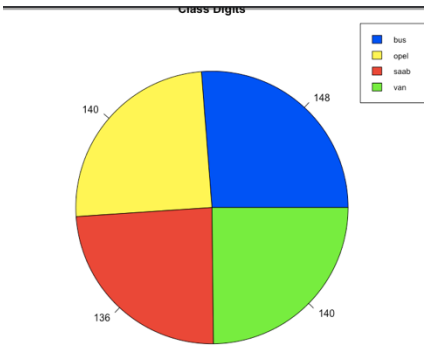
Homework#5
Farhana Afroze
Ub person# 50158114

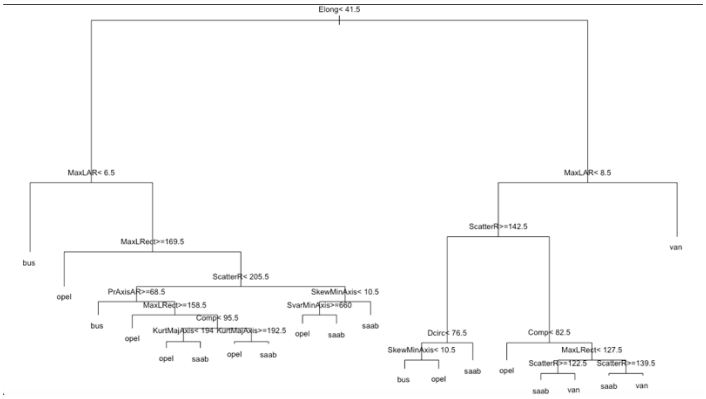
Problem#1
Exploring the dataset and visualization:
First, we will explore the vehicle dataset. It has 564 rows with 20 columns. There are no missing values in the dataset. There is one column called class-digit and it has 4 types of classes 1,2,3 and 4. There is another categorical column called 'class' and it 4 types of vehicle called bus, opel, saab and van. So, we would like to predict the 'class' column. I would like to see how many data points for each classes.



As we can see we have 148 bus, 140 opel, 136 saab and 140 van.

Splitting dataset and constructing classification tree:
Next we will split our data date into 80% for training and 20% for testing. In the training dataset we got 452 rows with 20 columns now. And for testing we got 112 rows with 20 columns. Next we would like to drop 'classdigit' categorical column and keep the 'class' categorical column for both training and testing set as we will predict that one.

Next we will build the classification tree. We chose 20 min-splits with 10 cross validation as it is default parameter. After building the model, fully grown classification tree looks like this:



As we can see from tree variable elong has the highest association with categorical response variable. Here we see if 'elong' is less than 41.5 then it goes to 'MaxLar' which has cutoff point

6.5 and this how it goes on and predict the classes. Next would like to see 'cp'- complexity parameter value.

	CP	nsplit	rel error	xerror	xstd
1	0.225225	0	1.00000	1.02402	0.027481
2	0.111111	1	0.77477	0.78679	0.031515
3	0.096096	3	0.55255	0.55856	0.031418
4	0.048048	4	0.45646	0.49550	0.030738
5	0.024024	5	0.40841	0.50751	0.030890
6	0.021021	7	0.36036	0.49850	0.030777
7	0.012012	8	0.33934	0.48048	0.030531
8	0.010010	10	0.31532	0.45345	0.030113
9	0.009009	13	0.28529	0.44745	0.030012
10	0.006006	15	0.26727	0.44444	0.029961
11	0.003003	17	0.25526	0.43243	0.029747
12	0.000000	19	0.24925	0.42342	0.029579

Here we can see after 12th cp the tree has stopped growing. The lowest cross validation error is cp 12.
Next we will predict the model using test data. After predicting we got testing accuracy is 62% and testing error is 38% which is not so bad.
Next we would like to see confusion matrix of our prediction.

