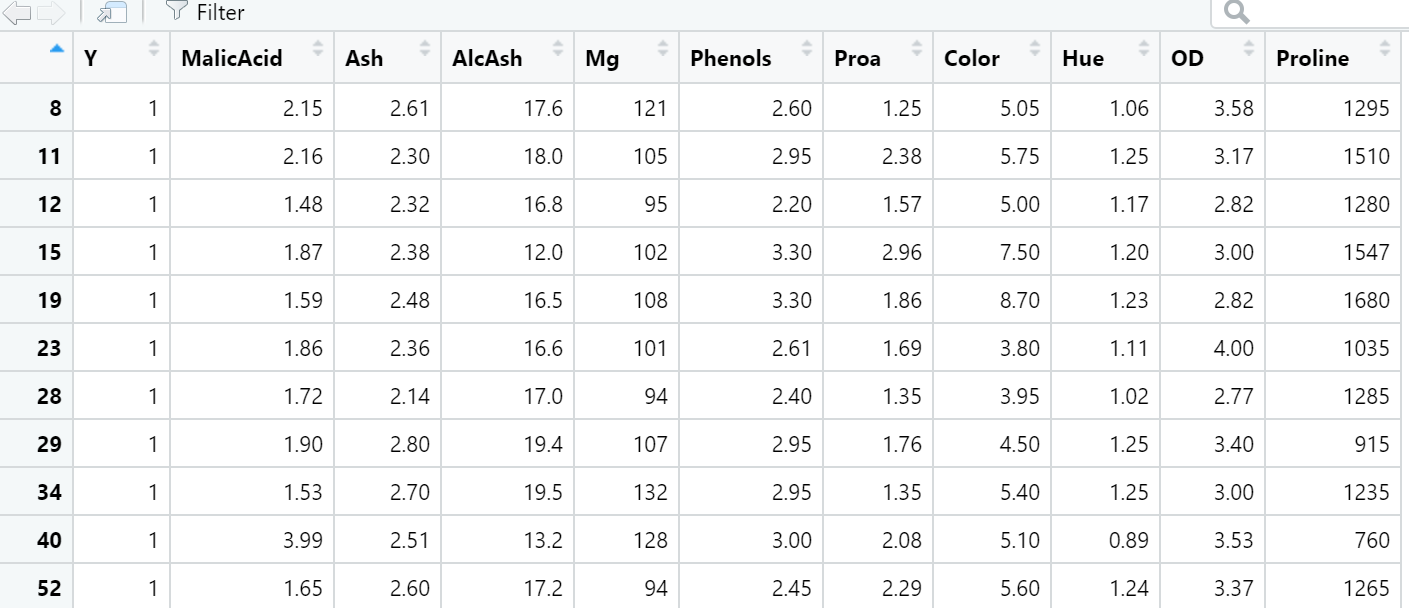
a)

Process:

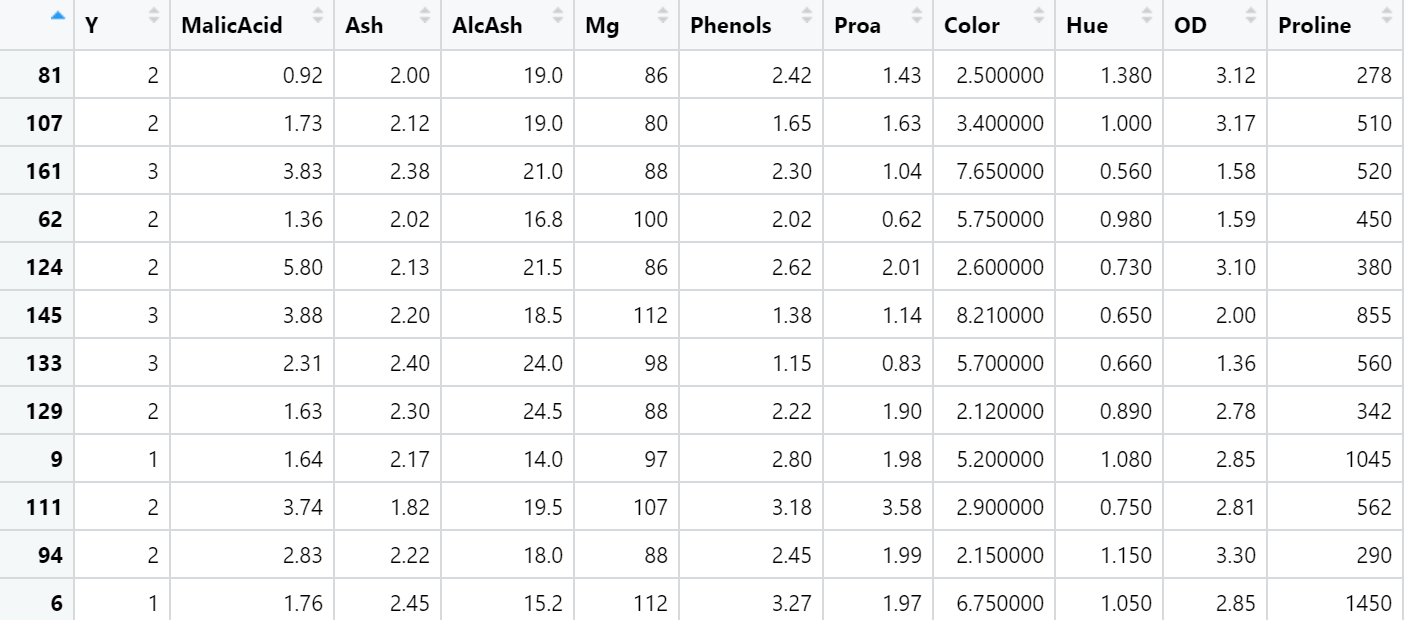
1. **Load data**
2. **Split the Train and Test data**: I used the method ‘sample’ to split the data into the train and test data.
3. **Made a pruned tree model:** I made a tree model and show the cp value. With the cp value, I can findout the tree size that maximizes the model’s explanation. Also, I compared the pruned tree with the full tree.
4. **Made a Random forest model:** I made a random forest model, and check the important features in the model. To check the performance of the model, I made a confusion matrix with some performance values.

Outputs:

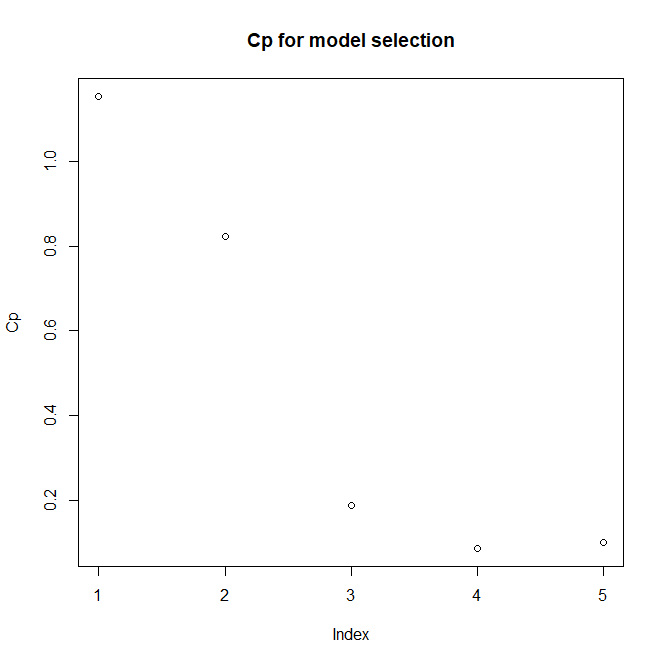
**Test data**



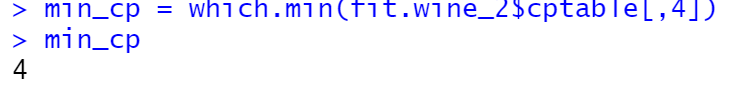
**Train data**



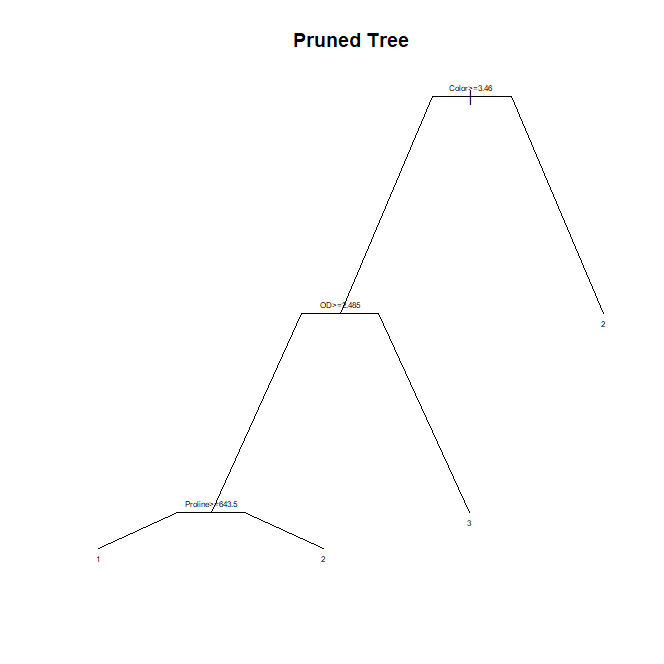
**Decision Tree cp for the tree model**



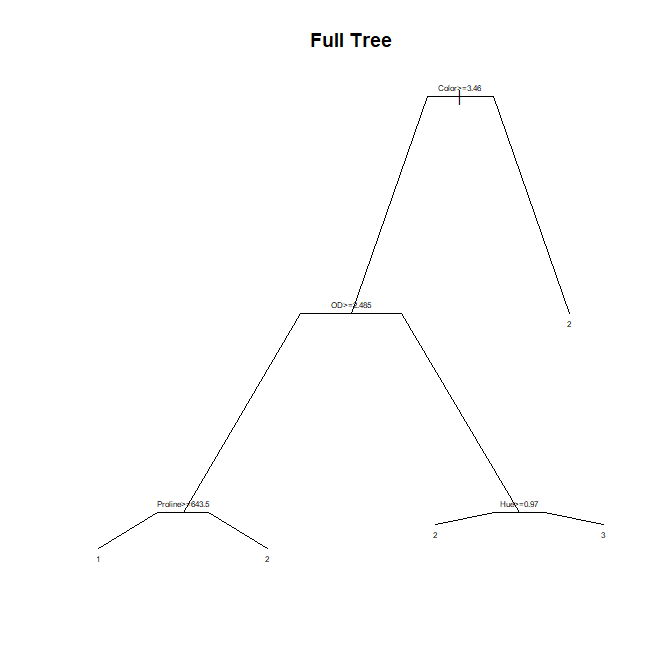
**Min CP value for the tree model.**



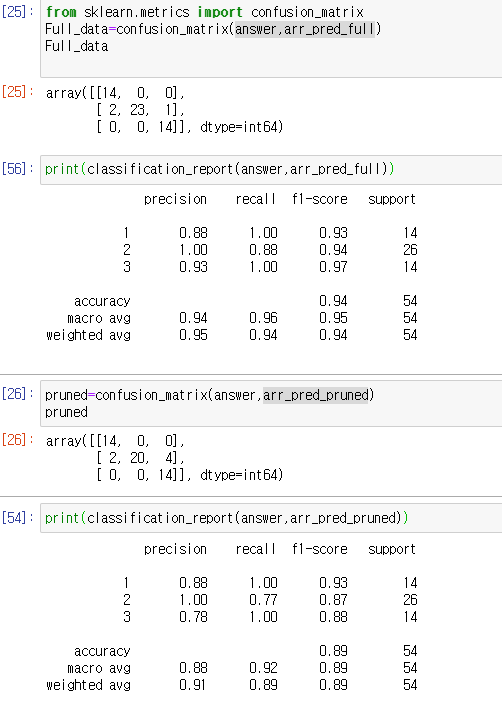
**Pruned decision tree shape (MIN CP values)**



