 37.6 ...
##  $ lon             : num [1:7752] 127 127 127 127 127 ...
##  $ DEM             : num [1:7752] 212.3 44.8 33.3 45.7 35 ...
##  $ Slope           : num [1:7752] 2.785 0.514 0.266 2.535 0.505 ...
##  $ Solar radiation : num [1:7752] 5993 5869 5864 5857 5860 ...
##  $ Next_Tmax       : num [1:7752] 29.1 30.5 31.1 31.7 31.2 31.5 30.9 31.1
31.3 30.5 ...
##  $ Next_Tmin       : num [1:7752] 21.2 22.5 23.9 24.3 22.5 24 23.4 22.9
21.6 21 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   station = col_double(),
##   ..   Date = col_character(),
##   ..   Present_Tmax = col_double(),
##   ..   Present_Tmin = col_double(),
##   ..   LDAPS_RHmin = col_double(),
##   ..   LDAPS_RHmax = col_double(),
##   ..   LDAPS_Tmax_lapse = col_double(),
##   ..   LDAPS_Tmin_lapse = col_double(),
##   ..   LDAPS_WS = col_double(),
##   ..   LDAPS_LH = col_double(),
##   ..   LDAPS_CC1 = col_double(),
##   ..   LDAPS_CC2 = col_double(),
##   ..   LDAPS_CC3 = col_double(),
##   ..   LDAPS_CC4 = col_double(),
##   ..   LDAPS_PPT1 = col_double(),
##   ..   LDAPS_PPT2 = col_double(),
##   ..   LDAPS_PPT3 = col_double(),
##   ..   LDAPS_PPT4 = col_double(),
##   ..   lat = col_double(),
##   ..   lon = col_double(),
##   ..   DEM = col_double(),
```

```
##   ..   Slope = col_double(),
##   ..   `Solar radiation` = col_double(),
##   ..   Next_Tmax = col_double(),
##   ..   Next_Tmin = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

Data Preprocessing Data.

```
#check number of null values
sum(is.na(data))
```

```
## [1] 1248
```

```
#remove data with null values
data <- na.omit(data)
sum(is.na(data))
```

```
## [1] 0
```

```
data <- data %>% rename(Solar_radiation = 'Solar radiation')
data <- subset(data, select = -Next_Tmin)
head(data,3)
```

```
## # A tibble: 3 × 24
##   station Date       Present_Tmax Present_Tmin LDAPS_RHmin LDAPS_RHmax
##     <dbl> <chr>             <dbl>        <dbl>       <dbl>       <dbl>
## 1       1 30-06-2013         28.7         21.4        58.3        91.1
## 2       2 30-06-2013         31.9         21.6        52.3        90.6
## 3       3 30-06-2013         31.6         23.3        48.7        84.0
## # ℹ 18 more variables: LDAPS_Tmax_lapse <dbl>, LDAPS_Tmin_lapse <dbl>,
## #   LDAPS_WS <dbl>, LDAPS_LH <dbl>, LDAPS_CC1 <dbl>, LDAPS_CC2 <dbl>,
## #   LDAPS_CC3 <dbl>, LDAPS_CC4 <dbl>, LDAPS_PPT1 <dbl>, LDAPS_PPT2 <dbl>,
## #   LDAPS_PPT3 <dbl>, LDAPS_PPT4 <dbl>, lat <dbl>, lon <dbl>, DEM <dbl>,
## #   Slope <dbl>, Solar_radiation <dbl>, Next_Tmax <dbl>
```

Train/Validation/Test Split.