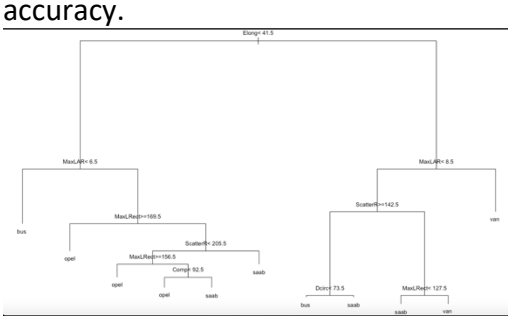
pred	true bus	opel	saab	van
bus	24	0	3	0
opel	3	12	13	0
saab	0	13	8	3
van	2	3	3	25

>

From this we can see 24 'bus' class, 12 'opel' class, 8 'saab' class and 25 'van' class were correctly predicted, and 38 data points were misclassified from their true values.

But as we see fully grown tree overfit the training data and thus do poor performance on the testing set. For limit this overfitting problem we can prune the tree. With pruning we can reduce overfitting problem and thus can do good on testing set. For pruning the tree, we will select the cp which has lowest cross validation error. We found cp = 12 has lowest cross validation error so we will select 12 for pruning. So, after pruning we got the same accuracy score 62% which we found using fully grown trees. So, it didn't change.

Next I changed minsplit to 50 to see how the classification tree will look like and to know the accuracy.



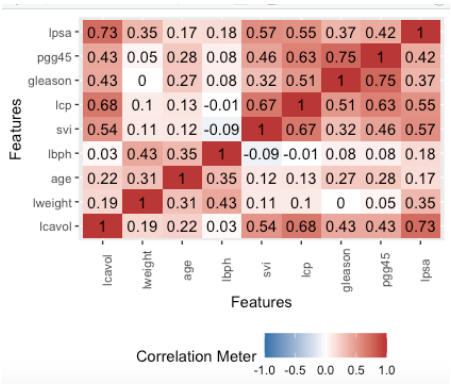
Here we see tree looks less complicated and simple but the mean accuracy we got is the same as with fully grown tree which is 62% and error is 28%. And we found out that cp 5 has the

lowest cross validation error. Next we will prune the tree using cp-5. Next we predicted and found mean accuracy is 61% which is less than our non- prune tree. So, we will ignore this tree. So, we can see pruning not always gives you more accuracy than non- prune tree.

Problem#2

Exploring the dataset and visualization:

In this problem we will look through the prostate data set. In the prostate data set we have 97 rows with 10 columns. There are no missing values in the dataset. We would like to see correlation of the data set.



Here we see 'pgg45' and 'gleason' has highest positive correlation and there are some zero correlation between variables we can see.

Splitting the dataset and normalization:

In our data set we have one column called 'train' which includes 'true' and 'false' two categorical values. For all the data points which includes 'true' it would be used for training set and all the data points which includes 'false' it would be used for testing set. So, after splitting the dataset for the training set, we got 67 rows with 10 columns and for the testing set we got 30 rows with 10 columns. Next we would like to drop the 'train' column from our training and testing data set. After dropping the datasets looks like this:

```
> head(trains)
      lcavol  lweight age      lbph svi       lcp gleason pgg45       lpsa
1 -0.5798185 2.769459  50 -1.386294   0 -1.386294       6     0 -0.4307829
2 -0.9942523 3.319626  58 -1.386294   0 -1.386294       6     0 -0.1625189
3 -0.5108256 2.691243  74 -1.386294   0 -1.386294       7    20 -0.1625189
4 -1.2039728 3.282789  58 -1.386294   0 -1.386294       6     0 -0.1625189
5  0.7514161 3.432373  62 -1.386294   0 -1.386294       6     0  0.3715636
6 -1.0498221 3.228826  50 -1.386294   0 -1.386294       6     0  0.7654678
```

Now we got 9 columns for both training and testing set. We will use 'lpsa' for our dependent variable while keeping others as our predicting variables.

