Data Mining - Assignment-2

Anirudh K M - anikamal

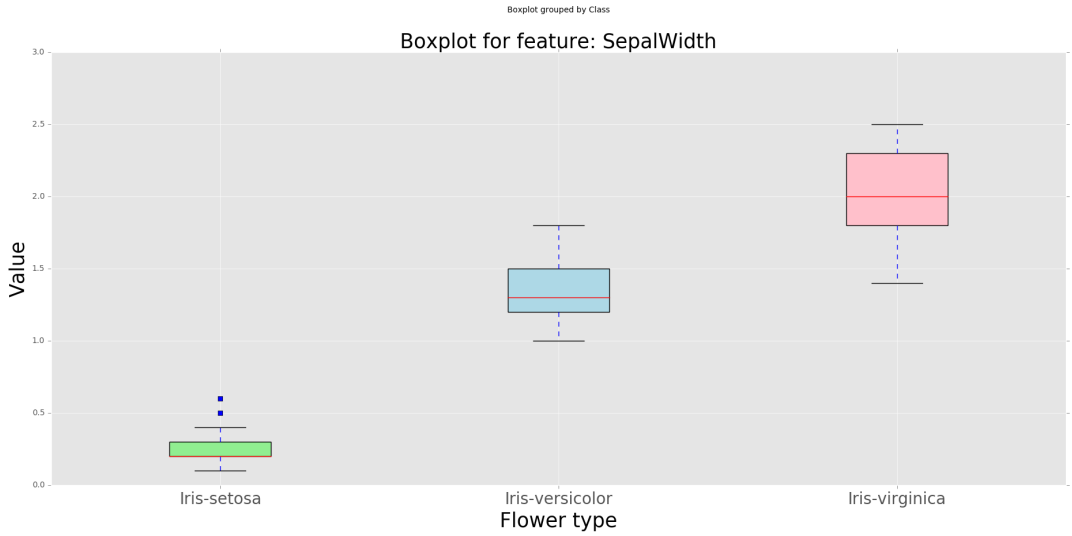
25th February, 2016

1 Problem 1

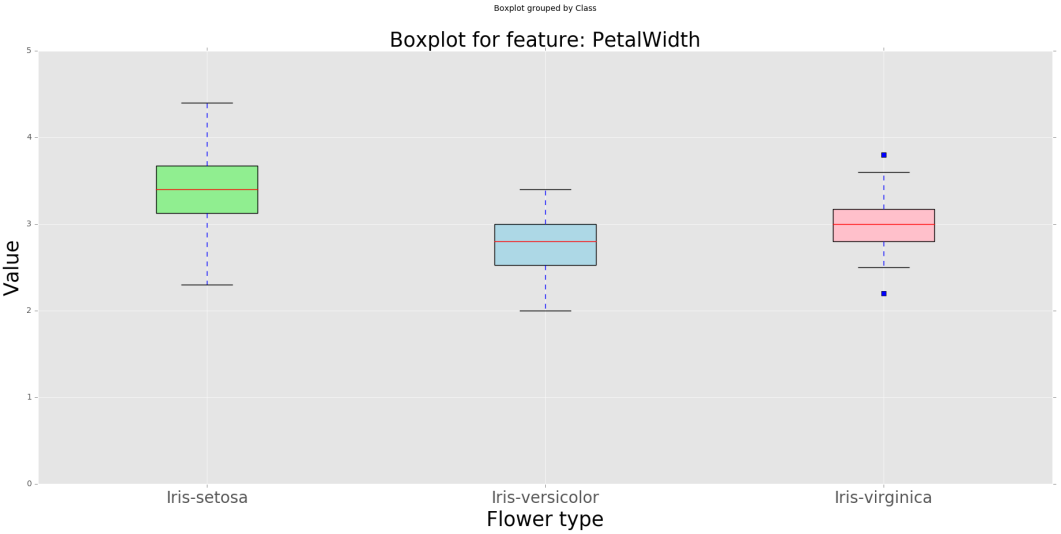
The average value and the standard deviation for each of the four features are as follows

```
student@student:~/Dropbox/hw2/1$ python iris_analysis.py
Results which includes all flower types
=======================================
Feature: PetalLength
Mean: 5.843333
SD: 0.825301

Feature: PetalWidth
Mean: 3.054000
SD: 0.432147

Feature: SepalLength
Mean: 3.758667
SD: 1.758529

Feature: SepalWidth
Mean: 1.198667
SD: 0.760613

Result for flower type: Iris-virginica
=======================================
Feature: PetalLength
Mean: 6.588000
SD: 0.629489

Feature: PetalWidth
Mean: 2.974000
SD: 0.319255

Feature: SepalLength
Mean: 5.552000
SD: 0.546348

Feature: SepalWidth
Mean: 2.026000
SD: 0.271890

Result for flower type: Iris-setosa
===================================
Feature: PetalLength
Mean: 5.006000
SD: 0.348947

Feature: PetalWidth
Mean: 3.418000
SD: 0.377195

Feature: SepalLength
Mean: 1.464000
SD: 0.171767

Feature: SepalWidth
Mean: 0.244000
SD: 0.106132

Result for flower type: Iris-versicolor
=======================================
Feature: PetalLength
Mean: 5.936000
SD: 0.510983

Feature: PetalWidth
Mean: 2.770000
SD: 0.310644

Feature: SepalLength
Mean: 4.260000
SD: 0.465188

Feature: SepalWidth
Mean: 1.326000
SD: 0.195765
```

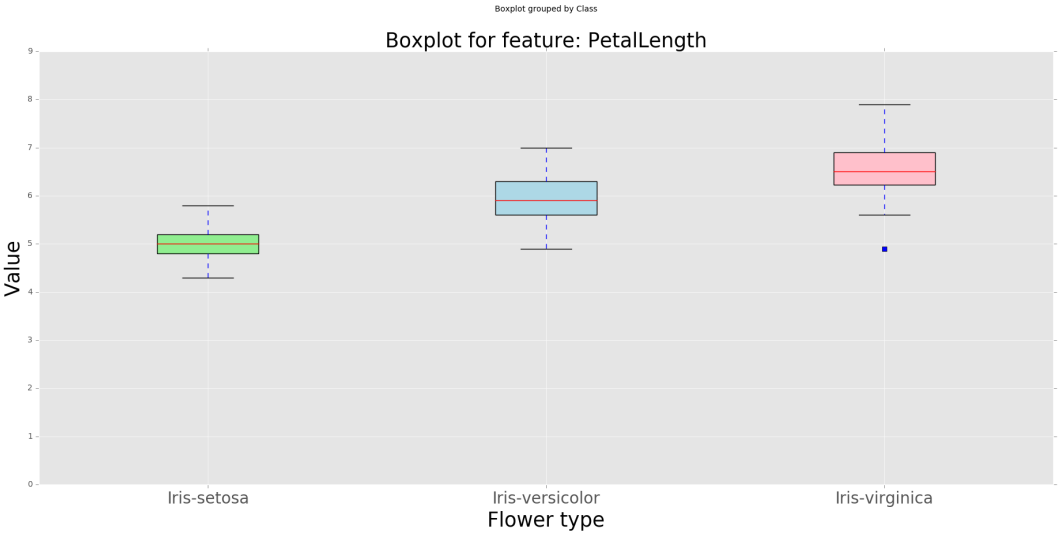
The box plot of the data one for each feature is as follows



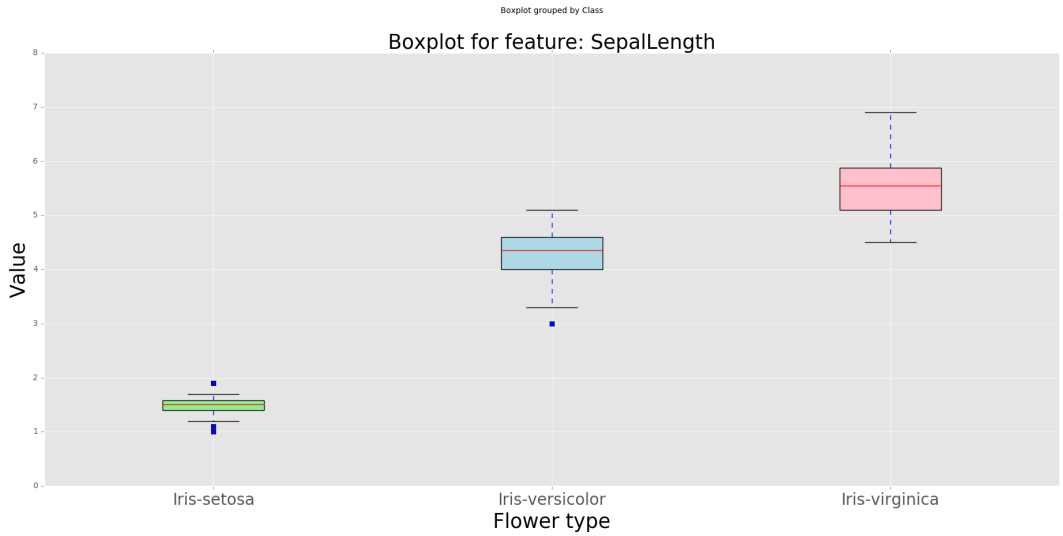
For the feature *Sepal Width* we see that the box of virginica is higher compared to rest and Setosa the lowest.