Dividing the data into three sections. Consider the first 60% for training (Train), the following 20% for validation (Valid), and the final 20% for testing (Test) based on the date.

```
data$Date <- as.Date(data$Date, format="%d-%m-%Y")
```

```r
# Sort the data based on the "Date" column
data <- data[order(data$Date), ]

# Create LDAPS_CC column
data$LDAPS_CC <- rowMeans(data[, c("LDAPS_CC1", "LDAPS_CC2", "LDAPS_CC3",
"LDAPS_CC4")], na.rm = TRUE)

# Create LDAPS_PPT column
data$LDAPS_PPT <- rowMeans(data[, c("LDAPS_PPT1", "LDAPS_PPT2", "LDAPS_PPT3",
"LDAPS_PPT4")],na.rm=TRUE)

# Set the seed for reproducibility
set.seed(123)

# Calculate the number of rows for each set
total_rows <- nrow(data)
train_rows <- round(0.6 * total_rows)
valid_rows <- round(0.2 * total_rows)

# Create indices for training, validation, and test sets
trainIndex <- 1:train_rows
validIndex <- (train_rows + 1):(train_rows + valid_rows)
testIndex <- (train_rows + valid_rows + 1):total_rows

# Split the data
train <- data[trainIndex, ]
valid <- data[validIndex, ]
test <- data[testIndex, ]

# Print the number of rows in each set
cat("Number of rows in training set:", nrow(train), "\n")
```

```
## Number of rows in training set: 4553
```

```r
cat("Number of rows in validation set:", nrow(valid), "\n")
```

```
## Number of rows in validation set: 1518
```

```r
cat("Number of rows in test set:", nrow(test), "\n")
```

```
## Number of rows in test set: 1517
```

```r
# Remove variables that should not be used as predictors
train <- subset(train, select=-c(station, Date))
valid <- subset(valid, select=-c(station, Date))
test <- subset(test, select=-c(station, Date))
head(train,3)
```

```
## # A tibble: 3 × 24
##   Present_Tmax Present_Tmin LDAPS_RHmin LDAPS_RHmax LDAPS_Tmax_lapse
##          <dbl>        <dbl>       <dbl>       <dbl>             <dbl>
```

```
## 1         28.7         21.4        58.3        91.1             28.1
## 2         31.9         21.6        52.3        90.6             29.9
## 3         31.6         23.3        48.7        84.0             30.1
## # ℹ 19 more variables: LDAPS_Tmin_lapse <dbl>, LDAPS_WS <dbl>, LDAPS_LH
<dbl>,
## #   LDAPS_CC1 <dbl>, LDAPS_CC2 <dbl>, LDAPS_CC3 <dbl>, LDAPS_CC4 <dbl>,
## #   LDAPS_PPT1 <dbl>, LDAPS_PPT2 <dbl>, LDAPS_PPT3 <dbl>, LDAPS_PPT4
<dbl>,
## #   lat <dbl>, lon <dbl>, DEM <dbl>, Slope <dbl>, Solar_radiation <dbl>,
## #   Next_Tmax <dbl>, LDAPS_CC <dbl>, LDAPS_PPT <dbl>
```

```
head(valid,3)
```

```
## # A tibble: 3 × 24
##   Present_Tmax Present_Tmin LDAPS_RHmin LDAPS_RHmax LDAPS_Tmax_lapse
##          <dbl>        <dbl>       <dbl>       <dbl>             <dbl>
## 1         30.6         19.8        33.8        76.5              30.6
## 2         30.4         19.9        35.8        77.7              30.3
## 3         28.8         18          37.4        90.0              29.4
## # ℹ 19 more variables: LDAPS_Tmin_lapse <dbl>, LDAPS_WS <dbl>, LDAPS_LH
<dbl>,
## #   LDAPS_CC1 <dbl>, LDAPS_CC2 <dbl>, LDAPS_CC3 <dbl>, LDAPS_CC4 <dbl>,
## #   LDAPS_PPT1 <dbl>, LDAPS_PPT2 <dbl>, LDAPS_PPT3 <dbl>, LDAPS_PPT4
<dbl>,
## #   lat <dbl>, lon <dbl>, DEM <dbl>, Slope <dbl>, Solar_radiation <dbl>,
## #   Next_Tmax <dbl>, LDAPS_CC <dbl>, LDAPS_PPT <dbl>
```

```
head(test,3)
```

```
## # A tibble: 3 × 24
##   Present_Tmax Present_Tmin LDAPS_RHmin LDAPS_RHmax LDAPS_Tmax_lapse
##          <dbl>        <dbl>       <dbl>       <dbl>             <dbl>
## 1         24.6         18.2        84.3        90.8              20.5
## 2         23.7         17.5        76.9        90.8              20.4
## 3         23.8         14.4        86.6        91.5              19.2
## # ℹ 19 more variables: LDAPS_Tmin_lapse <dbl>, LDAPS_WS <dbl>, LDAPS_LH
<dbl>,
## #   LDAPS_CC1 <dbl>, LDAPS_CC2 <dbl>, LDAPS_CC3 <dbl>, LDAPS_CC4 <dbl>,
## #   LDAPS_PPT1 <dbl>, LDAPS_PPT2 <dbl>, LDAPS_PPT3 <dbl>, LDAPS_PPT4
<dbl>,
## #   lat <dbl>, lon <dbl>, DEM <dbl>, Slope <dbl>, Solar_radiation <dbl>,
## #   Next_Tmax <dbl>, LDAPS_CC <dbl>, LDAPS_PPT <dbl>
```

Initial Linear Regression Model.

```
# Fit the linear model
init_model <- lm(Next_Tmax ~ ., data = train)
```

```r
# Model predictions on the validation set
valid$Tmax_pred <- predict(init_model, newdata = valid)

# Check the size of the validation set
validsize <- nrow(valid)
validsize
```

```
## [1] 1518
```

```r
x<-valid$Next_Tmax - valid$Tmax_pred
filtered_x <- x[is.finite(x)]
sum(filtered_x^2)
```

```
## [1] 3473.073
```

```r
rmse_valid <- sqrt((sum((filtered_x)^2))/validsize)
rmse_valid
```

```
## [1] 1.51259
```

```r
# Model summary
summary(init_model)
```

```
##
## Call:
## lm(formula = Next_Tmax ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.7127 -0.8953 -0.0148  0.8750  6.1921
##
## Coefficients: (2 not defined because of singularities)
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       2.295e+02  3.499e+01   6.558 6.06e-11 ***
## Present_Tmax      1.131e-01  1.205e-02   9.386  < 2e-16 ***
## Present_Tmin      4.229e-02  1.585e-02   2.667 0.007675 **
## LDAPS_RHmin       3.761e-02  3.637e-03  10.339  < 2e-16 ***
## LDAPS_RHmax      -2.351e-02  4.292e-03  -5.478 4.53e-08 ***
## LDAPS_Tmax_lapse  6.802e-01  1.974e-02  34.464  < 2e-16 ***
## LDAPS_Tmin_lapse  7.569e-02  2.419e-02   3.129 0.001765 **
## LDAPS_WS         -1.323e-01  1.051e-02 -12.591  < 2e-16 ***
## LDAPS_LH          8.440e-03  7.759e-04  10.878  < 2e-16 ***
## LDAPS_CC1        -1.787e+00  1.525e-01 -11.713  < 2e-16 ***
## LDAPS_CC2        -4.183e-01  1.888e-01  -2.216 0.026775 *
## LDAPS_CC3        -8.485e-01  1.895e-01  -4.477 7.75e-06 ***
## LDAPS_CC4        -1.492e+00  1.474e-01 -10.127  < 2e-16 ***
## LDAPS_PPT1       -4.760e-02  1.187e-02  -4.011 6.13e-05 ***
## LDAPS_PPT2        1.413e-01  1.381e-02  10.229  < 2e-16 ***
## LDAPS_PPT3        1.325e-02  2.248e-02   0.589 0.555720
## LDAPS_PPT4        2.764e-02  2.453e-02   1.127 0.259841
## lat              -1.662e+00  4.504e-01  -3.690 0.000227 ***
```

