wine quality analysis

Omid Lavakhamseh

16/12/2021

data description and research objective:

In this Lab, we've chosen wine quality-white dataset, which has 12 features with 4898 observations. Our aim is to predict the quality of wine (classify quality from 1 to 10). Therefore, we separated data in 2 different ways and different models, which each one can be shown below:

At first, let's take a look to the data: Due to privacy and logistic issues, only physicochemical (inputs) and sensory (the output) variables are available (e.g. there is no data about grape types, wine brand, wine selling price, etc.).

Input variables (based on physicochemical tests):

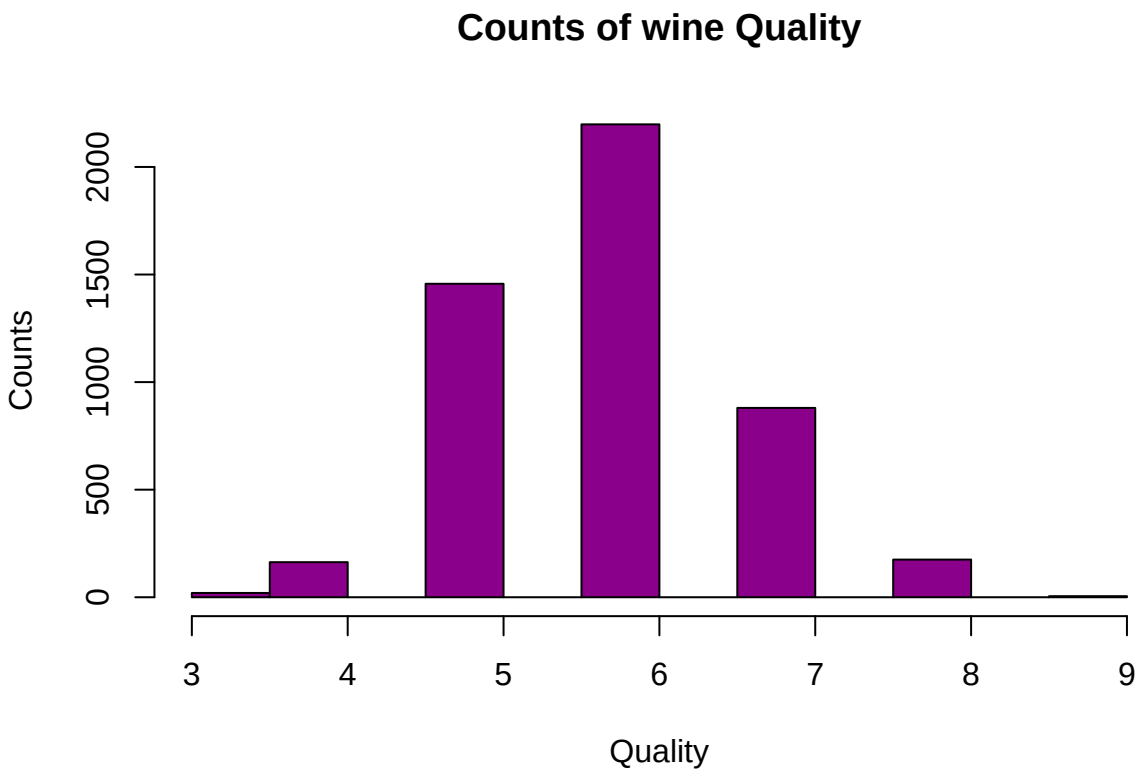
1. fixed acidity
2. volatile acidity
3. citric acid
4. residual sugar
5. chlorides
6. free sulfur dioxide
7. total sulfur dioxide
8. density
9. pH
10. sulphates
11. alcohol

Output variable (based on sensory data):

12. quality (score between 0 and 10)

.

