Final Project

Ovie Soman, Qianli Wu, Sebastian Rivera-Chepetla

30 May, 2022

Contents

Question 1. Propose a fractional factorial design for the problem. In addition, propose an experimental design constructed using the optimal design approach. **3**

The goal of this project is to identify the tuning parameters of random forest that affect the cross-validation accuracy. 3

Fractional Factorial Design 3

An Experimental Design Constructed using the Optimal Design Approach. 6

Question 2. Compare the optimal design with the fractional factorial design in practical and statistical terms. For instance, what is the performance of the designs for studying the main effects of the tuning parameters only? Can they estimate all two-parameter interactions? Why or why not? How do they compare in terms of multicollinearity? 7

1.1 Color Map for Fractional Factorial Design 7

2.1 Color Map for D-optimal Design 9

2.2 VIF for D-Optimal Design 10

3. Compare 11

Question 3. Recommend one experimental design between the two options in Question 1. Motivate your decision. 12

Question 4. Using a commercial software, the TAs and I came up with the experimental design shown in Table 2. How does your recommended design in the previous question compare with this one? 13

4.1 Color Map for Table 2 Design 14

Question 5. Collect data using your recommended design in Question 3. 15

Question 6. Conduct a detailed data analysis. What are the influential tuning parameters? What is the final model that links the tuning parameters to the cross-validation accuracy? Does the final model provide a good fit to the data? 17

Question 1. Propose a fractional factorial design for the problem. In addition, propose an experimental design constructed using the optimal design approach.

The goal of this project is to identify the tuning parameters of random forest that affect the cross-validation accuracy.

- We have seven tuning parameters:

	Parameter	Low Level	High Level
A	ntree	100	1000
B	replace	0	1
C	mtry	2	6
D	nodesize	1	11
E	maxnodes	10	1000
F	classwt	0.5	0.9
G	cutoff	0.2	0.8

Fractional Factorial Design

Since there are 7 factors each with 2 levels, a complete factorial design 2^7 requires 128 runs. Only 7 degrees of freedom correspond to main effects, 21 correspond to two-factor interactions, and the rest of 99 are associated with three-factor and higher interactions.

Due to limitation of resources, we can only run at most 35 tests. We assume that certain high-order interactions are negligible.

Since

- The effect sparsity principle
- The projection property
- Sequential experimentation

We choose the One-Quarter Fraction of the 2^7 Design, i.e., 2^{7-2} Design.

Letting the 'FrF2' function recommend one design. Generally, the recommended designs are those in Table 8.14 in Chapter 8. The designs recommended by the function have the largest possible resolution. They also have the smallest aberration as defined in Section 8.4.1. Fractional factorial designs with minimum aberration are preferred because they minimize the aliasing among the factors' effects.

```
factors <- list(ntree = c(100, 1000),
                mtry = c(2, 6),
                replace = c(0, 1),
                nodesize = c(1,11),
                classwt = c(0.5, 0.9),
                cutoff = c(0.2, 0.8),
                maxnodes = c(10, 1000))
my.design <- FrF2(nruns = 32, nfactors = 7, randomize = F, factor.names = factors)
print(my.design)
```

```
##    ntree mtry replace nodesize classwt cutoff maxnodes
## 1    100    2       0        1     0.5    0.2     1000
## 2   1000    2       0        1     0.5    0.8       10
## 3    100    6       0        1     0.5    0.8       10
## 4   1000    6       0        1     0.5    0.2     1000
## 5    100    2       1        1     0.5    0.8     1000
## 6   1000    2       1        1     0.5    0.2       10
## 7    100    6       1        1     0.5    0.2       10
## 8   1000    6       1        1     0.5    0.8     1000
## 9    100    2       0       11     0.5    0.2       10
## 10  1000    2       0       11     0.5    0.8     1000
## 11   100    6       0       11     0.5    0.8     1000
## 12  1000    6       0       11     0.5    0.2       10
## 13   100    2       1       11     0.5    0.8       10
## 14  1000    2       1       11     0.5    0.2     1000
## 15   100    6       1       11     0.5    0.2     1000
## 16  1000    6       1       11     0.5    0.8       10
## 17   100    2       0        1     0.9    0.2       10
## 18  1000    2       0        1     0.9    0.8     1000
## 19   100    6       0        1     0.9    0.8     1000
## 20  1000    6       0        1     0.9    0.2       10
## 21   100    2       1        1     0.9    0.8       10
## 22  1000    2       1        1     0.9    0.2     1000
## 23   100    6       1        1     0.9    0.2     1000
## 24  1000    6       1        1     0.9    0.8       10
## 25   100    2       0       11     0.9    0.2     1000
## 26  1000    2       0       11     0.9    0.8       10
## 27   100    6       0       11     0.9    0.8       10
## 28  1000    6       0       11     0.9    0.2     1000
## 29   100    2       1       11     0.9    0.8     1000
## 30  1000    2       1       11     0.9    0.2       10
## 31   100    6       1       11     0.9    0.2       10
## 32  1000    6       1       11     0.9    0.8     1000
## class=design, type= FrF2
```

```
design.info(my.design)$catlg.entry
```

```
## Design:  7-2.1
##    32  runs,  7  factors,
##    Resolution  IV
##    Generating columns:  7 27
##    WLP (3plus):  0 1 2 0 0 ,  15  clear 2fis
##  Factors with all 2fis clear:  D E G
```

```
design.info(my.design)$aliased
```

```
## $legend
## [1] "A=ntree"    "B=mtry"     "C=replace"  "D=nodesize" "E=classwt"
## [6] "F=cutoff"   "G=maxnodes"
##
## $main
## character(0)
##
## $fi2
## [1] "AB=CF" "AC=BF" "AF=BC"
```

