DMWA Lab – 5

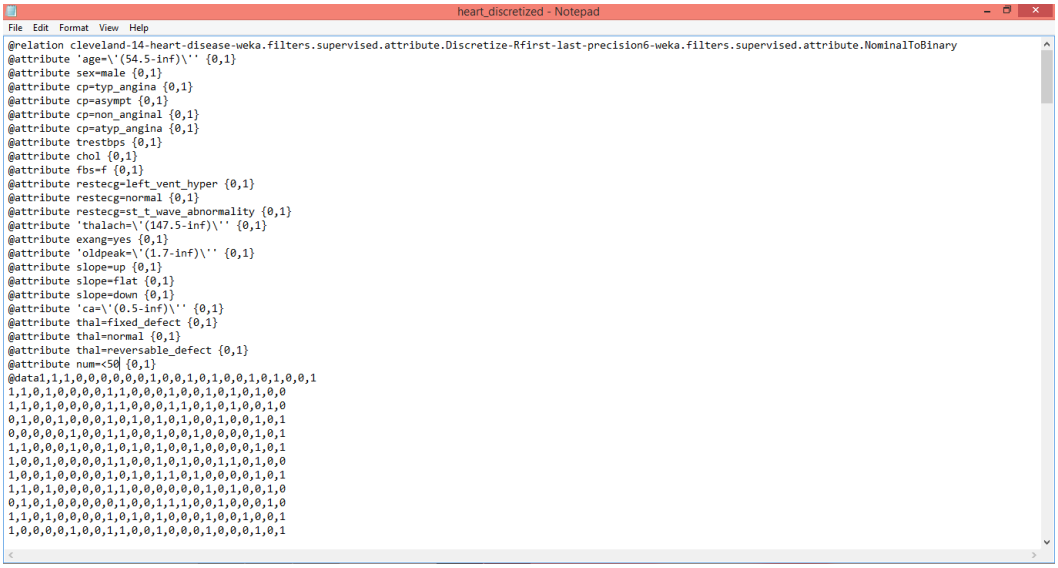
Teghdeep Kapoor

18104050

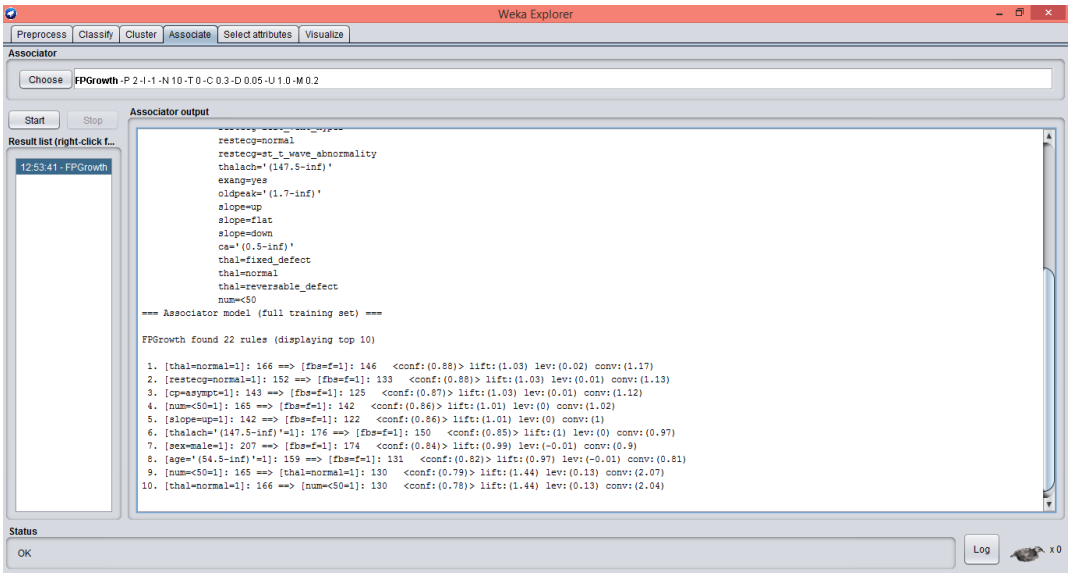
B12

Q1

heart_discretized.arff



FP_growth Rule in weka



Rules generated by FP_growth Weka

```
FPGrowth found 22 rules (displaying top 10)

 1. [thal=normal=1]: 166 ==> [fbs=f=1]: 146   <conf:(0.88)> lift:(1.03) lev:(0.02) conv:(1.17)
 2. [restecg=normal=1]: 152 ==> [fbs=f=1]: 133   <conf:(0.88)> lift:(1.03) lev:(0.01) conv:(1.13)
 3. [cp=asympt=1]: 143 ==> [fbs=f=1]: 125   <conf:(0.87)> lift:(1.03) lev:(0.01) conv:(1.12)
 4. [num=<50=1]: 165 ==> [fbs=f=1]: 142   <conf:(0.86)> lift:(1.01) lev:(0) conv:(1.02)
 5. [slope=up=1]: 142 ==> [fbs=f=1]: 122   <conf:(0.86)> lift:(1.01) lev:(0) conv:(1)
 6. [thalach='(147.5-inf)'=1]: 176 ==> [fbs=f=1]: 150   <conf:(0.85)> lift:(1) lev:(0) conv:(0.97)