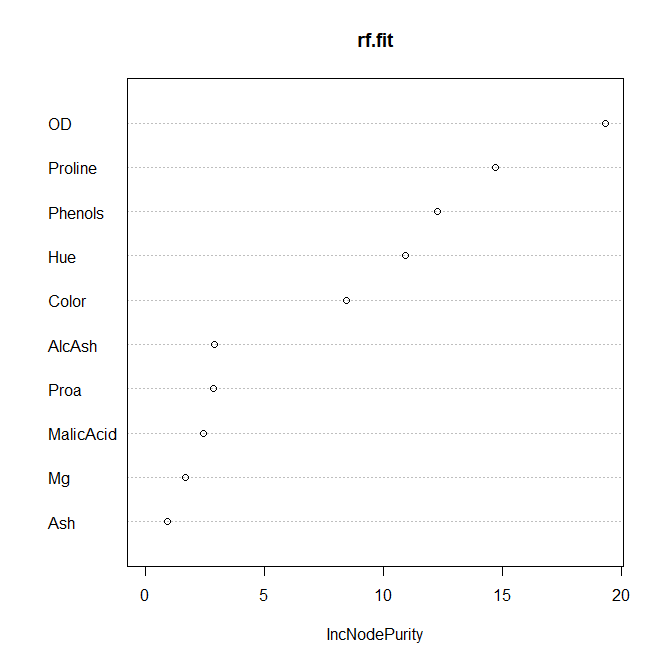
**Full decision tree shape**

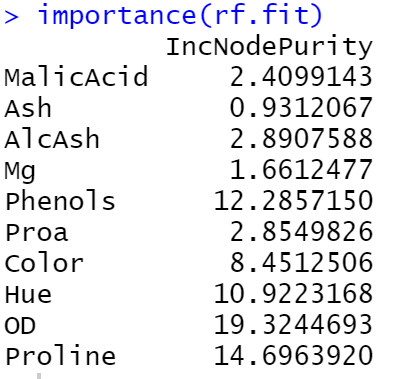


**Confusion matrix and perfomance for the Full tree and the pruned tree**

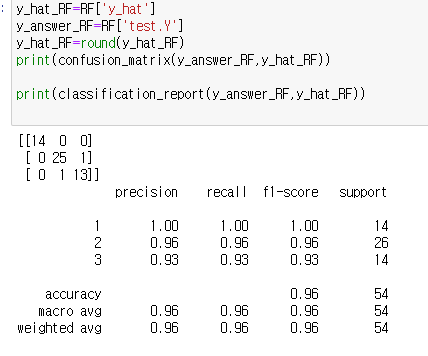


**Important feature from the Random Forest model**





**Confusion matrix for the Random Forest model and it’s performance.**



Discussion:

***Performance***

The random forest model: 96% accuracy

The pruned tree model: 89% accuracy

The full tree model: 94% accuracy

Let’s look at the performance of the tree model. The random forest model got the highest accuracy for the test data prediction. The pruned tree model got the lowest accuracy for the data prediction. All Three models show high accuracy in the test data classification.

***Partitioning***

When I look at the full tree model shape, it is possible to say that there are some useful variables in the data for making a decision boundary to classify the data. This is because the full tree model is not complicated although there are lots of variables in the train data.

The tree model classifies the data so that the same class of the data supposed to be in the same area. The tree model tries to reduce the impurity of each area. In other words, a low impurity area consists of objects in the same class. To calculate the impurity, the model uses some indicators such as ‘Entropy’, ‘Gini’, and ‘misclassification error’. The tree model recursively divides the data with criteria that minimize the area’s impurity. However, in our full tree data model, there are only 5 terminal nodes which imply 4 criteria (the number of decision nodes for the full data) are enough to minimize the impurity of each classifying area. Considering there are 10 variables in the training data set, this is an interesting point.

According to the ‘Important feature from the Random Forest model’, about 5 features have much more high importance values than the other features.

***Pruning***

In general, a full tree model is overfitted in the train data. To generalize the model, we need to prune the tree model. To find out the fitness pruned tree, we used the minimum cp value in the model which maximizes the model’s explanation. However, in our case, the difference cp value between the full tree model and pruned tree model is trivial. Futhermore the full tree looks so simple that pruning in this model enables to reduce information for the classification instead of generalizing the model. This can be one of the reasons that the full tree has better accuracy than the prune tree in the test prediction.