An Experimental Design Constructed using the Optimal Design Approach.

Suggestion is to start with 32 runs

3 runs are saved for confirmation experiments or follow-up experiment.

```
# Create a full design for 7 factors each with 2 levels
full.design <- FrF2(nruns = 128, nfactors = 7, randomize = F, factor.names = factors)
```

```
## creating full factorial with 128 runs ...
```

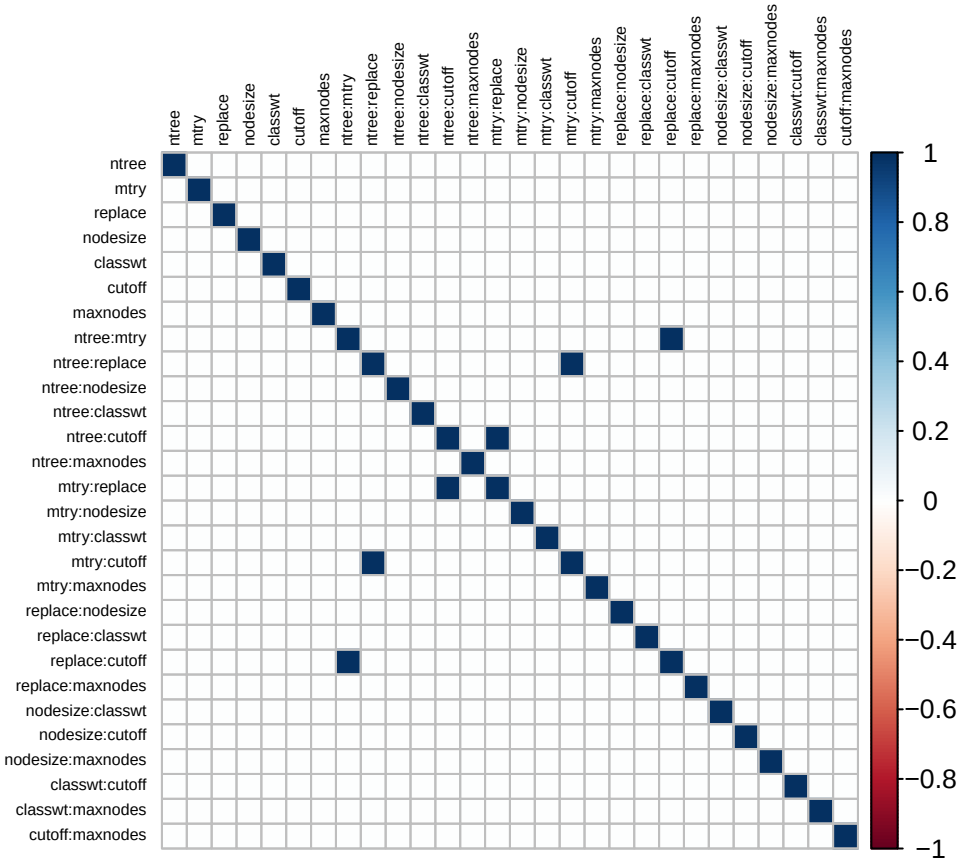
```
# Consider a model with 35 runs
alternative.design <- optFederov(~.^2,
                                 full.design, nTrials = 32, nRepeats = 1000)
print.data.frame(alternative.design$design)
```

```
##     ntree mtry replace nodesize classwt cutoff maxnodes
## 1     100    2       0        1     0.5    0.2       10
## 4    1000    6       0        1     0.5    0.2       10
## 6    1000    2       1        1     0.5    0.2       10
## 10   1000    2       0       11     0.5    0.2       10
## 15    100    6       1       11     0.5    0.2       10
## 18   1000    2       0        1     0.9    0.2       10
## 27    100    6       0       11     0.9    0.2       10
## 29    100    2       1       11     0.9    0.2       10
## 34   1000    2       0        1     0.5    0.8       10
## 43    100    6       0       11     0.5    0.8       10
## 45    100    2       1       11     0.5    0.8       10
## 48   1000    6       1       11     0.5    0.8       10
## 49    100    2       0        1     0.9    0.8       10
## 55    100    6       1        1     0.9    0.8       10
## 60   1000    6       0       11     0.9    0.8       10
## 62   1000    2       1       11     0.9    0.8       10
## 66   1000    2       0        1     0.5    0.2     1000
## 71    100    6       1        1     0.5    0.2     1000
## 73    100    2       0       11     0.5    0.2     1000
## 76   1000    6       0       11     0.5    0.2     1000
## 78   1000    2       1       11     0.5    0.2     1000
## 83    100    6       0        1     0.9    0.2     1000
## 85    100    2       1        1     0.9    0.2     1000
## 88   1000    6       1        1     0.9    0.2     1000
## 90   1000    2       0       11     0.9    0.2     1000
## 97    100    2       0        1     0.5    0.8     1000
## 104  1000    6       1        1     0.5    0.8     1000
## 106  1000    2       0       11     0.5    0.8     1000
## 116  1000    6       0        1     0.9    0.8     1000
## 118  1000    2       1        1     0.9    0.8     1000
## 121   100    2       0       11     0.9    0.8     1000
## 127   100    6       1       11     0.9    0.8     1000
```