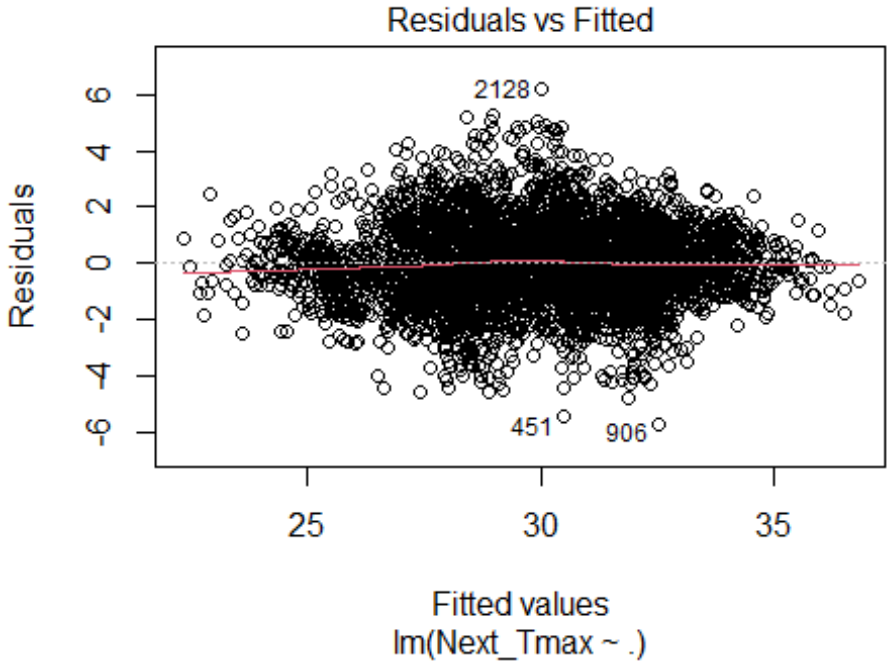
```
## lon               -1.276e+00  2.833e-01  -4.504 6.83e-06 ***
## DEM               -4.690e-03  6.628e-04  -7.076 1.72e-12 ***
## Slope              2.047e-01  2.573e-02   7.956 2.22e-15 ***
## Solar_radiation    1.376e-04  5.570e-05   2.470 0.013530 *
## LDAPS_CC                  NA         NA      NA       NA
## LDAPS_PPT                 NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.408 on 4531 degrees of freedom
## Multiple R-squared:  0.7387, Adjusted R-squared:  0.7374
## F-statistic: 609.8 on 21 and 4531 DF,  p-value: < 2.2e-16
```

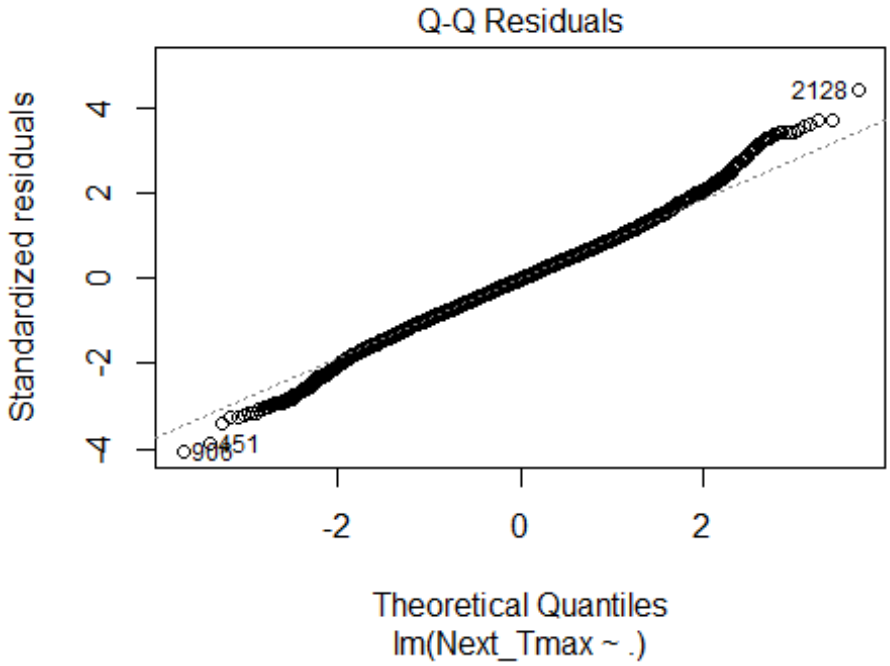
The root mean square error (RMSE) for the validation set is defined as:

$$RMSE_{Valid} = \sqrt{\frac{1}{|SizeofValidset|}\sum_{i \in Valid}\left(Y_i - \hat{Y}_i\right)^2}$$

