Milestone4_CODE

Akhil Havaldar

11/10/2022

Section 2

```
library(readr)
```

```
## Warning: package 'readr' was built under R version 4.1.3
```

```
library(tree)
```

```
## Warning: package 'tree' was built under R version 4.1.3
```

```
library(janitor)
```

```
## Warning: package 'janitor' was built under R version 4.1.3
```

```
##
## Attaching package: 'janitor'
```

```
## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.1.3
```

```
library(reshape2)
```

```
## Warning: package 'reshape2' was built under R version 4.1.3
```

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.1.3
```

```
##
## 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
```

```
library(tidyr)
```

```
## Warning: package 'tidyr' was built under R version 4.1.3
```

```
##
## Attaching package: 'tidyr'

## The following object is masked from 'package:reshape2':
##
##     smiths
```

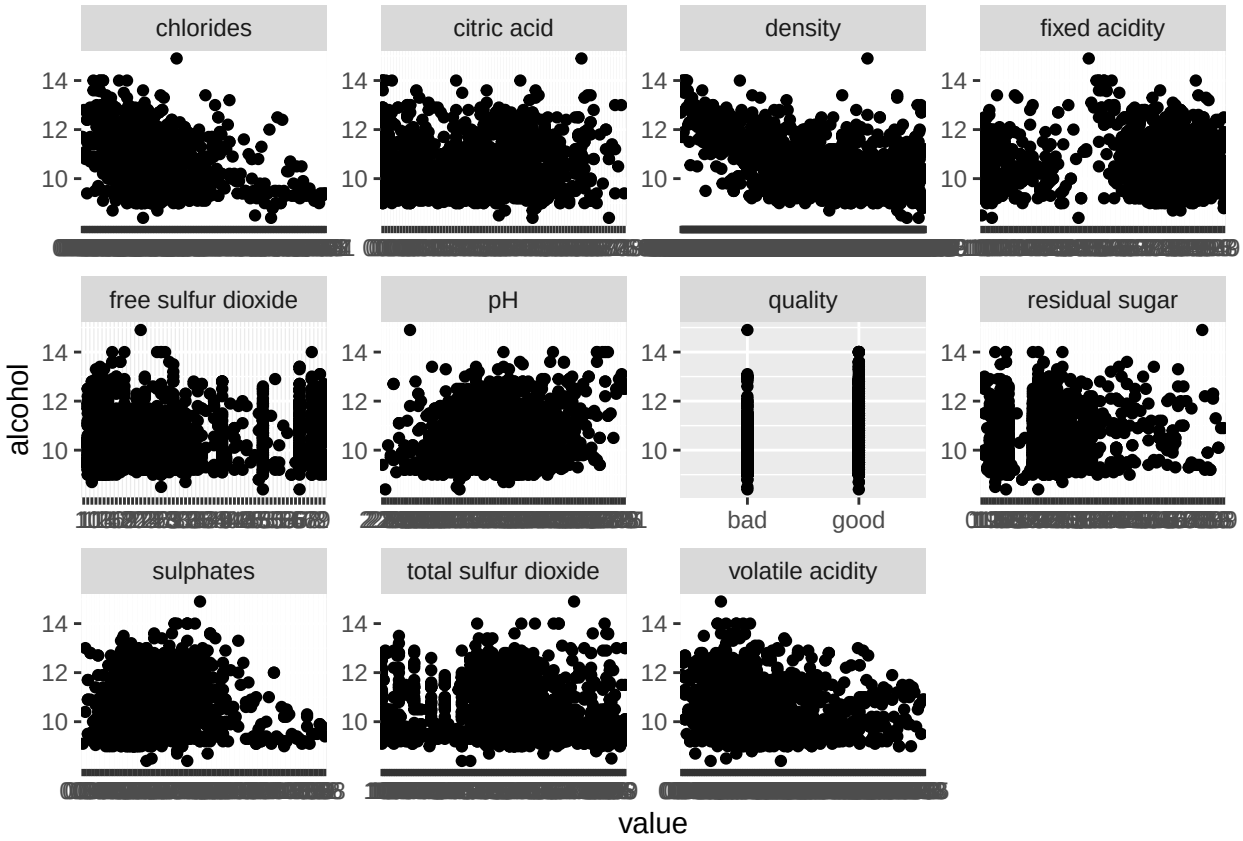
```
wine <- read_csv("wine.csv")
```

```
## Rows: 1599 Columns: 12

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (1): quality
## dbl (11): fixed acidity, volatile acidity, citric acid, residual sugar, chlo...
##
## 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.
```

