Assignments_4 Course

Mathematical and Computational Statistics

Achladianakis Minas

A report presented for the Course Mathematical and Computational Statistics

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
UNIVERSITY OF CRETE

Department of Applied Mathematics
University of Crete

Greece, October 27, 2023

Abstract

This assignment is part of my evaluation for the course: Mathematical and Computational Statistics.

Assignment 1

1.1 Part 1

From the build-in R dataset "air quality" Develop 3 predictive models (M1), (M2), and (M3) for Ozone using just the intercept, the Solar.R, Wind, Temp and all the second order terms based on Solar.R, Wind, Temp respectively. Then report estimated coefficients using least squares (LS) and least absolute deviations (LAD):

Our Dataset:

```
# Load the air quality dataset
data("airquality")

# Display the first few rows of the dataset
head(airquality)

# Check for missing values
summary(airquality)
```

There are missing data in our dataset:

```
> summary(airquality)
     Ozone           Solar.R          Wind             Temp
      ↪  Month            Day
 Min.   :  1.00   Min.   :  7.0   Min.   : 1.700   Min.   :56.00   Min.
 ↪  :5.000   Min.   : 1.0
 1st Qu.: 18.00   1st Qu.:115.8   1st Qu.: 7.400   1st Qu.:72.00   1st
 ↪  Qu.:6.000   1st Qu.: 8.0
 Median : 31.50   Median :205.0   Median : 9.700   Median :79.00   Median
 ↪  :7.000   Median :16.0
 Mean   : 42.13   Mean   :185.9   Mean   : 9.958   Mean   :77.88   Mean
 ↪  :6.993   Mean   :15.8
 3rd Qu.: 63.25   3rd Qu.:258.8   3rd Qu.:11.500   3rd Qu.:85.00   3rd
 ↪  Qu.:8.000   3rd Qu.:23.0
 Max.   :168.00   Max.   :334.0   Max.   :20.700   Max.   :97.00   Max.
 ↪  :9.000   Max.   :31.0
 NA's   :37       NA's   :7
```

, which I intend to fill with the mean value of each respected column using the below function:

```r
# Function to replace missing values with mean of the column
replace_na_with_mean <- function(x) {
  if (is.numeric(x)) {
    x[is.na(x)] <- mean(x, na.rm = TRUE)
  }
  return(x)
}

# Apply the function to each column of the airquality dataset
airquality_clean <- apply(airquality, 2, replace_na_with_mean)

# Convert the result back to a data frame (apply returns a matrix)
airquality_clean <- as.data.frame(airquality_clean)

# Check the result
head(airquality_clean)
```

our dataset now has the form:

```
> summary(airquality_clean)
     Ozone           Solar.R          Wind             Temp
      ↪  Month            Day
 Min.   :  1.00   Min.   :  7.0   Min.   : 1.700   Min.   :56.00   Min.
 ↪  :5.000   Min.   : 1.0
 1st Qu.: 21.00   1st Qu.:120.0   1st Qu.: 7.400   1st Qu.:72.00   1st
 ↪  Qu.:6.000   1st Qu.: 8.0
 Median : 42.13   Median :194.0   Median : 9.700   Median :79.00   Median
 ↪  :7.000   Median :16.0
 Mean   : 42.13   Mean   :185.9   Mean   : 9.958   Mean   :77.88   Mean
 ↪  :6.993   Mean   :15.8
 3rd Qu.: 46.00   3rd Qu.:256.0   3rd Qu.:11.500   3rd Qu.:85.00   3rd
 ↪  Qu.:8.000   3rd Qu.:23.0
 Max.   :168.00   Max.   :334.0   Max.   :20.700   Max.   :97.00   Max.
 ↪  :9.000   Max.   :31.0
```

Our models:

```r
# Load required library for linear modeling
library(lmtest)

# Model 1: Only intercept
M1 <- lm(Ozone ~ 1, data = airquality_clean)

# Model 2: Ozone ~ Solar.R + Wind + Temp
M2 <- lm(Ozone ~ Solar.R + Wind + Temp, data = airquality_clean)

# Model 3: Including second-order terms and interactions
M3 <- lm(Ozone ~ poly(Solar.R, 2) + poly(Wind, 2) + poly(Temp, 2) +
 ↪  Solar.R:Wind + Solar.R:Temp + Wind:Temp, data = airquality_clean)
```

with summaries:

```
> # Summary of Model 1
> summary(M1)

Call:
lm(formula = Ozone ~ 1, data = airquality_clean)

Residuals:
    Min      1Q  Median      3Q     Max
-41.129 -21.129   0.000   3.871 125.871

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    42.13       2.32   18.16   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 28.69 on 152 degrees of freedom

> summary(M2)

Call:
lm(formula = Ozone ~ Solar.R + Wind + Temp, data = airquality_clean)

Residuals:
    Min      1Q  Median      3Q     Max
-38.618 -14.491  -5.054  12.270 101.176

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -38.22315   18.88338  -2.024  0.04474 *
Solar.R       0.05775    0.02003   2.883  0.00452 **
Wind         -2.71725    0.54280  -5.006 1.55e-06 ***
Temp          1.24126    0.20906   5.937 1.96e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 20.9 on 149 degrees of freedom
Multiple R-squared:   0.48,        Adjusted R-squared:  0.4696
F-statistic: 45.85 on 3 and 149 DF,  p-value: < 2.2e-16

> # Summary of Model 3
> summary(M3)

Call:
lm(formula = Ozone ~ poly(Solar.R, 2) + poly(Wind, 2) + poly(Temp,
    2) + Solar.R:Wind + Solar.R:Temp + Wind:Temp, data = airquality_clean)

Residuals:
    Min      1Q  Median      3Q     Max
-37.242 -11.536  -2.992   9.319  84.164
```

```
Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)         5.058e+01  5.904e+01   0.857  0.39303
poly(Solar.R, 2)1   1.310e+02  2.111e+02   0.621  0.53589
poly(Solar.R, 2)2  -2.721e+01  2.059e+01  -1.322  0.18841
poly(Wind, 2)1     -2.249e+01  2.261e+02  -0.099  0.92089
poly(Wind, 2)2      8.194e+01  2.461e+01   3.329  0.00111 **
poly(Temp, 2)1      1.148e+02  9.933e+01   1.156  0.24960
poly(Temp, 2)2      5.837e+01  2.441e+01   2.391  0.01811 *
Solar.R:Wind       -1.115e-02  5.592e-03  -1.994  0.04808 *
Solar.R:Temp        8.931e-04  2.265e-03   0.394  0.69396
Wind:Temp          -1.486e-03  7.155e-02  -0.021  0.98346
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 18.25 on 143 degrees of freedom
Multiple R-squared:  0.6195,        Adjusted R-squared:  0.5956
F-statistic: 25.87 on 9 and 143 DF,  p-value: < 2.2e-16
```