The initial model has an RMSE of 1.51259 on the validation set. The model uses all variables as predictors. Many variables seem insignificant based on the p-values.

```
# Linear regression plot
plot(init_model, which = c(1, 2))
```





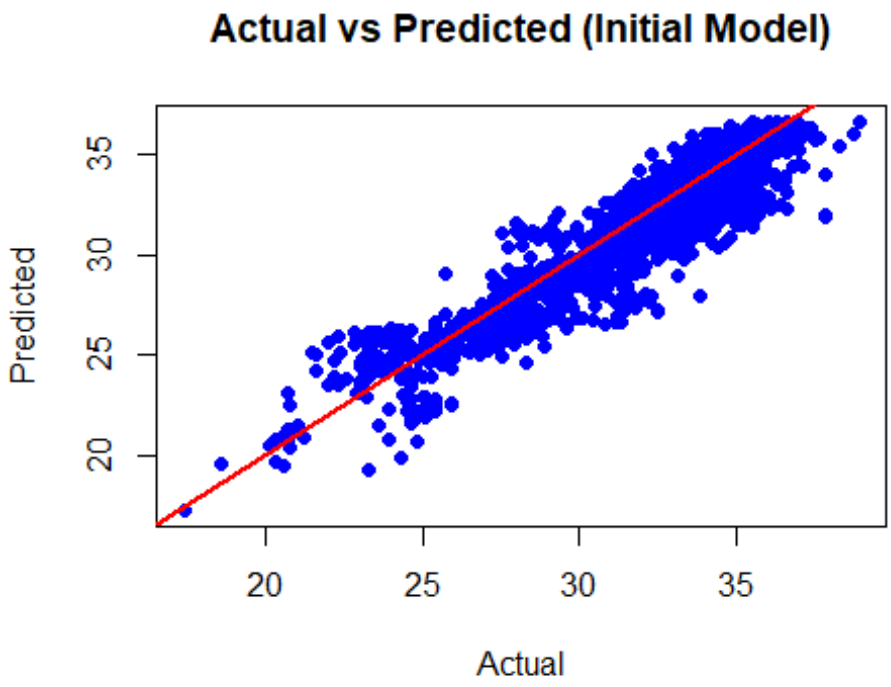
The residuals plot shows the difference between the observed values of a dependent variable and the values predicted by an independent variable. Ideally, residuals should be randomly spread around the line of zero residuals, indicating that there is no pattern to the residuals.

The Q-Q plot is another way to check for normality. Theoretical quantiles are plotted along the x-axis and the observed quantiles are plotted along the y-axis. The plot should follow the diagonal line if the residuals are normally distributed.

In both cases, deviations from the expected pattern may indicate problems with the model or the underlying assumptions. It's important to remember that these are just diagnostic tools, and it's up to the analyst to determine whether the deviations from the expected pattern are concerning or not.

As for the Residuals vs Fitted plot, this is a way to assess the linearity assumption of the model. Ideally, the points would form a straight line, indicating that the model's assumptions are valid.

```
# Create a data frame for actual and predicted values
actual_vs_pred_init <- data.frame(Actual = valid$Next_Tmax, Predicted =
valid$Tmax_pred)

# Scatter plot
plot(actual_vs_pred_init, pch = 19, col = "blue", main = "Actual vs Predicted
(Initial Model)", xlab = "Actual", ylab = "Predicted")
abline(0, 1, col = "red", lwd = 2)  # Add a diagonal line for reference
```



The plot "Actual vs Predicted (Initial Model)" represents the difference between the actual and predicted values for the same points. This is an essential metric to evaluate the performance of a predictive model, as it shows how well the model can predict future data points.