Next we will normalize our training and testing data set so that all the data points have range between 0 and 1. After normalizing the dataset looks like this:

```
> head(tests)
     lcavol   lweight       age      lbph svi       lcp   gleason      pgg45       lpsa
1 0.3562892 0.1876512 0.7777778 0.5625878   0 0.0000000 0.0000000 0.00000000 0.00000000
2 0.0000000 0.2080030 0.1481481 0.0000000   0 0.0000000 0.0000000 0.00000000 0.05850613
3 0.2353003 0.1170353 0.7407407 0.0000000   0 0.0000000 0.0000000 0.00000000 0.05850613
4 0.4666357 0.1779369 0.5185185 0.0000000   0 0.2227061 0.3333333 0.05555556 0.13144863
5 0.6674758 0.1961400 0.6296296 0.8042029   0 0.6373135 0.3333333 0.22222222 0.18531747
6 0.2734595 0.2474447 0.9629630 0.8392331   0 0.0000000 0.0000000 0.00000000 0.20055939
```

Next we will do best subset selection on our data set.

Best subset selection:

After doing best subset selection on eight predicting variables the outcome looks like this:

```
Selection Algorithm: exhaustive
         lcavol lweight age lbph svi lcp gleason pgg45
1  ( 1 ) "*"    " "     " " " "  " " " " " "     " "
2  ( 1 ) "*"    "*"     " " " "  " " " " " "     " "
3  ( 1 ) "*"    "*"     " " " "  "*" " " " "     " "
4  ( 1 ) "*"    "*"     " " "*"  "*" " " " "     " "
5  ( 1 ) "*"    "*"     " " "*"  "*" " " " "     "*"
6  ( 1 ) "*"    "*"     " " "*"  "*" "*" " "     "*"
7  ( 1 ) "*"    "*"     "*" "*"  "*" "*" " "     "*"
8  ( 1 ) "*"    "*"     "*" "*"  "*" "*" "*"     "*"
```

Here we see best 1 variable model is 'lcavol' and best 2 variable models are lcavol and lweight. We would like to see coefficient for all the eight predicting variables.

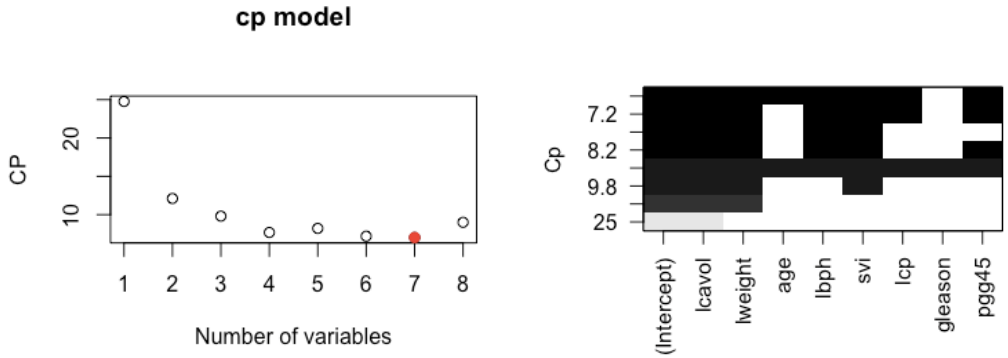
```
> coef(best_subset_model,8)
(Intercept)      lcavol     lweight         age        lbph         svi         lcp     gleason
 0.11352038  0.50431153  0.24998951 -0.12220771  0.09101825  0.12477526 -0.14118792 -0.01498041
      pgg45
 0.16020133
>
```

As we see for our dependent variable 'lpsa' has significant effect on predicting variable 'lcavol' and 'lweight'. But this can't tell that these two are the best variable model. So, we would like to choose optimal model from this using various algorithm.

Selecting optimal model using AIC and BIC:

AIC:

From CP/AIC we get the optimal mode is 7 which is 7 variable model. Here is the picture:



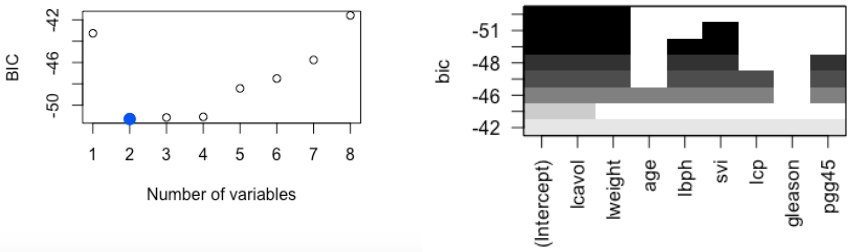
Here as we see for 7 variable model cp is lowest. In the right plot we see except predicting variable 'gleason' all other variables making 7 variables model is the best. Next I built the regression model using this 7 variables model to see how it's doing on the test data.

```
Call:
lm(formula = lpsa ~ pgg45 + gleason + lcp + svi + lbph + age +
    lweight + lcavol, data = trains)

Residuals:
     Min       1Q   Median       3Q      Max
-0.27905 -0.05779 -0.00918  0.07606  0.25164

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.11352    0.05963   1.904  0.06192 .
pgg45        0.16020    0.09218   1.738  0.08755 .
gleason     -0.01498    0.10213  -0.147  0.88389
lcp         -0.14119    0.07563  -1.867  0.06697 .
svi          0.12478    0.05053   2.469  0.01651 *
lbph         0.09102    0.04427   2.056  0.04431 *
age         -0.12221    0.08755  -1.396  0.16806
lweight      0.24999    0.09088   2.751  0.00792 **
lcavol       0.50431    0.09398   5.366 1.47e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1206 on 58 degrees of freedom
Multiple R-squared:  0.6944,    Adjusted R-squared:  0.6522
F-statistic: 16.47 on 8 and 58 DF,  p-value: 2.042e-12
```

As we can see all the predicting variables has good effect on 'lpsa' dependent variable. Next we calculated MSE error and we got mse 0.023% which is low and out linear model is doing good on this 7 variables model.

BIC:

Next we would like to select BIC for choosing best model.



As we can see BIC chose 2 variable model as 'BIC' has lowest rate on 2 variables model. We can see in the right picture that predicting variable 'lcavol' and 'lweight' is the most important variables to BIC.

Next we would like to bulid linear regression model on this two variables model from BIC. After building the model we test on the test set and MSE we got 0.024 which is almost same as AIC. So, this model is choosing less variable but doing good on testing set while AIC chooses 7 variable model but also did good on test data.

Five-fold cross validation:

Next we will do five folds cross validation into our dataset to select best model. The error we got after testing on the test data is:

```
> cross_errors
          [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]      [,8]
[1,] 0.02538318 0.01874114 0.02135591 0.01783601 0.01905253 0.01829982 0.01676866 0.01677190
[2,] 0.02867319 0.02271794 0.02811326 0.02419544 0.02491534 0.02494832 0.02321152 0.02393069
[3,] 0.01588220 0.01342635 0.01864649 0.01899162 0.02056422 0.01977480 0.01700297 0.01700375
[4,] 0.01238292 0.01468204 0.01582934 0.01657114 0.01452257 0.01459294 0.01196520 0.01197627
[5,] 0.02049697 0.01706562 0.01591535 0.01851443 0.01590377 0.01404142 0.01297500 0.01432428
>
```

Here we see 5 cross validation errors on eight predicting variables. We would like average all the column for each predicting variable to see which model is doing best.

```
> cross_error_ave
[1] 0.02056369 0.01732662 0.01997207 0.01922173 0.01899168 0.01833146 0.01638467 0.01680138
>
```

Here we see 7 variable model has the lowest error. So, we can tell five cross validation choosing 7 variables model as it has lowest error.

Ten-fold cross validation:

Here we will do ten folds cross validation to select the best variable model. The error we got after predicting on the test data here:

```
           [,1]       [,2]       [,3]       [,4]       [,5]       [,6]       [,7]       [,8]
 [1,] 0.011905054 0.012040997 0.012075102 0.008767947 0.013505960 0.012337893 0.012531257 0.013663392
 [2,] 0.026530453 0.025340732 0.022558590 0.023646429 0.026915149 0.024154463 0.025291167 0.026102404
 [3,] 0.014981940 0.018643572 0.029584490 0.030150580 0.031272312 0.028336602 0.026003283 0.025986121
 [4,] 0.015544117 0.025565810 0.030828006 0.035059029 0.043143841 0.039023274 0.037388777 0.038807277
 [5,] 0.033792110 0.019741134 0.014578741 0.010692411 0.013106240 0.010426390 0.009452008 0.009556971
 [6,] 0.009062183 0.012810590 0.018879609 0.019626838 0.018601265 0.015233951 0.013253813 0.013381226
 [7,] 0.013655047 0.009418461 0.012489227 0.009991660 0.012628382 0.010685167 0.008462718 0.008493439
 [8,] 0.012792731 0.011201289 0.006956017 0.005771942 0.005325921 0.004412083 0.003196132 0.003162390
 [9,] 0.030458519 0.022876761 0.021709106 0.018390065 0.023484020 0.020820856 0.019758522 0.019736991
[10,] 0.032109728 0.021787415 0.028029945 0.021234020 0.023707007 0.021365472 0.018334195 0.019093949
```