To estimate our coefficients (in R), we can follow the steps below:

```r
# Load required library for quantile regression
library(quantreg)

# LAD regression for Model 1
M1_LAD <- rq(Ozone ~ 1, data = airquality_clean)

# LAD regression for Model 2
M2_LAD <- rq(Ozone ~ Solar.R + Wind + Temp, data = airquality_clean)

# LAD regression for Model 3
M3_LAD <- rq(Ozone ~ poly(Solar.R, 2) + poly(Wind, 2) + poly(Temp, 2) + Solar.R:Wind
```

with:

```
> # Summary of LAD regression for Model 1
> summary(M1_LAD)

Call: rq(formula = Ozone ~ 1, data = airquality_clean)

tau: [1] 0.5

Coefficients:
(Intercept)
   42.12931

> # Summary of LAD regression for Model 2
> summary(M2_LAD)

Call: rq(formula = Ozone ~ Solar.R + Wind + Temp, data = airquality_clean)
```

```
tau: [1] 0.5

Coefficients:
            coefficients lower bd  upper bd
(Intercept) -59.39010    -86.46472 -28.91446
Solar.R       0.03886      0.00763   0.06820
Wind         -1.95360     -3.06336  -1.07138
Temp          1.40590      0.99392   1.84861
```

```
> # Summary of LAD regression for Model 3
> summary(M3_LAD)

Call: rq(formula = Ozone ~ poly(Solar.R, 2) + poly(Wind, 2) + poly(Temp,
    2) + Solar.R:Wind + Solar.R:Temp + Wind:Temp, data = airquality_clean)

tau: [1] 0.5

Coefficients:
                    coefficients lower bd   upper bd
(Intercept)           87.08141    -43.07564  172.17927
poly(Solar.R, 2)1    -14.01817   -221.31128  525.36794
poly(Solar.R, 2)2    -33.58878    -50.19975   12.25193
poly(Wind, 2)1       204.51227   -253.32552  537.28237
poly(Wind, 2)2        57.81626     16.59135   81.56295
poly(Temp, 2)1       213.68468     17.57138  355.43205
poly(Temp, 2)2        57.62070     35.21886  122.69884
Solar.R:Wind          -0.00251     -0.00940    0.00345
Solar.R:Temp           0.00128     -0.00403    0.00332
Wind:Temp             -0.08130     -0.19031    0.06450
```

We can follow the same approach for the (LAD):

```
# Load required library for quantile regression
library(quantreg)

# LAD regression for Model 1
M1_LAD <- rq(Ozone ~ 1, data = airquality_clean)

# LAD regression for Model 2
M2_LAD <- rq(Ozone ~ Solar.R + Wind + Temp, data = airquality_clean)

# LAD regression for Model 3
M3_LAD <- rq(Ozone ~ poly(Solar.R, 2) + poly(Wind, 2) + poly(Temp, 2) +
↪  Solar.R:Wind + Solar.R:Temp + Wind:Temp, data = airquality_clean)
```

with coefficients:

```
> # Summary of LAD regression for Model 1
> summary(M1_LAD)
```

```
Call: rq(formula = Ozone ~ 1, data = airquality_clean)

tau: [1] 0.5

Coefficients:
(Intercept)
   42.12931

> # Summary of LAD regression for Model 2
> summary(M2_LAD)

Call: rq(formula = Ozone ~ Solar.R + Wind + Temp, data = airquality_clean)

tau: [1] 0.5

Coefficients:
            coefficients lower bd  upper bd
(Intercept) -59.39010    -86.46472 -28.91446
Solar.R       0.03886      0.00763   0.06820
Wind         -1.95360     -3.06336  -1.07138
Temp          1.40590      0.99392   1.84861

> # Summary of LAD regression for Model 3
> summary(M3_LAD)

Call: rq(formula = Ozone ~ poly(Solar.R, 2) + poly(Wind, 2) + poly(Temp,
    2) + Solar.R:Wind + Solar.R:Temp + Wind:Temp, data = airquality_clean)

tau: [1] 0.5

Coefficients:
                  coefficients lower bd   upper bd
(Intercept)         87.08141    -43.07564  172.17927
poly(Solar.R, 2)1  -14.01817   -221.31128  525.36794
poly(Solar.R, 2)2  -33.58878    -50.19975   12.25193
poly(Wind, 2)1     204.51227   -253.32552  537.28237
poly(Wind, 2)2      57.81626     16.59135   81.56295
poly(Temp, 2)1     213.68468     17.57138  355.43205
poly(Temp, 2)2      57.62070     35.21886  122.69884
Solar.R:Wind        -0.00251     -0.00940    0.00345
Solar.R:Temp         0.00128     -0.00403    0.00332
Wind:Temp           -0.08130     -0.19031    0.06450
```

There is a substantial difference in coefficient estimates, as expected, and based on the results I would trust the forecasts of M3 mainly because the training dataset was small, thus the more predictors we keep the better the more informative our forecast is (reasonably high number of predictors to avoid overfitting and get results that can be better globalized). To provide more evidence backing our intuition we can assess the predictive performance, of the six alternative models, by performing