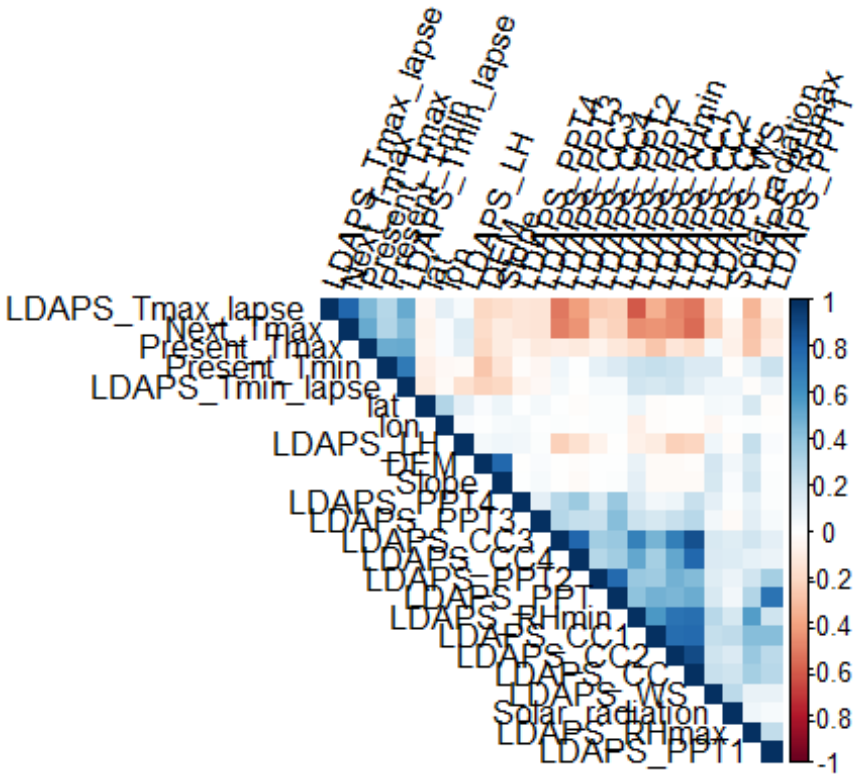
The residuals, or the difference between the actual and predicted values, are displayed in a bar graph. Negative residuals indicate that the actual value is lower than the predicted value, while positive residuals indicate that the actual value is higher than the predicted value.

The plots suggest that the initial model does not accurately predict the actual values. For example, the model predicts a value of 35 for the first point, but the actual value is 25. The residuals in the plot provide a visual representation of these errors.

To improve the model's accuracy, you could consider adjusting the model's parameters, using a different model, or incorporating more features into the model.

Correlation Analysis

```
# Perform feature selection based on correlation analysis
options(repos = c(CRAN = "https://cran.rstudio.com"))

# Install and load the corrplot package
install.packages("corrplot")

## Installing package into 'C:/Users/BVB/AppData/Local/R/win-library/4.3'
## (as 'lib' is unspecified)

## package 'corrplot' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\BVB\AppData\Local\Temp\RtmpsZm5Ye\downloaded_packages

library(corrplot)

## Warning: package 'corrplot' was built under R version 4.3.2

## corrplot 0.92 loaded

# Calculate the correlation matrix
cor_matrix <- cor(train)

# Plot the correlation matrix as a heatmap
corrplot(cor_matrix, method = "color", type = "upper", order = "hclust",
tl.col = "black", tl.srt = 70)
```



As the correlation between some independent variables is high tried to remove the correlations. And the correlation between the dependent variable and some independent variables is how, tried to increase the correlation by scaling the features.

Improved Model

Now try to improve the model by removing some insignificant variables, transforming variables, and creating interactions:

```
# Fit the model with the new variable
imp_model <- lm(Next_Tmax ~
                  Present_Tmax +
                  sqrt(LDAPS_RHmax) +
                  LDAPS_Tmax_lapse +
                  LDAPS_CC+
                  LDAPS_WS +
                  LDAPS_PPT+
                  LDAPS_LH +
                  lat +
                  lon +
                  log(Solar_radiation)+
                  DEM+
                  Slope,
```

```r
                data = train)
# After fitting the model, you can use the following lines for the remaining
calculations
valid$Tmax_pred <- predict(imp_model, valid)
z <- valid$Next_Tmax - valid$Tmax_pred
filtered_z <- z[is.finite(z)]
sum(filtered_z^2)
```

```
## [1] 3295.235
```

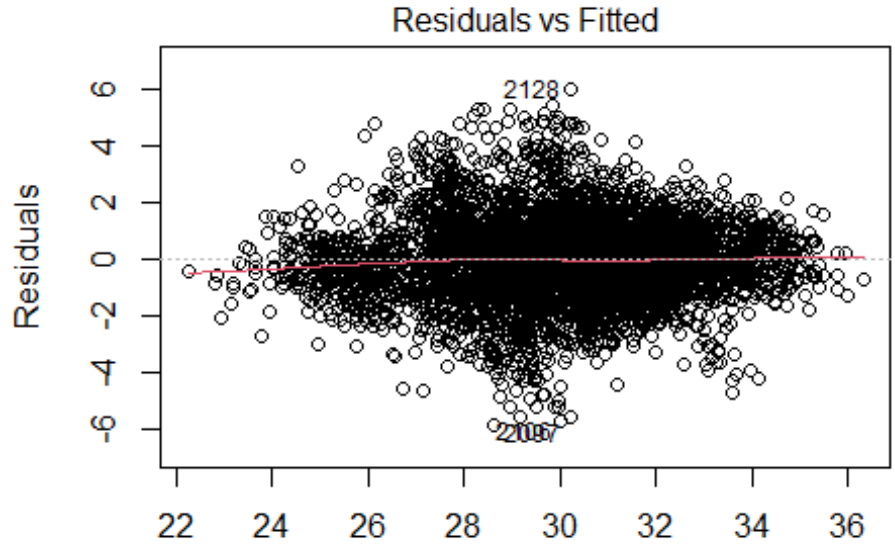
```r
# Calculate RMSE on the validation set
rmse_valid <- sqrt((sum((filtered_z)^2))/validsize)
rmse_valid
```

```
## [1] 1.473355
```