For the feature *Petal Width* we can see that the box of Setosa is higher compared to the rest.



For the feature *Petal Length* we can see that the box of Virginica is higher compared to the rest and all of them lie in the same range.



For the feature *Sepal Length* we see that the box of virginica is higher compared to rest and Setosa the lowest.

The experimentation I have tried is changed the color of the boxes, changes the border thickness and gave fontsize to the labels of the boxplots.

Also, I have applied *ggplot* style from the matplotlib library.

2 Problem 2

The plots of the four most correlating features are as follows

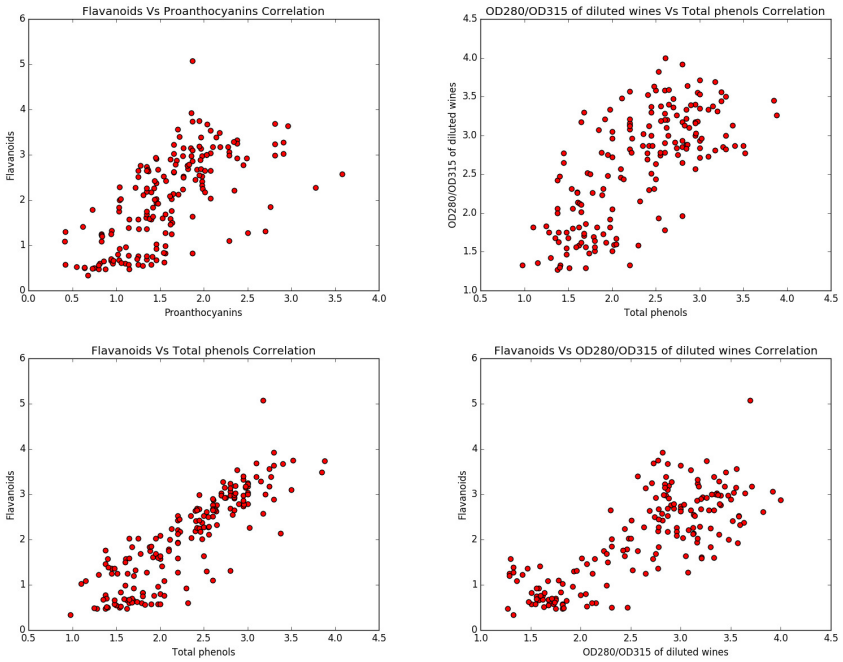


Figure 1: Most correlated plots based on features

The plots of the four least correlating features are as follows

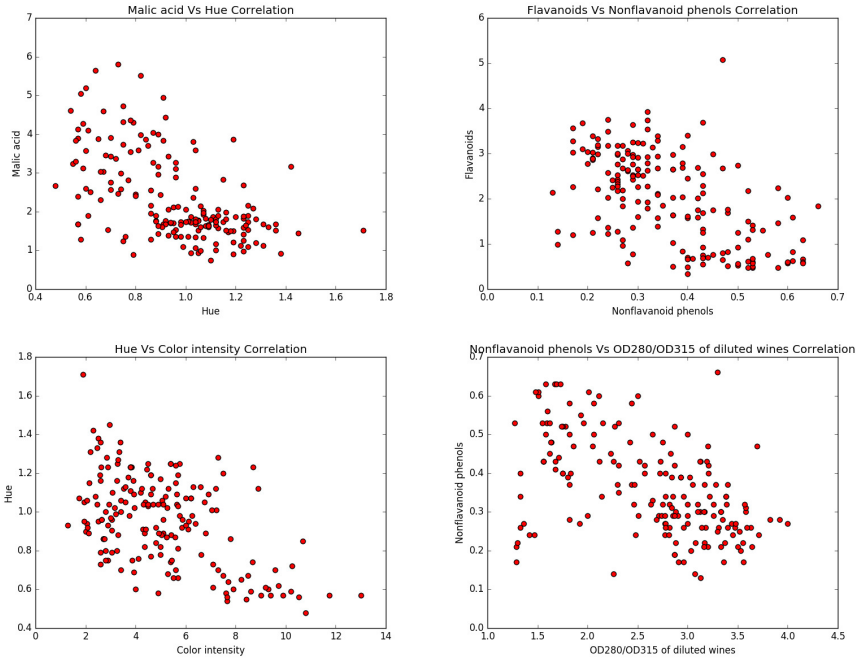


Figure 2: Least correlated plots based on features

Results to find the percent of close data points

```
Select C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.10240]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\anirudhkm\Box Sync\education\spring_2016\data_mining\hw2\hw2\2>python wine_data_analysis.py wine.data wine_headers.data
Actual data
Percentage of points with class neighbors is 76.9663%
For class 1
Percentage of points with class neighbors is 88.1356%
For class 2
Percentage of points with class neighbors is 76.0563%
For class 3
Percentage of points with class neighbors is 64.5833%


0-1 normalized data
Percentage of points with class neighbors is 100.0%
For class 1
Percentage of points with class neighbors is 100.0%
For class 2
Percentage of points with class neighbors is 100.0%
For class 3
Percentage of points with class neighbors is 100.0%

z-score normalized data
Percentage of points with class neighbors is 96.0674%
For class 1
Percentage of points with class neighbors is 100.0%
For class 2
Percentage of points with class neighbors is 90.1408%
For class 3
Percentage of points with class neighbors is 100.0%

C:\Users\anirudhkm\Box Sync\education\spring_2016\data_mining\hw2\hw2\2>
```

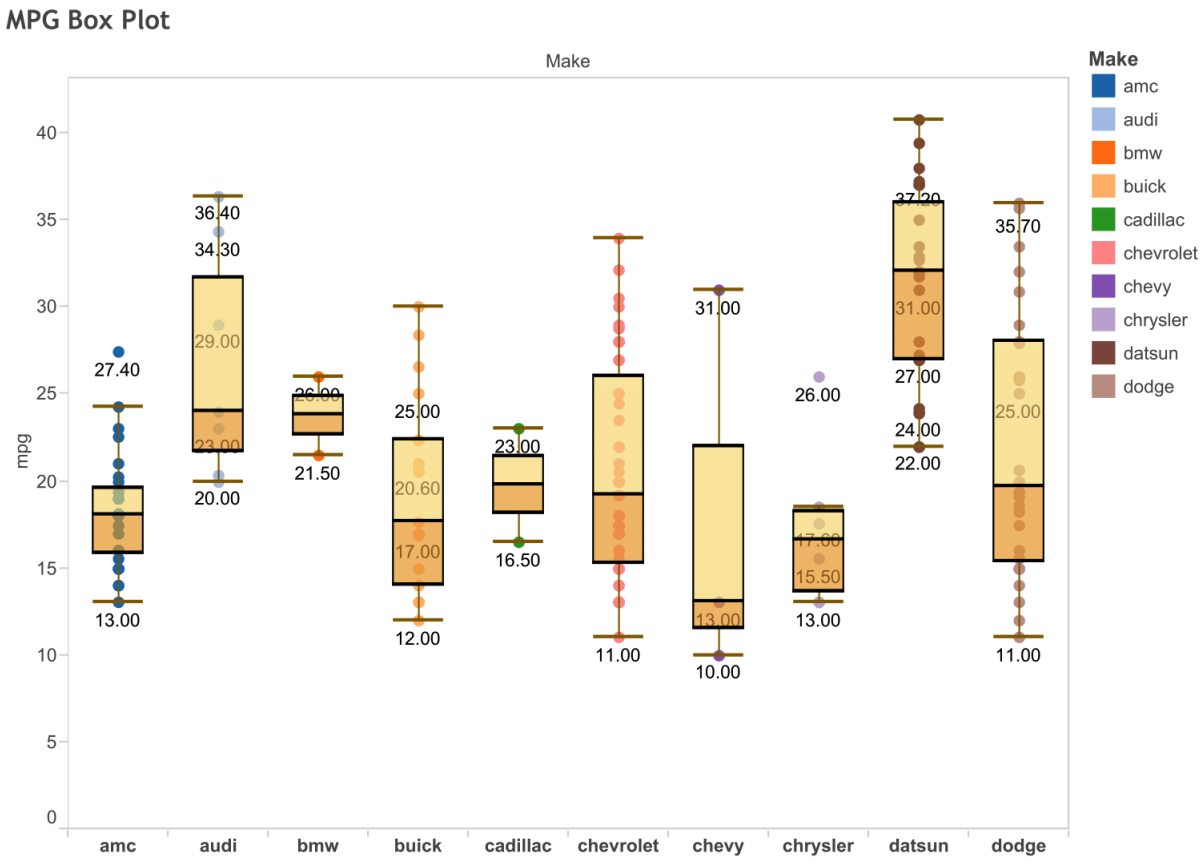
We observe that after normalization the percentage of close points increases by a greater deal.