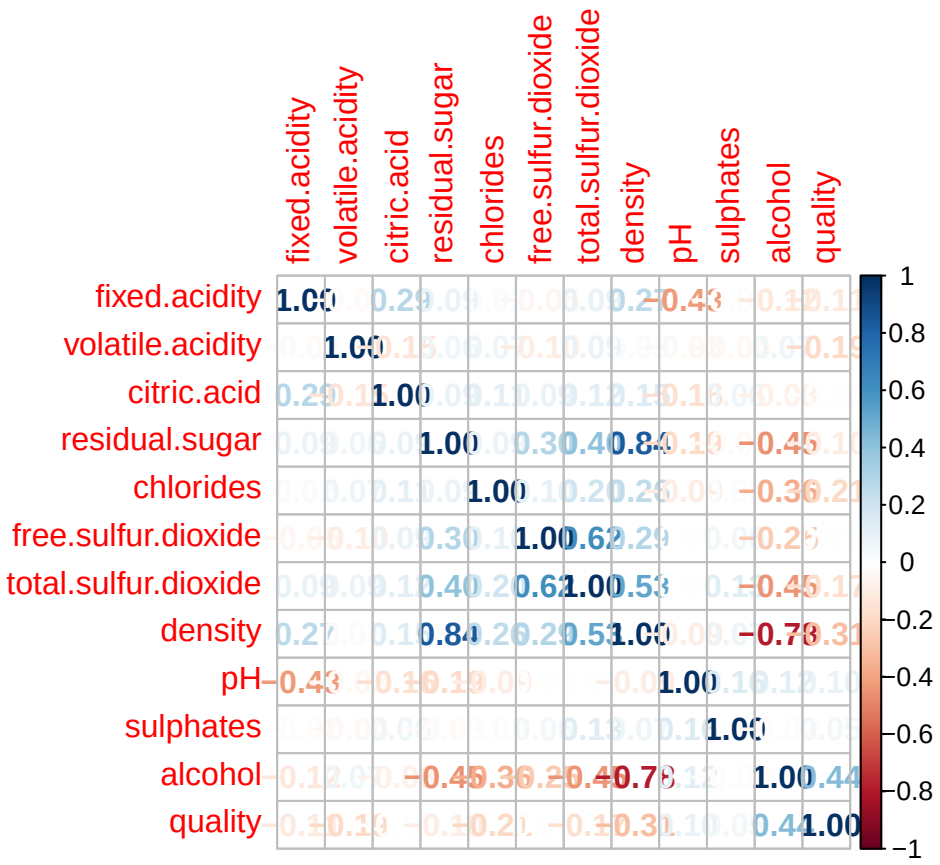
In below column chart we can see that qualities 5, 6 and 7, have the most quality classes among our data.




```
##
##    3    4    5    6    7    8    9
##   20  163 1457 2198  880  175    5
```

Next step is assessing our data more by checking correlation among features and between features and quality.

```
## corrplot 0.92 loaded
```



here we can see that, correlation between our target which is quality, is weak through our features and it affects accuracy of our prediction. In addition, there is a relation between features, for instance the correlation between residual.sugar and density is 0.84 and the correlation between density and alcohol is -0.78 (as our research, when the correlation of features is approximately 0.8, this correlation is high and it affects our prediction).

So in feature selection, between using PCA or removing a feature, we decided to remove a feature from our data. the correlation between quality and residual.sugar is 0.1 and between quality and density is -0.31, so we decide to remove residual.sugar.

Based on Lab block 2 instructions, we've done our work based on methods below:

different data partitioning: our data partitioning is different into 2 ways, which we used set.seed(123456) and splitted data into train, validation and test set by 60, 20 and 20 percent respectively, and in the second method, we used set.seed(0), and splitted data by 50, 25 and 25 percent respectively.

test wide ranges of hyper parameter values: in 1st model which is KNN, Omid used 50 different K values in validation part in order to find out the best hyper parameter, and in 2nd model which is Decision Tree, Hamed studied the trees up to 20 leaves to find the optimal tree depth.

feature/model selection: in model selection based on what we've learned in lectures, we used hold out method, and in feature selection we've used corrplot and decided to remove a feature.

Descriptions of models used and experimental design

our data set is wine quality which we should predict the quality of wine and it's a multi class classification. there are various models to use for multi class classification, but by doing some research in this case, we

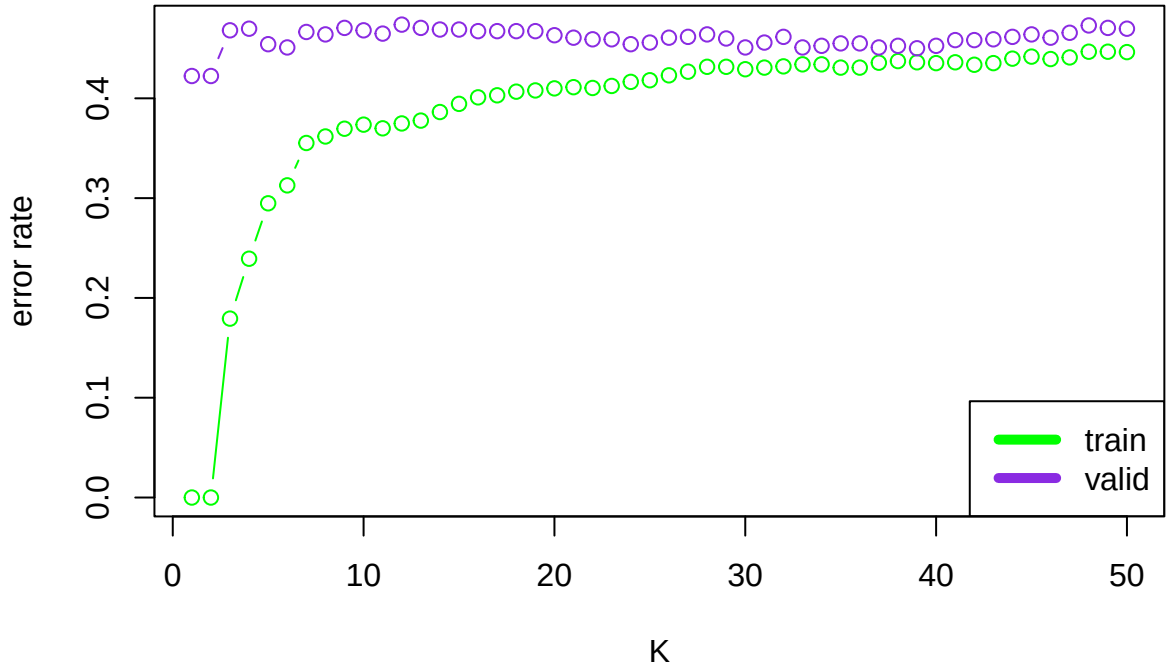
understand that KNN and Decision Tree sound good, so we tried to do use KNN and Decision Tree for predicting the class of wine quality.