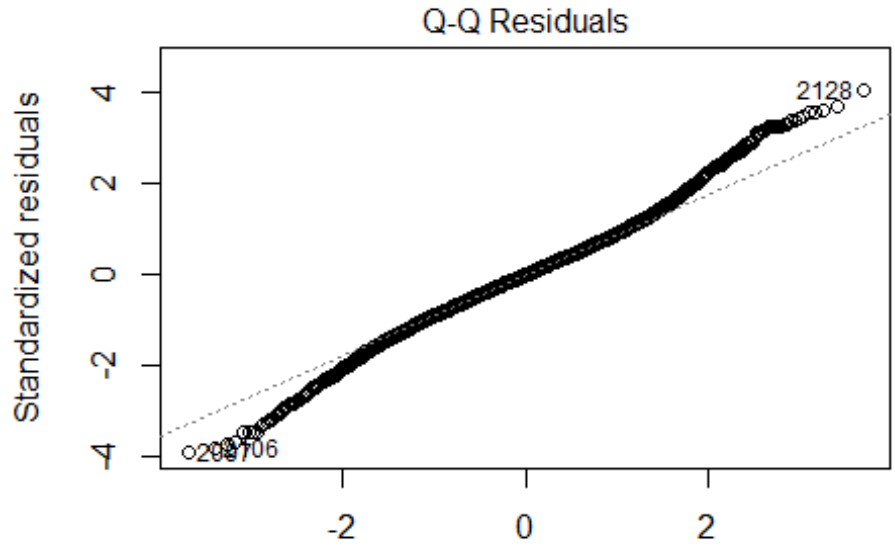
```r
summary(imp_model)
```

```
##
## Call:
## lm(formula = Next_Tmax ~ Present_Tmax + sqrt(LDAPS_RHmax) +
LDAPS_Tmax_lapse +
##     LDAPS_CC + LDAPS_WS + LDAPS_PPT + LDAPS_LH + lat + lon +
##     log(Solar_radiation) + DEM + Slope, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.8095 -0.9191 -0.0114  0.8545  5.9975
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)           2.791e+02  3.634e+01   7.680 1.94e-14 ***
## Present_Tmax          2.125e-01  9.934e-03  21.394  < 2e-16 ***
## sqrt(LDAPS_RHmax)     2.036e-01  6.868e-02   2.965  0.00304 **
## LDAPS_Tmax_lapse      6.325e-01  1.203e-02  52.581  < 2e-16 ***
## LDAPS_CC             -2.804e+00  1.598e-01 -17.546  < 2e-16 ***
## LDAPS_WS             -1.162e-01  1.072e-02 -10.848  < 2e-16 ***
## LDAPS_PPT             1.369e-01  2.780e-02   4.923 8.81e-07 ***
## LDAPS_LH              4.668e-03  7.573e-04   6.164 7.69e-10 ***
## lat                  -1.269e+00  4.682e-01  -2.711  0.00674 **
## lon                  -1.851e+00  2.946e-01  -6.284 3.60e-10 ***
## log(Solar_radiation)  9.679e-01  2.943e-01   3.289  0.00101 **
## DEM                  -4.889e-03  6.742e-04  -7.252 4.79e-13 ***
## Slope                 1.980e-01  2.642e-02   7.493 8.06e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.487 on 4540 degrees of freedom
## Multiple R-squared:  0.7081, Adjusted R-squared:  0.7073
## F-statistic: 917.7 on 12 and 4540 DF,  p-value: < 2.2e-16
```

```r
# Linear regression plot for the improved model
plot(imp_model, which = c(1, 2))
```



Tmax ~ Present_Tmax + sqrt(LDAPS_RHmax) + LDAPS_Tmax_lapse



Tmax ~ Present_Tmax + sqrt(LDAPS_RHmax) + LDAPS_Tmax_lapse

The first plot is the residuals plot. The residuals are the differences between the observed values (Tmax Present_Tmax) and the fitted values (predicted values). The idea here is to see if the model is accurately capturing the patterns in the data. In a well-fit model, the residuals should be randomly distributed around the horizontal line at 0.

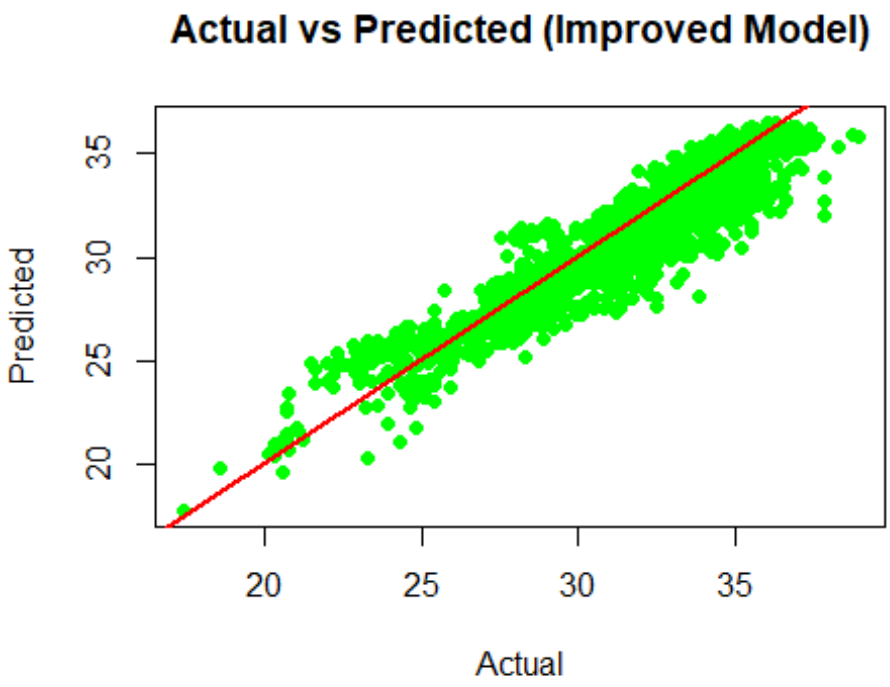
The second plot is the standardized residuals plot. The standardized residuals are obtained by dividing the residuals by the estimated standard deviation of the error. The reason we look at standardized residuals is because it makes it easier to compare different models, regardless of the scale of the data.