This is because there is a higher difference between the range of features. Features such as Magnesium, Alcalinity of Ash have higher range and features such as Hue has lower range.

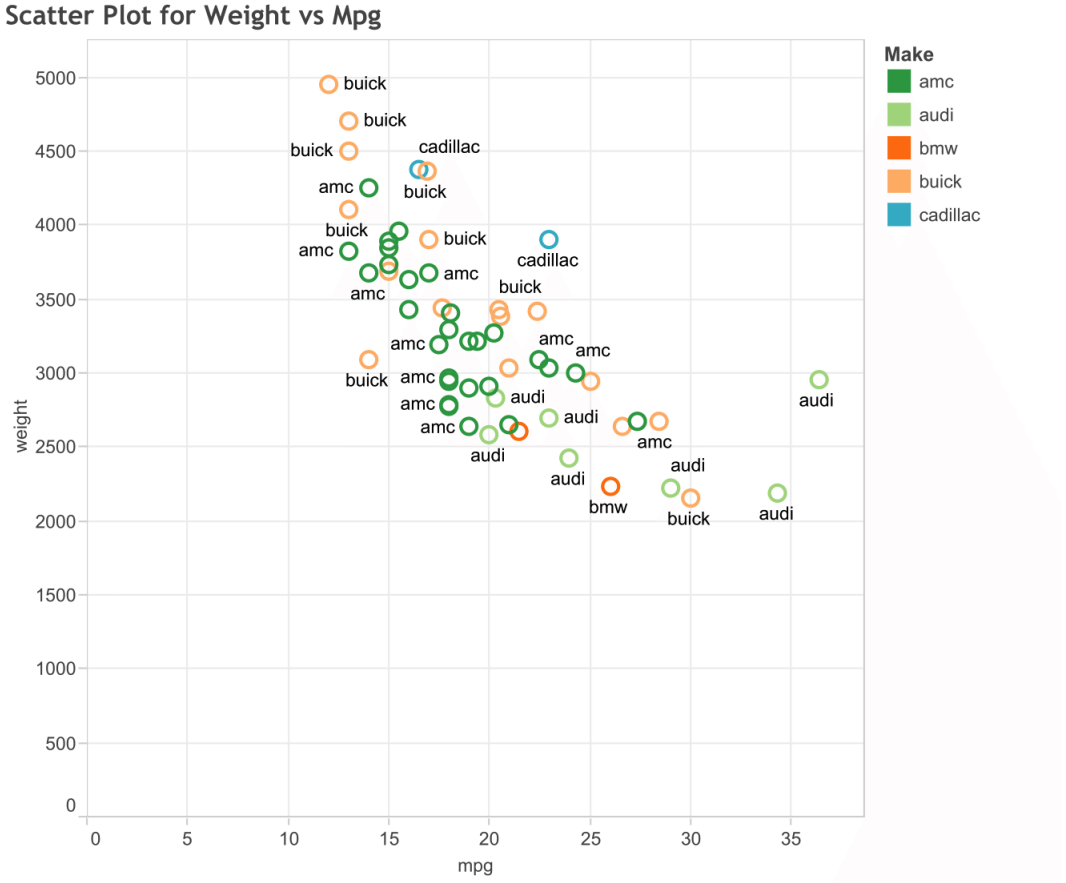
Because of these factors we observe discrepancy in the data, but after normalization, we bring the range of the each feature such close which helps in achieving higher percentage of closer points.

3 Problem 3

a



Mpg for each Make. Color shows details about Make. The marks are labeled by mpg. The view is filtered on Make, which keeps 10 of 37 members.



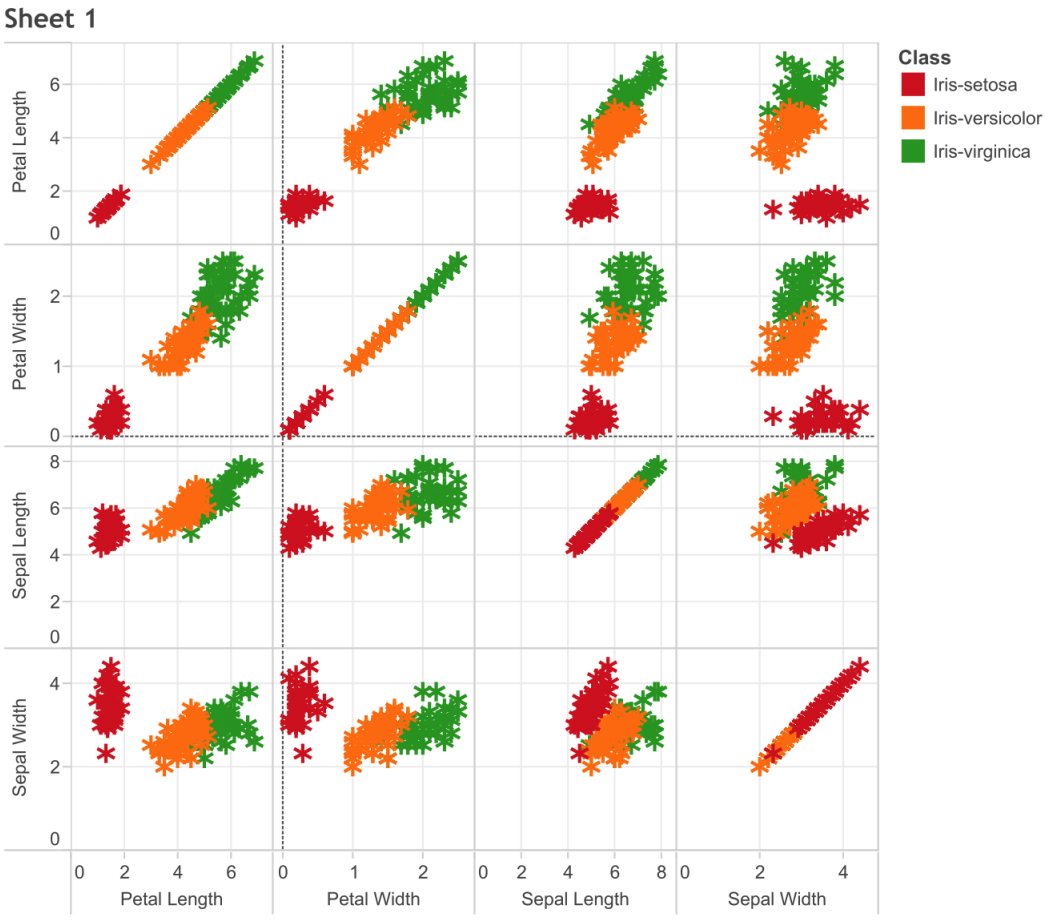
Mpg vs. weight. Color shows details about Make. The marks are labeled by Make. The view is filtered on Make, which keeps amc, audi, bmw, buick and cadillac.

From the boxplot we observe that the **datasun** has the highest mpg of the 10 makes.