Here we see 10 folds cross validation errors on the 5 predicting variables testing set. Next we will average each columns to get each variables ten- fold CV error rate.

```
> cross_error_ave
[1] 0.02008319 0.01794268 0.01976888 0.01833309 0.02116901 0.01867962 0.01736719 0.01779842
> minimum = which.min(cross_error_ave)
> minimum
[1] 7
```

Here as we see 7 variable model has the lowest cross validation error. Ten folds cross validation also choosing 7 variable model as a best model because it has lowest CV error.

Bootsrap:

Now we are going to use bootsrap for our model selection. We will do bootstrap .632 estimates of prediction error. For prediction we took the original prostate data and did 100 resamples and predict on that. The errors we get:

```
> error_stores
[1] 0.6370963 0.5873591 0.5432592 0.5417074 0.5387179 0.5446464 0.5403472 0.5493005
>
```

Here we see 5 is the best

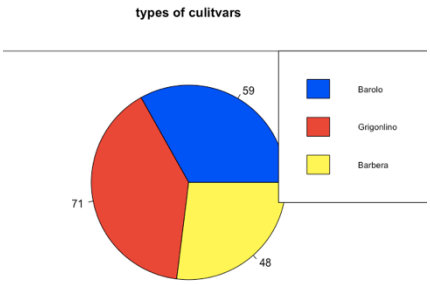
Problem#3

Exploring the dataset and visualization and splitting dataset:

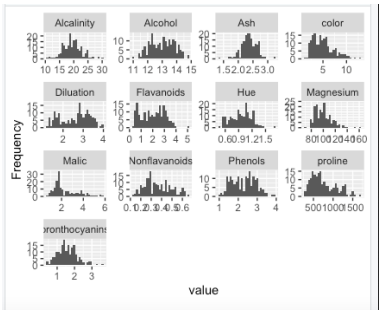
In the wine data set we first rename all the columns and made the categorical column to factor. Next we rename those categorical variables as 1-‘Barolo’, 2-‘Grigonlino’ and 3-‘barbera’. Here is our data set looks like now:

```
    Type Alcohol Malic  Ash Alcalinity Magnesium Phenols Flavanoids Nonflavanoids pronthocyanins
1 Barolo   14.23  1.71 2.43       15.6       127    2.80       3.06          0.28           2.29
2 Barolo   13.20  1.78 2.14       11.2       100    2.65       2.76          0.26           1.28
3 Barolo   13.16  2.36 2.67       18.6       101    2.80       3.24          0.30           2.81
4 Barolo   14.37  1.95 2.50       16.8       113    3.85       3.49          0.24           2.18
5 Barolo   13.24  2.59 2.87       21.0       118    2.80       2.69          0.39           1.82
6 Barolo   14.20  1.76 2.45       15.2       112    3.27       3.39          0.34           1.97
  color  Hue Dilution proline
1  5.64 1.04      3.92    1065
2  4.38 1.05      3.40    1050
3  5.68 1.03      3.17    1185
4  7.80 0.86      3.45    1480
5  4.32 1.04      2.93     735
6  6.75 1.05      2.85    1450
>
```

Here we have 178 rows with 14 columns. In our dataset there are no missing values. Next we would like to see how many observations for each categorical value.