Question 2. Compare the optimal design with the fractional factorial design in practical and statistical terms. For instance, what is the performance of the designs for studying the main effects of the tuning parameters only? Can they estimate all two-parameter interactions? Why or why not? How do they compare in terms of multicollinearity?

1.1 Color Map for Fractional Factorial Design

```
# Visualize the aliasing in the design.
D.alt <- (desnum(my.design)) # Extract the design.
# Create the model matrix including main effects and two-factor interactions.
X.alt <- model.matrix(~.^2-1, data.frame(D.alt))

# Create color map on pairwise correlations.
contrast.vectors.correlations.alt <- cor(X.alt)
corrplot(contrast.vectors.correlations.alt, type = "full", addgrid.col = "gray",
         tl.col = "black", tl.srt = 90, method = "color", tl.cex=0.5)
```



From the color map above, we get there's no alias among main effects of the tuning parameters, which means the performance of our design for studying the main effects of the tuning parameters only will be great.

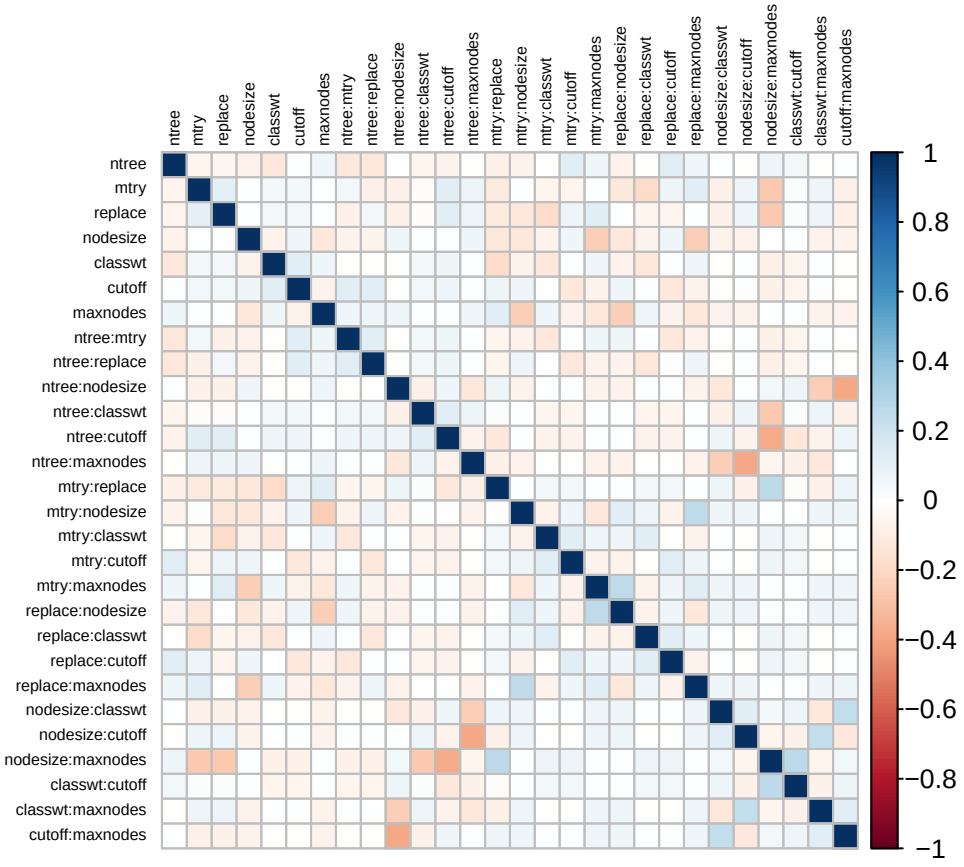
From the results of `design.info` in Question 1, we have AB=CF, AC=BF, and AF=BC, where `A=ntree`, `B=mtry`, `C=replace`, `D=nodesize`, `E=classwt`, `F=cutoff`, and `G=maxnodes`. The aliasing is also shown on the