***Ensemble***

The random forest model made lots of tree models by sampling among the train data to find several good tree models. This sampling is based on contingency and this will give more chance for the ensemble model to select better tree models. After finding several good tree models, the random forest model conbine each tree model’s results(voting) and get the best result the prediction. This process enables for the random forest model to get 96% of accuracy in the test prediction.

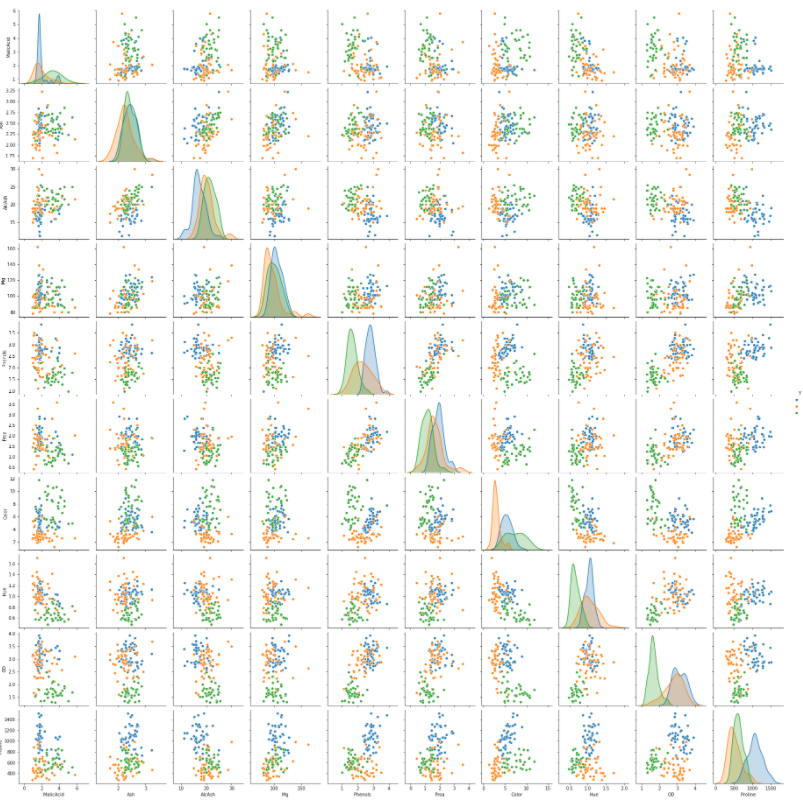
b)

Process:

1. **Load data:** Same with the part (a)
2. **Split the Train and Test data**: Same with the part (a)
3. **Made a Random forest model:** I plotted pair polt and create covariance matrix of features group by each Y(Barolo, Grignolino, Barbera). After check distributions, I made an LDA model. To check the performance of the model, I made a confusion matrix with some performance values.

Outputs:

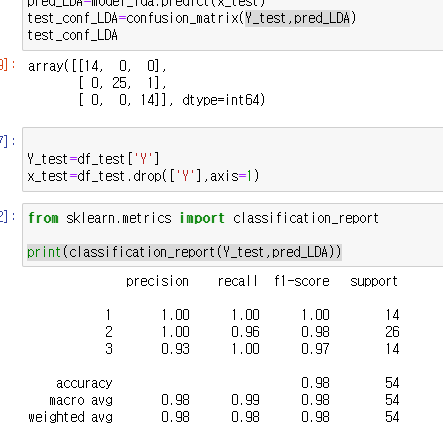
**Pair plot**



**Covariance matrix by Barolo, Grignolino, Barbera**



**Confusion matrix for the LDA model.**



Consideration:

According to the confusion matrix for the LDA model, the LDA model got 98% accuracy which is the best performance among the models. I tried to guess this result.

**Linear decision boundary**

The LDA model use a linear decision boundary for the classification. It is able to say that the simple tree model tends to be more linear decision boundary than the complex tree model. Especially our full and pruned tree models from the previous question are simple. There are only a few decision tree nodes in our tree model. This implies that there is a posibility that the decision boundary looks linear. However, we need a more direct reason to understand this result.