First Method:KNN

KNN is a simple model, its only issue is being slow in large data sets because it must compare new points with all stored samples. here we have an almost small data set, so KNN can be a good choice. we used KNN in both our partitioning methods.

next step is calculating optimal k. I have considered 1<k<50 to calculate optimal k. optimal k would be the least validation error. Below plot describes trend of train and validation errors for various values of k.

```
## Warning: package 'kknn' was built under R version 4.2.3
```



As we see in the plot , train error is always less than valid and we expected this because this is the part of data that has been used to fit the model. Another obvious point to mention is about the trend of training and validation error by increasing k. As we know by increasing k model complexity would be decreased and model would be simpler.So as we expect the trend of training error is very low at k=1,2 (about zero) and then by increasing the value of k training error increases too. by calculating minimum value of validation error optimal k would be K=1, although it is complex but it works well for unseen data, too. Then I have used train validation data to train the model in K=1(optimal k) and test it on test data. Using more data to train the model would increase accuracy , it is exactly the reason that I have used the summation of train and validation to fit my optimal model and lower test error rate rather than what we achieved in validation error rate in K=1(which was 0.409) is expected.

here you can find confusion matrix for optimal model:

```
##    prediction_test_optim
```

```
##       3   4   5   6   7   8   9
##   3   0   1   0   3   0   1   0
##   4   0  14  17  12   2   0   0
##   5   1  10 238 100  10   3   0
##   6   1   9  95 375  73   4   0
##   7   0   1  10  53 131  13   1
##   8   0   0   1  11   8  26   0
##   9   0   1   0   0   0   0   0
```

and below we can see accuracy and error , respectively.

```
accuracy_test_optim
```

```
## [1] 0.64
```

```
Missclassification_error_test_optim
```

```
## [1] 0.36
```

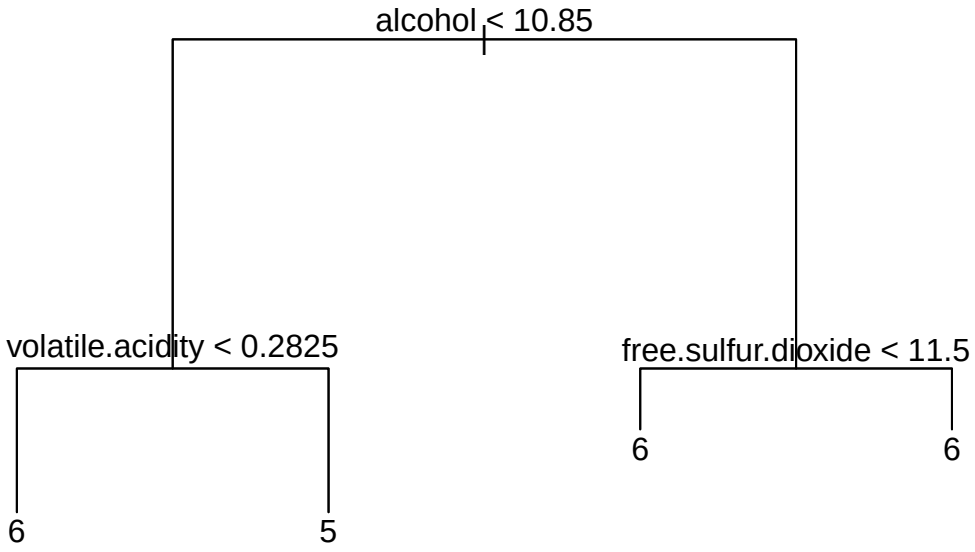
As we expected test error rate is 0.362 and it is lower than valid error rate in K=1 (which was 0.409) and this is because of using more data (trva) to fit the optimal model.

Now, we train Decision Tree on our our 1st partition method.