map above, as the corresponding dark cells for `mtry:classwt` with `ntree:replace`, `replace:classwt` with `ntree:mtry`, and `replace:mtry` with `ntree:classwt`.

Therefore, we have the fractional factorial design's performance is best for all main effects and two-factor interactions except these aliasing factors (AB, CF, AC, BF, AF, and BC).

2.1 Color Map for D-optimal Design

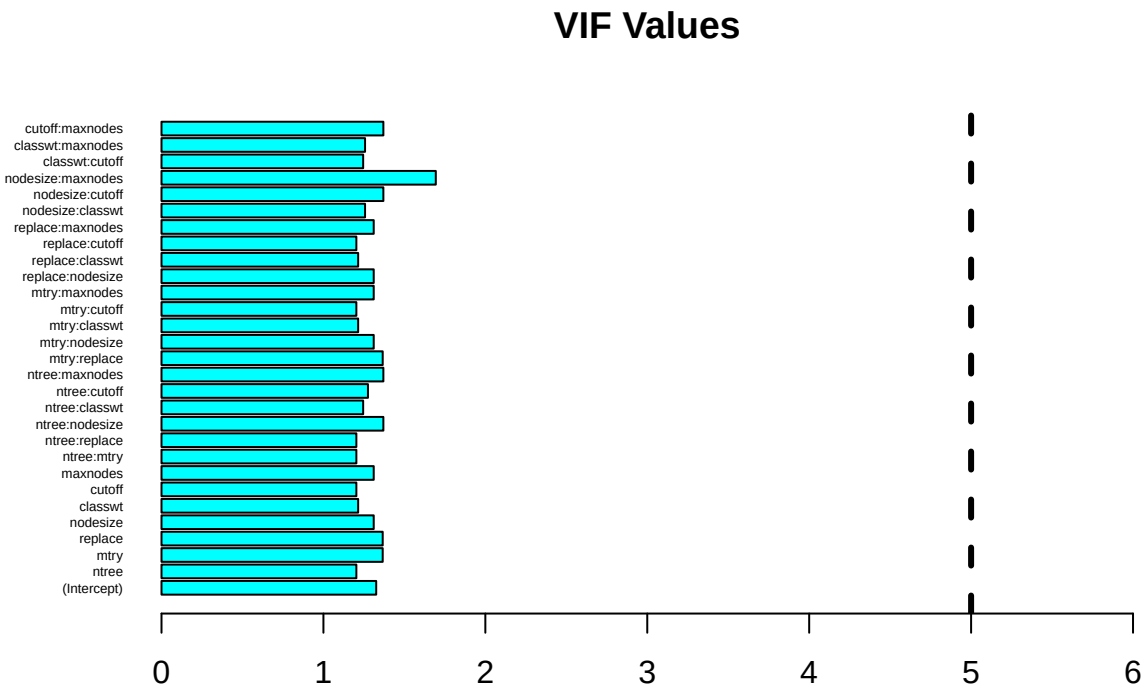
```
# Visualize the aliasing in the design.
D.alt <- alternative.design$design # Extract the design.

# Convert factors to -1 and 1
D.alt <- sapply(D.alt,function(x)(as.integer(x) -1.5)*2 )
X.alt <- model.matrix(~.^2-1, data.frame(D.alt))

# Create color map on pairwise correlations.
contrast.vectors.correlations.alt <- cor(X.alt)
corrplot(contrast.vectors.correlations.alt, type = "full", addgrid.col = "gray",
         tl.col = "black", tl.srt = 90, method = "color", tl.cex=0.5)
```



From the D-optimal Design above, we have most cells with light colors, which means the strength of correlation between the main effects and interactions are weak. Therefore, we have the degree of aliasing among the main effects,between the main effects and the specific interactions, and among the specific interactions are fairly low.