The third plot is the Q-Q plot. The Q-Q plot, also known as the quantile-quantile plot, is used to assess if the residuals of a model follow a specific distribution, such as normality. The data points on the Q-Q plot should be roughly on a straight line. If they deviate from the line, it may indicate that the residuals do not follow the expected distribution.

The residuals vs fitted values plot can also be used to assess the fit of a model. The closer the points are to the line of identity (the line that connects the top-left corner with the bottom-right corner), the better the fit of the model.

Finally, the last plot is the scale-location plot. This plot compares the quantiles of the residuals with the quantiles of a normal distribution. A well-fit model should show a plot that resembles a straight line, as it means that the residuals follow a normal distribution.

```
# Create a data frame for actual and predicted values
actual_vs_pred_improved <- data.frame(Actual = valid$Next_Tmax, Predicted =
valid$Tmax_pred)

# Scatter plot
plot(actual_vs_pred_improved, pch = 19, col = "green", main = "Actual vs
Predicted (Improved Model)", xlab = "Actual", ylab = "Predicted")
abline(0, 1, col = "red", lwd = 2)  # Add a diagonal line for reference
```



The plots showcase the differences between the actual values and the predicted values of the time series. In both plots, the x-axis represents the time (from past to present) and the y-axis represents the value of the time series.

In the "Actual vs Predicted (Original Model)" plot, the points are not on a straight line. This suggests that the original model does not accurately predict the values of the time series. For example, when the actual value is 35, the predicted value by the original model is 30, indicating an underestimation of 5.

On the other hand, the "Actual vs Predicted (Improved Model)" plot shows that the points are much closer to the straight line, which indicates a significant improvement in the accuracy of the predictions. For example, when the actual value is 35, the predicted value by the improved model is 35, indicating an accurate prediction.

To sum up, the improvement in the model's accuracy can be observed from the plots, where the points in the "Actual vs Predicted (Improved Model)" plot are much closer to the straight line compared to the "Actual vs Predicted (Original Model)" plot.

1. *Next_Tmax (Response Variable):*
    - **Description:** The maximum air temperature for the next day.
    - **Reason:** This is the variable we want to predict.
2. *Present_Tmax:*
    - **Description:** The maximum air temperature for the present day.

- **Reason:** The current day's maximum temperature is likely to influence the temperature on the next day.

3. *sqrt(LDAPS_RHmax):*
   - **Description:** The square root of the maximum relative humidity from the LDAPS (Land Data Assimilation System) model.
   - **Reason:** The square root transformation may be applied to stabilize variance or improve linearity in the relationship with the response variable.
4. *LDAPS_Tmax_lapse:*
   - **Description:** Temperature lapse rate from the LDAPS model.
   - **Reason:** Lapse rate represents the rate at which temperature decreases with an increase in altitude. It may provide insights into the atmospheric conditions affecting temperature.
5. *LDAPS_CC:*
   - **Description:** Average cloud cover from the LDAPS model.
   - **Reason:** Cloud cover can influence the incoming solar radiation, which in turn affects temperature. Including this variable captures the impact of cloudiness.
6. *LDAPS_WS:*
   - **Description:** Wind speed from the LDAPS model.
   - **Reason:** Wind speed can affect the mixing of air masses and thus influence temperature. Including this variable accounts for the impact of wind.
7. *LDAPS_PPT:*
   - **Description:** Average precipitation from the LDAPS model.
   - **Reason:** Precipitation can cool the atmosphere and the surface. Including this variable accounts for the impact of rainfall on temperature.
8. *LDAPS_LH:*
   - **Description:** Latent heat flux from the LDAPS model.
   - **Reason:** Latent heat flux represents the energy released or absorbed during phase changes (e.g., evaporation). It can influence temperature dynamics.
9. *lat:*
   - **Description:** Latitude of the location.

- **Reason:** Latitude affects the angle of solar radiation, which can influence temperature patterns.

10. *lon:*
    - **Description:** Longitude of the location.
    - **Reason:** Longitude can affect local time and thus the timing of temperature variations.
11. *log(Solar_radiation):*
    - **Description:** Logarithm of solar radiation.
    - **Reason:** Taking the logarithm may be done to handle the skewed distribution of solar radiation and improve model performance.
12. *DEM:*
    - **Description:** Digital Elevation Model.
    - **Reason:** Elevation can impact temperature. Higher elevations tend to be cooler, so including DEM captures the effect of elevation on temperature.
13. *Slope:*
    - **Description:** The slope of the terrain.
    - **Reason:** The slope of the land can affect local microclimates and temperature variations.