```
wine$quality <- factor(wine$quality)
```

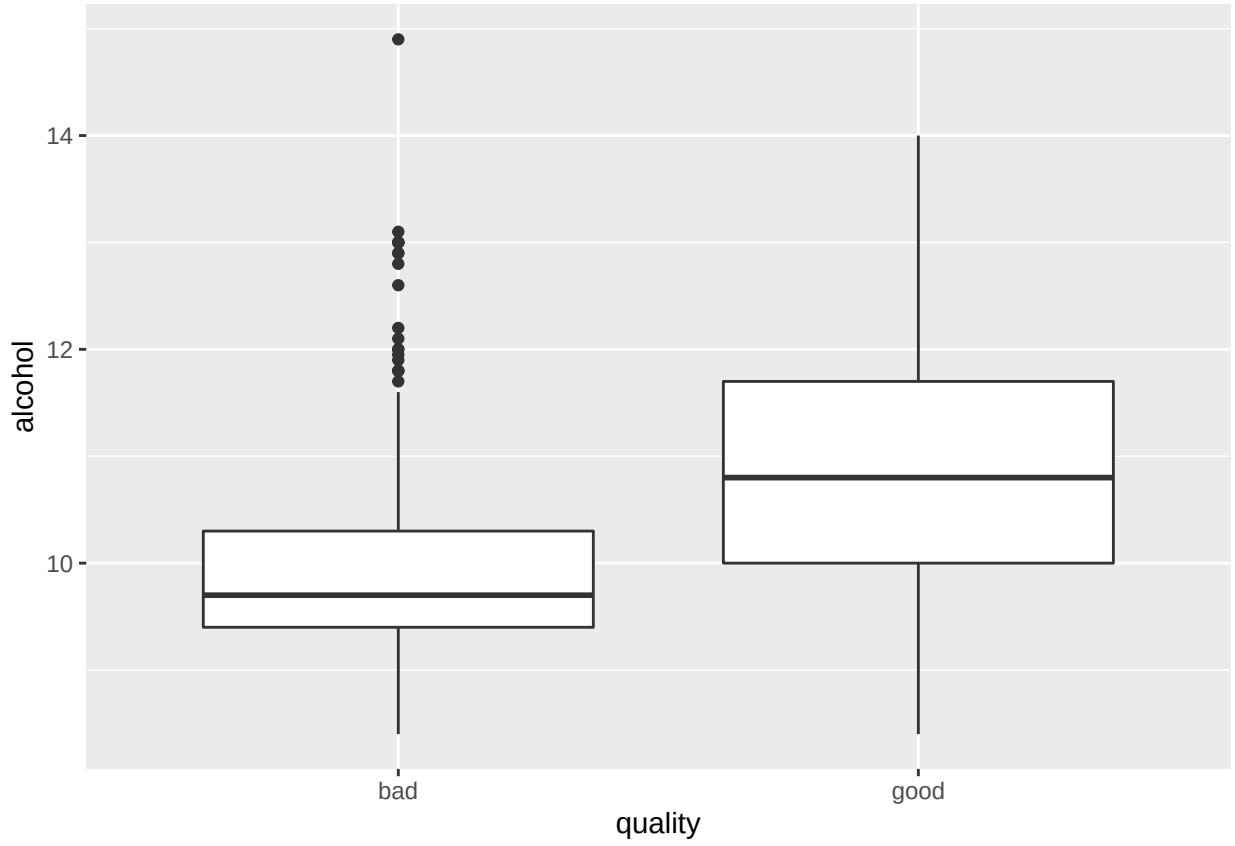
```
wine %>%
  gather(-alcohol, key = "var", value = "value") %>%
  ggplot(aes(x=value, y=alcohol)) + geom_point() + facet_wrap(~var, scales = "free")
```




```
cor(wine$alcohol, wine[,1:10])
```

```
##      fixed acidity volatile acidity citric acid residual sugar  chlorides
## [1,]   -0.06166827        -0.202288   0.1099032     0.04207544 -0.2211405
##      free sulfur dioxide total sulfur dioxide    density        pH  sulphates
## [1,]         -0.06940835           -0.2056539 -0.4961798 0.2056325 0.09359475
```

```
ggplot(data = wine, aes(x=quality, y=alcohol)) + geom_boxplot()
```



Section 3: Shrinkage Methods

a) Data Cleaning

```
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 4.1.3
```

```
## Loading required package: Matrix
```

```
## Warning: package 'Matrix' was built under R version 4.1.3
```

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
## Loaded glmnet 4.1-4
```

```
Data <-read.csv("wine.csv", header=T)
Data$quality <- factor(Data$quality)

x<-model.matrix(alcohol~.,data=Data)
x<-model.matrix(alcohol~.,data=Data)[,-1]

y<-Data$alcohol
```

b) Threshold Value

```
ridge.r<-glmnet::glmnet(x,y,alpha=0, lambda=0)
##compare with OLS
result<-lm(alcohol~.,data=Data)
cbind(coefficients(result), coefficients(ridge.r))
```

```
## 12 x 2 sparse Matrix of class "dgCMatrix"
##                                           s0
## (Intercept)           5.761122e+02  5.758798e+02
## fixed.acidity         4.997269e-01  4.991646e-01
## volatile.acidity      5.200129e-01  5.210394e-01
## citric.acid           8.500316e-01  8.515211e-01
## residual.sugar        2.683924e-01  2.682860e-01
## chlorides            -1.148642e+00 -1.153835e+00
## free.sulfur.dioxide  -3.433308e-03 -3.413963e-03
## total.sulfur.dioxide -1.203663e-03 -1.212415e-03
## density              -5.855023e+02 -5.852541e+02
## pH                    3.603756e+00  3.600546e+00
## sulphates             1.027259e+00  1.027485e+00
## qualitygood           3.119742e-01  3.121530e-01
```

```
## not the same values, lower threshold

ridge.r<-glmnet::glmnet(x,y,alpha=0, lambda=0, thresh = 1e-23)
##compare with OLS
cbind(coefficients(result), coefficients(ridge.r))
```

```
## 12 x 2 sparse Matrix of class "dgCMatrix"
##                                           s0
## (Intercept)           5.761122e+02  5.761122e+02
## fixed.acidity         4.997269e-01  4.997269e-01
## volatile.acidity      5.200129e-01  5.200129e-01
## citric.acid           8.500316e-01  8.500316e-01
## residual.sugar        2.683924e-01  2.683924e-01
## chlorides            -1.148642e+00 -1.148642e+00
## free.sulfur.dioxide  -3.433308e-03 -3.433308e-03
## total.sulfur.dioxide -1.203663e-03 -1.203663e-03
## density              -5.855023e+02 -5.855023e+02
## pH                    3.603756e+00  3.603756e+00
## sulphates             1.027259e+00  1.027259e+00
## qualitygood           3.119742e-01  3.119742e-01
```

