Data Mining: Assignment 2

==========================

Krishna Mahajan, 0003572903

Q1

Module:q1/q1.rmd

(a) Calculate the average value and standard deviation for each of the four features. (soln)

MEAN	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
	5.8433	3.057	3.758000	1.99333

SD	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
	0.82	0.43	1.76	1.199

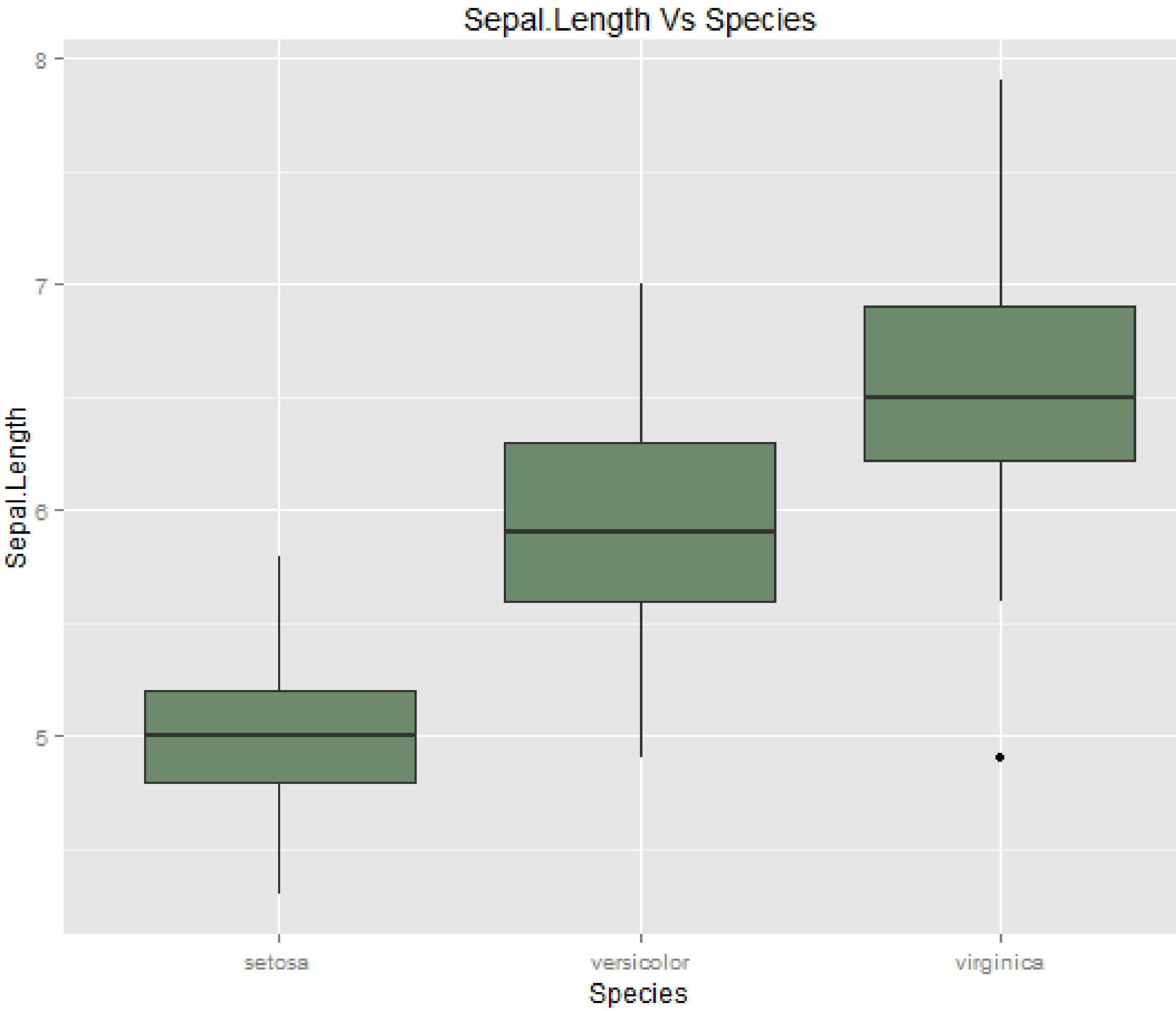
(b) Repeat the previous step but separately for each type of flower (soln)

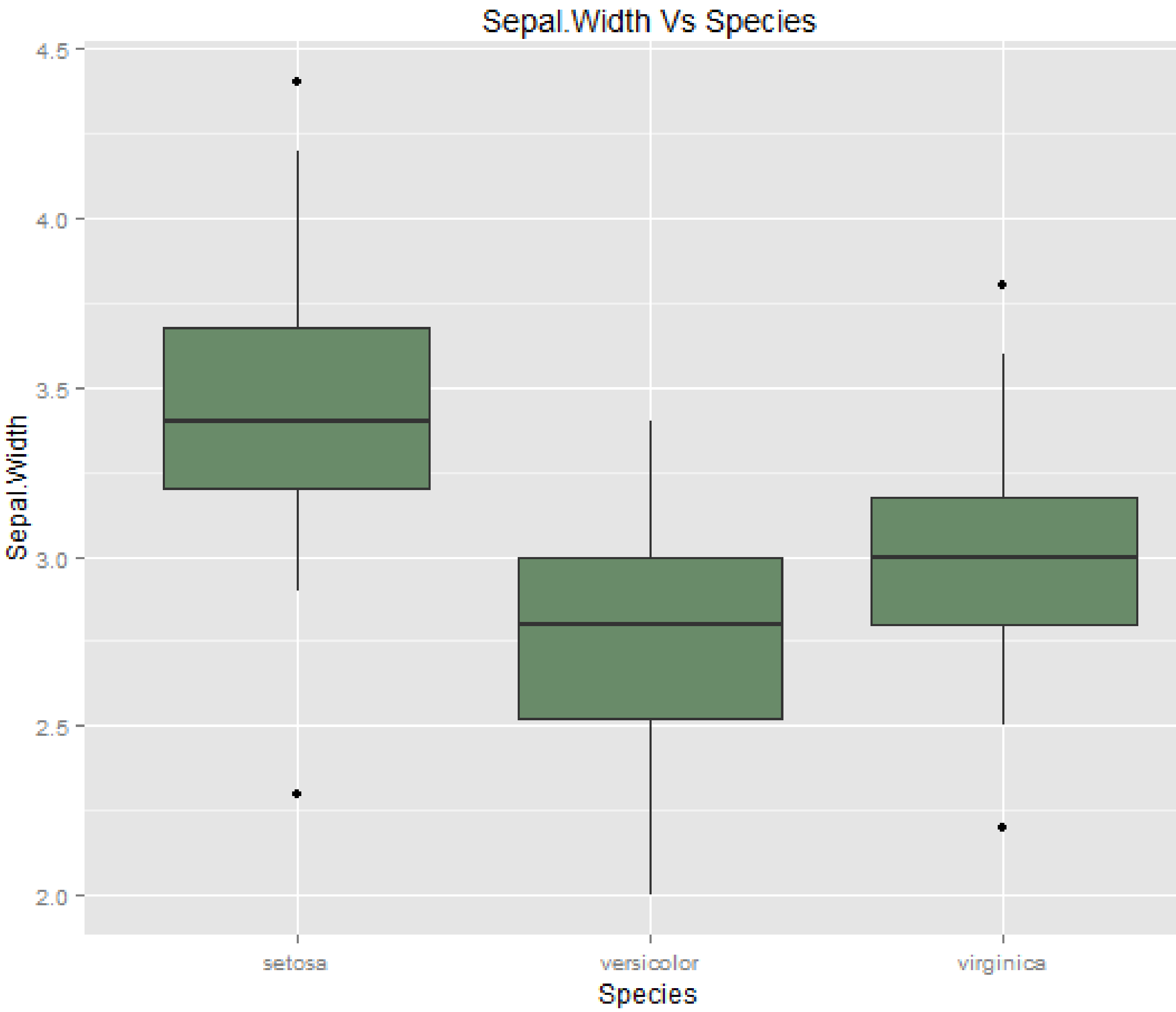
MEAN	setosa	versicolor	virginica
Sepal_Length	5.006	5.936	6.588
Sepal_Width	3.428	2.770	2.974
Petal_Length	1.462	4.260	5.552
Petal_Width	0.246	1.326	2.026