2nd method Decision Tree

We used decision tree as our second model. Decision Tree is used to solve both classification and regression problems. But the main drawback of Decision Tree is that it generally leads to overfitting of the data. It has clear visualization. The algorithm is simple to understand, interpret and visualize as the idea is mostly used in our daily lives. Output of a Decision Tree can be easily interpreted by humans. and it does not need feature scaling.

we train our decision tree model based on our 1st method of splitting data and plot the tree.

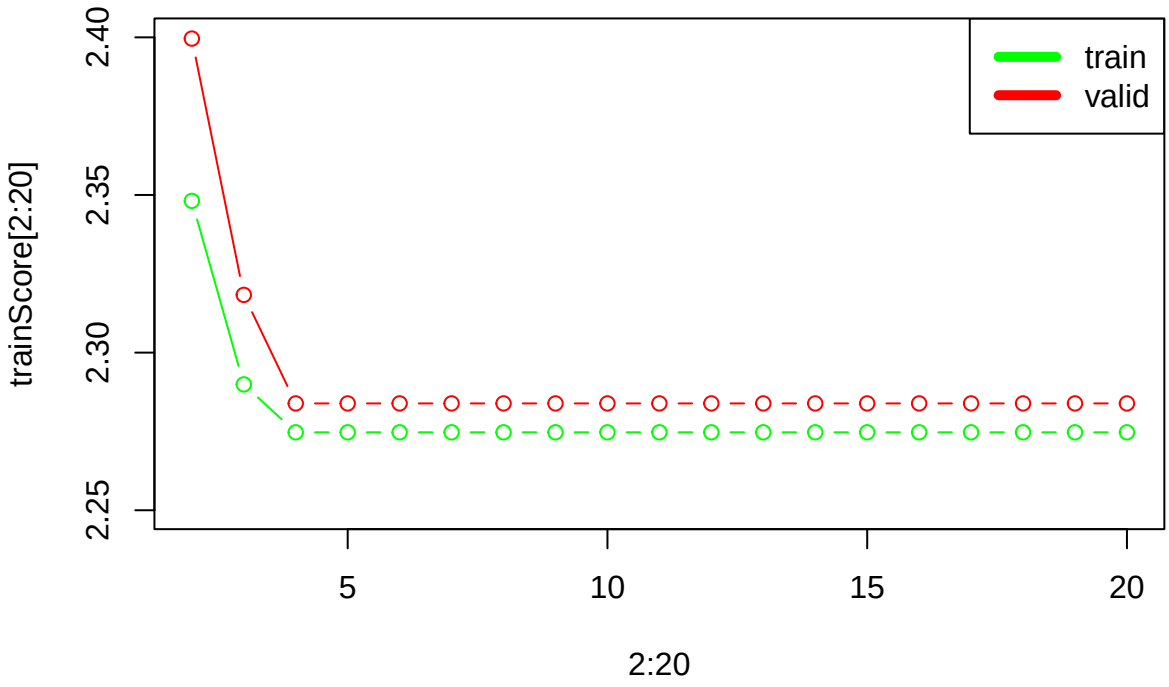


and now we can see that our miss classification error for training and validation data.

```
## MCE of train data is:  0.4887709
```

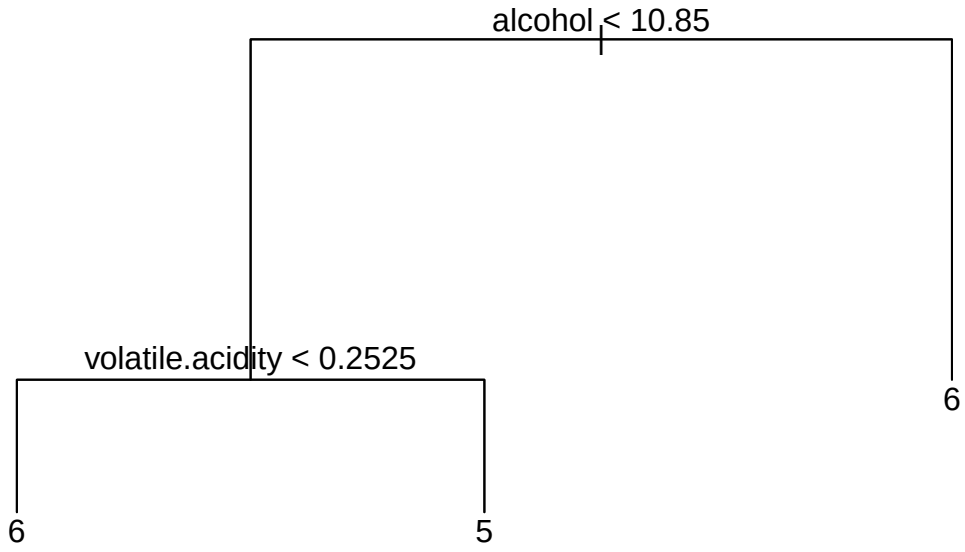
```
## MCE of validation data is 0.496732
```

in this part we try to prune our tree and find our hyper parameter (tree depth) in validation part. We studied the trees up to 20 leaves to find the optimal tree depth in the model.




```
## optimal hyper parameter is:  3
```

after finding the optimal hyper parameter (tree depth), we train (train plus validation part).