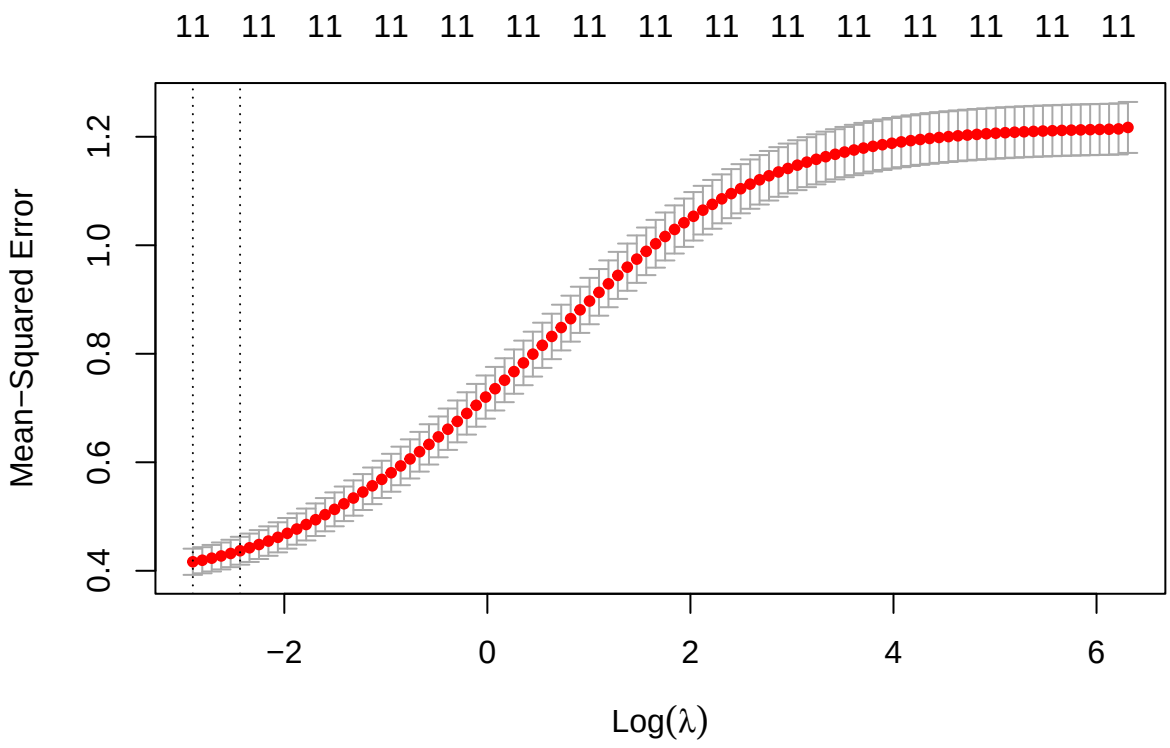
c) Ridge Regression

```
set.seed(4630)
sample.data<-sample.int(nrow(Data), floor(.50*nrow(Data)), replace = F)
x.train<-x[sample.data,]
x.test<-x[-sample.data,]
y.train<-y[sample.data]
y.test<-y[-sample.data]
train<-Data[sample.data, ]
test<-Data[-sample.data, ]

# i)
set.seed(4630)
cv.out<-glmnet::cv.glmnet(x.train,y.train,alpha=0, thresh = 1e-23)
bestlam<-cv.out$lambda.min
bestlam
```

```
## [1] 0.05497221
```

```
# ii)
plot(cv.out)
```



```
# iii)
## All 11 predictors are left in the model.

# iv)
## Fixed acidity, Volatile acidity, Citric acid, Residual sugar, Chlorides,
## Free sulfur dioxide, Total sulfur dioxide, Density, pH, Sulphates, and Quality

# v)
ridge.mod<-glmnet::glmnet(x.train,y.train,alpha=0,lambda=bestlam, thresh = 1e-25)
ridge.pred<-predict(ridge.mod,newx=x.test)
mean((ridge.pred-y.test)^2)
```

```
## [1] 0.354758
```

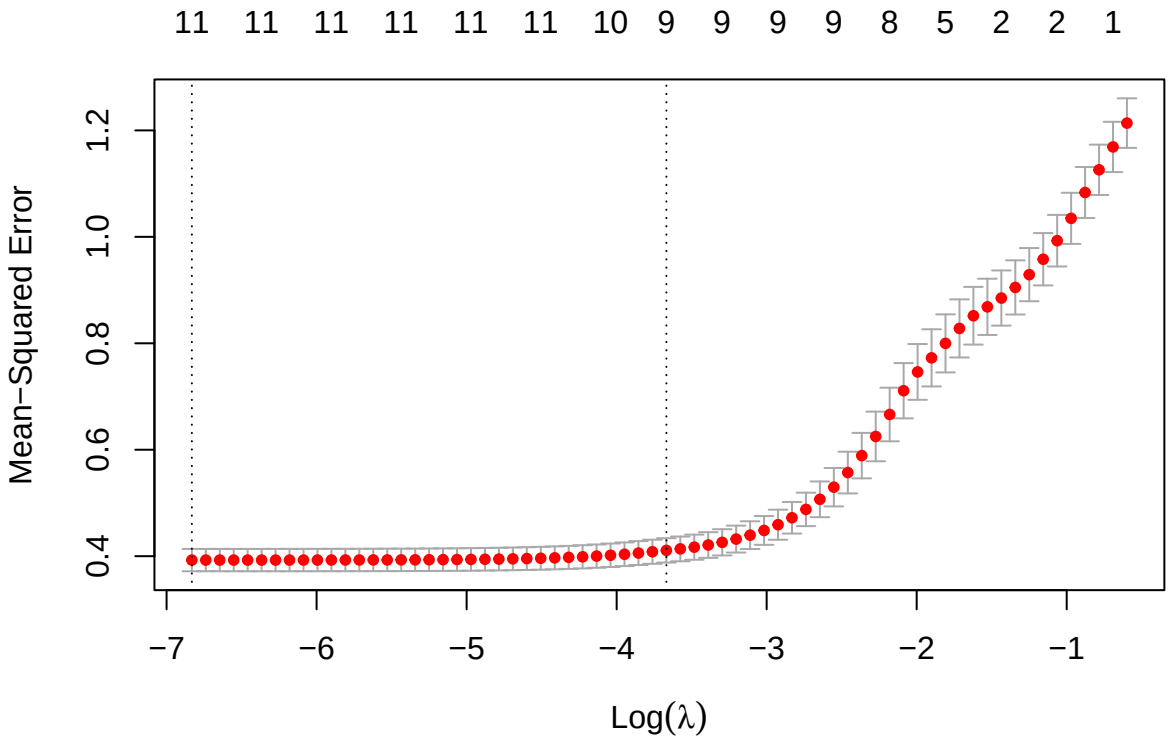
d) Lasso Regression

```
# i)
lasso.r<-glmnet::glmnet(x,y,alpha=1, lambda=0, thresh = 1e-23)

set.seed(4630)
cv.out.lasso<-glmnet::cv.glmnet(x.train,y.train,alpha=1, thresh = 1e-23)
bestlam.lasso<-cv.out.lasso$lambda.min
bestlam.lasso
```

```
## [1] 0.001079127
```

```
# ii)
plot(cv.out.lasso)
```




```
# iii)
coef(cv.out.lasso)
```

```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                                  s1
## (Intercept)           5.107144e+02
## fixed.acidity         4.210995e-01
## volatile.acidity      .
## citric.acid           6.192904e-01
## residual.sugar        2.142813e-01
## chlorides            -9.209318e-01
## free.sulfur.dioxide   .
## total.sulfur.dioxide -1.538457e-03
## density              -5.169071e+02
## pH                    3.098842e+00
## sulphates             7.624665e-01
## qualitygood           2.558054e-01
```

```
## 9 predictors are left in the model.

# iv)
## Fixed acidity, Citric acid, Residual sugar, Chlorides,
## Total sulfur dioxide, Density, pH, Sulphates, and Quality

# v)
```

```
lasso.mod<-glmnet::glmnet(x.train,y.train,alpha=1,lambda=bestlam.lasso, thresh = 1e-23)
lasso.pred<-predict(lasso.mod,newx=x.test)
mean((lasso.pred-y.test)^2)
```

```
## [1] 0.3416871
```