10-fold cross-validation. This process involves dividing the dataset into 10 equal parts, using 9 parts for training the model and the remaining part for testing. This process is repeated 10 times, with each fold serving as the test set once. The performance metrics used for evaluation are Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE).

```
# Load required library for bootstrapping
library(boot)

# Function to calculate RMSE
rmse <- function(actual, predicted) {
  sqrt(mean((actual - predicted)^2))
}

# Function to calculate MAE
mae <- function(actual, predicted) {
  mean(abs(actual - predicted))
}

# 10-fold cross-validation for Least Squares models
set.seed(1) # for reproducibility
cv_M1 <- cv.glm(airquality_clean, M1, K = 10)
cv_M2 <- cv.glm(airquality_clean, M2, K = 10)
cv_M3 <- cv.glm(airquality_clean, M3, K = 10)

# Calculate RMSE and MAE for Model 1 (Least Squares)
rmse_M1_LS <- rmse(airquality_clean$Ozone, predict(M1))
mae_M1_LS <- mae(airquality_clean$Ozone, predict(M1))

# Calculate RMSE and MAE for Model 2 (Least Squares)
rmse_M2_LS <- rmse(airquality_clean$Ozone, predict(M2))
mae_M2_LS <- mae(airquality_clean$Ozone, predict(M2))

# Calculate RMSE and MAE for Model 3 (Least Squares)
rmse_M3_LS <- rmse(airquality_clean$Ozone, predict(M3))
mae_M3_LS <- mae(airquality_clean$Ozone, predict(M3))

# Print cross-validation results for Least Squares models
cat("10-Fold CV for Least Squares Models:\n")
cat("Model 1: RMSE =", rmse_M1_LS, ", MAE =", mae_M1_LS, "\n")
cat("Model 2: RMSE =", rmse_M2_LS, ", MAE =", mae_M2_LS, "\n")
cat("Model 3: RMSE =", rmse_M3_LS, ", MAE =", mae_M3_LS, "\n")
```

results:

```
> # Print cross-validation results for Least Squares models
> cat("10-Fold CV for Least Squares Models:\n")
10-Fold CV for Least Squares Models:
> cat("Model 1: RMSE =", rmse_M1_LS, ", MAE =", mae_M1_LS, "\n")
Model 1: RMSE = 28.59945 , MAE = 19.97791
> cat("Model 2: RMSE =", rmse_M2_LS, ", MAE =", mae_M2_LS, "\n")
```

```
Model 2: RMSE = 20.62285 , MAE = 15.7517
> cat("Model 3: RMSE =", rmse_M3_LS, ", MAE =", mae_M3_LS, "\n")
Model 3: RMSE = 17.64133 , MAE = 13.21901
```

For a better illustration we can create a matrix using our models as columns and the folds as rows, using the R code presented further below:

- cv_results_rmse_ls: Stores the RMSE values for the Least Squares models across the 10 folds.
- cv_results_mae_ls: Stores the MAE values for the Least Squares models across the 10 folds.
- cv_results_rmse_lad: Stores the RMSE values for the Least Absolute Deviations models across the 10 folds.
- cv_results_mae_lad: Stores the MAE values for the Least Absolute Deviations models across the 10 folds.

```
# Load required library for model fitting
library(quantreg)

# Manually implement 10-fold cross-validation
set.seed(1)  # Set seed for reproducibility
n <- nrow(airquality_clean)
folds <- sample(rep(1:10, length.out = n))

# Initialize matrices to store results
cv_results_rmse_ls <- matrix(NA, nrow = 10, ncol = 3)
colnames(cv_results_rmse_ls) <- c("Model 1 (LS)", "Model 2 (LS)", "Model 3
↪  (LS)")

cv_results_mae_ls <- matrix(NA, nrow = 10, ncol = 3)
colnames(cv_results_mae_ls) <- c("Model 1 (LS)", "Model 2 (LS)", "Model 3
↪  (LS)")

cv_results_rmse_lad <- matrix(NA, nrow = 10, ncol = 3)
colnames(cv_results_rmse_lad) <- c("Model 1 (LAD)", "Model 2 (LAD)",
↪  "Model 3 (LAD)")

cv_results_mae_lad <- matrix(NA, nrow = 10, ncol = 3)
colnames(cv_results_mae_lad) <- c("Model 1 (LAD)", "Model 2 (LAD)", "Model
↪  3 (LAD)")

# Perform 10-fold cross-validation
for (i in 1:10) {
  # Split data into training and testing sets
  train <- airquality_clean[folds != i, ]
  test <- airquality_clean[folds == i, ]
```

```r
  # Fit models on training data
  M1_ls <- lm(Ozone ~ 1, data = train)
  M2_ls <- lm(Ozone ~ Solar.R + Wind + Temp, data = train)
  M3_ls <- lm(Ozone ~ poly(Solar.R, 2) + poly(Wind, 2) + poly(Temp, 2) +
  ↪  I(Solar.R * Wind) + I(Solar.R * Temp) + I(Wind * Temp), data =
  ↪  train)

  M1_lad <- rq(Ozone ~ 1, data = train, tau = 0.5)
  M2_lad <- rq(Ozone ~ Solar.R + Wind + Temp, data = train, tau = 0.5)
  M3_lad <- rq(Ozone ~ poly(Solar.R, 2) + poly(Wind, 2) + poly(Temp, 2) +
  ↪  I(Solar.R * Wind) + I(Solar.R * Temp) + I(Wind * Temp), data =
  ↪  train, tau = 0.5)

  # Predict on testing data
  pred_M1_ls <- predict(M1_ls, newdata = test)
  pred_M2_ls <- predict(M2_ls, newdata = test)
  pred_M3_ls <- predict(M3_ls, newdata = test)

  pred_M1_lad <- predict(M1_lad, newdata = test)
  pred_M2_lad <- predict(M2_lad, newdata = test)
  pred_M3_lad <- predict(M3_lad, newdata = test)

  # Calculate RMSE and MAE for Least Squares models
  cv_results_rmse_ls[i, 1] <- rmse(test$Ozone, pred_M1_ls)
  cv_results_rmse_ls[i, 2] <- rmse(test$Ozone, pred_M2_ls)
  cv_results_rmse_ls[i, 3] <- rmse(test$Ozone, pred_M3_ls)

  cv_results_mae_ls[i, 1] <- mae(test$Ozone, pred_M1_ls)
  cv_results_mae_ls[i, 2] <- mae(test$Ozone, pred_M2_ls)
  cv_results_mae_ls[i, 3] <- mae(test$Ozone, pred_M3_ls)

  # Calculate RMSE and MAE for Least Absolute Deviations models
  cv_results_rmse_lad[i, 1] <- rmse(test$Ozone, pred_M1_lad)
  cv_results_rmse_lad[i, 2] <- rmse(test$Ozone, pred_M2_lad)
  cv_results_rmse_lad[i, 3] <- rmse(test$Ozone, pred_M3_lad)

  cv_results_mae_lad[i, 1] <- mae(test$Ozone, pred_M1_lad)
  cv_results_mae_lad[i, 2] <- mae(test$Ozone, pred_M2_lad)
  cv_results_mae_lad[i, 3] <- mae(test$Ozone, pred_M3_lad)
}

# Print the results matrices
cat("RMSE Results for Least Squares Models:\n")
print(cv_results_rmse_ls)
cat("MAE Results for Least Squares Models:\n")
print(cv_results_mae_ls)

cat("RMSE Results for Least Absolute Deviations Models:\n")
print(cv_results_rmse_lad)
cat("MAE Results for Least Absolute Deviations Models:\n")
```

```
print(cv_results_mae_lad)
```