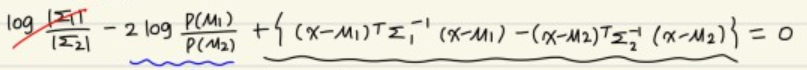
Why the LDA model used Linear decision boundary for the classification?

* **The two group of data have same distribution (Σ1 = Σ2)**

(assumption for the LDA model)

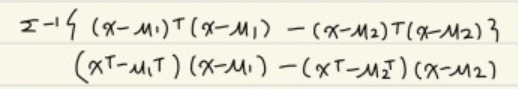
Σ1 = Σ2 = pooled variance = ((n1-1) Σ1 +(n2-1) Σ2)/ (n1 + n2 -2) = Σ

Apply it into discriminant function

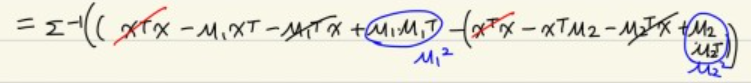


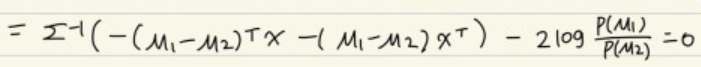
Blue line is a constant.

Rearrange the equation for the Black line with Σ-1 (notice that Σ1 = Σ**2**)



Multiply each equation and make it simpler.





The equation becomes like the picture above.

This arranged equation became the linear function form as like **AX + B = 0.**

This is because the two groups have in pooled variance **(Σ1 = Σ2)**

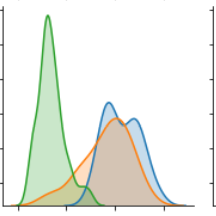
In this case we can see the division boundary became linear.

**Coveriance and distribution for some important variables by Y (barolo, grignolino, barbera).**

Based on the random forest model we found some important features. I supposed that the feature gives information to the LDA model as well.

**BLUE: barolo ORANGE: grignolino GREEN: barbera**

**OD**

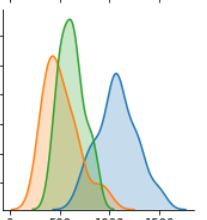


**BLUE: barolo ORANGE: grignolino GREEN: barbera**

**Cov (0.13, 0.24, 0.07) => acceptable.**

**There is a huge distance between the distribution center of ‘barbera’ and the others’ which is a good feature for the LDA model.**

**Proline**

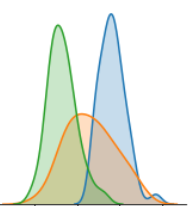


**BLUE: barolo ORANGE: grignolino GREEN: barbera**

**Cov (39000, 30000, 14800) => acceptable.**

**There is huge distance between distribution center of ‘barbera’ and ‘barolo’ which is good feature for the LDA model and between ‘barbera and ‘grignolino’ as well.**

**Phenols**

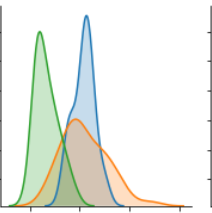


**BLUE: barolo ORANGE: grignolino GREEN: barbera**

**Cov (0.1, 0.3, 0.09) => acceptable.**

**There is a huge distance between the distribution center of ‘barbera’ and ‘barolo’ which is a good feature for the LDA model.**

**Hue**



**BLUE: barolo ORANGE: grignolino GREEN: barbera**

**Cov(0.01, 0.05, 0.01)**

**There is a huge distance between the distribution center of ‘barbera’ and ‘barolo’ which is a good feature for the LDA model.**

For this reason, the LDA shows high accuracy in the test prediction.

Reference:

#http://lijiancheng0614.github.io/scikit-learn/modules/generated/sklearn.decomposition.LatentDirichletAllocation.html#sklearn.decomposition.LatentDirichletAllocation.fit\_transform

#https://blog.naver.com/jaehong7719/221926671654

#https://blog.naver.com/powerparan/221867153428

#\*\*\*https://blog.naver.com/sanghan1990/221126257295\*\*\*