and now we can see our miss classification error for our unseen data (test data) after finding the best hyper parameter.

```
## MCE of test after finding optimal hyper parameter:  0.4840816
```

comparison

by comparing these 2 models (KNN and decision tree) onto our 1st partitioning method, we will find below information:

It's obvious from the table that, KNN has the least error.

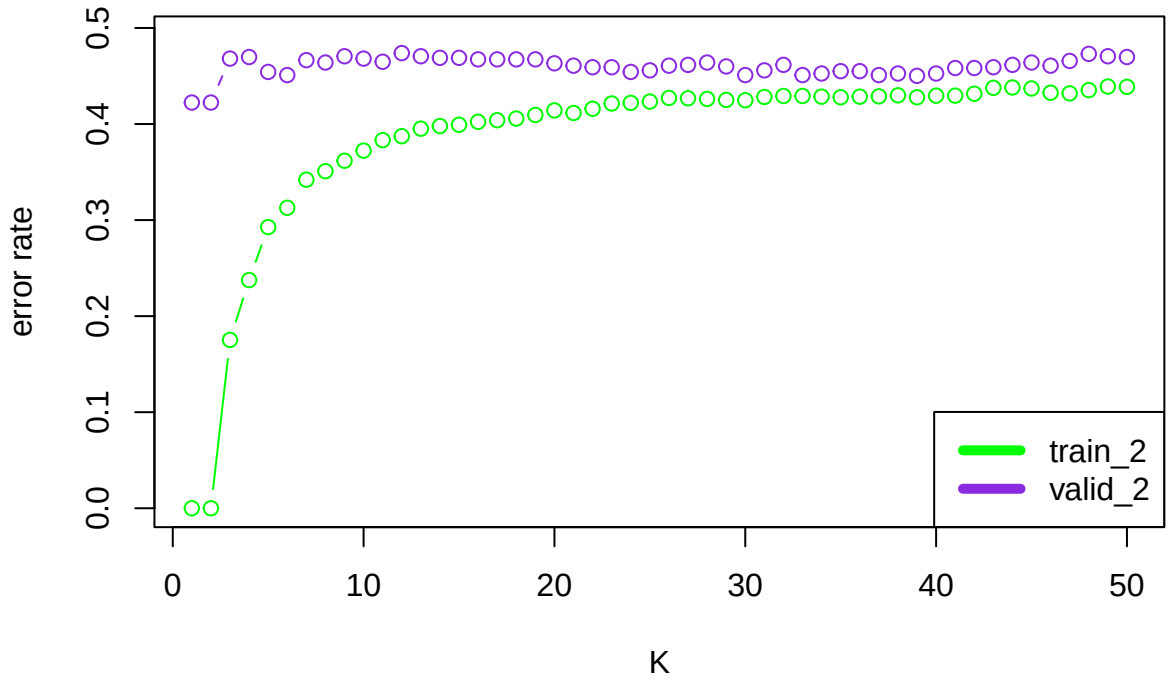
```
## MCE of KNN:  0.36
```

```
## MCE of Decision Tree:  0.4840816
```

2nd partitioning

now, we train out both models (KNN, Decision Tree) onto our 2nd partitioning method, and compare the missclassification errors.

train and valid error rate, confusion matrix and accuracy for optimal k are presented in below.