e) OLS Regression Test MSE

```
result<-lm(alcohol~.,data=train)
predicty <- predict(result,test)

mean((predicty-test$alcohol)^2)
```

```
## [1] 0.3419983
```

- Conclusion found in google doc file

Section 4: Regression Trees

a)

- Need to change the quality column to a factor.
- Need to rename columns to remove spaces from the variable names

b)

```
library(readr)
library(tree)
library(janitor)

wine <- read_csv("wine.csv")
```

```
## Rows: 1599 Columns: 12
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (1): quality
## dbl (11): fixed acidity, volatile acidity, citric acid, residual sugar, chlo...
##
## 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.
```

```
wine$quality <- factor(wine$quality)
wine <- clean_names(wine)       # removes spaces from names

set.seed(4630)
sample.data<-sample.int(nrow(wine), floor(0.5*nrow(wine)), replace = F)
```

```
train<-wine[sample.data, ]
test<-wine[-sample.data, ]

y.test<-test[,"alcohol"]
```

b - output of summary function

```
tree1 <- tree(alcohol~. , train)
summary(tree1)
```

```
##
## Regression tree:
## tree(formula = alcohol ~ ., data = train)
## Variables actually used in tree construction:
## [1] "density"        "fixed_acidity"  "residual_sugar" "sulphates"
## [5] "quality"        "citric_acid"
## Number of terminal nodes:  13
## Residual mean deviance:  0.4656 = 365.9 / 786
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -2.32200 -0.43140 -0.04444  0.00000  0.41370  3.57800
```

b - terminal nodes

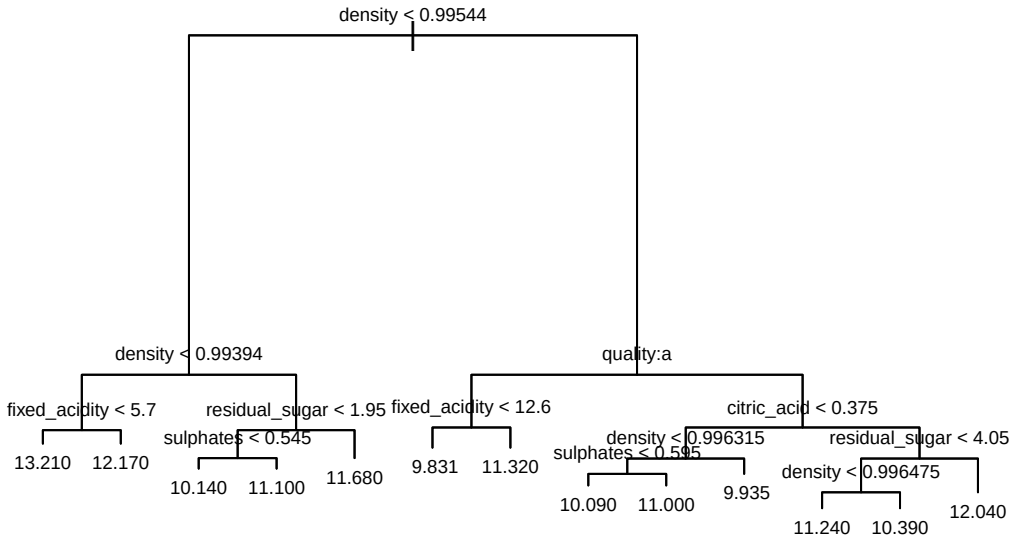
- there are 13 terminal nodes

b - predictors used

- density, fixed acidity, residual sugar, suplhates, quality, and citric acid

b - graph

```
plot(tree1)
text(tree1, cex=0.6)
```



b - test mse

```
tree1.pred<-predict(tree1, newdata =test)
mse.tree1 <- mean((tree1.pred-y.test$alcohol)^2)
print(mse.tree1)
```

```
## [1] 0.5121195
```