which provides:

```
RMSE Results for Least Squares Models:
> print(cv_results_rmse_ls)
      Model 1 (LS) Model 2 (LS) Model 3 (LS)
 [1,]     20.22012     23.15897     16.64409
 [2,]     19.05469     14.61688     13.21152
 [3,]     31.57308     23.27691     20.02789
 [4,]     36.89512     20.81176     22.14959
 [5,]     21.49937     14.92730     15.46934
 [6,]     25.98757     16.35519     14.37452
 [7,]     38.69387     30.30394     26.81094
 [8,]     34.53485     23.93236     21.14016
 [9,]     24.86381     17.54000     16.48996
[10,]     26.62580     22.39673     21.79812
> cat("MAE Results for Least Squares Models:\n")
MAE Results for Least Squares Models:
> print(cv_results_mae_ls)
      Model 1 (LS) Model 2 (LS) Model 3 (LS)
 [1,]     15.67418     19.27599     13.89379
 [2,]     13.24395     12.28912     11.25530
 [3,]     23.53363     16.32419     14.35418
 [4,]     30.29229     16.24626     16.50530
 [5,]     14.94529     13.69496     13.92987
 [6,]     16.82137     14.08985     11.56937
 [7,]     23.03711     18.38294     15.20779
 [8,]     25.06049     18.56525     16.81541
 [9,]     20.08070     14.72085     12.24956
[10,]     19.59544     19.38446     17.93607
>
> cat("RMSE Results for Least Absolute Deviations Models:\n")
RMSE Results for Least Absolute Deviations Models:
> print(cv_results_rmse_lad)
      Model 1 (LAD) Model 2 (LAD) Model 3 (LAD)
 [1,]      20.10135      22.52195      14.35730
 [2,]      19.01892      14.56392      13.66063
 [3,]      31.44391      24.07382      20.94453
 [4,]      36.68024      24.07368      22.47468
 [5,]      21.37949      12.54339      13.63970
 [6,]      25.96279      16.52016      12.45868
 [7,]      38.54650      32.29638      28.90356
 [8,]      34.46992      26.70703      23.73352
 [9,]      24.81114      16.50057      15.57203
[10,]      26.55581      23.16836      20.47901
> cat("MAE Results for Least Absolute Deviations Models:\n")
MAE Results for Least Absolute Deviations Models:
> print(cv_results_mae_lad)
      Model 1 (LAD) Model 2 (LAD) Model 3 (LAD)
 [1,]      15.35291      17.69466     11.876390
 [2,]      13.14116      12.12661     11.514173
```

```
 [3,]      23.28233      16.06197      14.470229
 [4,]      29.99138      17.64560      16.686952
 [5,]      14.56782      10.67669      11.621795
 [6,]      16.70115      13.40687       9.819413
 [7,]      22.88391      18.20767      16.328141
 [8,]      25.09253      20.38717      19.616293
 [9,]      19.86034      12.52307      12.169726
[10,]      19.44943      18.84321      15.982535
```

To create a better model we can review our M3 and start dropping predictors in a backward elimination fashion using the corrected old-fashioned thresholding of the p-value, suggested by the American statistic Association (0.005)[1]:

```
# Model 3: Including second-order terms and interactions
> M3 <- lm(Ozone ~ poly(Solar.R, 2) + poly(Wind, 2) + poly(Temp, 2) +
↪  Solar.R:Wind + Solar.R:Temp + Wind:Temp, data = airquality_clean)
> # Summary of Model 3
> summary(M3)

Call:
lm(formula = Ozone ~ poly(Solar.R, 2) + poly(Wind, 2) + poly(Temp,
    2) + Solar.R:Wind + Solar.R:Temp + Wind:Temp, data = airquality_clean)

Residuals:
    Min      1Q  Median      3Q     Max
-37.242 -11.536  -2.992   9.319  84.164

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)         5.058e+01  5.904e+01   0.857  0.39303
poly(Solar.R, 2)1   1.310e+02  2.111e+02   0.621  0.53589
poly(Solar.R, 2)2  -2.721e+01  2.059e+01  -1.322  0.18841
poly(Wind, 2)1     -2.249e+01  2.261e+02  -0.099  0.92089
poly(Wind, 2)2      8.194e+01  2.461e+01   3.329  0.00111 **
poly(Temp, 2)1      1.148e+02  9.933e+01   1.156  0.24960
poly(Temp, 2)2      5.837e+01  2.441e+01   2.391  0.01811 *
Solar.R:Wind       -1.115e-02  5.592e-03  -1.994  0.04808 *
Solar.R:Temp        8.931e-04  2.265e-03   0.394  0.69396
Wind:Temp          -1.486e-03  7.155e-02  -0.021  0.98346
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 18.25 on 143 degrees of freedom
Multiple R-squared:  0.6195,        Adjusted R-squared:  0.5956
F-statistic: 25.87 on 9 and 143 DF,  p-value: < 2.2e-16
```