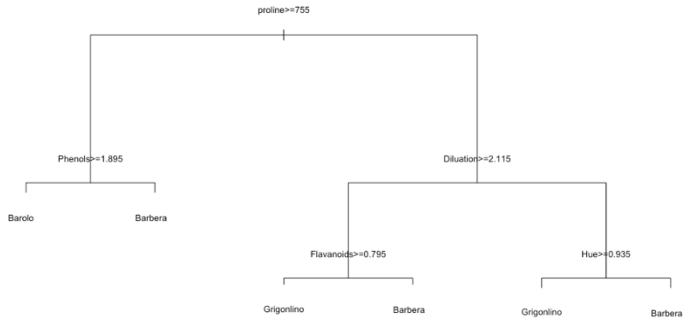


As we see there are 59 Barolo observation, 71 Grigonlino and 48 Barbera observation.



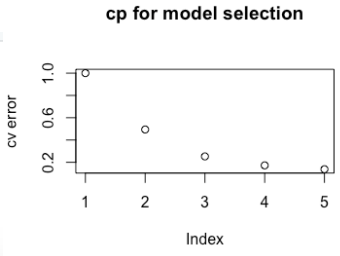
From the histogram we can see not all variables are normally distributed.
Next we would like to splitting our data set into training and testing. We kept 80% for our training data. So, in our training data set we have now 144 rows with 14 columns and our testing set we have now 34 rows with 14 columns.
Next we will construct appropriate size of classification tree.

Classification tree:
So, for creating classification tree we select minsplit =5 and xval = 10. Here is our classification tree looks like:



As we see here predictive variable ‘proline’ has the highest association with the response variable. Next we would like to see ‘cp’ parameter:

```
n= 144

        CP nsplit rel error  xerror     xstd
1 0.517241      0  1.000000 1.00000 0.067452
2 0.333333      1  0.482759 0.49425 0.063124
3 0.034483      2  0.149425 0.25287 0.049624
4 0.022989      4  0.080460 0.17241 0.042135
5 0.000000      5  0.057471 0.13793 0.038122
> 
```



As we see after 5th cp tree stopped growing. Here we see 5th cp has lowest cross validation error. Next we will predict our model using testing set. After prediction model we got the accuracy 88% which means the classification tree able to classify into 3 classes correctly about 88% and the classification error we got 12% which is not bad. Let's see the confusion matrix:

```
             true
pred          Barolo Grigonlino Barbera
  Barolo          10          2       1
  Grigonlino       1         12       0
  Barbera          0          0       8
>
```

Here we see we predicted 10 observation for 'barolo' correctly, 12 observation for 'Grigonlino' correctly and 8 observation for 'Barbera' correctly.
Next we will prune tree to see if it does more good than non-prune one. We select 5th cp value to prune. We got 88% accuracy and 12% error which is same as non-prune tree. So, we can see there is no effect doing pruning the tree on this tree.

Ensemble technique-Random Forest:
Here we will do one ensemble technique called random forest. Random forest works by ensembling all the decision trees. First, we will use the default random forest. After building the model on the training set it looks like this:

```
Call:
 randomForest(formula = Type ~ ., data = wine_training, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 3

        OOB estimate of  error rate: 2.78%
Confusion matrix:
           Barolo Grigonlino Barbera class.error
Barolo         47          1       0  0.02083333
Grigonlino      1         54       2  0.05263158
Barbera         0          0      39  0.00000000
>
```

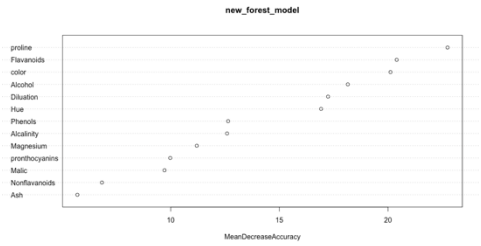
Here we see number of decision trees used is 500 and we see out of bag error is 2.78% and we can also see each class error. As we can see the model predicted correctly for wine class 'Barbera'
Next we would like to tune parameter for the random forest. For tuning we will keep ntree = 500 and mtry = 2 which is number of variables available for splitting at each tree node. So we run the model and let's see the output:

```
Call:
 randomForest(formula = Type ~ ., data = wine_training, ntree = 500,      mtry = 2, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 2

        OOB estimate of  error rate: 2.08%
Confusion matrix:
           Barolo Grigonlino Barbera class.error
Barolo         48          0       0  0.00000000
Grigonlino      1         54       2  0.05263158
Barbera         0          0      39  0.00000000
>
```