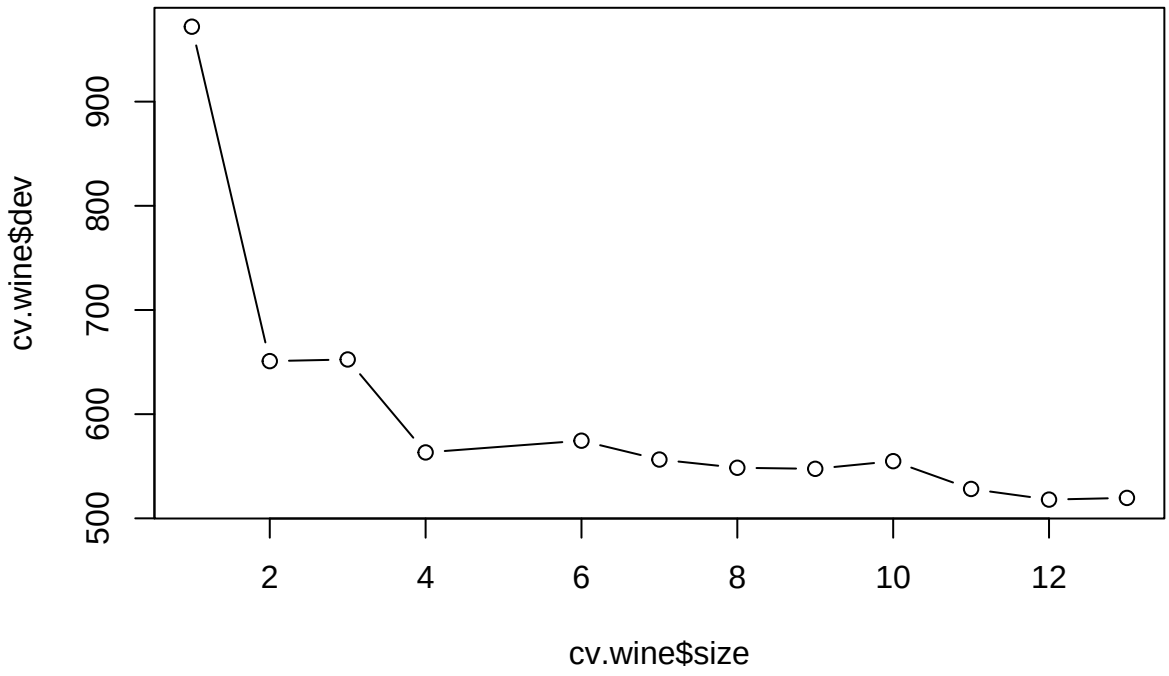
c - Pruned tree

```
set.seed(4630)
cv.wine<-cv.tree(tree1, K=10)
cv.wine
```

```
## $size
##  [1] 13 12 11 10  9  8  7  6  4  3  2  1
##
## $dev
##  [1] 519.6275 518.0125 528.2267 554.8960 547.5422 548.5763 556.4566 574.5341
##  [9] 563.2866 652.5016 650.9141 971.9087
##
```

```
## $k
##  [1]      -Inf  10.10262  11.16693  12.99857  14.76357  15.32526  19.46448
##  [8]  26.94100  31.19840  50.72324  53.31209 326.57416
##
## $method
## [1] "deviance"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```
plot(cv.wine$size, cv.wine$dev,type='b')
```




```
trees.num.wine<-cv.wine$size[which.min(cv.wine$dev)]
trees.num.wine
```

```
## [1] 12
```

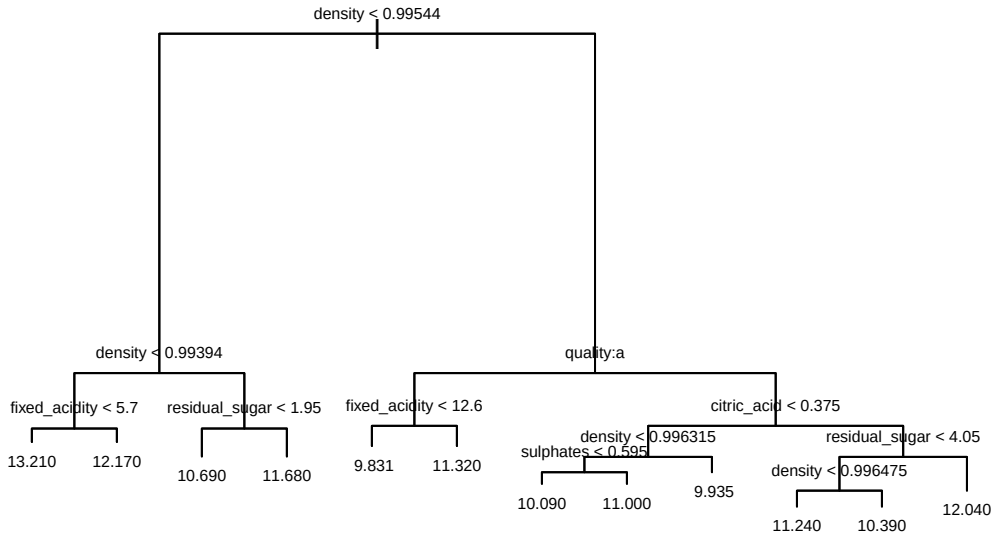
```
tree.full<-tree::tree(alcohol~., data = train)
prune.full<-tree::prune.tree(tree.full, best=trees.num.wine)
summary(prune.full)
```

```
##
## Regression tree:
## snip.tree(tree = tree.full, nodes = 10L)
```

```
## Variables actually used in tree construction:
## [1] "density"        "fixed_acidity"  "residual_sugar" "quality"
## [5] "citric_acid"    "sulphates"
## Number of terminal nodes:  12
## Residual mean deviance:  0.4778 = 376 / 787
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -2.32200 -0.43540 -0.03929  0.00000  0.43410  3.57800
```

c - graph

```
plot(prune.full)
text(prune.full, cex=0.5)
```



c - test mse

```
tree.prune.pred<-predict(prune.full, newdata =test)
mse.tree2 <- mean((tree.prune.pred-y.test$alcohol)^2)
print(mse.tree1)
```

```
## [1] 0.5121195
```

d - random forest

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.1.3
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