From the scatter plot we can see that there is always a negative correlation between weight and mpg for all the 5 makes analyzed.

b

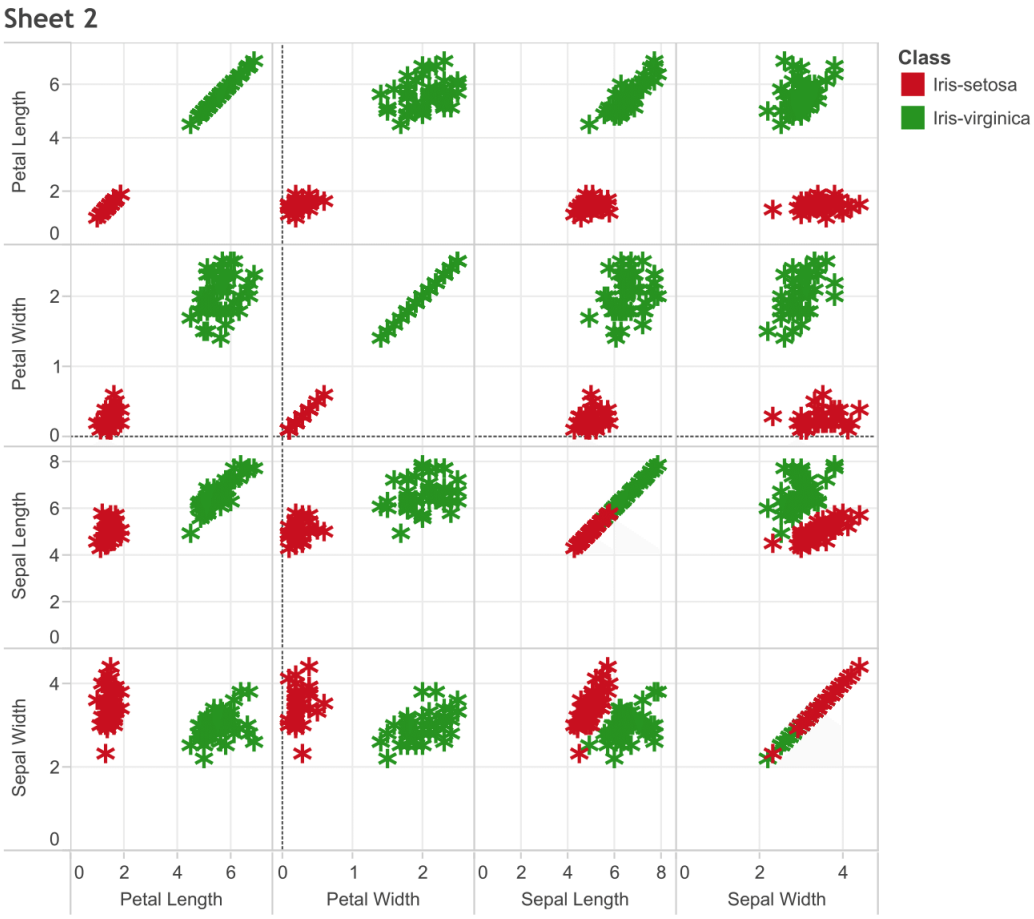
IRIS dataset



This graphs shows the pairs correlation between the features of iris dataset.

We could see that the data is very much separable between the classes i.e the flower type.

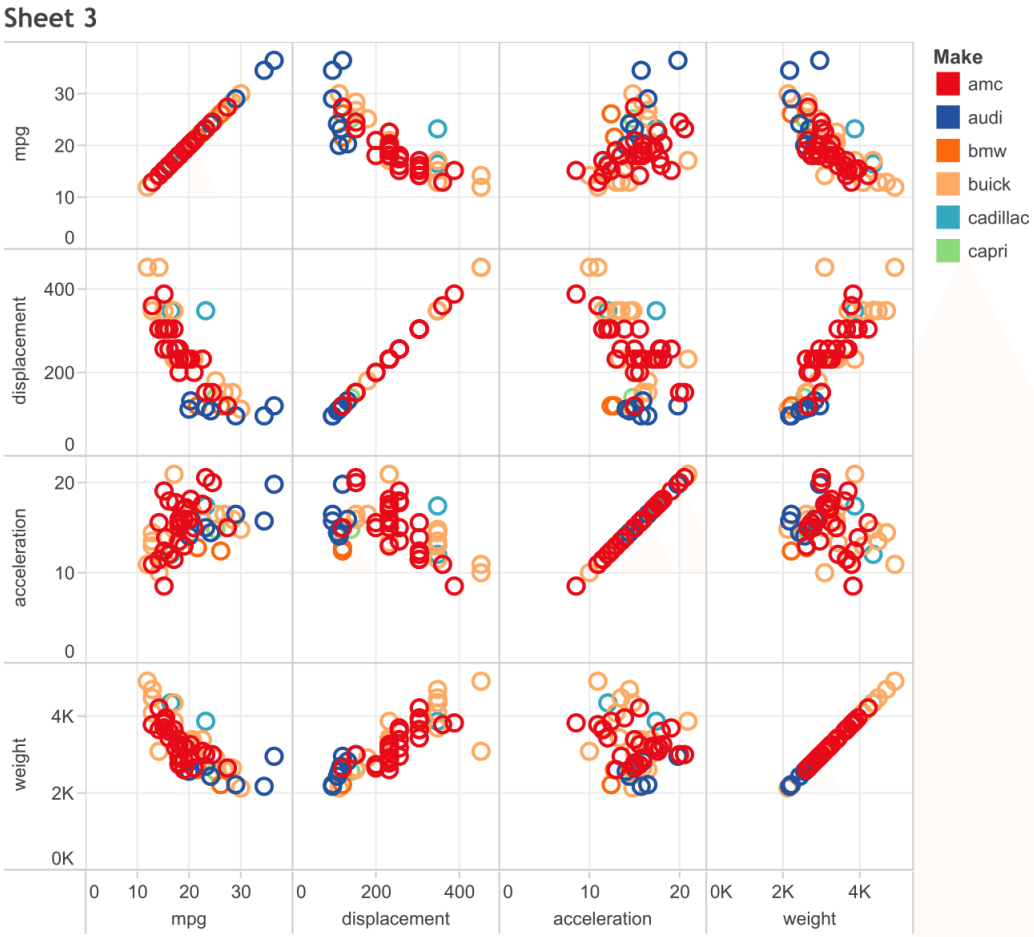
I have tried experimenting with the colors, size, shapes, fiters and groups on this dataset.



Petal Length, Petal Width, Sepal Length and Sepal Width vs. Petal Length, Petal Width, Sepal Length and Sepal Width. Color shows details about Class. The view is filtered on Class, which keeps Iris-setosa and Iris-virginica.

This graph represent with the same graph as above but with only two classes.