2.2 VIF for D-Optimal Design

```
# We need to include the intercept when computing the VIF.
X.alt <- model.matrix(~.^2, data.frame(D.alt))
# Variance-covariance matrix of 46-run design. Assuming sigma^2 = 1
var.eff.one <- diag(solve(t(X.alt)%*%X.alt))
# Set the left margin of plot be 1
par(oma=c(0,1,0,0))
#create horizontal bar chart to display each VIF value
barplot(nrow(X.alt)*var.eff.one, main = "VIF Values", horiz = TRUE, col = "cyan", las=1, cex.names=0.4,
#add vertical line at 5
abline(v = 5, lwd = 3, lty = 2)
```



From the VIF plot above, we have there's no factor has VIF greater than 5. Thus, we don't have a severe multi-collineairty or aliasing issue.

3. Compare

- We have the fractional factorial design only suffers from multi-collinearity problem of these three pairs of aliased interactions. It will be great design if `mtry:classwt`, `ntree:replace`, `replace:classwt`, `ntree:mtry`, `replace:mtry`, and `ntree:classwt` are not potentially important effects.
- However, there's no serious multi-collinearity problem or aliasing issue for the D-optimal design. But the variances of more coefficient estimates are inflated trivially due to collinearity.

Question 3. Recommend one experimental design between the two options in Question 1. Motivate your decision.

- We select the D-optimal Design since there's no series multi-collinearity problem or aliasing issue for all main effects and two-factor interactions.

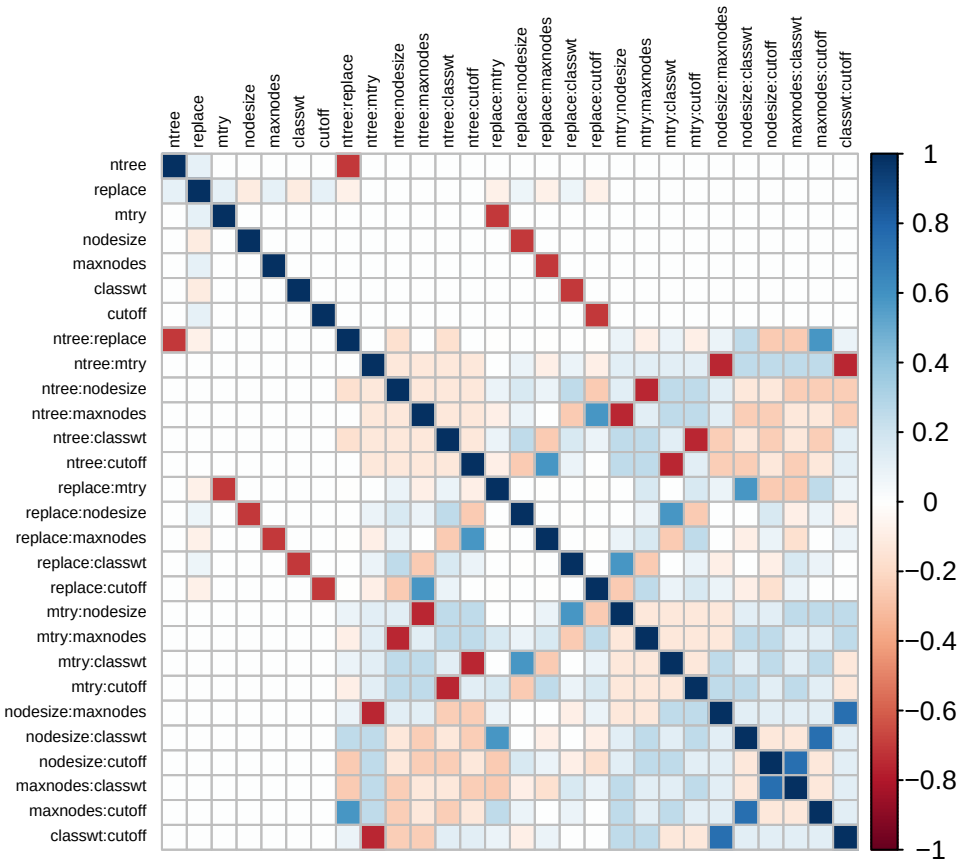
Question 4. Using a commercial software, the TAs and I came up with the experimental design shown in Table 2. How does your recommended design in the previous question compare with this one?

```
table2 <- read.csv("Data/Alternative_Experimental_Design.csv")
head(table2)
```

```
##   Run ntree replace mtry nodesize maxnodes classwt cutoff
## 1   1   100       1    2       11       10     0.5    0.8
## 2   2   550       0    2        1       10     0.5    0.2
## 3   3  1000       1    4        1       10     0.5    0.2
## 4   4  1000       1    6        1     1000     0.5    0.8
## 5   5  1000       0    6        1     1000     0.9    0.2
## 6   6   100       0    2        1     1000     0.5    0.8
```