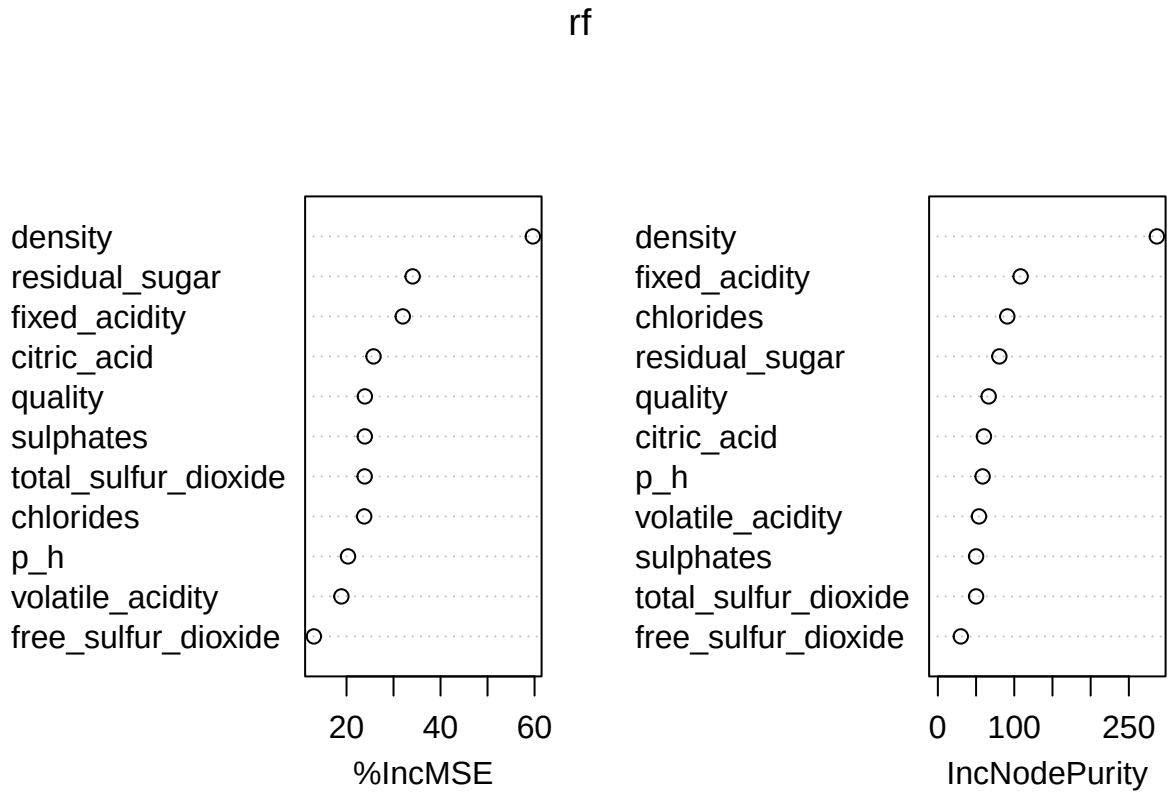
```
rf<-randomForest::randomForest(alcohol~., data=train, mtry=3,importance=TRUE)
rf
```

```
##
## Call:
##  randomForest(formula = alcohol ~ ., data = train, mtry = 3, importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##           Mean of squared residuals: 0.344575
##                     % Var explained: 71.61
```

```
importance(rf)
```

```
##                       %IncMSE IncNodePurity
## fixed_acidity        31.96819     108.31021
## volatile_acidity     18.91228      53.80856
## citric_acid          25.74406      60.26742
## residual_sugar       34.06240      80.47612
## chlorides            23.76029      90.87450
## free_sulfur_dioxide  13.05502      30.00289
## total_sulfur_dioxide 23.86067      50.03898
## density              59.63640     286.62080
## p_h                  20.31392      58.55422
## sulphates            23.88765      50.04959
## quality              23.90743      66.48568
```

```
varImpPlot(rf)
```



d- test mse

```
rf.pred<-predict(rf, newdata =test)
mse.rf <- mean((rf.pred-y.test$alcohol)^2)
print(mse.rf)
```

```
## [1] 0.303565
```

e - table

```
out <- data.frame(mse.tree1, mse.tree2, mse.rf)
colnames(out) <- c("MSE.RecBinary", "MSE.Pruned", "MSE.RandomForest")
out
```

```
##   MSE.RecBinary MSE.Pruned MSE.RandomForest
## 1     0.5121195  0.5164452         0.303565
```

- Main conclusions found in PDF

Section 5: Classification Trees

a)

```
library(readr)
library(tree)

Data<-read.csv("wine.csv", header=T)
Data$quality<- factor(Data$quality)

set.seed(4630)

sample.data<-sample.int(nrow(Data), floor(.50*nrow(Data)), replace = F)
train<-Data[sample.data, ]
test<-Data[-sample.data, ]
y.test<-test$quality
```

b)

i)

```
tree.class.train<-tree::tree(quality~., data=train)
summary(tree.class.train)
```

```
##
## Classification tree:
## tree::tree(formula = quality ~ ., data = train)
## Variables actually used in tree construction:
## [1] "alcohol"              "volatile.acidity"     "sulphates"
## [4] "total.sulfur.dioxide" "free.sulfur.dioxide"
## Number of terminal nodes:  11
## Residual mean deviance:  0.9965 = 785.3 / 788
## Misclassification error rate: 0.219 = 175 / 799
```

ii)

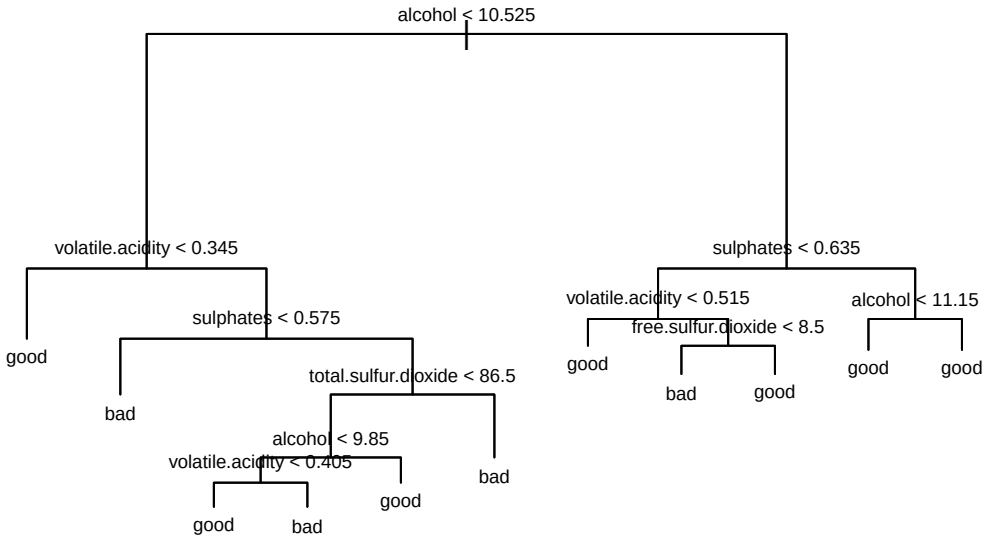
11 terminal nodes were used in the tree creation.