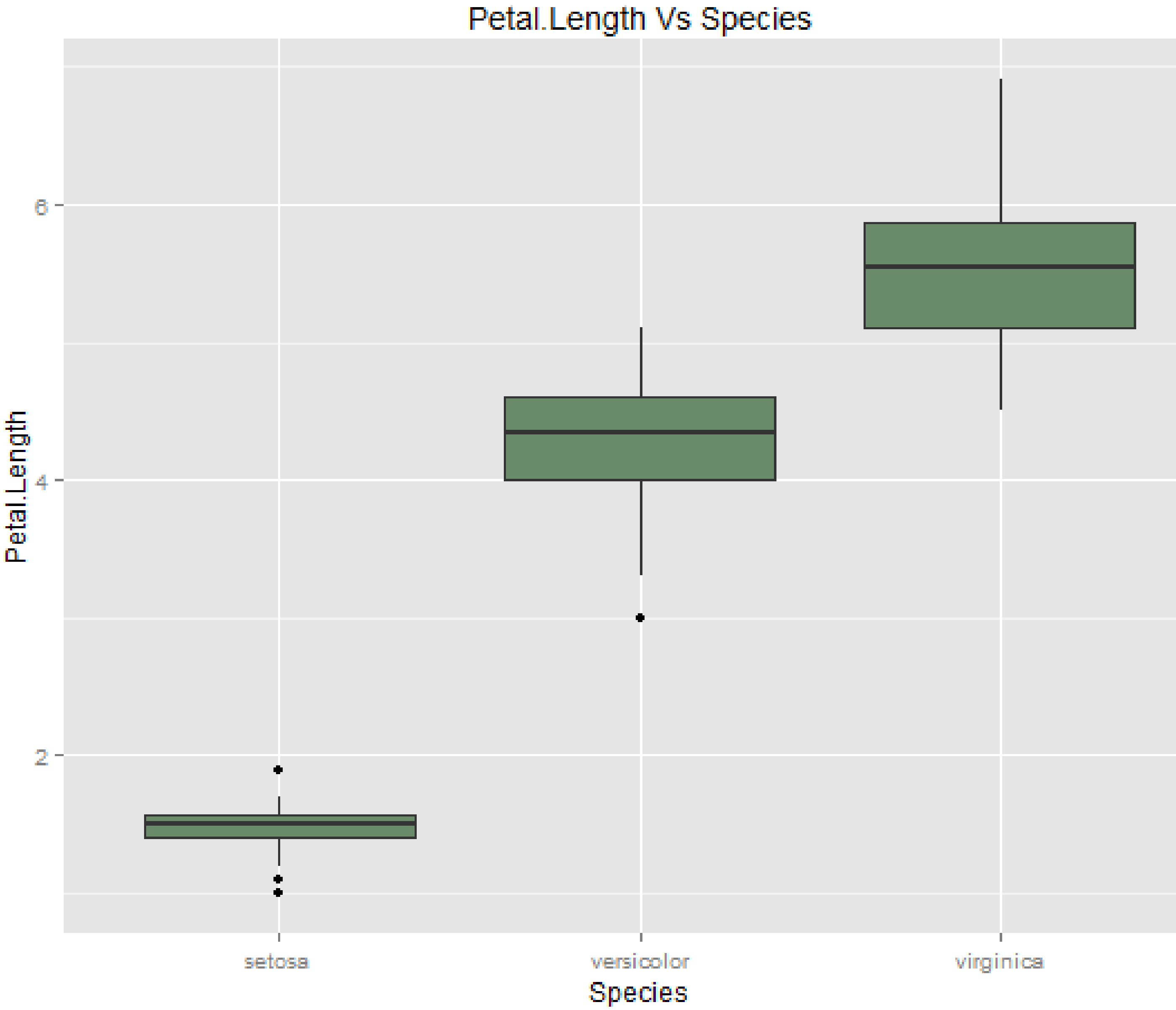
SD	setosa	versicolor	virginica
Sepal_Length	0.3524	0.5161	0.63
Sepal_Width	3.428	2.770	2.974
Petal_Length	0.1736	0.469	0.551
Petal_Width	0.105	0.197	0.274

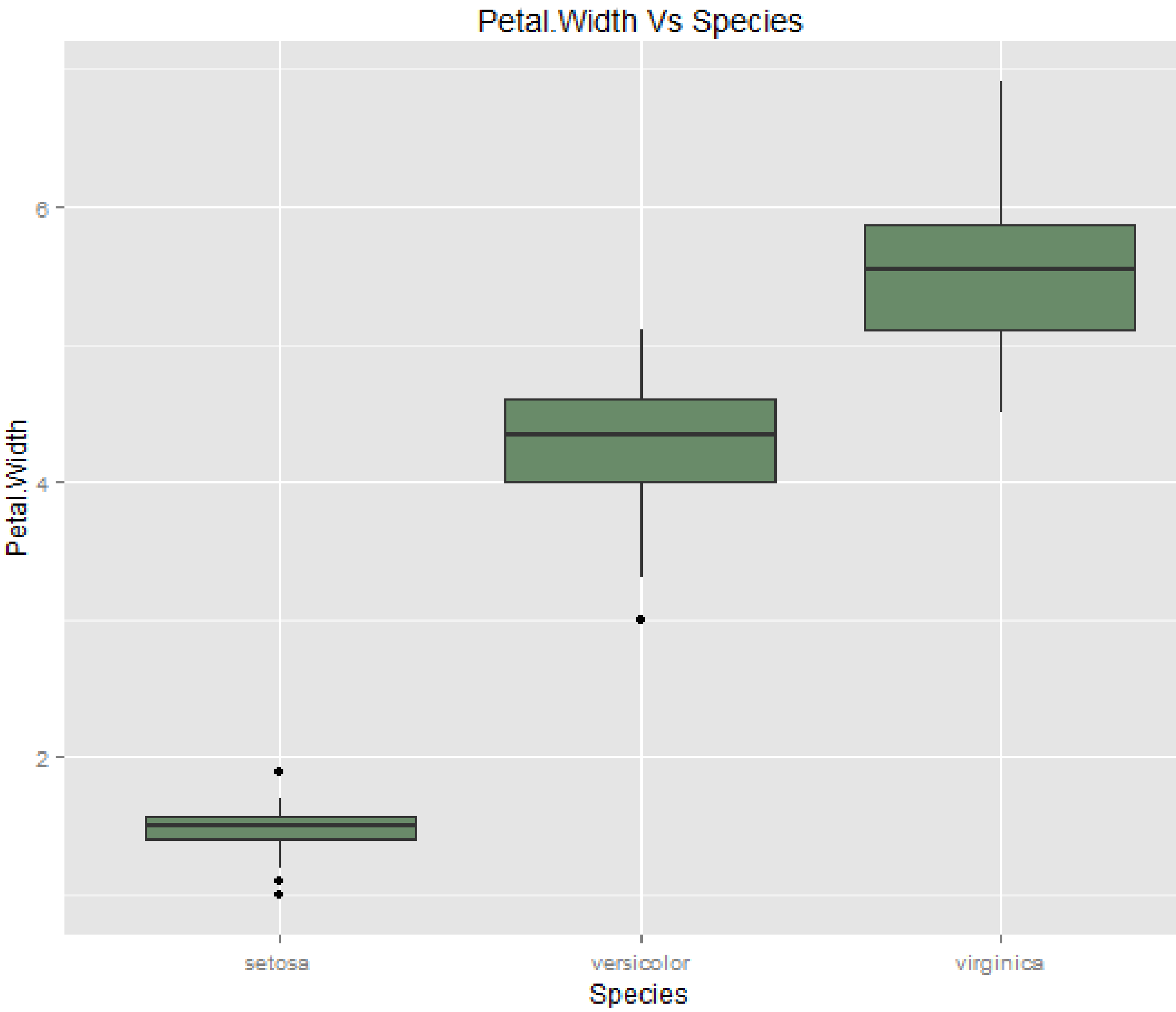
(c) Draw four box plots, one for each feature, such that each figure shows three boxes, one for each type of flower. Properly label your figures and axes in all box plots. Make sure that the box plots look professional and appear in high resolution. Experiment with thickness of lines, font styles/sizes,etc. and describe what you tried and what looked the most professional

(soln)


Sepal.Length Vs Species
8
7
6
5
Sepal.Length
setosa
versicolor
virginica
Species


Sepal.Width Vs Species
4.5
4.0
3.5
3.0
2.5
2.0
Sepal.Width
setosa
versicolor
virginica
Species