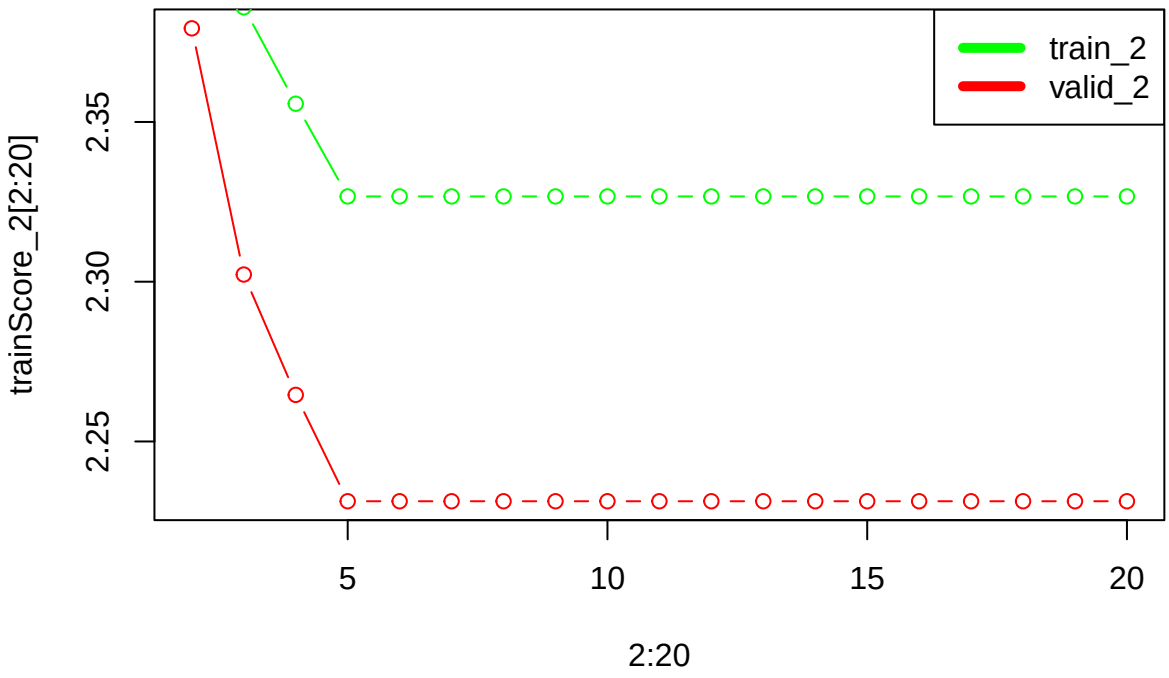



```
##    prediction_test_optim_2
##       3   4   5   6   7   8   9
##   3   0   0   2   2   0   0   0
##   4   0   9   6   8   2   0   0
##   5   0  18 198  72  10   1   0
##   6   0   6  69 305  56   8   0
##   7   0   1  10  47 110   4   0
##   8   0   0   1   2  18  15   0
##   9   0   0   0   0   1   0   0
```

```
## [1] 0.6493374
```

```
## [1] 0.3506626
```

and below is the result of fitting decision tree model on 2nd partitioning.


train_2
valid_2
trainScore_2[2:20]
2.25
2.30
2.35
5
10
15
20
2:20

comparison

by comparing these 2 models (KNN and decision tree) onto our 2nd partitioning method, we will find below information:

It's obvious from the table that, KNN has the least error.

```
## MCE of KNN:  0.3506626
```

```
## MCE of Decision Tree:  0.4872579
```

conclusion

So based on above comparison in terms of misclassification error, KNN is the best model in both partitioning methods with about 0.362895 error rate in 1st partitioning method and 0.35 in 2nd partitioning method.As we have mentioned in data description part ,quality has a low correlation with features so we have expected this high error rate. furthermore, we've used such other methods such as PCA to remove correlation between features, and other ML models which can be used for multi class classification and achieve higher accuracy, but as mentioned above, the correlation of our target with features is not high, thus we encounter high MCE.