Looking at the p-values, we can see that the Wind:Temp interaction term has the highest p-value (0.98346), which is much higher than the conventional significance

[1] A lot of people still use the 0.05 thresholding, though it has been proven that if you create a stricter arbitrary thresholding value (like 0,005) the results are more trustworthy and this is the reason a lot of statisticians now prefer thresholding using 0.005

level of 0.005. This suggests that this term is not significantly contributing to the model and could be considered for removal. So we run again without it:

```
> M3 <- lm(Ozone ~ poly(Solar.R, 2) + poly(Wind, 2) + poly(Temp, 2) +
↪  Solar.R:Wind + Solar.R:Temp, data = airquality_clean)
> summary(M3)

Call:
lm(formula = Ozone ~ poly(Solar.R, 2) + poly(Wind, 2) + poly(Temp,
    2) + Solar.R:Wind + Solar.R:Temp, data = airquality_clean)

Residuals:
    Min      1Q  Median      3Q     Max
-37.270 -11.532  -2.969   9.331  84.114

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)         4.962e+01  3.653e+01   1.359  0.17643
poly(Solar.R, 2)1   1.320e+02  2.048e+02   0.645  0.52012
poly(Solar.R, 2)2  -2.725e+01  2.040e+01  -1.336  0.18377
poly(Wind, 2)1     -2.710e+01  4.479e+01  -0.605  0.54617
poly(Wind, 2)2      8.227e+01  1.866e+01   4.409 2.01e-05 ***
poly(Temp, 2)1      1.131e+02  5.534e+01   2.044  0.04275 *
poly(Temp, 2)2      5.864e+01  2.049e+01   2.862  0.00484 **
Solar.R:Wind       -1.120e-02  5.060e-03  -2.213  0.02849 *
Solar.R:Temp        8.877e-04  2.242e-03   0.396  0.69278
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 18.18 on 144 degrees of freedom
Multiple R-squared:  0.6195,        Adjusted R-squared:  0.5984
F-statistic: 29.31 on 8 and 144 DF,  p-value: < 2.2e-16
```

Again Looking at the summary of the updated M3 model, the Solar.R:Temp interaction term has the highest p-value (0.69278), which is much higher than the conventional significance level of 0.005. This suggests that this term is not significantly contributing to the model and could be considered for removal. We perform the same procedure again:

```
> M3 <- lm(Ozone ~ poly(Solar.R, 2) + poly(Wind, 2) + poly(Temp, 2) +
↪  Solar.R:Wind, data = airquality_clean)
> summary(M3)

Call:
lm(formula = Ozone ~ poly(Solar.R, 2) + poly(Wind, 2) + poly(Temp,
    2) + Solar.R:Wind, data = airquality_clean)

Residuals:
    Min      1Q  Median      3Q     Max
-37.022 -11.303  -3.064   9.576  84.199

Coefficients:
```

```
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)         63.628943   9.054398   7.027 7.60e-11 ***
poly(Solar.R, 2)1 209.364555  61.292631   3.416 0.000825 ***
poly(Solar.R, 2)2 -24.833089  19.410181  -1.279 0.202805
poly(Wind, 2)1    -22.723442  43.280437  -0.525 0.600366
poly(Wind, 2)2     82.726094  18.568718   4.455 1.66e-05 ***
poly(Temp, 2)1    133.264728  21.705539   6.140 7.52e-09 ***
poly(Temp, 2)2     61.515985  19.107831   3.219 0.001585 **
Solar.R:Wind       -0.011720   0.004871  -2.406 0.017377 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 18.13 on 145 degrees of freedom
Multiple R-squared:  0.6191,        Adjusted R-squared:  0.6007
F-statistic: 33.67 on 7 and 145 DF,  p-value: < 2.2e-16
```

The summary of the updated M3 model shows that the poly(Wind, 2)1 term has the highest p-value (0.600366), which is much higher than the conventional significance level of 0.005. This indicates that this term is not significantly contributing to the model and could be considered for removal, thus we repeat the same procedure once more:

```
# Creating the quadratic term for Wind
airquality_clean$Wind2 <- airquality_clean$Wind^2

# Model without the linear term of Wind
M3 <- lm(Ozone ~ poly(Solar.R, 2) + Wind2 + poly(Temp, 2) + Solar.R:Wind,
↪  data = airquality_clean)

> summary(M3)

Call:
lm(formula = Ozone ~ poly(Solar.R, 2) + Wind2 + poly(Temp, 2) +
    Solar.R:Wind, data = airquality_clean)

Residuals:
    Min      1Q  Median      3Q     Max
-35.874 -11.851  -3.449   9.808  95.809

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)         71.783929   5.309798  13.519  < 2e-16 ***
poly(Solar.R, 2)1 311.271351  58.921246   5.283 4.53e-07 ***
poly(Solar.R, 2)2 -27.218557  20.417594  -1.333  0.18458
Wind2               0.073217   0.042748   1.713  0.08888 .
poly(Temp, 2)1    144.178694  22.668795   6.360 2.44e-09 ***
poly(Temp, 2)2     66.399663  20.069270   3.309  0.00118 **
Solar.R:Wind       -0.020615   0.004585  -4.496 1.40e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
Residual standard error: 19.08 on 146 degrees of freedom
Multiple R-squared:  0.5752,        Adjusted R-squared:  0.5578
F-statistic: 32.95 on 6 and 146 DF,  p-value: < 2.2e-16
```