Petal.Length Vs Species
Petal.Length
6
4
2
setosa
versicolor
virginica
Species



Q2

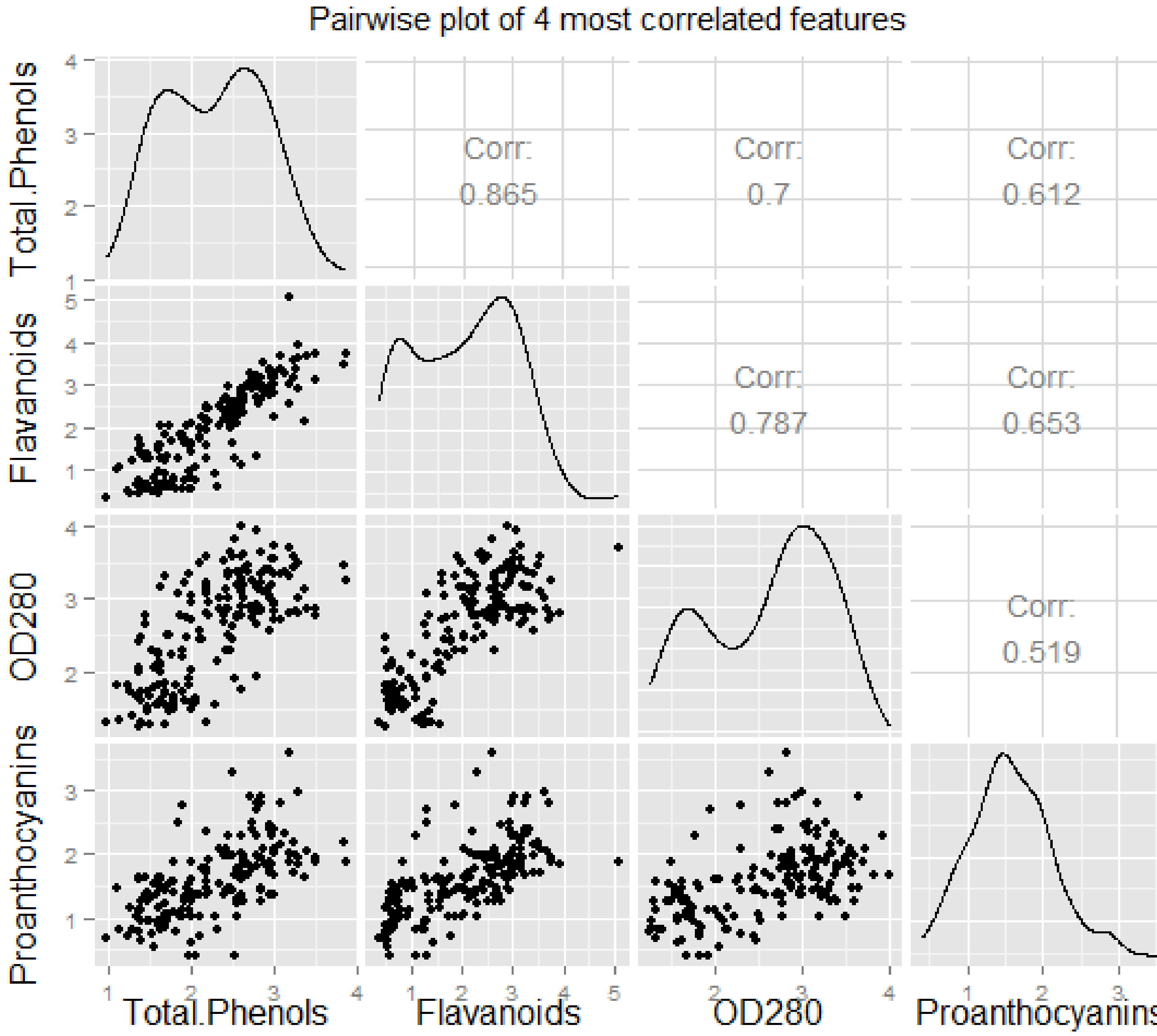
Module:q2/q2.rmd

(a) Provide pairwise scatter plots for four most correlated and four least correlated pairs of features, using Pearson's correlation coefficient. Label all axes in all your plots and select fonts of appropriate style and size. Experiment with different ways to plot these scatter plots and choose the one most visually appealing and most professionally looking

(soln)

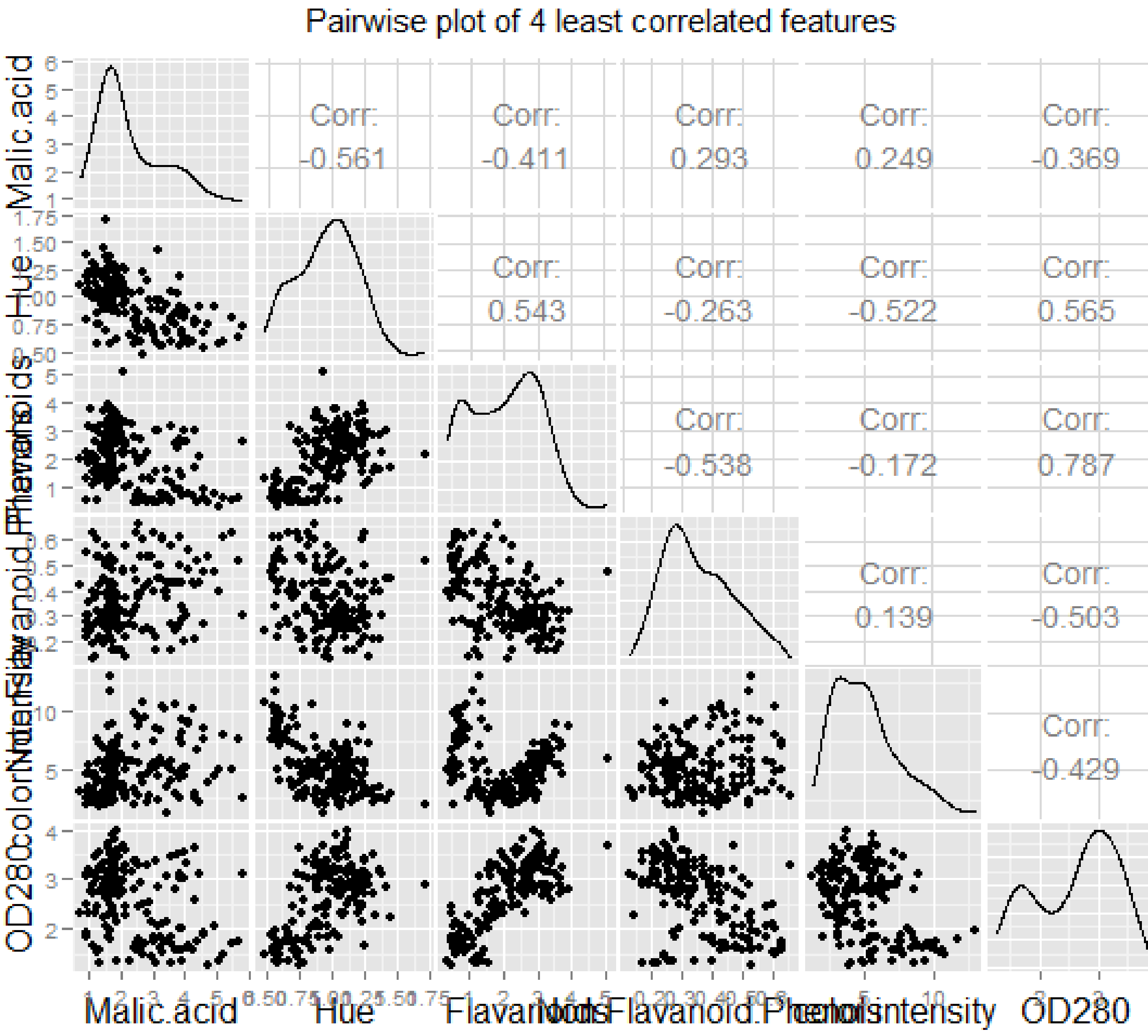
Four most correlated pairs are:

Feature1	Feature2	cor
Total.Phenols	Flavanoids	0.86
Flavanoids	OD280	0.78
Total.Phenols	OD280	0.69
Flavanoids	Proanthocyanis	0.65



Four least correlated pairs are:

Feature1	Feature2	cor
Malic.acid	Hue	-0.56
Flavanoids	Non Flavanoids Phenols	-0.53
color.intensity	Hue	-0.52
Non Flavanoids Phenols	OD280	-0.50



(b) Use Euclidean distance to find the closest example to every example available in the data set (exclude the class variable). Calculate the percentage of points whose closest neighbors have the same class label (for data set as a whole and also for each class).

(soln)
I used Following Algorithm to compute this ques:
1)For every observation i find out it closest Euclidean neighbour and checked if they have same class(Type of wine)
2) % of closest points with same class label on whole dataset=**Total pairs with same classes/size**(**data**)
3) % of closest points with particular class label C_i for class C_i= $\textbf{Total pairs with same class C_i /Total number of observation with class=C_i}$

Results:

whole_dataset	class 1	class2	class3
76.96%	88.1%	76.05%	64.58%

(c) Repeat the previous step but after the data set is normalized using first 0-1 normalization and then z-score normalization. Investigate the reasons for discrepancy and provide evidence to support every one of your claims. Provide the code you used for normalizing and visualizing the data.
(soln)
I used Following Algorithm to compute this ques:

1)First i normalized(Z score) all the columns of the wine dataset so that each column's mean=0 and standard deviation=1
2) Then i passed this normalized dataset to Algorithm developed in above in part.
3) Then i did 0-1 transfromation of the original dataset and passed this dataset to the alogrithm in above in part.

Results:

Z Score

whole_dataset	class 1	class2	class3
95.1%	100%	88.7%	100%

0-1 Normalization

whole_dataset	class 1	class2	class3
94.9%	100%	87.3%	100%

Normalization or standardization is very important techniques especially while dealing with parameters of different units and scale.For example in calculating Euclidean distance all the parameters should have same scale for a fair comparison between them. Both of these techniques have their drawbacks.For ex, if we have outliers in our original data then normalizing(0-1 transformation) 'll scale the normal data to very narrow interval.In practice most of the datasets have outliers. Also drawback using standardization(Z score) is that it don't bound the data as in 0-1 normalization.However, standardizing is the preferred method because it produces meaningful information about each data point, and where it falls within its normal distribution, plus provides a crude indicator of outliers (i.e.,

anything above or below a Z-Score of ±4.
Here since wine data have no outliers or missing value so both the techniques are giving almost same results.

Q4

Implementing classification trees and evaluating their accuracy

(a) Implement the greedy algorithm that learns a classification tree given a data set. Assume that all features are numerical and properly find the best threshold for each split. Use Gini and information gain, as specified by user, to decide on the best attribute to split in every step. Stop growing the tree when all examples in a node belong to the same class or the remaining examples contain identical features

(soln)
Here are following explanation of python Modules that i coded while implementing the decision tree ,in sequence.