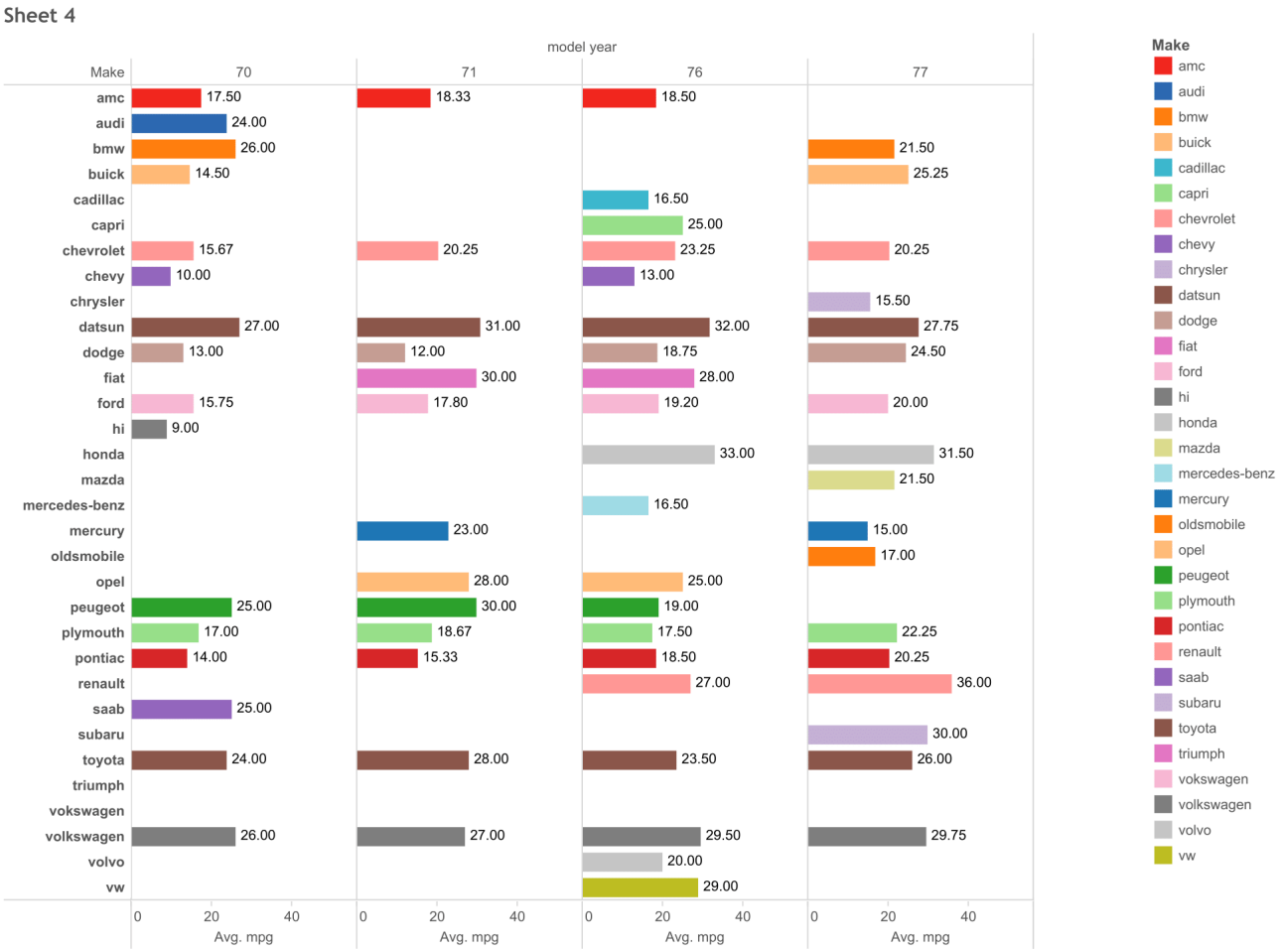
Auto MPG dataset



Mpg, displacement, acceleration and weight vs. mpg, displacement, acceleration and weight. Color shows details about Make. The view is filtered on Make, which keeps 6 of 37 members.

This plot shows the correlation between features such as weight, acceleration, displacement and mpg for 5 car makes. We observe that there is negative correlation between displacement and mpg, mpg and weight and few more.

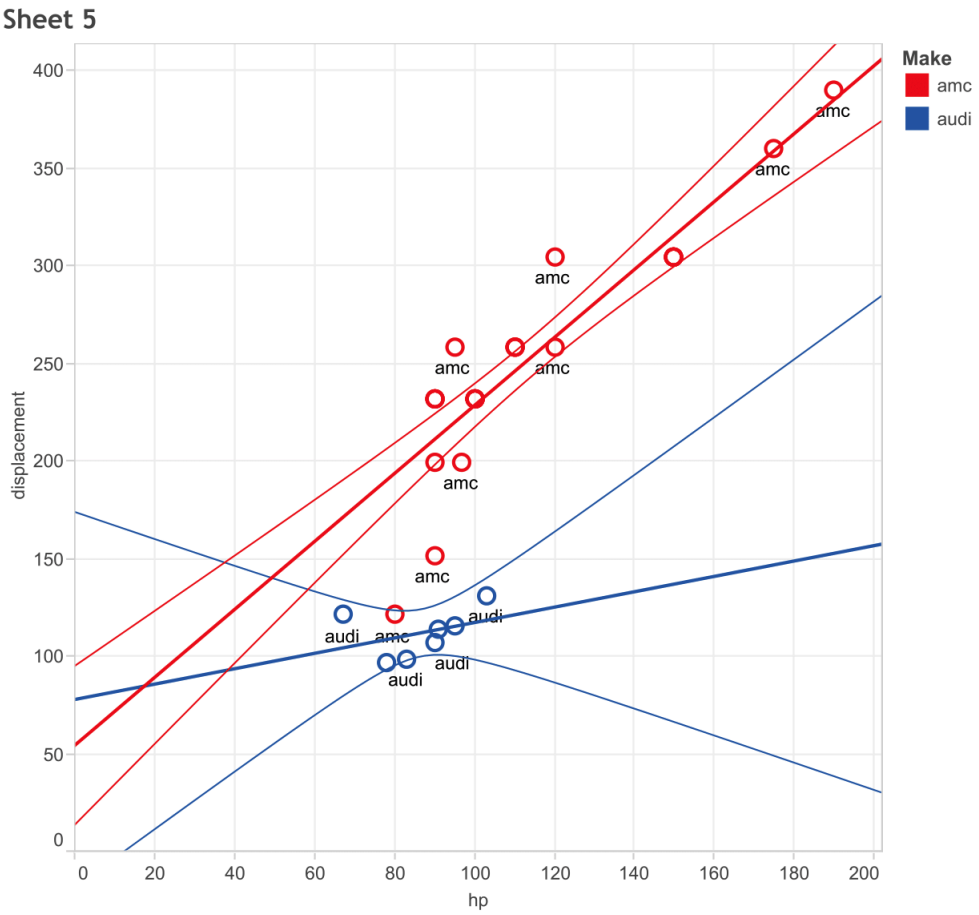
Also, we see positive correlation between displacement and weight, weight and displacement and etc.



Average of mpg for each Make broken down by model year. Color shows details about Make. The marks are labeled by average of mpg. The view is filtered on model year, which keeps 70, 71, 76, 77 and 80.

This plot shows us the the mpg value for all makes over the years.

From this we can see how companies have improved their mpg over the time.

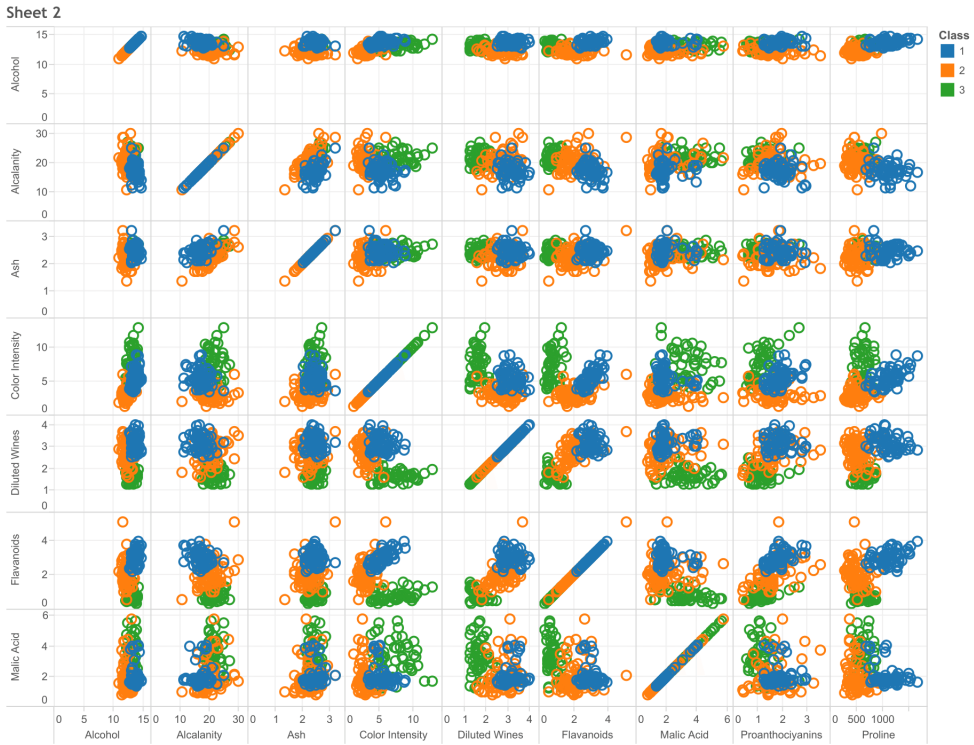


Hp vs. displacement. Color shows details about Make. The marks are labeled by Make. The view is filtered on Make, which keeps amc and audi.

This graph shows the linear regression line between displacement and horsepower of the data for make audi and amc.

We see amc has a better correlation between displacement and horsepower.

Wine data set



This plot shows the correlation between various features of the wine data set highlighted by the classes.

From this graph we can see that their no proper correlation between any of the features in the wine data set.

c

This is my first experience with tableau and I found it really easy and interested.

The interface is good and it helps me to produce results in a easier way. Also, the plots are much more visually applealing compared to matplotlib in Python and the basic plot in R.

I did really like tableau and would love to make use of this in my future projects.