iii)

The predictors used in the tree include alcohol, volatile acidity, sulphates, total.sulfur.dioxide, and free.sulfur.dioxide.

iv)

```
plot(tree.class.train)
text(tree.class.train, cex=0.6, pretty=0)
```



v)

The classification tree shows us the most important predictors in the classification of a wine as good or bad. The tree demonstrates that alcohol, volatile.acidity, sulphates, total.sulfur.dioxide, and free.sulfur.dioxide are the most important predictors of a wines classification. Specifically, we can see that alcohol content is the most important predictor, followed by sulphates and volatile.acidity.

vi)

```
tree.pred.test<-predict(tree.class.train, newdata=test, type="class")
table(y.test, tree.pred.test)
```

```
##       tree.pred.test
## y.test bad good
##   bad  258  113
##   good  98  331
```

vii)

```
error_rate<-(113+98)/(113+258+98+331)
error_rate
```

```
## [1] 0.26375
```

viii)

```
fpr<-(113)/(113+258)
fpr
```

```
## [1] 0.3045822
```

ix)

```
fnr<-(98)/(98+331)
fnr
```

```
## [1] 0.2284382
```

x)

Lowering the threshold will increase the false positive rate, while increasing the threshold will increase the false negative rate. Because we are concerned with classifying a wine correctly as good or bad, we would not want to have significantly higher false positive rates or false negative rates, indicating a wine would be misclassified overly as good or bad respectively. Therefore, it is best to keep the threshold at 0.5.

c)

i)

```
set.seed(4630)
cv.class<-tree::cv.tree(tree.class.train, K=10, FUN=prune.misclass)
tree.num.class<-cv.class$size[which.min(cv.class$dev)]
prune.class<-tree::prune.misclass(tree.class.train, best=tree.num.class)
summary(prune.class)
```

```
##
## Classification tree:
## snip.tree(tree = tree.class.train, nodes = 3L)
## Variables actually used in tree construction:
## [1] "alcohol"              "volatile.acidity"     "sulphates"
## [4] "total.sulfur.dioxide"
## Number of terminal nodes:  7
## Residual mean deviance:  1.082 = 856.7 / 792
## Misclassification error rate: 0.239 = 191 / 799
```

ii)

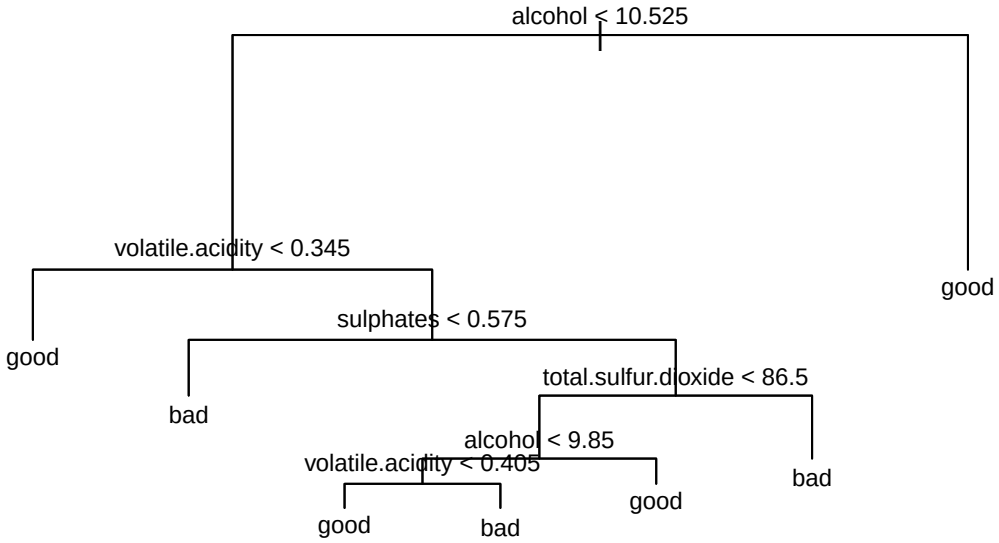
The tree has 7 terminal nodes.

iii)

The predictors used in the tree include alcohol, volatile acidity, sulphates, and total.sulfur.dioxide, as shown in the summary output.

iv)

```
plot(prune.class)
text(prune.class, cex=0.75, pretty=0)
```



v)

The classification tree shows us the most important predictors in the classification of a wine as good or bad. The tree demonstrates that alcohol, volatile.acidity, sulphates, and total.sulfur.dioxide, are the most important predictors of a wines classification. Specifically, we can see that alcohol content is the most important predictor, followed by volatile.acidity and sulphates. Unlike the unpruned tree, free sulfur dioxide was not an important predictor.