 7. [sex=male=1]: 207 ==> [fbs=f=1]: 174   <conf:(0.84)> lift:(0.99) lev:(-0.01) conv:(0.9)
 8. [age='(54.5-inf)'=1]: 159 ==> [fbs=f=1]: 131   <conf:(0.82)> lift:(0.97) lev:(-0.01) conv:(0.81)
 9. [num=<50=1]: 165 ==> [thal=normal=1]: 130   <conf:(0.79)> lift:(1.44) lev:(0.13) conv:(2.07)
10. [thal=normal=1]: 166 ==> [num=<50=1]: 130   <conf:(0.78)> lift:(1.44) lev:(0.13) conv:(2.04)
```

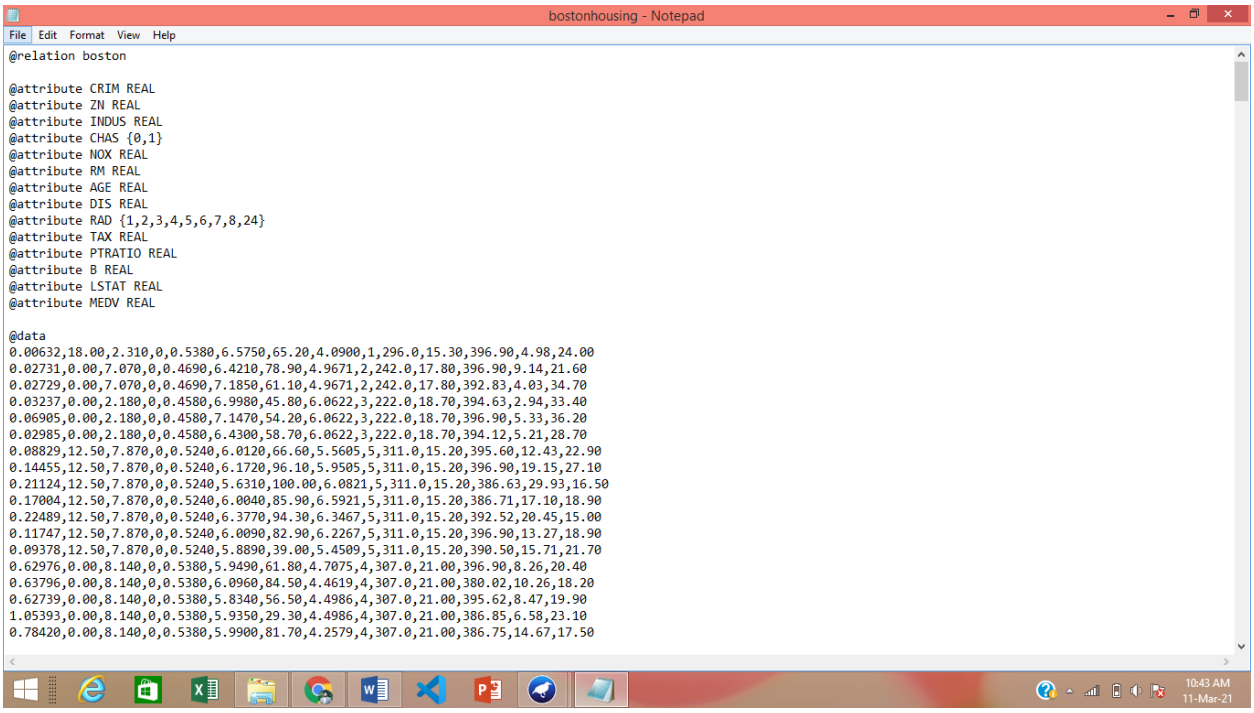
Q2

```
import pandas as pd
import numpy as np

import pyfpgrowth

df= pd.read_csv(“ transaction_data.csv”)

patterns = pyfpgrowth. find_frequent_patterns(transactions, 10)

rules = pyfpgrowth. generate_association_rules(patterns,0.8)


def support_count(rhs):

count=0

rhs= set(rhs)

for j in df[‘items’]:

j=set(j)

if(rhs.issubset(j)):

count=count+1

return count
```