Here we see out of bag(OOB) error got less. Here our model improved than the default one. And we can also see the classification for each of the classes. Next we will predict this model using test set to see how it's doing on unseen data. After prediction we got 100% accuracy rate. Now let see the confusion matrix:

```
forest_predict Barolo Grigonlino Barbera
    Barolo         11           0       0
    Grigonlino      0          14       0
    Barbera         0           0       9
>
```

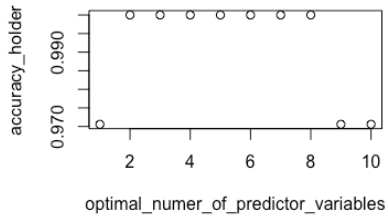
Here we see our tuning random forest did great job on classifying. We got 100% accuracy and all the classes are predicted correctly. Next we would like to see important variables in our dataset.



In this plot we can see proline, flavonoids and color are the most important variables. As if we remove these three variables it will result big misclassification error.

Next I would like set multiple parameter for mtry and build the model and see how the random forest is doing on all those mtry values.

```
> accuracy_holder
 [1] 0.9705882 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 0.9705882
[10] 0.9705882
```



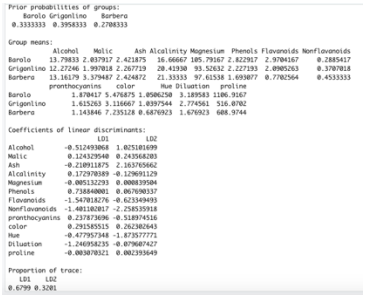
As we can see when mtry = 1 accuracy is low then from 2 to 8 mtry values accuracy stays same after 8 accuracy got less again.

Compare the performance:

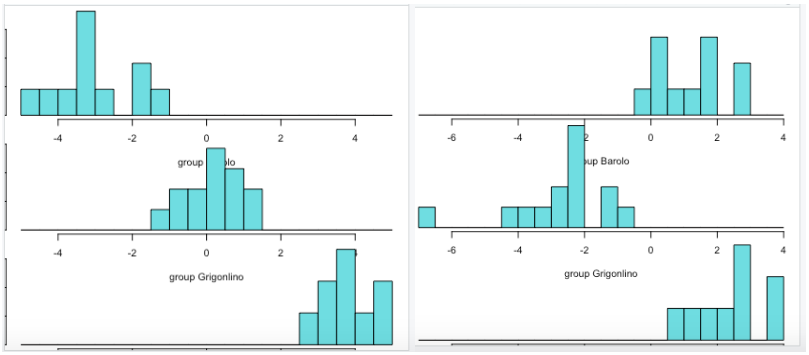
Classification tree accuracy: 88%	Random forest accuracy: 100%
Classification tree error: 12%	Random forest error: 0%

LDA model:

Next we will construct LDA model. After building LDA model on the training set the summary looks like this:



Here we see prior probabilities for our LDA model then we see LD1 and LD2 values. Here LD1 is 68% and LD2 is 32%.

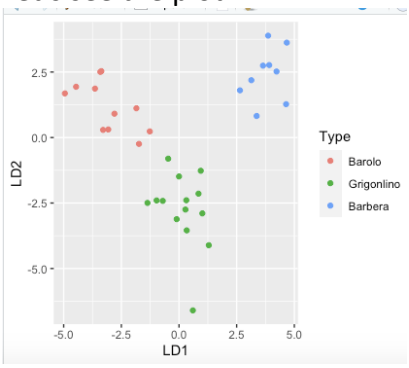


Left: LD1 Right: LD2

As we can see in LD2 model all the class groups are well separated and in LD1 model there is very slight overlapping on the first two groups. Now let's predict this model using testing set. After prediction we got the accuracy 100%. Here is the confusion matrix:

	Barolo	Grigonlino	Barbera
Barolo	11	0	0
Grigonlino	0	14	0
Barbera	0	0	9

Here we see LDA model giving 100% classification error by classifying all the classes correctly. Let's see the plot.