Appendix:

```
#wine_data=read.csv('M:/LIU/ML/uni/labs/lab block2/winequality-white.csv',header = TRUE)
wine_data=read.csv('M:/LIU/ML/uni/labs/lab block2/winequality-white.csv',
                   header = TRUE)
hist(wine_data$quality,
     main = 'Counts of wine Quality',
     xlab = 'Quality',
     ylab = 'Counts',
     col = 'darkmagenta')

table(wine_data$quality)
library(corrplot)
corrplot(cor(wine_data),
         method = "number")

df=read.csv('M:/LIU/ML/uni/labs/lab block2/winequality-white.csv',
            header = TRUE)

#str(df)
#summary(df)

df$quality <- as.factor(df$quality)

# splitting data into 2 different ways
n=dim(df)[1]
set.seed(0)
id=sample(1:n, floor(n*0.5))
train=df[id,]
id1=setdiff(1:n, id)
set.seed(0)
id2=sample(id1, floor(n*0.25))
valid=df[id2,]
id3=setdiff(id1,id2)
test=df[id3,]
id4=setdiff(1:n,id3)
trva <- df[id4,]

set.seed(123456)
id=sample(1:n, floor(n*0.6))
train_2=df[id,]
id1=setdiff(1:n, id)
set.seed(123456)
id2=sample(id1, floor(n*0.2))
valid_2=df[id2,]
id3=setdiff(id1,id2)
test_2=df[id3,]
id4=setdiff(1:n,id3)
trva_2 <- df[id4,]

# 1st model - KNN - by partition 1

library(kknn)
```

```
errors1 = numeric(50)
errors2 = numeric(50)
for (i in 1:50){

  model <- kknn(formula = as.factor(quality) ~ ., train=train, test=train,
                kernel = "rectangular", k = i)
  Xfit1 <- model$fitted.values
  X1 <- c(train[,12])
  CM1 <- table(X1,Xfit1)
  errors1[i] <- 1-((sum(diag(CM1)))/sum(CM1))
  model_valid <- kknn(formula = as.factor(quality) ~ ., train=train, test=valid,
                      kernel = "rectangular", k = i)
  Xfit2 <- model_valid$fitted.values
  X2 <- c(valid[,12])
  CM2 <- table(X2,Xfit2)
  errors2[i] <- 1-((sum(diag(CM2)))/sum(CM2))
}
plot(c(1:50), errors1, ylim=range(errors1,errors2), type = "b",
     col="green", ylab = "error rate", xlab = "K")
points(errors2, type = "b", col="blueviolet")
legend("bottomright", legend = c('train',"valid"), lwd = 5,
       col = c('green','blueviolet'))
optimal_k=which.min(errors2)


model_test_optim <- kknn(as.factor(quality)~., trva,test,k= 1,
                         kernel = "rectangular", scale = TRUE)
prediction_test_optim <- predict(model_test_optim,newdata=test[,-12])
CM_test_optim <- table(test[,12],prediction_test_optim);CM_test_optim
accuracy_test_optim <- (sum(diag(CM_test_optim)))/sum(CM_test_optim)
Missclassification_error_test_optim <-
  1 - (sum(diag(CM_test_optim)) / sum(CM_test_optim))

library("tree")

# 2nd model - Decision Tree by partition 1

N_train = dim(train)[1]
fit=tree(as.factor(quality)~.-residual.sugar, data=train,
         control = tree.control(nobs = N_train))
plot(fit)
text(fit, pretty = 0)

missclass=function(X,X1){
  n=length(X)
  return(1-sum(diag(table(X,X1)))/n)
}

Yfit=predict(fit, newdata=train, type="class")
MC_train = missclass(train$quality,Yfit)
cat('MCE of train data is: ', MC_train)
Yfit=predict(fit, newdata=valid, type="class")
MC_valid = missclass(valid$quality,Yfit)
```

```
cat('MCE of validation data is', MC_valid)


nrow_train <- nrow(train)
nrow_valid <- nrow(valid)
trainScore=rep(0,20)
validScore=rep(0,20)
for(i in 2:20) {
  prunedTree=prune.tree(fit,best=i)
  pred=predict(prunedTree, newdata=valid,type="tree")
  trainScore[i]=deviance(prunedTree)/nrow_train
  validScore[i]=deviance(pred)/nrow_valid
}
plot(2:20, trainScore[2:20], type="b", col="red", ylim = c(2.25,2.4))
points(2:20, validScore[2:20], type="b", col="green")
legend("topright", legend = c('train',"valid"), lwd = 5, col = c('green','red'))

best <- which.min(validScore[2:20])
cat('optimal hyper parameter is: ', best)


n_trva <- dim(trva)[1]

fit2 <- tree(as.factor(quality)~.-residual.sugar, data=trva,
             control = tree.control(nobs = n_trva))

finalTree=prune.tree(fit2, best=best)
plot(finalTree)
text(finalTree, pretty = 0)

Yfit_test = predict(finalTree, newdata=test, type="class")
Confustion_matrix = table(test$quality,Yfit_test)
mce = 1 - sum(diag(Confustion_matrix))/length(Yfit_test)
cat('MCE of test after finding optimal hyper parameter: ', mce)


cat('MCE of KNN: ', Missclassification_error_test_optim) # 0.362895
cat('MCE of Decision Tree: ', mce) # 0.4840816


# 1st model - KNN - by partition 2

errors1_2 = numeric(50)
errors2_2 = numeric(50)
for (i in 1:50){

  model2 <- kknn(formula = as.factor(quality) ~ ., train=train_2, test=train_2,
                 kernel = "rectangular", k = i)