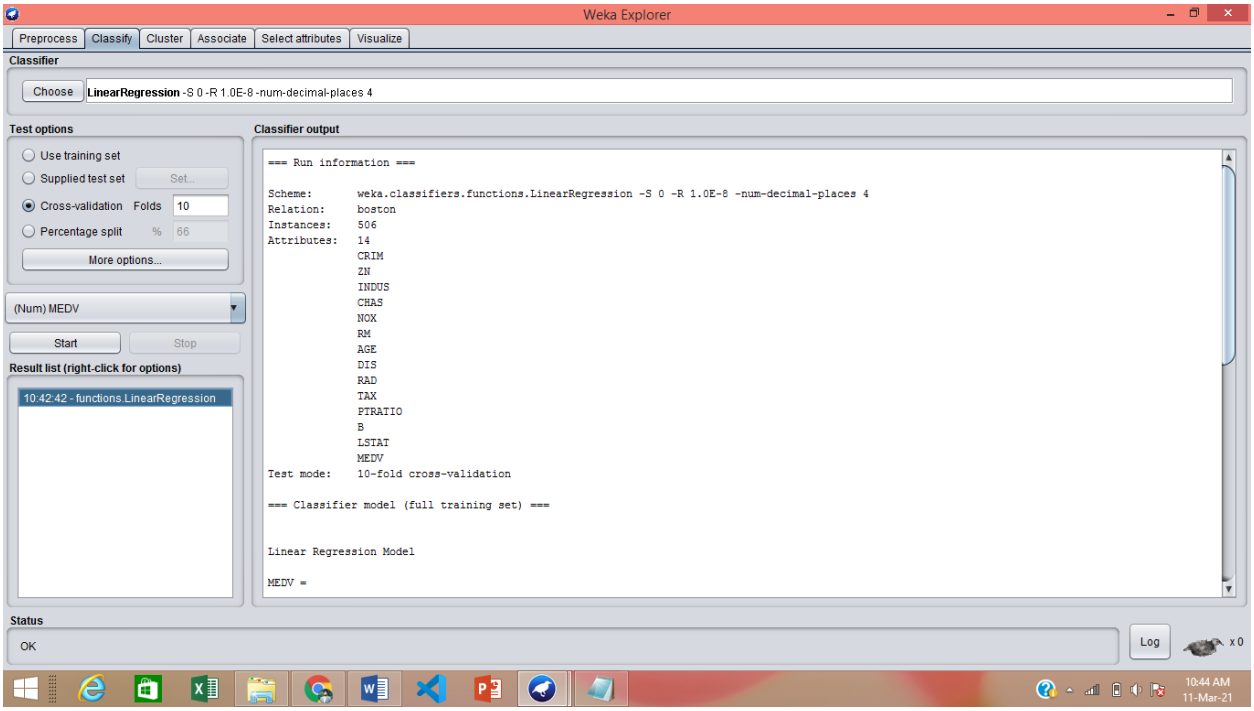
```
rhs_support = []
for I in rules_df['Consequent']:
a=support_count(i)
rhs_support.append(a/len(df))
rules_df['RHS_support'] = rhs_support
rules_df['lift'] = rules_df['Confidence']/rules_df['RHS_support']
rules_df['Conviction'] = (1-rules_df['RHS_support'])/(1-rules_df['Confidence'])
```