And again for the removal of the quadratic Wind estimator with p_value 0.08888

```
> M3_updated <- lm(Ozone ~ poly(Solar.R, 2) + poly(Temp, 2) +
↪  Solar.R:Wind, data = airquality_clean)
> summary(M3_updated)

Call:
lm(formula = Ozone ~ poly(Solar.R, 2) + poly(Temp, 2) + Solar.R:Wind,
    data = airquality_clean)

Residuals:
    Min      1Q  Median      3Q     Max
-35.929 -12.166  -3.419   9.906  99.823

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)         67.789825   4.801582  14.118  < 2e-16 ***
poly(Solar.R, 2)1 231.952820  36.667715   6.326 2.86e-09 ***
poly(Solar.R, 2)2 -24.154461  20.472396  -1.180 0.239964
poly(Temp, 2)1    139.342960  22.639728   6.155 6.80e-09 ***
poly(Temp, 2)2     75.995962  19.397653   3.918 0.000136 ***
Solar.R:Wind       -0.013988   0.002477  -5.648 8.15e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 19.21 on 147 degrees of freedom
Multiple R-squared:  0.5667,        Adjusted R-squared:  0.552
F-statistic: 38.45 on 5 and 147 DF,  p-value: < 2.2e-16
```

The coefficient for the quadratic Solar.R has a p-value above the threshold of 0.005, indicating that it is not statistically significant at this level, so we repeat the same steps once again:

```
> M3_updated <- lm(Ozone ~ I(poly(Solar.R, 2)[,1]) + poly(Temp, 2) +
↪  Solar.R:Wind, data = airquality_clean)
> summary(M3_updated)

Call:
lm(formula = Ozone ~ I(poly(Solar.R, 2)[, 1]) + poly(Temp, 2) +
    Solar.R:Wind, data = airquality_clean)

Residuals:
    Min      1Q  Median      3Q     Max
-34.474 -11.584  -3.006   9.407  99.848

Coefficients:
                              Estimate Std. Error t value Pr(>|t|)
(Intercept)                  68.584804   4.760364  14.407  < 2e-16 ***
I(poly(Solar.R, 2)[, 1]) 235.308589  36.605628   6.428 1.67e-09 ***
```

```
poly(Temp, 2)1           145.568910  22.045314   6.603 6.76e-10 ***
poly(Temp, 2)2            76.162682  19.422813   3.921 0.000134 ***
Solar.R:Wind              -0.014422   0.002453  -5.880 2.62e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 19.23 on 148 degrees of freedom
Multiple R-squared:  0.5626,        Adjusted R-squared:  0.5508
F-statistic: 47.59 on 4 and 148 DF,  p-value: < 2.2e-16
```

Finally, the model M3_updated includes only predictors with p-values below the strict threshold of 0.005. All the predictors in this model have p-values well below this threshold, indicating that there is statistical significance in keeping them. Now we can proceed in performing cross-validation for our final M3_updated model:

```
set.seed(1)
n <- nrow(airquality_clean)
folds <- sample(rep(1:10, length.out = n))
fold_indices <- lapply(1:10, function(i) which(folds == i))

# Storage for results
cv_results_rmse_updated <- numeric(10)
cv_results_mae_updated <- numeric(10)

# Perform 10-fold cross-validation using the same partitions
for(i in 1:10) {
  # Use the same indices as before
  test_idx <- fold_indices[[i]]
  test_data <- airquality_clean[test_idx, ]
  train_data <- airquality_clean[-test_idx, ]

  # Fit the model on the training data
  model_updated <- lm(Ozone ~ I(poly(Solar.R, 2)[,1]) + poly(Temp, 2) +
  ↪  Solar.R:Wind, data = train_data)

  # Make predictions on the test data
  predictions_updated <- predict(model_updated, newdata = test_data)

  # Calculate RMSE and MAE
  cv_results_rmse_updated[i] <- sqrt(mean((predictions_updated -
  ↪  test_data$Ozone)^2))
  cv_results_mae_updated[i] <- mean(abs(predictions_updated -
  ↪  test_data$Ozone))
}

# Print the results
cat("RMSE Results for M3_updated Model (using same partitions):\n")
print(cv_results_rmse_updated)
cat("MAE Results for M3_updated Model (using same partitions):\n")
print(cv_results_mae_updated)
```

with results:

```
RMSE Results for M3_updated Model (using same partitions):
> print(cv_results_rmse_updated)
 [1] 39.61770 41.97702 41.19778 52.43441 47.57234 31.40722 38.91736
 ↪  40.69809 48.99549 47.91164
MAE Results for M3_updated Model (using same partitions):
> print(cv_results_mae_updated)
 [1] 29.64988 35.73945 36.65872 43.82158 38.42309 26.37949 35.45391
 ↪  35.42807 41.85031 41.40793
```

1.2 Part 2

In this part, we will compute bootstrap confidence intervals for LS and LAD-derived coefficients in M3 (in our case M3_updated). Compare to the ones derived from normality assumptions. Then, using the same folds created in your previous CV experiment we will perform Bagging to improve the predictive performance of M3.

Bootsraping The steps:

- Resample the dataset with replacement.
- Fit the M3_updated model on each resampled dataset.
- Compute the coefficients for each model.
- Calculate the 2.5th and 97.5th percentiles of the bootstrap samples to get the 95% confidence interval.