4 Problem 4

a

The following are example output for decision tree based on gini and information gain, side by side.

```
File  Edit  View  Terminal  Tabs  Help
student@student:~/Dropbox/hw2/4/combine$ python main.py iris.data entropy
                Decision tree displayed for iris.data

 ('PetalLength', (0.9183, 2.34))
        ('PetalWidth', (0.6900999999999999, 1.75))
                ('PetalLength', (0.0912, 4.8))
                        set(['Iris-virginica'])
                        ('SepalLength', (0.9183, 5.9))
                                set(['Iris-virginica'])
                                set(['Iris-versicolor'])
                ('PetalLength', (0.2132, 4.9))
                        ('PetalWidth', (0.4591, 1.58))
                                ('SepalLength', (0.9183, 6.9))
                                        set(['Iris-virginica'])
                                        set(['Iris-versicolor'])
                                set(['Iris-virginica'])
                        ('PetalWidth', (0.1461, 1.63))
                                set(['Iris-virginica'])
                                set(['Iris-versicolor'])
        set(['Iris-setosa'])
Results
=======
Impurity measure: entropy
Accuracy after 10-fold CV: 95.5833333333
student@student:~/Dropbox/hw2/4/combine$
```

```
File  Edit  View  Terminal  Tabs  Help
student@student:~/Dropbox/hw2/4/combine$ python main.py iris.data gini
                Decision tree displayed for iris.data

 ('PetalLength', (0.3333, 2.34))
        ('PetalWidth', (0.1103, 1.75))
                ('PetalLength', (0.029, 4.8))
                        set(['Iris-virginica'])
                        ('SepalWidth', (0.0, 3.0))
                                set(['Iris-versicolor'])
                                set(['Iris-virginica'])
                ('PetalLength', (0.0856, 4.9))
                        ('PetalWidth', (0.2222, 1.58))
                                ('PetalLength', (0.0, 5.38))
                                        set(['Iris-virginica'])
                                        set(['Iris-versicolor'])
                                set(['Iris-virginica'])
                        ('PetalWidth', (0.0, 1.63))
                                set(['Iris-virginica'])
                                set(['Iris-versicolor'])
        set(['Iris-setosa'])
Results
=======
Impurity measure: gini
Accuracy after 10-fold CV: 94.3333333333
student@student:~/Dropbox/hw2/4/combine$
```

```
File  Edit  View  Terminal  Tabs  Help
student@student:~/Dropbox/hw2/4/combine$ python main.py seeds.txt entropy
                Decision tree displayed for seeds.txt

 ('f7', (0.7756, 5.5272))
        ('f1', (0.10470000000000002, 16.824))
                set([2])
                ('f7', (0.39490000000000003, 5.5415))
                        ('f6', (0.4138, 2.04))
                                set([2])
                                set([1])
                        set([1])
        ('f1', (0.6119000000000001, 13.418))
                ('f5', (0.09409999999999999, 3.4508))
                        ('f6', (0.7219, 3.1696))
                                set([2])
                                set([1])
                        set([1])
                ('f6', (0.20270000000000005, 3.6277))
                        ('f1', (0.08610000000000001, 12.55))
                                ('f6', (0.684, 4.157))
                                        set([3])
                                        set([1])
                                set([3])
                        ('f7', (0.3647, 4.981))
                                ('f2', (0.6194, 13.746))
                                        set([1])
                                        set([3])
                                ('f5', (0.4204, 3.119))
                                        ('f2', (0.9183, 13.482))
                                                set([1])
                                                set([3])
                                        set([1])
Results
=======
Impurity measure: entropy
Accuracy after 10-fold CV: 93.5930735931
student@student:~/Dropbox/hw2/4/combine$
```

```
File  Edit  View  Terminal  Tabs  Help
student@student:~/Dropbox/hw2/4/combine$ python main.py seeds.txt gini
                Decision tree displayed for seeds.txt

('f7', (0.3674, 5.5272))
        ('f7', (0.0547, 5.528))
                ('f7', (0.0282, 5.533))
                        ('f6', (0.0248, 2.1008))
                                set([2])
                                ('f5', (0.0, 3.5904))
                                        set([2])
                                        set([1])
                        set([1])
                set([1])
        ('f1', (0.1956, 13.418))
                ('f5', (0.0582, 3.4508))
                        ('f6', (0.0, 3.1696))
                                set([2])
                                set([1])
                        set([1])
                ('f7', (0.1721, 4.7841))
                        ('f1', (0.1355, 12.72))
                                ('f6', (0.1191, 4.2596))
                                        set([3])
                                        ('f3', (0.0, 0.8822))
                                                set([3])
                                                set([1])
                                ('f6', (0.0317, 1.415))
                                        ('f7', (0.0262, 4.9392))
                                                set([3])
                                                ('f5', (0.0, 2.9946))
                                                        set([1])
                                                        set([3])
                                        set([1])
                        ('f3', (0.125, 0.8955))
                                ('f3', (0.0, 0.898))
                                        set([1])
                                        set([3])
                                set([1])
Results
=======
Impurity measure: gini
Accuracy after 10-fold CV: 89.5021645022
student@student:~/Dropbox/hw2/4/combine$
```

```
File  Edit  View  Terminal  Tabs  Help
student@student:~/Dropbox/hw2/4/combine$ python main.py /home/student/Dropbox/hw2/DataSets/
                Decision tree displayed for /home/student/Dropbox/hw2/DataSets/Wine.csv

 ('Flavanoids', (0.6313, 1.396))
        ('Proline', (0.7532, 750.0))
                ('Color Intensity', (0.18639999999999998, 3.73))
                        set([1])
                        ('Alcohol', (1.0, 13.02))
                                set([1])
                                set([2])
                ('Alcohol', (0.18460000000000001, 13.201))
                        ('Magnesium', (0.9999, 98.5))
                                ('Proline', (0.9183, 608.0))
                                        set([1])
                                        set([3])
                                set([2])
                        set([2])
        ('Color Intensity', (0.5852, 3.91))
                set([3])
                ('Color Intensity', (0.4395, 3.6))
                        set([3])
                        set([2])
Results
=======
Impurity measure: entropy
Accuracy after 10-fold CV: 93.0701754386
student@student:~/Dropbox/hw2/4/combine$
```

```
File  Edit  View  Terminal  Tabs  Help
student@student:~/Dropbox/hw2/4/combine$ python main.py /home/student/Dropbox/hw2/DataSets
                Decision tree displayed for /home/student/Dropbox/hw2/DataSets/Wine.csv

        ('Proline', (0.4308, 937.0))
                ('Color Intensity', (0.0, 3.35))
                        set([1])
                        set([2])
                ('Color Intensity', (0.2862, 3.81))
                        ('Flavanoids', (0.1776, 1.394))
                                ('Magnesium', (0.2174, 98.2))
                                        ('Proline', (0.0952, 608.0))
                                                set([1])
                                                ('Nonflavanoid phenols', (0.0, 0.22))
                                                        set([2])
                                                        set([3])
                                        set([2])
                                set([3])
                        ('Proline', (0.0393, 790.0))
                                ('Malic Acid ', (0.0, 1.69))
                                        set([1])
                                        set([2])
                                set([2])
    Results
    =======
    Impurity measure: gini
    Accuracy after 10-fold CV: 96.2865497076
    student@student:~/Dropbox/hw2/4/combine$
```

b

The 10 fold cross validation results are as follows with gini as the impurity measure.

```
Microsoft Windows [Version 10.0.10240]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\anirudhkm\Box Sync\education\spring_2016\data_mining\hw2\hw2\4>python main.py data/iris.data gini
data/iris.data
Results
=======
Impurity measure: gini
Accuracy after 10-fold CV: 94.9166666667

C:\Users\anirudhkm\Box Sync\education\spring_2016\data_mining\hw2\hw2\4>python main.py data/seeds.txt gini
data/seeds.txt
Results
=======
Impurity measure: gini
Accuracy after 10-fold CV: 91.2554112554

C:\Users\anirudhkm\Box Sync\education\spring_2016\data_mining\hw2\hw2\4>python main.py data/Wine.csv gini
data/Wine.csv
Results
=======
Impurity measure: gini
Accuracy after 10-fold CV: 88.8304093567

C:\Users\anirudhkm\Box Sync\education\spring_2016\data_mining\hw2\hw2\4>python main.py data/user_knowledge.data gini
data/user_knowledge.data
Results
=======
Impurity measure: gini
Accuracy after 10-fold CV: 92.150997151

C:\Users\anirudhkm\Box Sync\education\spring_2016\data_mining\hw2\hw2\4>python main.py data/BreastTissue.csv gini
data/BreastTissue.csv
Results
=======
Impurity measure: gini
Accuracy after 10-fold CV: 72.196969697

C:\Users\anirudhkm\Box Sync\education\spring_2016\data_mining\hw2\hw2\4>python main.py data/glass.data gini
data/glass.data
Results
=======
Impurity measure: gini
Accuracy after 10-fold CV: 68.6561264822

C:\Users\anirudhkm\Box Sync\education\spring_2016\data_mining\hw2\hw2\4>python main.py data/VertebralColumn.csv gini
data/VertebralColumn.csv
Results
=======
Impurity measure: gini
Accuracy after 10-fold CV: 80.8669354839
```

```
C:\Users\anirudhkm\Box Sync\education\spring_2016\data_mining\hw2\hw2\4>python main.py data/iris.data entropy
data/iris.data
Results
=======
Impurity measure: entropy
Accuracy after 10-fold CV: 94.9583333333

C:\Users\anirudhkm\Box Sync\education\spring_2016\data_mining\hw2\hw2\4>python main.py data/seeds.txt entropy
data/seeds.txt
Results
=======
Impurity measure: entropy
Accuracy after 10-fold CV: 89.8917748918

C:\Users\anirudhkm\Box Sync\education\spring_2016\data_mining\hw2\hw2\4>python main.py data/Wine.csv entropy
data/Wine.csv
Results
=======
Impurity measure: entropy
Accuracy after 10-fold CV: 93.5380116959

C:\Users\anirudhkm\Box Sync\education\spring_2016\data_mining\hw2\hw2\4>python main.py data/user_knowledge.data entropy
data/user_knowledge.data
Results
=======
Impurity measure: entropy
Accuracy after 10-fold CV: 95.5128205128

C:\Users\anirudhkm\Box Sync\education\spring_2016\data_mining\hw2\hw2\4>python main.py data/BreastTissue.csv entropy
data/BreastTissue.csv
Results
=======
Impurity measure: entropy
Accuracy after 10-fold CV: 64.3939393939

C:\Users\anirudhkm\Box Sync\education\spring_2016\data_mining\hw2\hw2\4>python main.py data/glass.data entropy
data/glass.data
Results
=======
Impurity measure: entropy
Accuracy after 10-fold CV: 66.8774703557

C:\Users\anirudhkm\Box Sync\education\spring_2016\data_mining\hw2\hw2\4>python main.py data/VertebralColumn.csv entropy
data/VertebralColumn.csv
Results
=======
Impurity measure: entropy
Accuracy after 10-fold CV: 83.3870967742

C:\Users\anirudhkm\Box Sync\education\spring_2016\data_mining\hw2\hw2\4>
```