As we can see all three classes are separated correctly.

Comparing the performance:

Classification tree accuracy- 88%	Random forest accuracy- 100%	LDA model accuracy- 100%
Classification tree error- 12%	Random forest error- 0%	LDA model error- 0%

Here we see Random forest and LDA both doing best on our wine data set. I will choose LDA if the dataset is small. But if we get larger data sets I will choose Random forest as it ensembles so gives higher accuracy.

Problem#4

Exploring the dataset, reducing the size of the dataset and splitting:

In the forest 'covertype' data set has 581,012 observations with 55 columns. There are no missing values in the data set. Next we like to see how many observations in each forest cover type has:

```
> forest_classes

     1      2      3      4      5      6      7
211840 283301  35754   2747   9493  17367  20510
>
```

1 -- Spruce/Fir
2 -- Lodgepole Pine
3 -- Ponderosa Pine
4 -- Cottonwood/Willow
5 -- Aspen
6 -- Douglas-fir
7 – Krummholz

As this data set is so big and unable to run in my computer, I would like to reduce the data size. So, what I would like to do is- choose the Rawah wilderness area. And we will select all the rows of the data set which has Rawah wildernes area -1(presence) and all the other columns. After doing subset of the dataset we got 260796 rows with 55 columns. Next we would like to split our dataset into 75% for training and 25% for testing. And we would like remove feature from 11 to 54 columns. '55' column forest cover type which we will predict. After doing these we have 195598 rows with 11 columns for the training data and for the testing data we have 65198 rows with 11 columns. Next we will implement random forest on our training data and predict using test data to see how it's doing on the model.

Random forest model:

After using default random forest method on training set this is the result:

```
Call:
 randomForest(formula = V55 ~ ., data = forest_training, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 3

        OOB estimate of  error rate: 3.64%
Confusion matrix:
      1      2    5    7 class.error
1 75409   3839   12   83  0.04958219
2  2135 107320  125   13  0.02074038
5    23    473 2361    0  0.17360868
7   377     40    0 3388  0.10959264
```

Here we see out of bag(OOB) error rate is 3.64% and we see this data set predicted 4 types of forest covertype. As we see class error for 1 and 2 is less than 5 and 7. Those two forests cover type has higher correct prediction than 5 and 7 forest cover type. Next we would like to predict using training data. We got 100% accuracy for the training data set prediction. Let's see predict on test data to see how random forest model doing on the unseen data. And the testing set accuracy we got 96% which is very good and error is 0.037. Next see the confusion matrix:

```
forest_predict_cover     1     2     5     7
                   1 25059   736     4   133
                   2  1290 35822   176    15
                   5     6    38   744     0
                   7    19     8     0  1148
> |
```

Here we see 2425 data points were misclassified out of 65198 data points. And total correctly predicted data points- 62,773.

Next we would like to tuning the parameter for the random forest. We would like to get ntree = 500 which is default value and mtry = 5. Now let's see how the random forest is doing on the training set.

```
No. of variables tried at each split: 5

        OOB estimate of  error rate: 3.03%
Confusion matrix:
      1      2    5    7 class.error
1 76181   3059   11   92  0.03985229
2  1886 107561  131   15  0.01854133
5    17    367 2473    0  0.13440672
7   298     41    0 3466  0.08909330
> |
```

Here with mtry = 5 we see OOB out of bag error estimate error rate is 3.03%, which is less than our default random forest. Next we will predict the model using test set to see how it's doing on test data. We found accuracy of the test data is 97% which is tiny improvement than our non- tuning random forest one and error rate is- 0.031. Here is the confusion matrix to see how the model did on classifying forest cover type.

```
forest_predict_cover_new     1     2     5     7
                         1 25325   671     3    94
                         2  1017 35882   146    15
                         5     9    41   775     0
                         7    23    10     0  1187
> |
```

Here our total misclassification points is- 2029 out of 65198 data points. Total correctly classified data is- 63,196 data points. Here we see most misclassification on 1017 data points. It should be in lodgepole forest types but random forest predicted as spruce forest cover type. But overall random forest doing really good on the forest cover type data set.