vi)

```
tree.prune.test<-predict(prune.class, newdata=test, type="class")
table(y.test, tree.prune.test)
```

```
##       tree.prune.test
## y.test bad good
##   bad  245  126
##   good  90  339
```

vii)

```
error_rate2<-(126+90)/(126+90+339+245)
error_rate2
```

```
## [1] 0.27
```

vii)

```
fpr2<-126/(126+245)
fpr2
```

```
## [1] 0.3396226
```

ix)

```
fnr2<-90/(90+339)
fnr2
```

```
## [1] 0.2097902
```

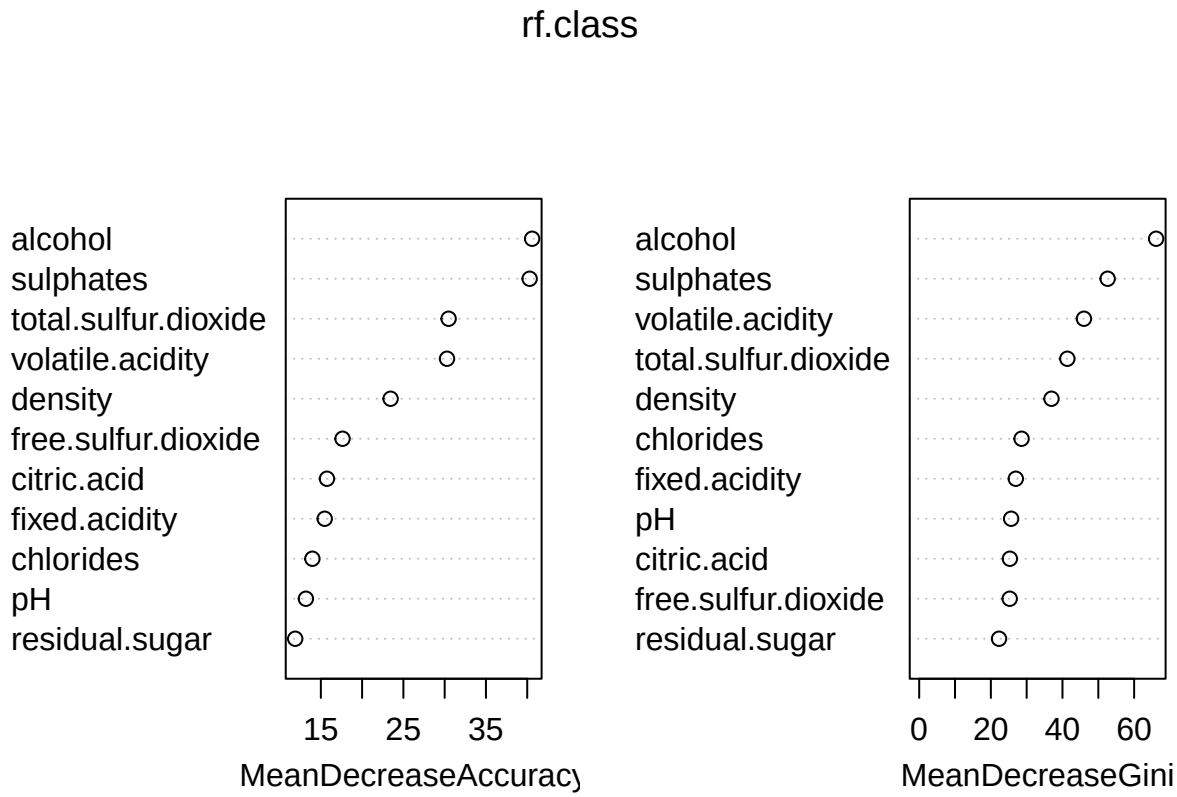
x)

Because the FPR is significantly higher than the FNR, it may be a good idea to reduce the threshold. However, after experimenting with lowering the threshold, any change to lower the FPR significantly increases the FNR and slightly reduces the accuracy. Therefore, even though the FPR is significantly higher than the FNR, any reduction in the FPR will not reduce the error rate and will just significantly increase the FPR. Because we are interested in classification of a wine as good and bad equally, reducing the threshold does not benefit us. Overall we can see that the pruned tree has a slightly higher error rate, FPR and a slightly lower FNR. Thus, overall the pruned tree does not perform better than the unpruned tree.

d)

i)

```
library(randomForest)
set.seed(4630)
rf.class<-randomForest::randomForest(quality~., data=train, mtry=3, importance=TRUE)
randomForest::varImpPlot(rf.class)
```



After using mtry=3, because we have 11 predictors, so the square root of 11 rounded down is 3, we found similar results as before. Alcohol was again the most important predictor followed by sulphates, volatile.acidity, and total.sulfur.dioxide. Free.sulfur.dioxide, and citric.acid were the least important.

ii)

```
pred.rf<-predict(rf.class,newdata=test)
table(y.test,pred.rf)
```

```
##       pred.rf
## y.test bad good
##   bad  279   92
##   good  76  353
```

iii)

```
error_rf<-(81+88)/(284+87+80+349)
error_rf
```

```
## [1] 0.21125
```

iv)

```
fpr_rf<-88/(88+283)
fpr_rf
```

```
## [1] 0.2371968
```

v)

```
fnr_rf<-81/(81+348)
fnr_rf
```

```
## [1] 0.1888112
```

vi)

Lowering the threshold will increase the false positive rate, while increasing the threshold will increase the false negative rate. Because we are concerned with classifying a wine correctly as good or bad, we would not want to have significantly higher false positive rates or false negative rates, indicating a wine would be misclassified overly as good or bad respectively. Because the FPR and FNR are pretty close together it is best to keep the threshold at 0.5.

e)

ii)

The models answer our question by showing us which predictors are most important in a wines classification. Overall, we found that alcohol was the most important predictor, followed by sulphates, volatile.acidity, and total.sulfur.dioxide.

iii)

Random Forests best answered our question because it had the highest accuracy, and also showed the most important predictors that I mentioned above. The findings were not surprising because we had initially suspected that alcohol content would be important in a wines classification. Additionally from our EDA we found that sulphates, volatile.acidity and total.sulfur.dioxide all had relatively different distributions for wines classification as good or bad, so it makes sense that varying values of these variables would influence a wines classification.

iv)

We were sometimes unsure of whether or not we should reduce or raise our threshold in order to change the accuracy, FPR, or FNR. Though in the end we decided to keep the threshold at 0.5 because any changes to it slightly reduced accuracy and increased the FNR a large amount. Because our research question is to find what predictors classify a wine as good AND bad, we are not especially interested in reducing the FPR or FNR, but rather reducing both of them. Besides this we did not face any challenges in this section.