Step 1) Reading raw data:

(1) Functionality of this module is to open the raw data file containing all the observations with final target variable(Class Variable).The module read all the data and do the necessarily cleaning (such as converting numerical attributed which are loaded as strings and convert them back to numerical type and adjusting end of line of character).
(2) Module 'll also add header to the raw data and colnames need to be passed as an input.
(3) Finally after all the cleaning ,Module convert the loaded the dataset into python list of list format(Data Frame) and dump this list in pickle format so that it can be later used readily while building the decision tree.
Module:readdata.py

Step 2) Gini Measure to split dataset:

I have used giniimpurity instead of entropy as asked to select the criteria to split the dataset. Giniimpurity is simply the expected error rate of assiging wrong class to a random observation. I have testing dataset with all three impurity measures(gini,information gain & variance)
Module:impurity.py

Step 3) Building the Tree:

(1) I have implemented decision tree in recursive manner.
Here are brief steps of the Algorithm.
1. The Algorithm iterate over all the features in data and within each feature find the best split among all the possible values of that feature such that information gain is maximum if the data is splitted on that particular feature's particular value.
2.So now all the observations are divided either into True Branch(which are greater or equal to feature value) or False Branch(which are less than feature value)

3.Now recursively the same Algorithm repeats for each branch until there's only one class variable in that branch.(stopping criteria)

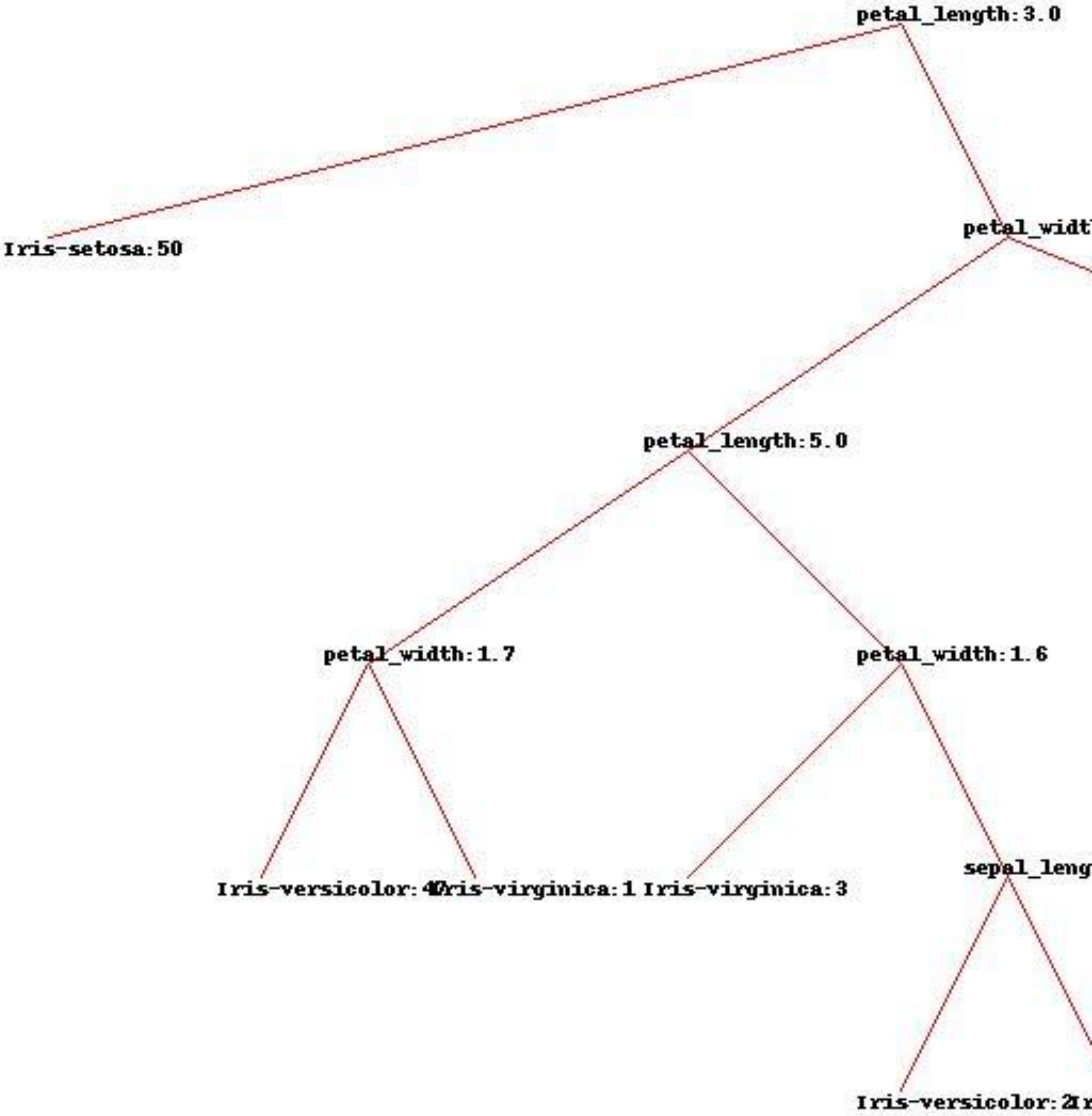
(2) At the completion ,the algorithm returns the root decision node which was initial node at which dataset was splitted first.If we the traverse through the root decision node we can reach all other decision node and the final leafs(classes).
please note left branch is falseBranch and rightBranch is TrueBranch.
Module:buildtree.py

Step 4)Printing the tree:

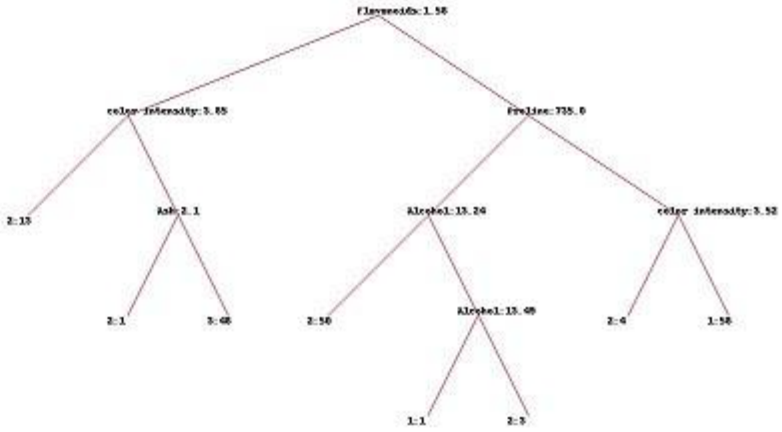
(1) I have used python's PIL library to print the decision tree as per colnames to visualise all the decision node and their corresponding branches.
Module:drawtree.py

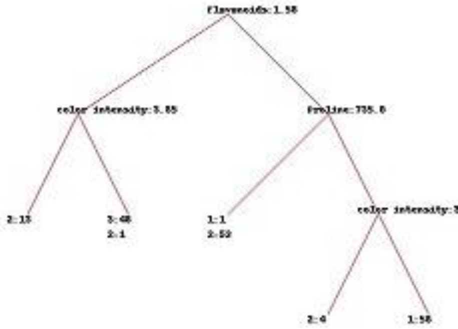
Here are some examples of final decision tree model when i ran them on the following famous datasets:

Iris

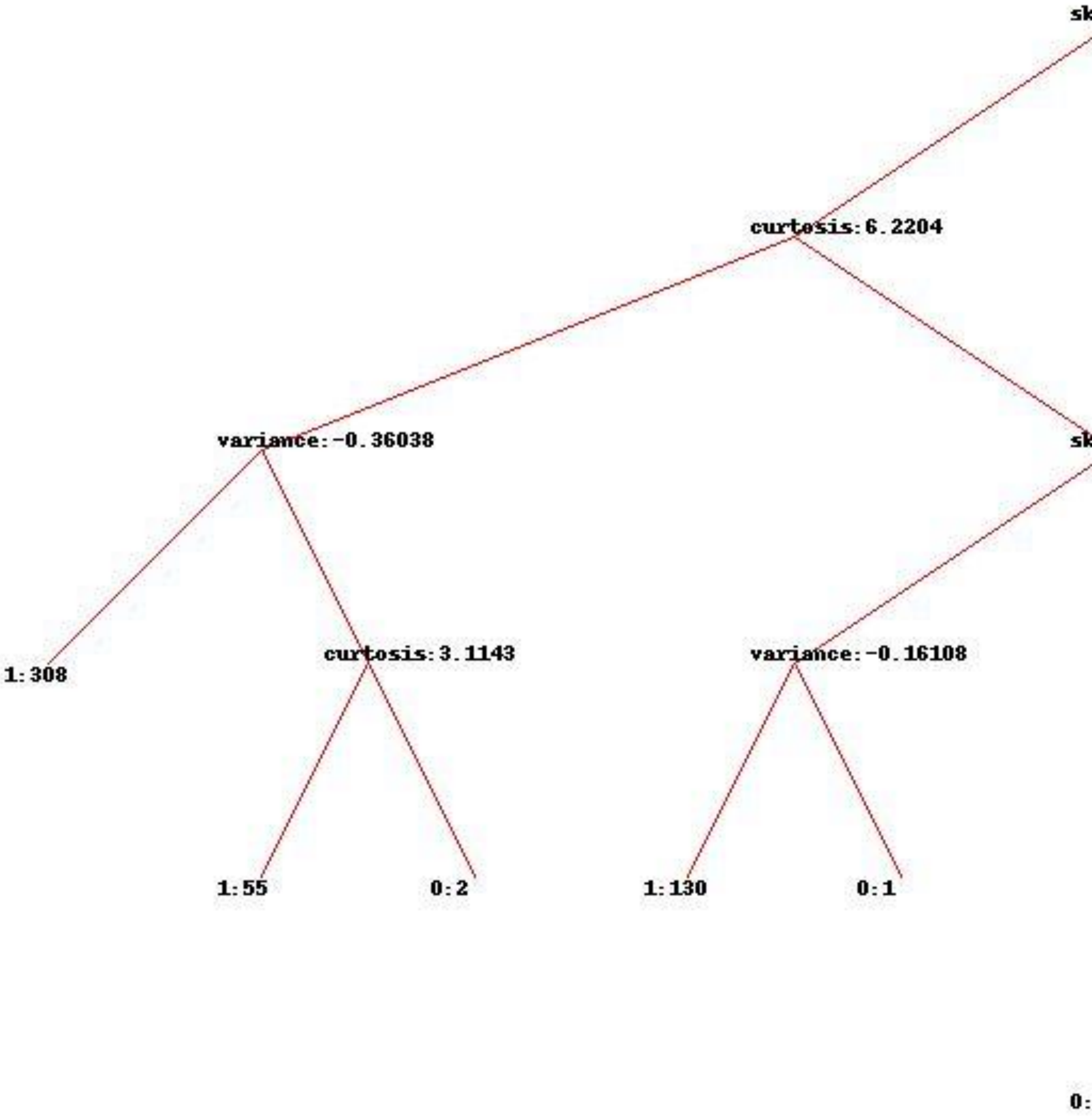

petal_length:3.0
Iris-setosa:50
petal_widt
petal_length:5.0
petal_width:1.7
petal_width:1.6
Iris-versicolor:47
Iris-virginica:1
Iris-virginica:3
sepal_leng
Iris-versicolor:2
Ir