4.1 Color Map for Table 2 Design

```
# Convert factors to -1 and 1
table2 <- sapply(table2,function(x)(as.integer(as.factor(x)) -2) )
X.alt <- model.matrix(~.^2-1, data.frame(table2[,-1]))

# Create color map on pairwise correlations.
contrast.vectors.correlations.alt <- cor(X.alt)
corrplot(contrast.vectors.correlations.alt, type = "full", addgrid.col = "gray",
         tl.col = "black", tl.srt = 90, method = "color", tl.cex=0.5)
```



- The experimental design shown in Table2 is 7 factor with 3 levels, however, the design we proposed in Question 1 has only 2 levels for each factor.
- From the color map above, we have the design in Table 2 has lighter cells for aliasing among main effects with other main effects or interactions. However, since the cells at bottom right corner are darker, we have the degree of aliasing is generally greater for the aliasing among two-factor interactions.

Question 5. Collect data using your recommended design in Question 3.

```
# import function cv.rf(design, y, X)
source("CrossValidation_RandomForest/CrossValidation_RF.R")
# import dataset Cardiovascular
load("Data/cardiovascular.Rdata")
```

```
# convert the design from factor to numerical values
design <- as.data.frame(sapply(alternative.design$design, function(x) as.numeric(as.character(x))))
result <- cv.rf(design, y, X)
```

```
## Collecting response on test combination  1
## Collecting response on test combination  2
## Collecting response on test combination  3
## Collecting response on test combination  4
## Collecting response on test combination  5
## Collecting response on test combination  6
## Collecting response on test combination  7
## Collecting response on test combination  8
## Collecting response on test combination  9
## Collecting response on test combination  10
## Collecting response on test combination  11
## Collecting response on test combination  12
## Collecting response on test combination  13
## Collecting response on test combination  14
## Collecting response on test combination  15
## Collecting response on test combination  16
## Collecting response on test combination  17
## Collecting response on test combination  18
## Collecting response on test combination  19
## Collecting response on test combination  20
## Collecting response on test combination  21
## Collecting response on test combination  22
## Collecting response on test combination  23
## Collecting response on test combination  24
## Collecting response on test combination  25
## Collecting response on test combination  26
## Collecting response on test combination  27
## Collecting response on test combination  28
## Collecting response on test combination  29
## Collecting response on test combination  30
## Collecting response on test combination  31
## Collecting response on test combination  32
```

```
print(result)
```

```
##    ntree mtry replace nodesize classwt cutoff maxnodes        CV
## 1    100    2       0        1     0.5    0.2       10 0.6624975
## 2   1000    6       0        1     0.5    0.2       10 0.7126758
## 3   1000    2       1        1     0.5    0.2       10 0.6673822
```

```
## 4   1000    2       0       11      0.5     0.2       10 0.6680928
## 5    100    6       1       11      0.5     0.2       10 0.7120758
## 6   1000    2       0        1      0.9     0.2       10 0.5000000
## 7    100    6       0       11      0.9     0.2       10 0.5000044
## 8    100    2       1       11      0.9     0.2       10 0.5000000
## 9   1000    2       0        1      0.5     0.8       10 0.6954486
## 10   100    6       0       11      0.5     0.8       10 0.7176979
## 11   100    2       1       11      0.5     0.8       10 0.6903382
## 12  1000    6       1       11      0.5     0.8       10 0.7181153
## 13   100    2       0        1      0.9     0.8       10 0.5071824
## 14   100    6       1        1      0.9     0.8       10 0.5749049
## 15  1000    6       0       11      0.9     0.8       10 0.5485835
## 16  1000    2       1       11      0.9     0.8       10 0.5131072
## 17  1000    2       0        1      0.5     0.2     1000 0.6557472
## 18   100    6       1        1      0.5     0.2     1000 0.6334354
## 19   100    2       0       11      0.5     0.2     1000 0.6745869
## 20  1000    6       0       11      0.5     0.2     1000 0.6658979
## 21  1000    2       1       11      0.5     0.2     1000 0.6706846
## 22   100    6       0        1      0.9     0.2     1000 0.6538935
## 23   100    2       1        1      0.9     0.2     1000 0.5284753
## 24  1000    6       1        1      0.9     0.2     1000 0.6535826
## 25  1000    2       0       11      0.9     0.2     1000 0.5009600
## 26   100    2       0        1      0.5     0.8     1000 0.6704707
## 27  1000    6       1        1      0.5     0.8     1000 0.6133425
## 28  1000    2       0       11      0.5     0.8     1000 0.6778841
## 29  1000    6       0        1      0.9     0.8     1000 0.6162309
## 30  1000    2       1        1      0.9     0.8     1000 0.7074668
## 31   100    2       0       11      0.9     0.8     1000 0.7151466
## 32   100    6       1       11      0.9     0.8     1000 0.7109554
```