c

From the cross validation results obtained, we see that there is not much difference in the accuracy measure with gini and information gain.

There were observations where one seems to perform better than the other, but when seeing from a broder angle, there doesn't seem to any difference in the usage of gini and information gain.

5 Problem 5

In this we add the pessimistic error and then calculate the cost of the decision tree using the minimum description length.

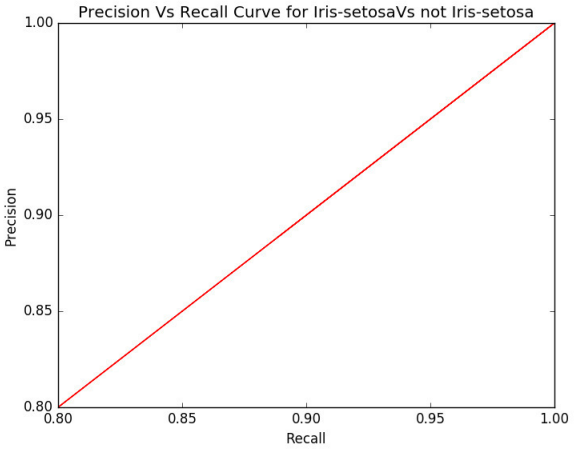
The results are as follows

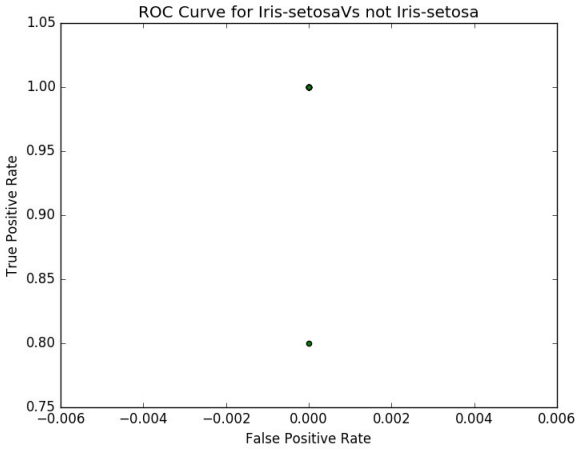
```
C:\Windows\System32\cmd.exe

C:\Users\anirudhkm\Box Sync\education\spring_2016\data_mining\hw2\hw2\5\new>python main.py data/iris.data gini
The cost of the trees based on 10 fold CV are
[37.3308, 44.5007, 40.9157, 33.7458, 19.4059, 48.0856, 40.9264]
The best tree of the 10 trees obtained is
   ('PetalLength', (0.3242, 1.9))
        ('PetalWidth', (0.0844, 1.6))
                set(['Iris-virginica'])
                ('PetalLength', (0.0, 4.9))
                        set(['Iris-virginica'])
                        set(['Iris-versicolor'])
        set(['Iris-setosa'])
None

The cost of the above tree is  19.4059

C:\Users\anirudhkm\Box Sync\education\spring_2016\data_mining\hw2\hw2\5\new>python main.py data/Wine.csv gini
The cost of the trees based on 10 fold CV are
[93.4733, 56.4665, 61.7519, 45.8957, 56.4755, 51.1811, 82.8935, 77.6171]
The best tree of the 10 trees obtained is
   ('Color Intensity', (0.3998, 3.816))
        ('Flavanoids', (0.1914, 1.39))
                ('Proline', (0.0662, 678.0))
                        ('Total phenols', (0.0, 2.2))
                                set([1])
                                set([2])
                        ('Hue', (0.0, 0.59))
                                set([2])
                                set([3])
                set([3])
        ('Diluted Wines', (0.0414, 3.434))
                ('Color Intensity', (0.0, 3.418))
                        set([1])
                        set([2])
                set([2])
None

The cost of the above tree is  45.8957
```

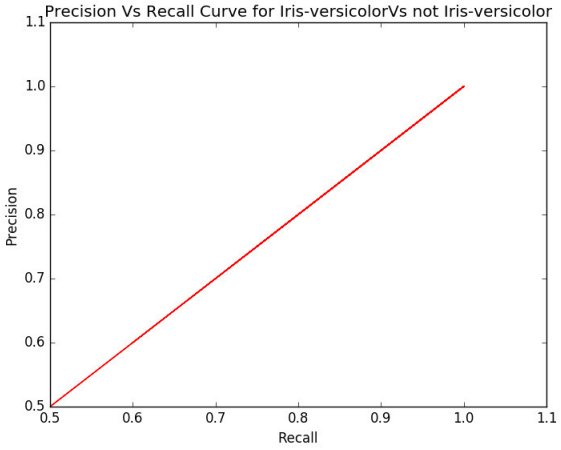
Here the validation data (25 percentage of test data) is used to calculate the tree error.

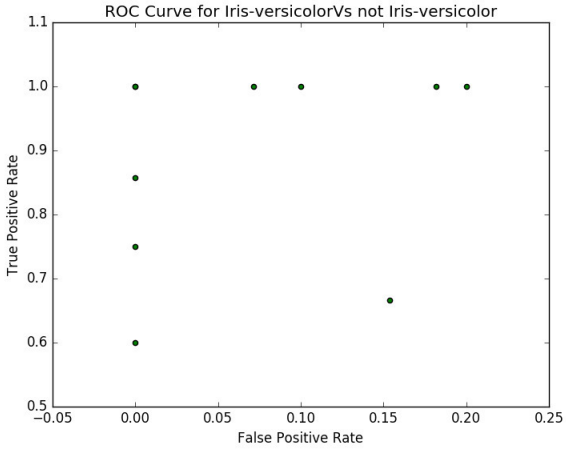
Results for performance evaluation

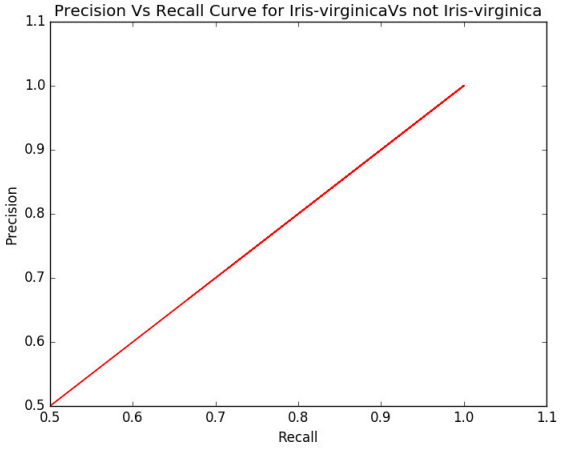
```
C:\Users\anirudhkm\Box Sync\education\spring_2016\data_mining\hw2\hw2\5\new>python main.py data/iris.data gini


Results for Iris-virginica Vs Non-Iris-virginica classifier
Simple accuracy: 0.942916666667
Balanced accuracy: 0.45836996337
F1 measure: 0.912388693178


Results for Iris-setosa Vs Non-Iris-setosa classifier
Simple accuracy: 0.99375
Balanced accuracy: 0.49
F1 measure: 0.98962962963


Results for Iris-versicolor Vs Non-Iris-versicolor classifier
Simple accuracy: 0.936666666667
Balanced accuracy: 0.488231074481
F1 measure: 0.904154271654

C:\Users\anirudhkm\Box Sync\education\spring_2016\data_mining\hw2\hw2\5\new>python main.py data/wine.csv gini


Results for 1 Vs Non-1 classifier
Simple accuracy: 0.95701754386
Balanced accuracy: 0.474874847375
F1 measure: 0.936811924459


Results for 2 Vs Non-2 classifier
Simple accuracy: 0.96783625731
Balanced accuracy: 0.524105339105
F1 measure: 0.977407203269


Results for 3 Vs Non-3 classifier
Simple accuracy: 0.925438596491
Balanced accuracy: 0.536517649018
F1 measure: 0.93923495667
```