Wine


flavonoids:1.58
color intensity:3.85
Proline:755.0
2:13
Ash:2.1
Alcohol:13.24
color intensity:3.52
2:1
3:48
2:50
Alcohol:13.49
2:4
1:58
1:1
2:3


flavonoids:1.58
color intensity:3.85
Proline:755.0
2:13
3:48
2:1
1:1
2:52
color intensity:3
2:4
1:58

BankNote


sk
curtosis:6.2204
variance:-0.36038
sk
1:308
curtosis:3.1143
variance:-0.16108
1:55
0:2
1:130
0:1
0:

(b) Implement 10-fold cross-validation to evaluate the accuracy of your algorithm on 10 different data sets from the UCI Machine Learning Repository. Select only those data sets where all features are numerical.

(soln)
Here are the following module which were used to implement 10 Fold cross validation functionality to check accuracy for the decision tree implemented .

Step 1) Build a classify Algorithm:

(1) Built a classifier function which predicts the class of any test observation based on the rules of decision node.It turned out to be simple recursive function which starts at root decision node and traves along the branches till the leaf node c is reached which is its final class.
This classifier has also the functionality to handle NA cases in which it simply traverse both the branches(True Branch+False Branch) and predict the final class as weigthed average of possible classes predicted in each branch.
Module:classifier.py

Step 2) Create stratified 10 Folds:

(1) This modules splits the dataset in 10 training,testing folds while preserving the original distribution of classes in the original data at each fold.
I used python's scikit library's stratified K Fold to implement this module.
Module:createfolds.py

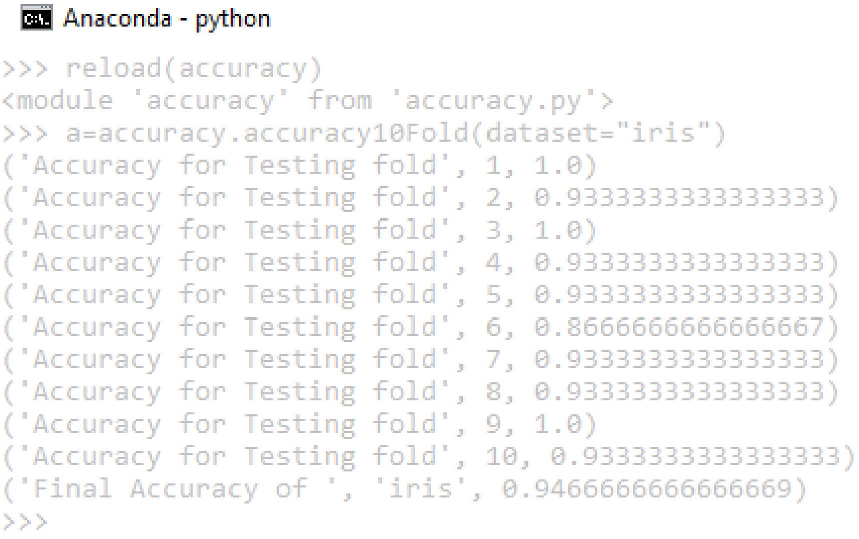
Step 3) Calculate Final efficiency:

(1) This module builds a decision tree on every folds training data and then predicts final classes of each observation in testing data and then calculate the efficiency.
(2) The Algorithm iterate 10 times for each folds and at the end calculate the final efficiency which is mean efficiency of all 10 folds.
Module:accuracy.py

$$\textbf{Efficiency} = (\textbf{observation predicted correctly in Test Set})/(\textbf{Total observation in test set})$$

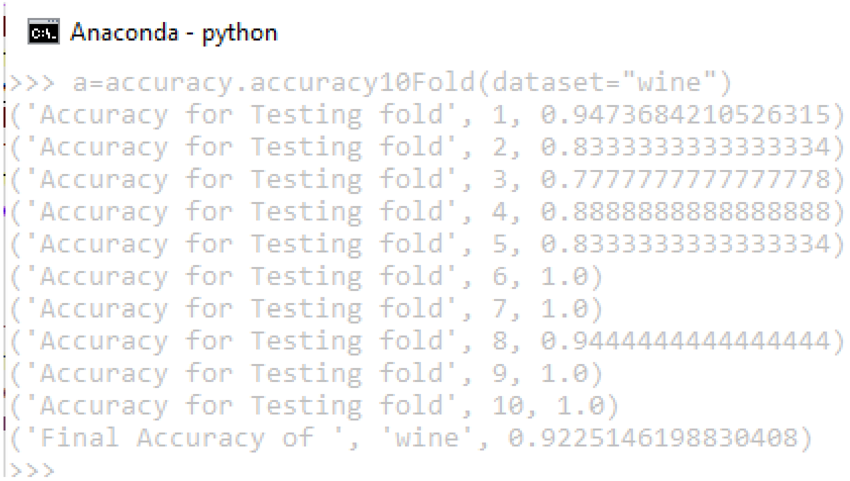
Here are the results when i tested Decision Tree on 10 different datasets with 10 fold cross validation .As obvious decision proved to be a bad classifier when the no of classes is large in a dataset.Please click on dataset to find information on it. All the dataset were obtained from UCI machine learning repository.

Iris



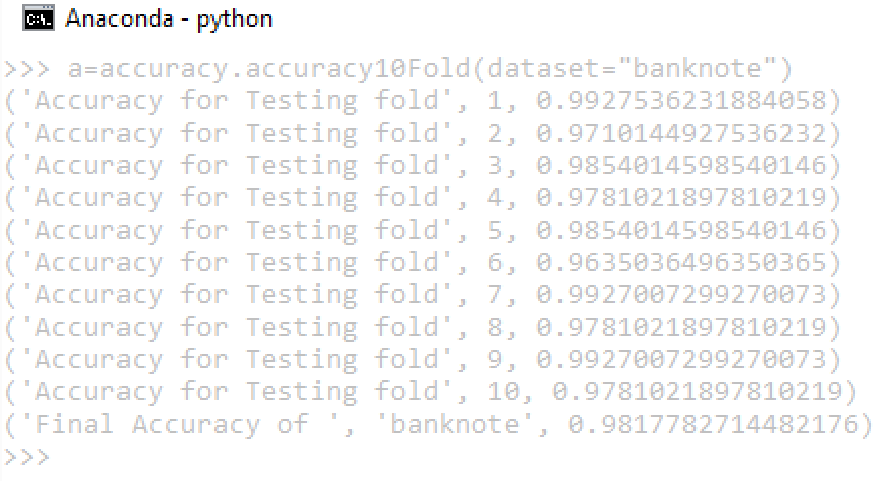
```
>>> reload(accuracy)
<module 'accuracy' from 'accuracy.py'>
>>> a=accuracy.accuracy10Fold(dataset="iris")
('Accuracy for Testing fold', 1, 1.0)
('Accuracy for Testing fold', 2, 0.9333333333333333)
('Accuracy for Testing fold', 3, 1.0)
('Accuracy for Testing fold', 4, 0.9333333333333333)
('Accuracy for Testing fold', 5, 0.9333333333333333)
('Accuracy for Testing fold', 6, 0.8666666666666667)
('Accuracy for Testing fold', 7, 0.9333333333333333)
('Accuracy for Testing fold', 8, 0.9333333333333333)
('Accuracy for Testing fold', 9, 1.0)
('Accuracy for Testing fold', 10, 0.9333333333333333)
('Final Accuracy of ', 'iris', 0.9466666666666669)
>>>
```

Wine



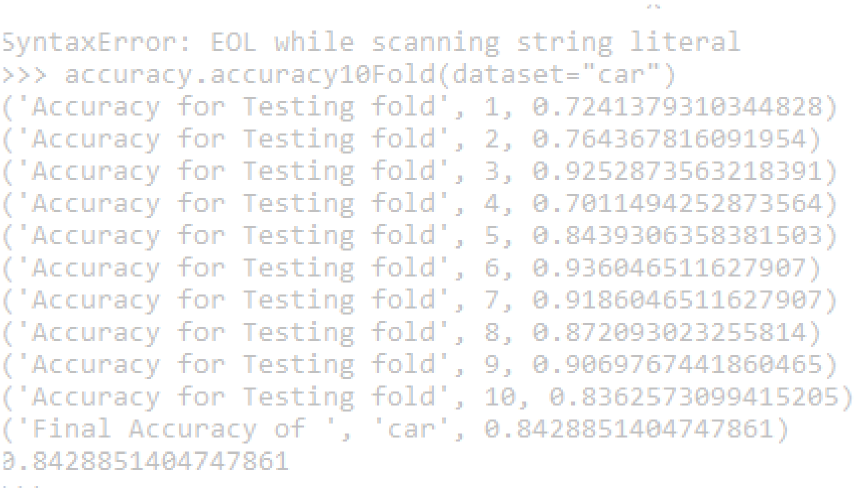
```
>>> a=accuracy.accuracy10Fold(dataset="wine")
('Accuracy for Testing fold', 1, 0.9473684210526315)
('Accuracy for Testing fold', 2, 0.8333333333333334)
('Accuracy for Testing fold', 3, 0.7777777777777778)
('Accuracy for Testing fold', 4, 0.8888888888888888)
('Accuracy for Testing fold', 5, 0.8333333333333334)
('Accuracy for Testing fold', 6, 1.0)
('Accuracy for Testing fold', 7, 1.0)
('Accuracy for Testing fold', 8, 0.9444444444444444)
('Accuracy for Testing fold', 9, 1.0)
('Accuracy for Testing fold', 10, 1.0)
('Final Accuracy of ', 'wine', 0.9225146198830408)
>>>
```