Question 6. Conduct a detailed data analysis. What are the influential tuning parameters? What is the final model that links the tuning parameters to the cross-validation accuracy? Does the final model provide a good fit to the data?

```
# Converting numerical variables back to factors
str(result)
```

```
## 'data.frame':    32 obs. of  8 variables:
##  $ ntree   : num  100 1000 1000 1000 100 1000 100 100 1000 100 ...
##  $ mtry    : num  2 6 2 2 6 2 6 2 2 6 ...
##  $ replace : num  0 0 1 0 1 0 0 1 0 0 ...
##  $ nodesize: num  1 1 1 11 11 1 11 11 1 11 ...
##  $ classwt : num  0.5 0.5 0.5 0.5 0.5 0.9 0.9 0.9 0.5 0.5 ...
##  $ cutoff  : num  0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.8 0.8 ...
##  $ maxnodes: num  10 10 10 10 10 10 10 10 10 10 ...
##  $ CV      : num  0.662 0.713 0.667 0.668 0.712 ...
```

```
result[,c("ntree","mtry","replace",
          "nodesize","classwt","cutoff","maxnodes")] <- lapply(result[,c("ntree","mtry","replace",
                                                                         "nodesize","classwt","cutoff",
```

```
# Re-encoding Factors
levels(result$ntree) = c(-1,1)
levels(result$mtry) = c(-1,1)
levels(result$replace) = c(-1,1)
levels(result$nodesize) = c(-1,1)
levels(result$classwt) = c(-1,1)
levels(result$cutoff) = c(-1,1)
levels(result$maxnodes) = c(-1,1)

str(result)
```

```
## 'data.frame':    32 obs. of  8 variables:
##  $ ntree   : Factor w/ 2 levels "-1","1": 1 2 2 2 1 2 1 1 2 1 ...
##  $ mtry    : Factor w/ 2 levels "-1","1": 1 2 1 1 2 1 2 1 1 2 ...
##  $ replace : Factor w/ 2 levels "-1","1": 1 1 2 1 2 1 1 2 1 1 ...
##  $ nodesize: Factor w/ 2 levels "-1","1": 1 1 1 2 2 1 2 2 1 2 ...
##  $ classwt : Factor w/ 2 levels "-1","1": 1 1 1 1 1 2 2 2 1 1 ...
##  $ cutoff  : Factor w/ 2 levels "-1","1": 1 1 1 1 1 1 1 1 2 2 ...
##  $ maxnodes: Factor w/ 2 levels "-1","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ CV      : num  0.662 0.713 0.667 0.668 0.712 ...
```

```
attach(result)
# Running Linear Model
cv.model <- lm(CV~ntree+mtry+replace+nodesize+classwt+cutoff+maxnodes)
summary(cv.model)
```

```
##
```

```
## Call:
## lm.default(formula = CV ~ ntree + mtry + replace + nodesize +
##     classwt + cutoff + maxnodes)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.118267 -0.039066 -0.003863  0.037441  0.105915
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.634080   0.029472  21.515  < 2e-16 ***
## ntree1      -0.009024   0.021796  -0.414   0.6825
## mtry1        0.025187   0.021812   1.155   0.2596
## replace1     0.005539   0.021812   0.254   0.8017
## nodesize1    0.003637   0.021729   0.167   0.8685
## classwt1    -0.104313   0.021964  -4.749 7.86e-05 ***
## cutoff1      0.037176   0.021796   1.706   0.1010
## maxnodes1    0.038653   0.021729   1.779   0.0879 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06063 on 24 degrees of freedom
## Multiple R-squared:  0.5298, Adjusted R-squared:  0.3927
## F-statistic: 3.864 on 7 and 24 DF,  p-value: 0.005954
```

As can be seen, the influential tuning parameters are classwt and, to a lesser extent, cutoff, per the significance level $\alpha = 0.05$. Their p-values for the t-test are lower than 0.05. We run a reduced model next with only these two factors.