  Xfit1_2 <- model2$fitted.values
  X1_2 <- c(train_2[,12])
  CM1_2 <- table(X1_2,Xfit1_2)
  errors1_2[i] <- 1-((sum(diag(CM1_2)))/sum(CM1_2))
  model_valid2 <- kknn(formula = as.factor(quality) ~ ., train=train_2, test=valid_2,
```

```
                        kernel = "rectangular", k = i)
  Xfit2_2 <- model_valid2$fitted.values
  X2_2 <- c(valid_2[,12])
  CM2_2 <- table(X2_2,Xfit2_2)
  errors2_2[i] <- 1-((sum(diag(CM2_2)))/sum(CM2_2))
}
plot(c(1:50), errors1_2, ylim=range(errors1_2,errors2_2), type = "b",
     col="green", ylab = "error rate", xlab = "K")
points(errors2, type = "b", col="blueviolet")
legend("bottomright", legend = c('train_2',"valid_2"), lwd = 5,
       col = c('green','blueviolet'))
optimal_k_2=which.min(errors2_2)

model_test_optim_2 <- kknn(as.factor(quality)~., trva_2,test_2,k= 1,
                           kernel = "rectangular", scale = TRUE)
prediction_test_optim_2 <- predict(model_test_optim_2,newdata=test_2[,-12])
CM_test_optim_2 <- table(test_2[,12],prediction_test_optim_2);CM_test_optim_2
accuracy_test_optim_2 <- (sum(diag(CM_test_optim_2)))/sum(CM_test_optim_2)
Missclassification_error_test_optim_2 <-
  1 - (sum(diag(CM_test_optim_2)) / sum(CM_test_optim_2))

accuracy_test_optim_2

Missclassification_error_test_optim_2


# 2nd model - Decision Tree by partition 1

N_train = dim(train)[1]
fit=tree(as.factor(quality)~.-residual.sugar, data=train,
         control = tree.control(nobs = N_train))
plot(fit)
text(fit, pretty = 0)

missclass=function(X,X1){
  n=length(X)
  return(1-sum(diag(table(X,X1)))/n)
}

Yfit=predict(fit, newdata=train, type="class")
MC_train = missclass(train$quality,Yfit)
Yfit=predict(fit, newdata=valid, type="class")
MC_valid = missclass(valid$quality,Yfit)

nrow_train <- nrow(train)
nrow_valid <- nrow(valid)
trainScore=rep(0,20)
validScore=rep(0,20)
for(i in 2:20) {
  prunedTree=prune.tree(fit,best=i)
  pred=predict(prunedTree, newdata=valid,type="tree")
  trainScore[i]=deviance(prunedTree)/nrow_train
  validScore[i]=deviance(pred)/nrow_valid
```

```
}
plot(2:20, trainScore[2:20], type="b", col="red")
points(2:20, validScore[2:20], type="b", col="green")
legend("topright", legend = c('train',"valid"), lwd = 5,
       col = c('green','red'))

best <- which.min(validScore[2:20])

n_trva <- dim(trva)[1]

fit2 <- tree(as.factor(quality)~.-residual.sugar, data=trva,
             control = tree.control(nobs = n_trva))

finalTree=prune.tree(fit2, best=best)
plot(finalTree)
text(finalTree, pretty = 0)

Yfit=predict(finalTree, newdata=valid, type="class")
MC_valid = missclass(valid$quality,Yfit)

Yfit_test = predict(finalTree, newdata=test, type="class")
Confustion_matrix = table(test$quality,Yfit_test)
mce = 1 - sum(diag(Confustion_matrix))/length(Yfit_test)


# 2nd model - Decision Tree by partition 2
N_train_2 = dim(train_2)[1]
fit_2=tree(as.factor(quality)~.-residual.sugar, data=train_2,
           control = tree.control(nobs = N_train_2))
plot(fit_2)
text(fit_2, pretty = 0)

Yfit_2=predict(fit_2, newdata=train_2, type="class")
MC_train_2 = missclass(train_2$quality,Yfit_2)
Yfit_2=predict(fit_2, newdata=valid_2, type="class")
MC_valid_2 = missclass(valid_2$quality,Yfit_2)

nrow_train_2 <- nrow(train_2)
nrow_valid_2 <- nrow(valid_2)
trainScore_2=rep(0,20)
validScore_2=rep(0,20)
for(i in 2:20) {
  prunedTree_2=prune.tree(fit_2,best=i)
  pred_2=predict(prunedTree_2, newdata=valid_2,type="tree")
  trainScore_2[i]=deviance(prunedTree_2)/nrow_train_2
  validScore_2[i]=deviance(pred_2)/nrow_valid_2
}
plot(2:20, trainScore_2[2:20], type="b", col="red")
points(2:20, validScore_2[2:20], type="b", col="green")
legend("topright", legend = c('train_2',"valid_2"), lwd = 5,
       col = c('green','red'))

best_2 <- which.min(validScore_2[2:20])
```

```
n_trva_2 <- dim(trva_2)[1]

fit2_2 <- tree(as.factor(quality)~.-residual.sugar, data=trva_2,
               control = tree.control(nobs = n_trva_2))

finalTree_2=prune.tree(fit2_2, best=best_2)
plot(finalTree_2)
text(finalTree_2, pretty = 0)

Yfit_2=predict(finalTree_2, newdata=valid_2, type="class")
MC_valid_2 = missclass(valid_2$quality,Yfit_2)

Yfit_test_2 = predict(finalTree_2, newdata=test_2, type="class")
Confustion_matrix_2 = table(test_2$quality,Yfit_test_2)
mce_2 = 1 - sum(diag(Confustion_matrix_2))/length(Yfit_test_2)


cat('MCE of KNN: ', Missclassification_error_test_optim_2) # 0.362895
cat('MCE of Decision Tree: ', mce_2) # 0.4840816
```