BankNote



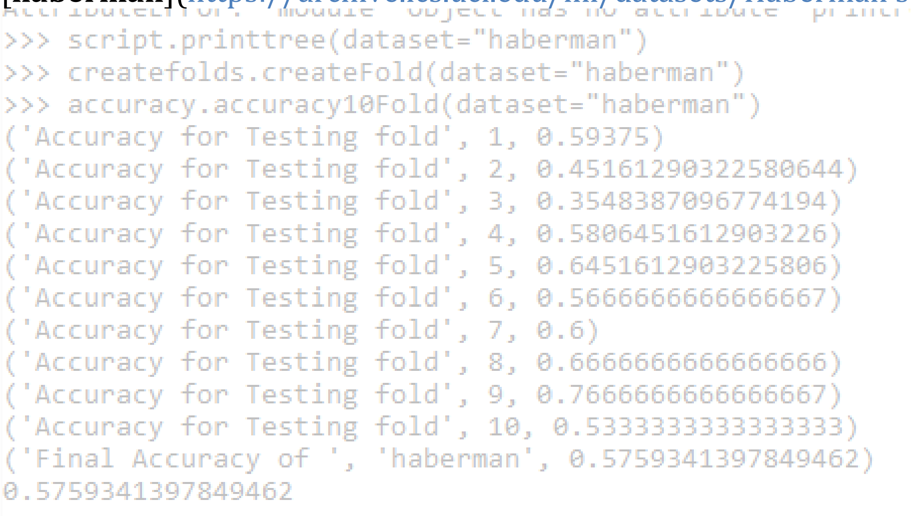
```
>>> a=accuracy.accuracy10Fold(dataset="banknote")
('Accuracy for Testing fold', 1, 0.9927536231884058)
('Accuracy for Testing fold', 2, 0.9710144927536232)
('Accuracy for Testing fold', 3, 0.9854014598540146)
('Accuracy for Testing fold', 4, 0.9781021897810219)
('Accuracy for Testing fold', 5, 0.9854014598540146)
('Accuracy for Testing fold', 6, 0.9635036496350365)
('Accuracy for Testing fold', 7, 0.9927007299270073)
('Accuracy for Testing fold', 8, 0.9781021897810219)
('Accuracy for Testing fold', 9, 0.9927007299270073)
('Accuracy for Testing fold', 10, 0.9781021897810219)
('Final Accuracy of ', 'banknote', 0.9817782714482176)
>>>
```

car



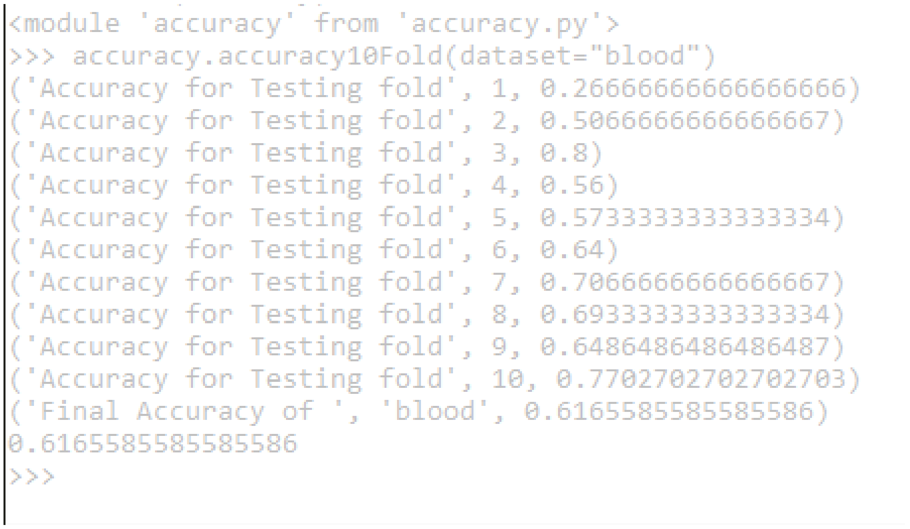
```
SyntaxError: EOL while scanning string literal
>>> accuracy.accuracy10Fold(dataset="car")
('Accuracy for Testing fold', 1, 0.7241379310344828)
('Accuracy for Testing fold', 2, 0.764367816091954)
('Accuracy for Testing fold', 3, 0.9252873563218391)
('Accuracy for Testing fold', 4, 0.7011494252873564)
('Accuracy for Testing fold', 5, 0.8439306358381503)
('Accuracy for Testing fold', 6, 0.936046511627907)
('Accuracy for Testing fold', 7, 0.9186046511627907)
('Accuracy for Testing fold', 8, 0.872093023255814)
('Accuracy for Testing fold', 9, 0.9069767441860465)
('Accuracy for Testing fold', 10, 0.8362573099415205)
('Final Accuracy of ', 'car', 0.8428851404747861)
0.8428851404747861
```

[**haberman**](https://archive.ics.uci.edu/ml/datasets/Haberman's+Survival)



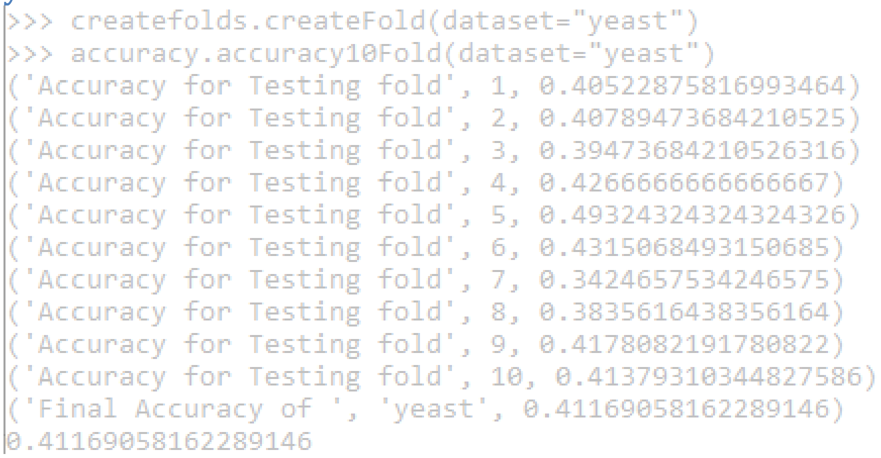
```
AttributeError: 'module' object has no attribute 'printt
>>> script.printtree(dataset="haberman")
>>> createfolds.createFold(dataset="haberman")
>>> accuracy.accuracy10Fold(dataset="haberman")
('Accuracy for Testing fold', 1, 0.59375)
('Accuracy for Testing fold', 2, 0.45161290322580644)
('Accuracy for Testing fold', 3, 0.3548387096774194)
('Accuracy for Testing fold', 4, 0.5806451612903226)
('Accuracy for Testing fold', 5, 0.6451612903225806)
('Accuracy for Testing fold', 6, 0.5666666666666667)
('Accuracy for Testing fold', 7, 0.6)
('Accuracy for Testing fold', 8, 0.6666666666666666)
('Accuracy for Testing fold', 9, 0.7666666666666667)
('Accuracy for Testing fold', 10, 0.5333333333333333)
('Final Accuracy of ', 'haberman', 0.5759341397849462)
0.5759341397849462
```