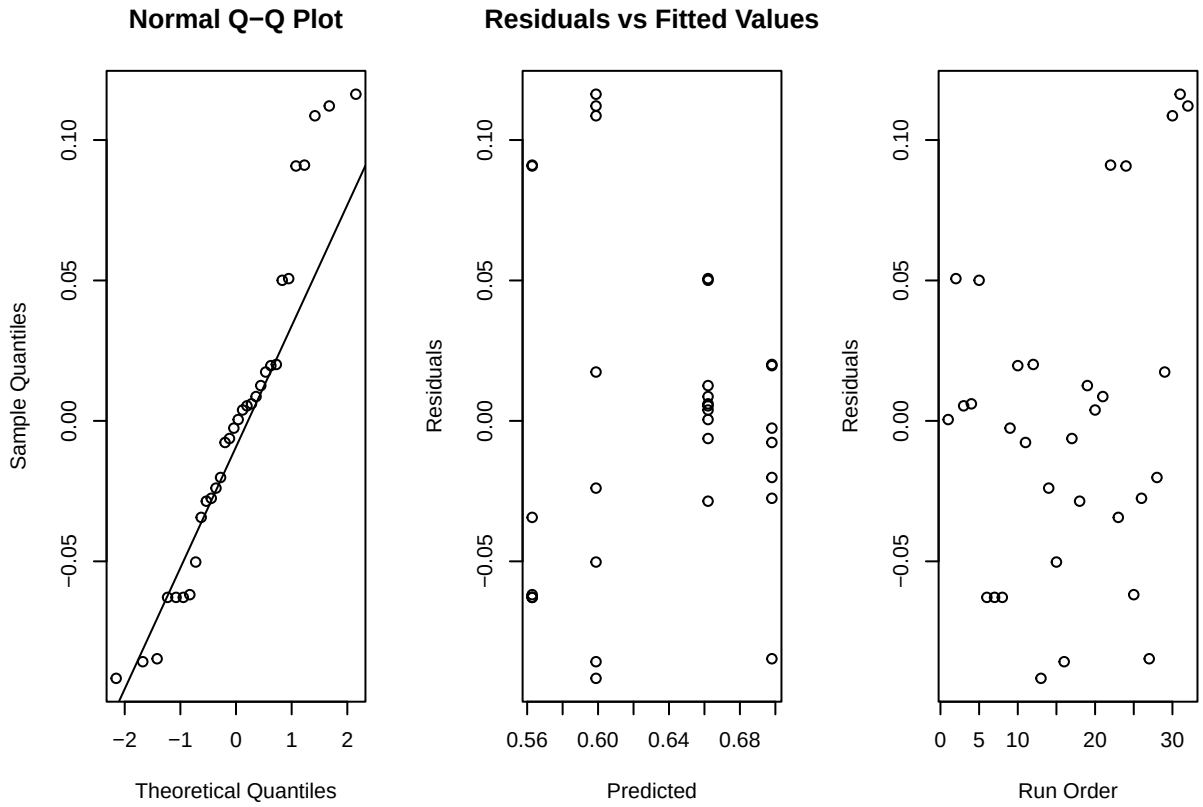
```
# Running Reduced Model
cv.model2 <- lm(CV~classwt+cutoff)
summary(cv.model2)
```

```
##
## Call:
## lm.default(formula = CV ~ classwt + cutoff)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.09165 -0.03833 -0.00105  0.01978  0.11631
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.66202    0.01713   38.64  < 2e-16 ***
## classwt1    -0.09919    0.02156   -4.60  7.7e-05 ***
## cutoff1      0.03601    0.02156    1.67    0.106
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06042 on 29 degrees of freedom
## Multiple R-squared:  0.4359, Adjusted R-squared:  0.397
## F-statistic: 11.21 on 2 and 29 DF,  p-value: 0.0002479
```

The classwt and cutoff continue to be significant per $\alpha = 0.05$. For the reduced model, the difference between

Multiple and Adjusted R^2 values is a lot smaller, indicating that the reduced model is generally a better fit for the dataset. We now analyse the residuals of this model.

```
# Analysis of Residuals
cv.res <- cv.model2$residuals
cv.fit <- cv.model2$fitted.values
par(mfrow = c(1,3))
qqnorm(cv.res)
qqline(cv.res)
plot(cv.fit,cv.res, main = "Residuals vs Fitted Values",
     xlab = "Predicted", ylab = "Residuals")

N <- nrow(result)
plot(x = 1:N, y = cv.res, xlab = "Run Order",
     ylab = "Residuals")
```



Points adhere to the qqline on the Normal Q-Q Plot, apart from at the higher theoretical quantile extremity. The plot of residuals vs fitted values is heteroskedastic around the horizontal, as is the plot of residuals vs run order. We thus conclude that the model does not satisfy all the assumptions of normality.