For LS:

```
set.seed(1)  # for reproducibility
n_bootstrap <- 1000
bootstrap_coefs <- matrix(NA, nrow = n_bootstrap, ncol = 5)  # 5
 ↪  coefficients in M3_updated

for (i in 1:n_bootstrap) {
  # Resample the data with replacement
  bootstrap_sample <- airquality_clean[sample(1:nrow(airquality_clean),
   ↪  replace = TRUE), ]

  # Fit the M3_updated model on the bootstrap sample
  M3_bootstrap <- lm(Ozone ~ I(poly(Solar.R, 2)[,1]) + poly(Temp, 2) +
   ↪  Solar.R:Wind, data = bootstrap_sample)

  # Store the coefficients
  bootstrap_coefs[i, ] <- coef(M3_bootstrap)
}

# Calculate 95% Confidence Intervals
bootstrap_ci <- apply(bootstrap_coefs, 2, function(x) quantile(x, c(0.025,
 ↪  0.975)))
```

the results are:

```
> bootstrap_ci
          [,1]     [,2]      [,3]      [,4]         [,5]
2.5%  52.05016 127.3633  97.88039  37.40619 -0.022990091
97.5% 86.93206 349.3274 192.46468 113.84170 -0.006541619
```

and for LAD:

```r
# Load necessary library
library(quantreg)

# Fit the M3_updated model using LAD
M3_updated_LAD <- rq(Ozone ~ I(poly(Solar.R, 2)[,1]) + poly(Temp, 2) +
↪  Solar.R:Wind, data = airquality_clean, tau = 0.5)

# Function to get LAD coefficients
get_LAD_coefs <- function(data, indices) {
  M3_updated_LAD <- rq(Ozone ~ I(poly(Solar.R, 2)[,1]) + poly(Temp, 2) +
  ↪  Solar.R:Wind, data = data[indices, ], tau = 0.5)
  return(coef(M3_updated_LAD))
}

# Apply bootstrap
set.seed(123) # for reproducibility
bootstrap_LAD_results <- boot(data = airquality_clean, statistic =
↪  get_LAD_coefs, R = 2000)

# Compute bootstrap confidence intervals for LAD coefficients
bootstrap_LAD_ci <- boot.ci(bootstrap_LAD_results, type = "bca")
```

with results:

```
> bootstrap_LAD_ci
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 2000 bootstrap replicates

CALL :
boot.ci(boot.out = bootstrap_LAD_results, type = "bca")

Intervals :
Level       BCa
95%   (45.53, 70.18 )
Calculations and Intervals on Original Scale
```

To get confidence intervals for all the coefficients, we can manually calculate them from the bootstrap results (as in the ls case):

```r
# Calculate percentiles for bootstrap confidence intervals
lad_ci_lower <- apply(bootstrap_LAD_results$t, 2, function(x) quantile(x,
↪  probs = 0.025))
lad_ci_upper <- apply(bootstrap_LAD_results$t, 2, function(x) quantile(x,
↪  probs = 0.975))

# Combine results
```

```
lad_ci <- cbind(lad_ci_lower, lad_ci_upper)
colnames(lad_ci) <- c("2.5%", "97.5%")
row.names(lad_ci) <- names(coef(M3_updated_LAD))
```

with results:

```
> lad_ci
                                2.5%         97.5%
(Intercept)                47.21473946  71.719186518
I(poly(Solar.R, 2)[, 1]) 105.40479016 260.124276409
poly(Temp, 2)1           112.15419898 220.590518221
poly(Temp, 2)2            39.80277557 154.105867482
Solar.R:Wind              -0.01651189  -0.004594733
```

The normality assumption: To compare the bootstrap confidence intervals to those derived from normality assumptions, we can calculate the latter using the standard errors of the coefficients and assuming a normal distribution. This is often done using the t-distribution, which approaches a normal distribution as the sample size increases.
The formula to calculate the confidence interval for a coefficient under the assumption of normality is:

$$coef + -(t_{\frac{a}{2},df} \times StandartError)$$

where $t_{\frac{a}{2},df}$ is the critical value from the t-distribution for a two-tailed test at the desired confidence level (here 95%) and the appropriate degrees of freedom. For LS

```
# Get the coefficients and their standard errors
coefficients_ls <- coef(summary(M3_updated))
coef_values_ls <- coefficients_ls[, "Estimate"]
std_error_ls <- coefficients_ls[, "Std. Error"]

# Get the critical t value
alpha <- 0.05
df <- df.residual(M3_updated)
t_critical_ls <- qt(1 - alpha/2, df)

# Calculate the confidence intervals
ci_lower_ls <- coef_values_ls - (t_critical_ls * std_error_ls)
ci_upper_ls <- coef_values_ls + (t_critical_ls * std_error_ls)

# Combine results
ci_ls <- cbind(ci_lower_ls, ci_upper_ls)
colnames(ci_ls) <- c("2.5%", "97.5%")
```

with results:

```
> ci_ls
                                2.5%         97.5%
(Intercept)                59.1777409  77.991866187
I(poly(Solar.R, 2)[, 1]) 162.9713852 307.645793199
poly(Temp, 2)1           102.0046692 189.133150328
poly(Temp, 2)2            37.7808250 114.544539849
Solar.R:Wind              -0.0192687  -0.009574922
```

For LAD:

```
#  Fit the LAD regression model
M3_updated_LAD_rq <- rq(Ozone ~ I(poly(Solar.R, 2)[,1]) + poly(Temp, 2) +
↪  Solar.R:Wind, data = airquality_clean, tau = 0.5)

#  Extract coefficients
coef_values_lad_rq <- as.numeric(coef(M3_updated_LAD_rq))

#  Calculate standard errors
lad_rq_summary <- summary(M3_updated_LAD_rq, se = "iid")
std_error_lad_rq <- lad_rq_summary$coefficients[, "Std. Error"]

# Calculate critical t-value
alpha <- 0.05
t_critical_lad_rq <- qt(1 - alpha/2, df = nrow(airquality_clean) -
↪  length(coef_values_lad_rq))

# Calculate confidence intervals
ci_lad_rq <- data.frame(
  "2.5%" = coef_values_lad_rq - t_critical_lad_rq * std_error_lad_rq,
  "97.5%" = coef_values_lad_rq + t_critical_lad_rq * std_error_lad_rq
)

#Present results
rownames(ci_lad_rq) <- names(coef_values_lad_rq)
```

with results:

```
> ci_lad_rq
          X2.5.         X97.5.
1  48.99614553  66.741011664
2 110.81039636 247.262554471
3 123.34978713 205.526513746
4  59.22298755 131.624006446
5  -0.01505769  -0.005914835
```