blood



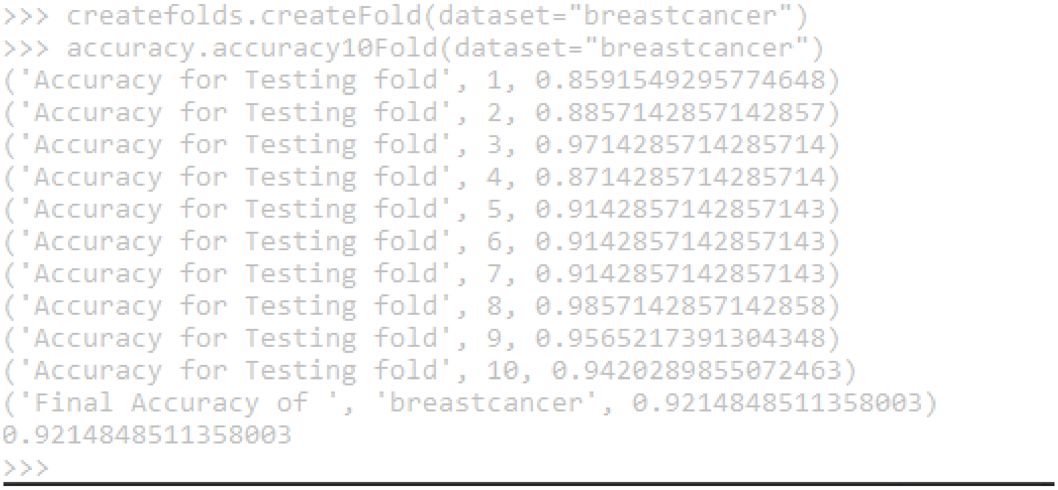
```
<module 'accuracy' from 'accuracy.py'>
>>> accuracy.accuracy10Fold(dataset="blood")
('Accuracy for Testing fold', 1, 0.26666666666666666)
('Accuracy for Testing fold', 2, 0.5066666666666667)
('Accuracy for Testing fold', 3, 0.8)
('Accuracy for Testing fold', 4, 0.56)
('Accuracy for Testing fold', 5, 0.5733333333333334)
('Accuracy for Testing fold', 6, 0.64)
('Accuracy for Testing fold', 7, 0.7066666666666667)
('Accuracy for Testing fold', 8, 0.6933333333333334)
('Accuracy for Testing fold', 9, 0.6486486486486487)
('Accuracy for Testing fold', 10, 0.7702702702702703)
('Final Accuracy of ', 'blood', 0.6165585585585586)
0.6165585585585586
>>>
```

yeast



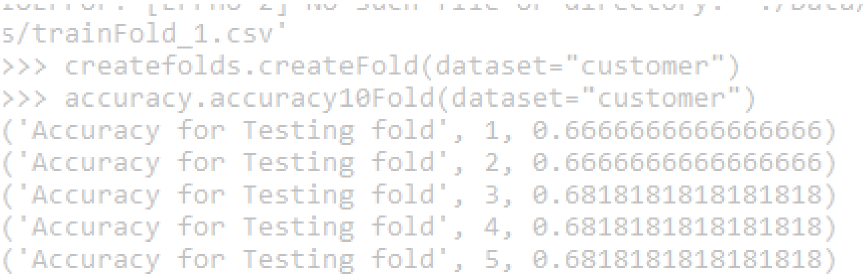
```
>>> createfolds.createFold(dataset="yeast")
>>> accuracy.accuracy10Fold(dataset="yeast")
('Accuracy for Testing fold', 1, 0.40522875816993464)
('Accuracy for Testing fold', 2, 0.40789473684210525)
('Accuracy for Testing fold', 3, 0.39473684210526316)
('Accuracy for Testing fold', 4, 0.4266666666666667)
('Accuracy for Testing fold', 5, 0.49324324324324326)
('Accuracy for Testing fold', 6, 0.4315068493150685)
('Accuracy for Testing fold', 7, 0.3424657534246575)
('Accuracy for Testing fold', 8, 0.3835616438356164)
('Accuracy for Testing fold', 9, 0.4178082191780822)
('Accuracy for Testing fold', 10, 0.41379310344827586)
('Final Accuracy of ', 'yeast', 0.41169058162289146)
0.41169058162289146
```

Breastcancer



```
>>> createfolds.createFold(dataset="breastcancer")
>>> accuracy.accuracy10Fold(dataset="breastcancer")
('Accuracy for Testing fold', 1, 0.8591549295774648)
('Accuracy for Testing fold', 2, 0.8857142857142857)
('Accuracy for Testing fold', 3, 0.9714285714285714)
('Accuracy for Testing fold', 4, 0.8714285714285714)
('Accuracy for Testing fold', 5, 0.9142857142857143)
('Accuracy for Testing fold', 6, 0.9142857142857143)
('Accuracy for Testing fold', 7, 0.9142857142857143)
('Accuracy for Testing fold', 8, 0.9857142857142858)
('Accuracy for Testing fold', 9, 0.9565217391304348)
('Accuracy for Testing fold', 10, 0.9420289855072463)
('Final Accuracy of ', 'breastcancer', 0.9214848511358003)
0.9214848511358003
>>>
```

customer



```
IOError: [Errno 2] No such file or directory: './Data/
s/trainFold_1.csv'
>>> createfolds.createFold(dataset="customer")
>>> accuracy.accuracy10Fold(dataset="customer")
('Accuracy for Testing fold', 1, 0.6666666666666666)
('Accuracy for Testing fold', 2, 0.6666666666666666)
('Accuracy for Testing fold', 3, 0.6818181818181818)
('Accuracy for Testing fold', 4, 0.6818181818181818)
('Accuracy for Testing fold', 5, 0.6818181818181818)
```

(c) Compare Gini and information gain as splitting criteria and discuss any observation on the quality of splitting

(soln) Based on testing Gini and Information gain,entropy on different datasets i found the following differences between them:
1)I observed that choice of impurity measure has little impact on performance of decision tree. (i,e max 2% difference on wine dataset when choosing entropy has impurity measure)
2)Gini seems to be more suitable for continuous attributes and information gain (entropy) for attributes that occur in classes

3)Entropy is little slower to calculate as it utilizes lambda function to calculate logarithm.
4)Gini is better to minimize misclassification error
5)Gini is more biased to find the largest class and "entropy" tends to find groups of classes that make up to ~50% of the data.
(http://paginas.fe.up.pt/~ec/files_1011/week%2008%20-%20Decision%20Trees.pdf)

Q5

(a) Use `pessimistic' estimates of the generalization error by adding a penalty factor 0.5 for each node in the tree (see Textbook page 181).