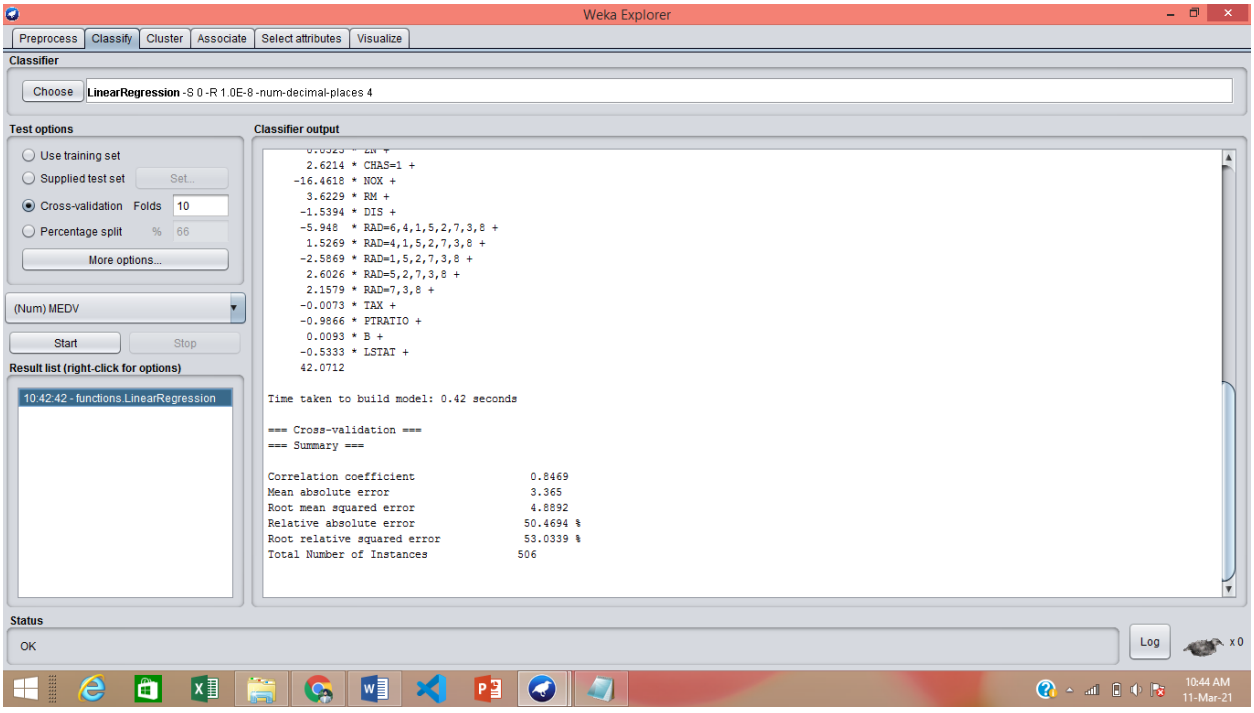
Q3

Bostonhousing.arff



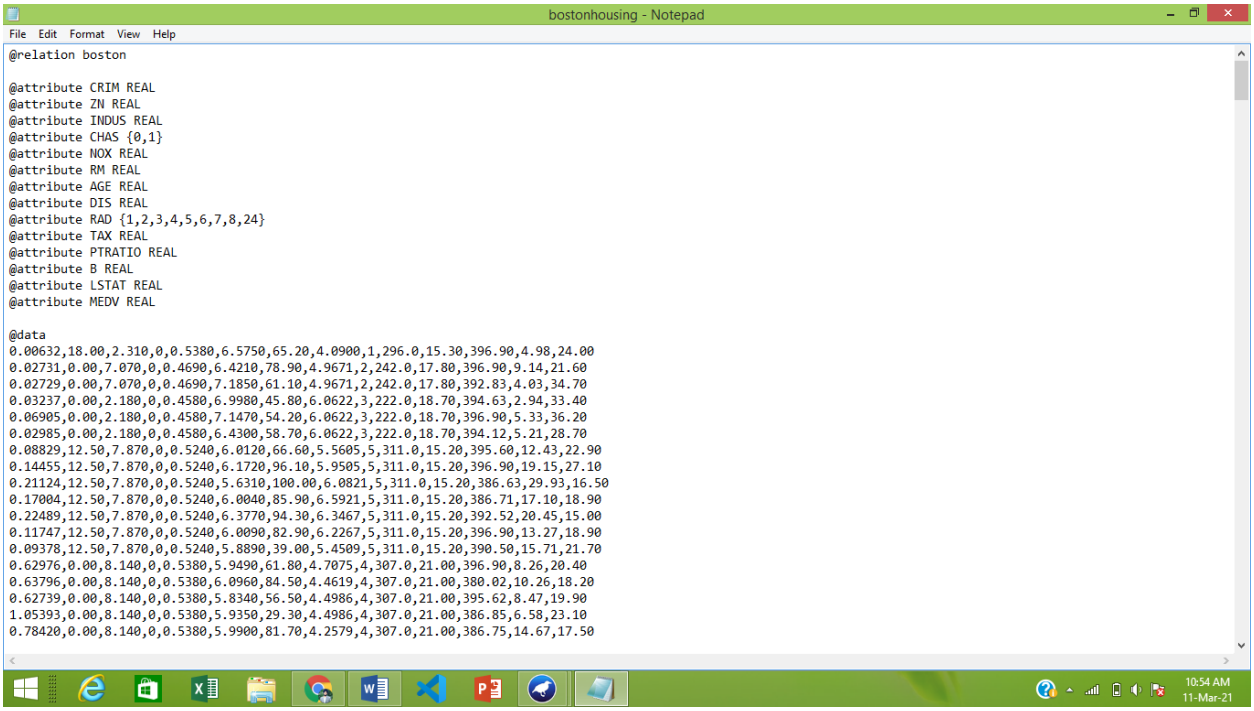
Weka Linear Regression



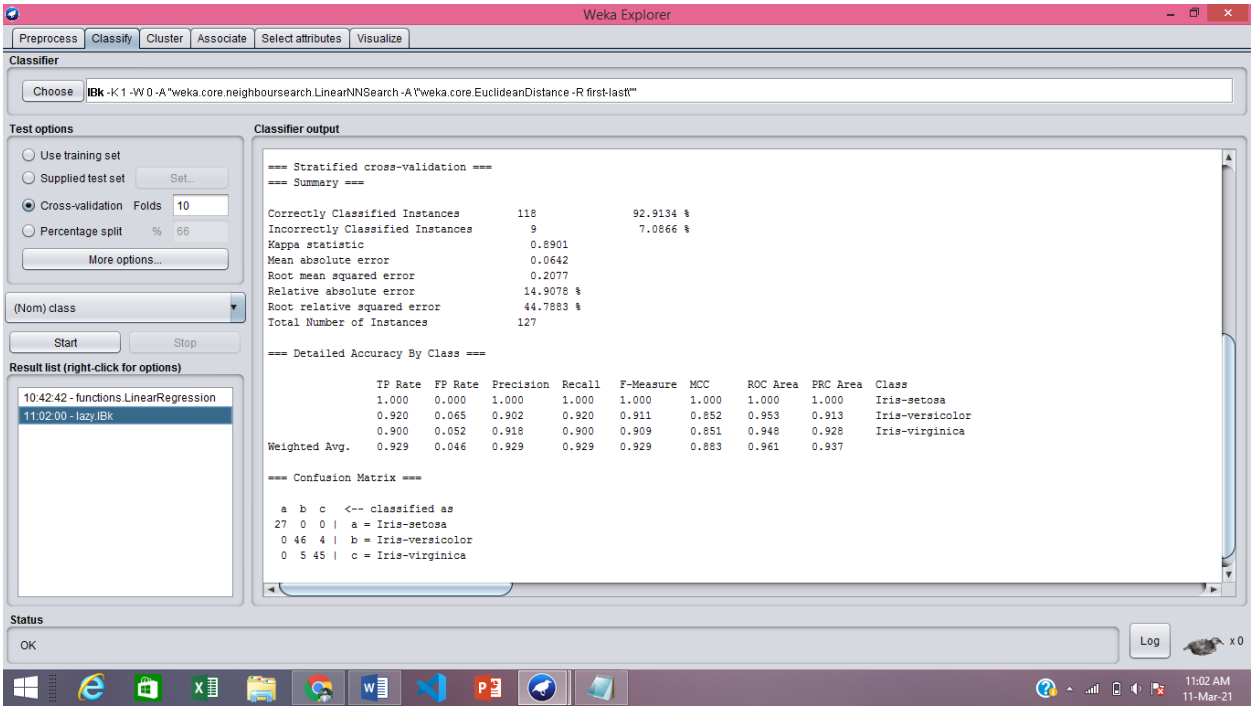


Q4

Iris.arff



KNN classifier using Weka



Confusion Matrix of Iris.arff using KNN

```
Correctly Classified Instances         118               92.9134 %
Incorrectly Classified Instances         9                7.0866 %
Kappa statistic                          0.8901
Mean absolute error                      0.0642
Root mean squared error                  0.2077
Relative absolute error                 14.9078 %
Root relative squared error             44.7883 %
Total Number of Instances              127