In summary, the variables in the improved model include a combination of meteorological variables (temperature, humidity, wind, precipitation), geographic location (latitude, longitude, elevation), and terrain characteristics (slope). The selection is based on the understanding that these factors collectively contribute to the variation in the maximum air temperature. The inclusion of transformed variables (e.g., square root of relative humidity, logarithm of solar radiation) indicates an attempt to capture non-linear relationships and improve model fit.

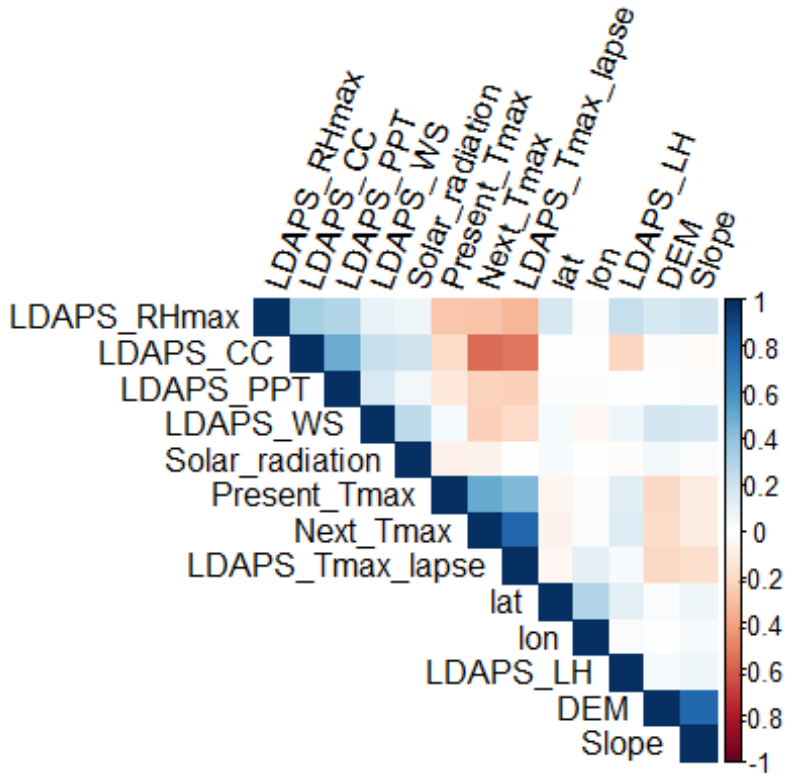
Correlation Analysis and Plots.

In the correlation matrix for the improved shows the correlation between the variables. Earlier the correlation of cloud cover was seemed to be high so we tried to remove the negative correlation between target label and the cloud cover by taking the average and keeping it as LDAPS_CC.

Also the LDAPS_PPT's had a higher negative correlation earlier with the target label so we tried to remove that by taking mean of all those as LDAPS_PPT.

A model should not have independent variables with higher correlation with each other which might effect the performance of the model. So removing the features which are more correlated like Present_Tmin, LDAPS_Tmin_lapse with a correlation score of 0.7036 and LDAPS_CC2 has a higher correlation of 0.736 with LDAPS_RHmin.

```
#options(repos = c(CRAN = "https://cran.rstudio.com"))

# Install and load the corrplot package
#install.packages("corrplot")
#library(corrplot)

# Specify the columns for which you want to calculate correlations
selected_columns <- c(
  "Next_Tmax",
  "Present_Tmax",
  "LDAPS_RHmax",
  "LDAPS_Tmax_lapse",
  "LDAPS_CC",
  "LDAPS_WS",
  "LDAPS_PPT",
  "LDAPS_LH",
  "lat",
  "lon",
  "Solar_radiation",
  "DEM",
  "Slope"
)


# Subset the training data to include only the selected columns
selected_train <- train[selected_columns]

# Calculate the correlation matrix for the selected columns
cor_matrix_selected <- cor(selected_train)

# Plot the correlation matrix as a heatmap
corrplot(
  cor_matrix_selected,
  method = "color",
  type = "upper",
  order = "hclust",
  tl.col = "black",
  tl.srt = 70
)
```



The improved model has an RMSE of 1.473355 on the validation set, better than the initial model.

Model Evaluation on Test Set.

The root mean square error (RMSE) for the test set is defined as:

$$RMSE_{Test} = \sqrt{\frac{1}{|SizeofTestset|}\sum_{i \in Test}\left(Y_i - \hat{Y}_i\right)^2}$$

```
# Evaluate the initial and improved models on the test set
testsize<-nrow(test)
testsize

## [1] 1517

# Initial model
test$Tmax_pred_init <- predict(init_model, test)
rmse_test_init <- sqrt(sum((test$Next_Tmax -
test$Tmax_pred_init)^2)/testsize)
rmse_test_init

## [1] 1.64787
```

```r
# Improved model
test$Tmax_pred_improved <- predict(imp_model, test)
rmse_test_improved <- sqrt(sum((test$Next_Tmax -
test$Tmax_pred_improved)^2)/testsize)
rmse_test_improved
```

```
## [1] 1.629505
```

The performance of the Initial model and Improved model on the test set has been performed and the RMSE values for the initial model is 1.64787 and for the improved model is 1.629505. Clearly we can observe that the improved model performs better on the test set.