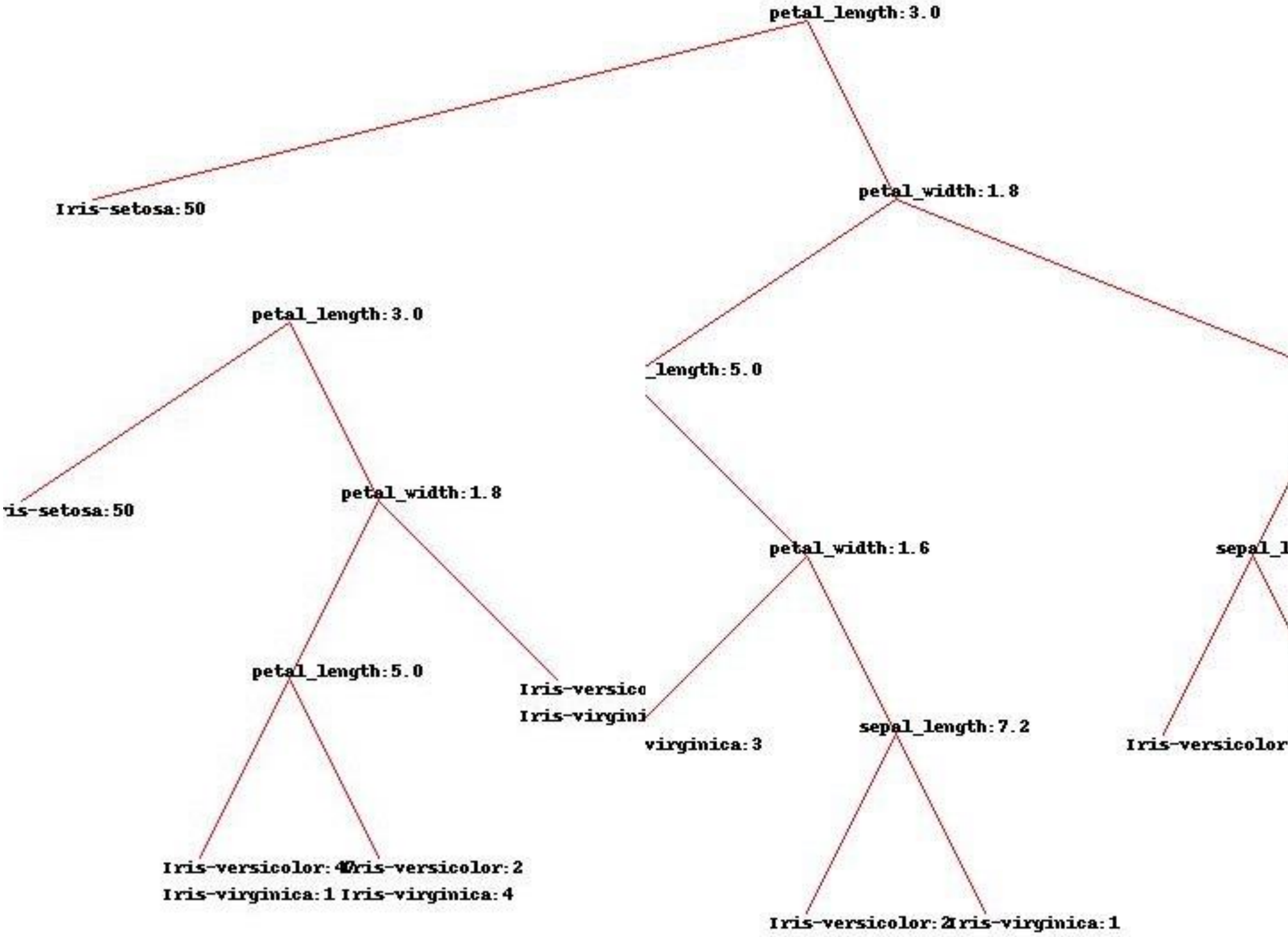
Module:prune.py

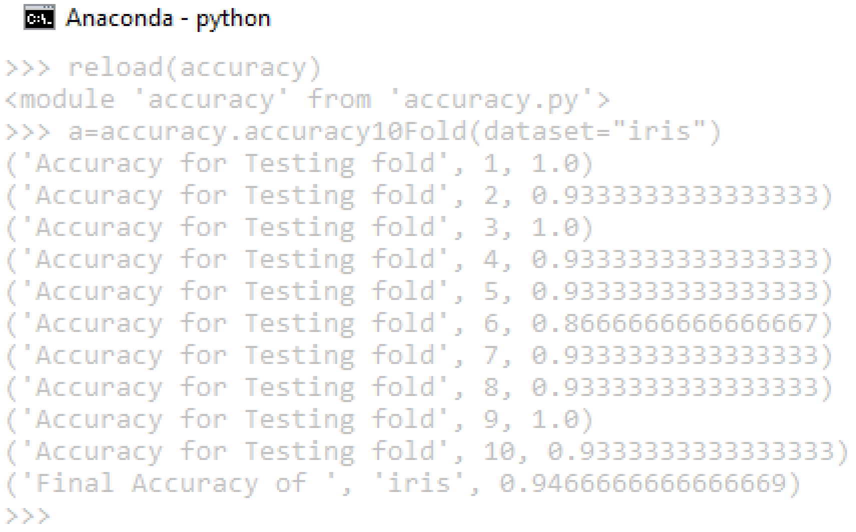
(soln) For this problem , i used pre pruning to incorporance pessimistic generlization error. Basically pruning principle is whenever a node is splitting in two leafs and if the pessimistic error is increasing then that splitting is nullified. Naturally This a tradeoff of model complexity vs accuracy.
I applied following formula to calculate pessimistic generalization error:

$$\mathbf{error} = \ (\mathbf{e}(\mathbf{t})\ +\ \mathbf{o}(\mathbf{t}))/(\mathbf{N})$$

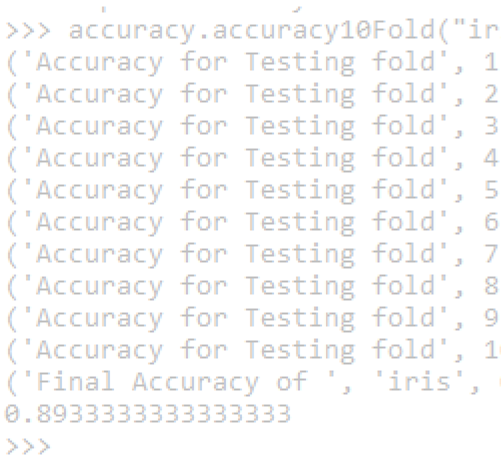
Following are the results when i applied pruning to following datasets(Right side is tree after pruning)

Iris


petal_length:3.0
Iris-setosa:50
petal_width:1.8
petal_length:3.0
_length:5.0
ris-setosa:50
petal_width:1.8
petal_width:1.6
sepal_l
petal_length:5.0
Iris-versico
Iris-virgini
virginica:3
sepal_length:7.2
Iris-versicolor
Iris-versicolor:47
Iris-virginica:1
Iris-versicolor:2
Iris-virginica:4
Iris-versicolor:2
Iris-virginica:1

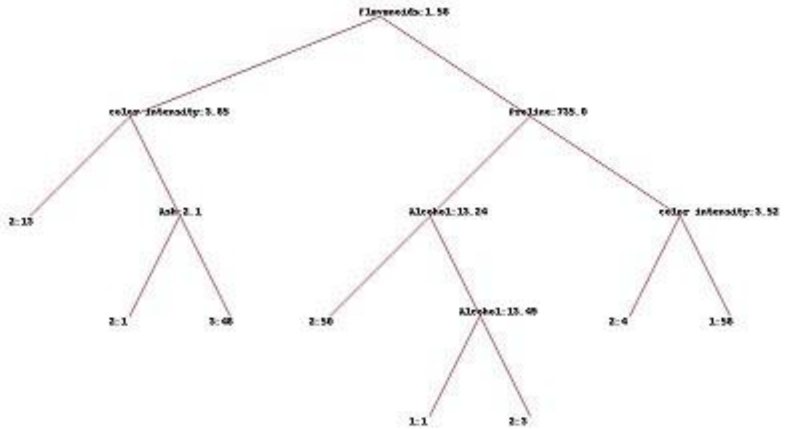


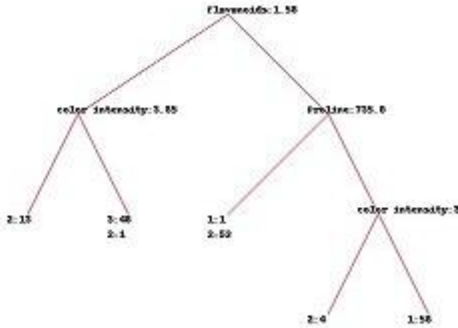
```
>>> reload(accuracy)
<module 'accuracy' from 'accuracy.py'>
>>> a=accuracy.accuracy10Fold(dataset="iris")
('Accuracy for Testing fold', 1, 1.0)
('Accuracy for Testing fold', 2, 0.9333333333333333)
('Accuracy for Testing fold', 3, 1.0)
('Accuracy for Testing fold', 4, 0.9333333333333333)
('Accuracy for Testing fold', 5, 0.9333333333333333)
('Accuracy for Testing fold', 6, 0.8666666666666667)
('Accuracy for Testing fold', 7, 0.9333333333333333)
('Accuracy for Testing fold', 8, 0.9333333333333333)
('Accuracy for Testing fold', 9, 1.0)
('Accuracy for Testing fold', 10, 0.9333333333333333)
('Final Accuracy of ', 'iris', 0.9466666666666669)
>>>
```

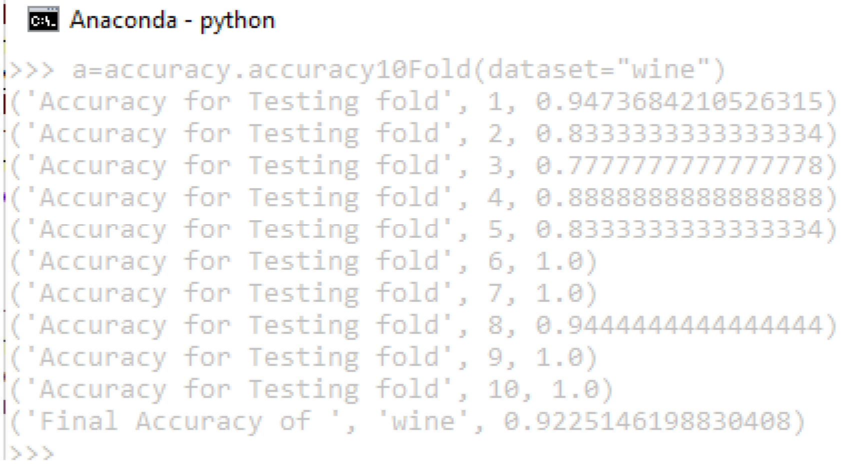


```
>>> accuracy.accuracy10Fold("ir
('Accuracy for Testing fold', 1
('Accuracy for Testing fold', 2
('Accuracy for Testing fold', 3
('Accuracy for Testing fold', 4
('Accuracy for Testing fold', 5
('Accuracy for Testing fold', 6
('Accuracy for Testing fold', 7
('Accuracy for Testing fold', 8
('Accuracy for Testing fold', 9
('Accuracy for Testing fold', 1
('Final Accuracy of ', 'iris', 
0.8933333333333333
>>>
```

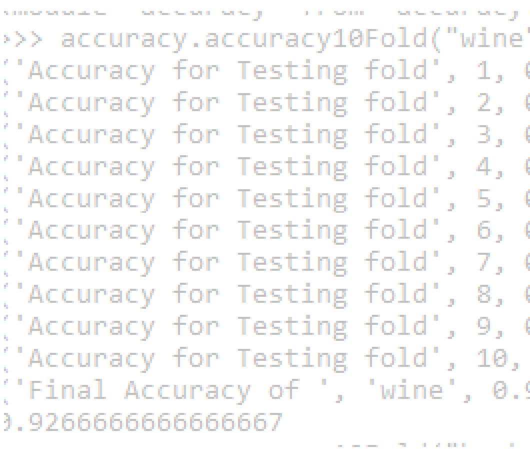
Wine







```
>>> a=accuracy.accuracy10Fold(dataset="wine")
('Accuracy for Testing fold', 1, 0.9473684210526315)
('Accuracy for Testing fold', 2, 0.8333333333333334)
('Accuracy for Testing fold', 3, 0.7777777777777778)
('Accuracy for Testing fold', 4, 0.8888888888888888)
('Accuracy for Testing fold', 5, 0.8333333333333334)
('Accuracy for Testing fold', 6, 1.0)
('Accuracy for Testing fold', 7, 1.0)
('Accuracy for Testing fold', 8, 0.9444444444444444)
('Accuracy for Testing fold', 9, 1.0)
('Accuracy for Testing fold', 10, 1.0)
('Final Accuracy of ', 'wine', 0.9225146198830408)
>>>
```



```
>> accuracy.accuracy10Fold("wine"
'Accuracy for Testing fold', 1, 0
'Accuracy for Testing fold', 2, 0
'Accuracy for Testing fold', 3, 0
'Accuracy for Testing fold', 4, 0
'Accuracy for Testing fold', 5, 0
'Accuracy for Testing fold', 6, 0
'Accuracy for Testing fold', 7, 0
'Accuracy for Testing fold', 8, 0
'Accuracy for Testing fold', 9, 0
'Accuracy for Testing fold', 10, 
'Final Accuracy of ', 'wine', 0.9
.9266666666666667
```

As you can see that efficiency is little decreased when pruning but model complexsity also hugely decreased