Bagging: Bagging (or Bootstrap Aggregation) is an ensemble learning method that aims to improve the stability and accuracy of machine learning algorithms. It works by training multiple models on different subsets of the training data (sampled with replacement) and then averaging their predictions.

```
set.seed(1)  # Set seed for reproducibility
n_bootstrap <- 100  # Number of bootstrap samples for Bagging
n_folds <- 10

# Storage for results
predictions_bagging <- numeric(nrow(airquality_clean))
rmse_bagging <- numeric(n_folds)
mae_bagging <- numeric(n_folds)
```

```r
# Perform Bagging using the saved fold indices
for (i in 1:n_folds) {
  # Get the training and test indices for the current fold
  test_idx <- fold_indices[[i]]
  train_idx <- setdiff(1:nrow(airquality_clean), test_idx)

  # Storage for bootstrap predictions
  bootstrap_predictions <- matrix(NA, nrow = length(test_idx), ncol =
  ↪  n_bootstrap)

  for (j in 1:n_bootstrap) {
    # Create a bootstrap sample of the training data
    bootstrap_idx <- sample(train_idx, replace = TRUE)
    bootstrap_data <- airquality_clean[bootstrap_idx, ]

    # Fit the M3_updated model on the bootstrap sample
    model_bagging <- lm(Ozone ~ I(poly(Solar.R, 2)[,1]) + poly(Temp, 2) +
    ↪  Solar.R:Wind, data = bootstrap_data)

    # Make predictions on the test data
    bootstrap_predictions[, j] <- predict(model_bagging, newdata =
    ↪  airquality_clean[test_idx, ])
  }

  # Aggregate predictions (mean of bootstrap predictions)
  predictions_bagging[test_idx] <- rowMeans(bootstrap_predictions, na.rm =
  ↪  TRUE)

  # Calculate RMSE and MAE for the current fold
  rmse_bagging[i] <- sqrt(mean((predictions_bagging[test_idx] -
  ↪  airquality_clean$Ozone[test_idx])^2))
  mae_bagging[i] <- mean(abs(predictions_bagging[test_idx] -
  ↪  airquality_clean$Ozone[test_idx]))
}

# Print the results
cat("Bagging RMSE Results for M3_updated Model (using same
↪  partitions):\n")
print(rmse_bagging)
cat("Bagging MAE Results for M3_updated Model (using same partitions):\n")
print(mae_bagging)
cat("Overall Bagging RMSE for M3_updated Model:\n")
print(sqrt(mean((predictions_bagging - airquality_clean$Ozone)^2)))
cat("Overall Bagging MAE for M3_updated Model:\n")
print(mean(abs(predictions_bagging - airquality_clean$Ozone)))
```

with results:

```
Bagging RMSE Results for M3_updated Model (using same partitions):
> print(rmse_bagging)
 [1] 39.97527 43.36830 40.11094 51.93515 46.60802 31.45209 41.43634
 ↪  40.93475 48.16328 49.07006
```

```
> cat("Bagging MAE Results for M3_updated Model (using same
↪  partitions):\n")
Bagging MAE Results for M3_updated Model (using same partitions):
> print(mae_bagging)
 [1] 30.11276 36.88648 35.66404 43.43432 37.82029 26.40443 37.79703
 ↪  35.70309 41.08089 42.26300
> cat("Overall Bagging RMSE for M3_updated Model:\n")
Overall Bagging RMSE for M3_updated Model:
> print(sqrt(mean((predictions_bagging - airquality_clean$Ozone)^2)))
[1] 43.61811
> cat("Overall Bagging MAE for M3_updated Model:\n")
Overall Bagging MAE for M3_updated Model:
> print(mean(abs(predictions_bagging - airquality_clean$Ozone)))
[1] 36.6677
```

1.3 Part 3

In this part, we will apply logistic regression to predict the probability of default, using income and balance on the Default data set (which is included on the ISLR library). We proceed by setting a random seed before beginning your analysis. Then we estimate the test error of this model, using the validation set approach. Then finally we repeat the process described above three times, with three different splits of the observations into training and validation sets.

Starting we will predict the probability of default based on income and balance:

```
library(ISLR)

# Load the Default dataset
data("Default")

# View the first few rows of the dataset
head(Default)

# Set a random seed for reproducibility
set.seed(1)

# Function to calculate test error
calculate_test_error <- function(actual, predicted) {
  mean(actual != predicted)
}

# Perform logistic regression and estimate test error for three different
↪  splits
for (i in 1:3) {
  cat("Iteration:", i, "\n")

  # Split the data into training (80%) and validation (20%) sets
  train_idx <- sample(1:nrow(Default), 0.8*nrow(Default))
  train_data <- Default[train_idx, ]
  validation_data <- Default[-train_idx, ]
```

```
# Fit logistic regression model on training data
glm_model <- glm(default ~ income + balance, data = train_data, family =
↪  "binomial")

# Make predictions on validation data
predictions <- predict(glm_model, newdata = validation_data, type =
↪  "response")
predicted_class <- ifelse(predictions > 0.5, "Yes", "No")

  # Calculate test error
  test_error <- calculate_test_error(validation_data$default,
  ↪  predicted_class)
  cat("Test Error:", test_error, "\n\n")
}
```

with result:

```
Iteration: 1
Test Error: 0.026

Iteration: 2
Test Error: 0.024

Iteration: 3
Test Error: 0.0265
```

The test errors we obtained from the three different splits of the observations into training and validation sets are quite low, which indicates that the logistic regression model has performed well in predicting the probability of default based on income and balance.