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Iris-setosa
                 0.920    0.065    0.902      0.920    0.911      0.852    0.953     0.913     Iris-versicolor
                 0.900    0.052    0.918      0.900    0.909      0.851    0.948     0.928     Iris-virginica
Weighted Avg.    0.929    0.046    0.929      0.929    0.929      0.883    0.961     0.937

=== Confusion Matrix ===

  a  b  c   <-- classified as
 27  0  0 |  a = Iris-setosa
  0 46  4 |  b = Iris-versicolor
  0  5 45 |  c = Iris-virginica
```

Code Of KNN using python-:

```
from math import sqrt


def euclidean_distance(row1, row2):
distance = 0.0
for i in range(len(row1)-1):
distance += (row1[i] - row2[i])**2
return sqrt(distance)

def get_neighbors(train, test_row, num_neighbors):
distances = list()
for train_row in train:
dist = euclidean_distance(test_row, train_row)
distances.append((train_row, dist))
distances.sort(key=lambda tup: tup[1])
neighbors = list()
for i in range(num_neighbors):
neighbors.append(distances[i][0])
return neighbors
```

```
def predict_classification(train, test_row, num_neighbors):
neighbors = get_neighbors(train, test_row, num_neighbors)
output_values = [row[-1] for row in neighbors]
prediction = max(set(output_values), key=output_values.count)
return prediction

dataset = [....]
prediction = predict_classification(dataset, dataset[0], 3)
print('Expected %d, Got %d.' % (dataset[0][-1], prediction))
```