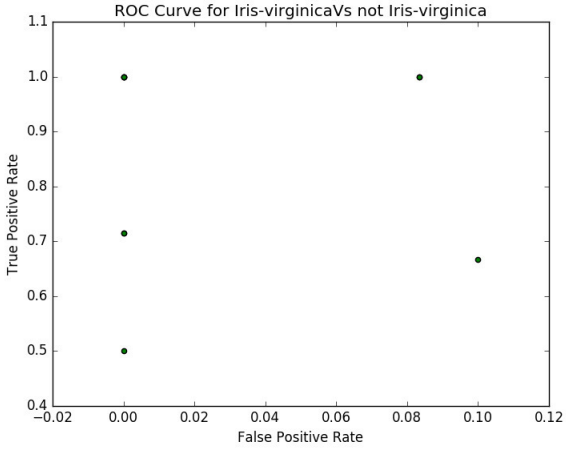

Precision Vs Recall Curve for Iris-setosaVs not Iris-setosa
Precision
Recall
1.00
0.95
0.90
0.85
0.80
0.80
0.85
0.90
0.95
1.00

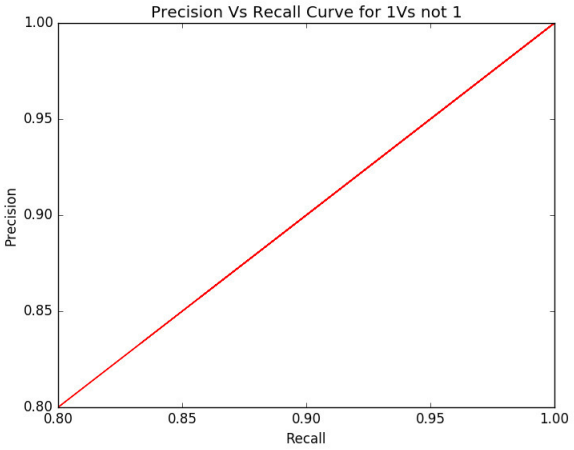

ROC Curve for Iris-setosaVs not Iris-setosa
True Positive Rate
False Positive Rate
1.05
1.00
0.95
0.90
0.85
0.80
0.75
−0.006
−0.004
−0.002
0.000
0.002
0.004
0.006

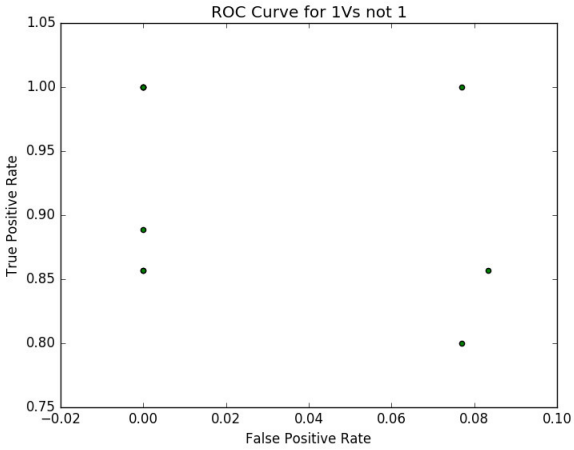

Precision Vs Recall Curve for Iris-versicolorVs not Iris-versicolor
Precision
Recall
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.5
0.6
0.7
0.8
0.9
1.0
1.1

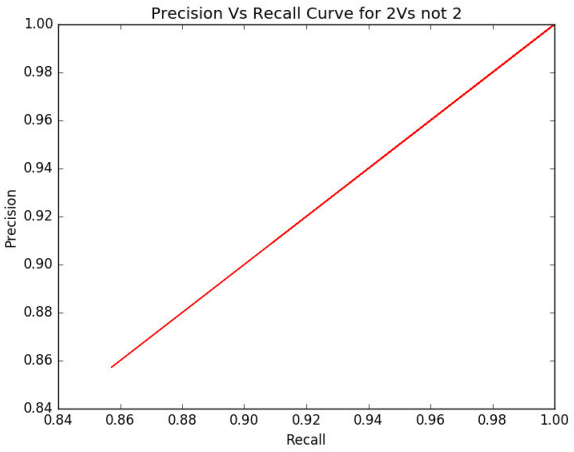

ROC Curve for Iris-versicolorVs not Iris-versicolor
True Positive Rate
False Positive Rate
1.1
1.0
0.9
0.8
0.7
0.6
0.5
−0.05
0.00
0.05
0.10
0.15
0.20
0.25

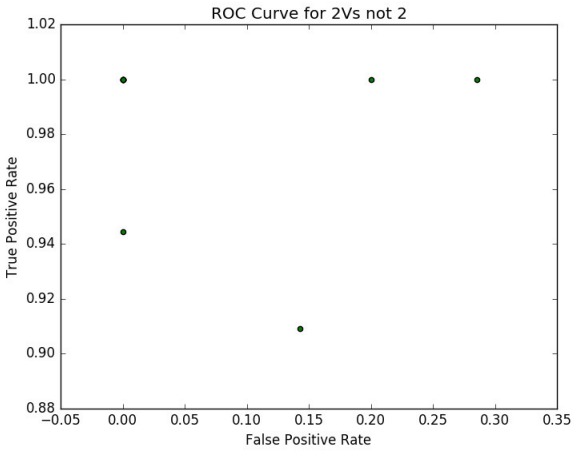

Precision Vs Recall Curve for Iris-virginicaVs not Iris-virginica
Precision
Recall
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.5
0.6
0.7
0.8
0.9
1.0
1.1

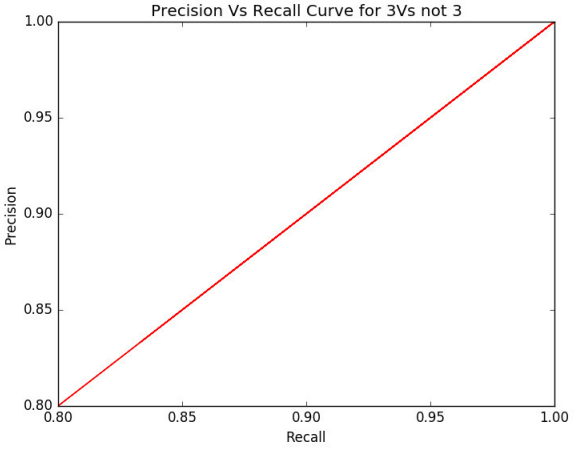

ROC Curve for Iris-virginicaVs not Iris-virginica
True Positive Rate
False Positive Rate
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
−0.02
0.00
0.02
0.04
0.06
0.08
0.10
0.12

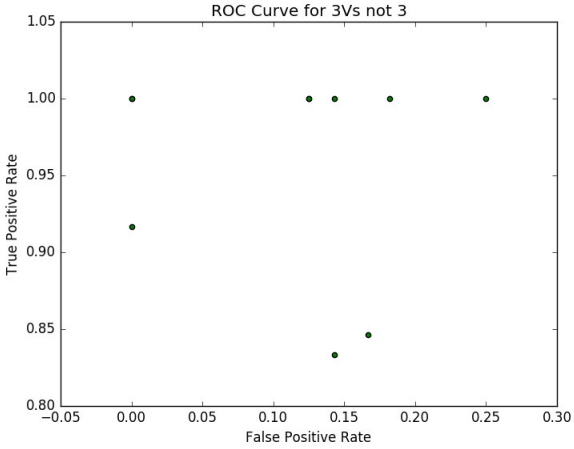












The graphs correspond to data Iris and wine.

6 Problem 6

Could complete only till the split, haven't achieved beyond that.

References

1. https://en.wikipedia.org/wiki/Receiver$_{o}perating_{c}haracteristic2.IntroductiontoDataminingPang - NingTan$