DM Assignment

April 20, 2020

1 1. DATA MINING ASSIGNMENT:python,pandas,jupyter notebook

https://www.kaggle.com/arshid/iris-flower-dataset

[1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
```

[2]:
```
data = pd.read_csv('Iris.csv')
```

2 Ques 2.1 How many attributes are there in the dataset? How many are Nominal, Ordinal and Numeric?

[3]:
```
data.columns
```

[3]:
```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
```

3 Ans: 2.1 there are 5 attributes and 1 id attributes in my Dataset (Iris-Flower-Dataset) 4 numeric and 1 nomial.

4 Ques 2.2 How many records in your dataset?

[4]:
```
data.shape
```

[4]:
```
(150, 6)
```

5 Ans :2.2 150*6 total records in my data set.

6 Ques 2.3 Are there any missing values? If yes, provide details like how many such records with missing values, which attribute has more number of missing values?

[5]:
```
data.isnull().sum()
```

[5]:
```
Id               0
SepalLengthCm    4
SepalWidthCm     1
PetalLengthCm    3
PetalWidthCm     1
Species          0
dtype: int64
```

7 Ans: 2.3 there is 9 missing values in my dataset..

8 Ques 2.4 Did you apply any data cleaning process in the dataset to improve its quality? Justify your answer.

[6]:
```
data.fillna(data.mean(),inplace=True)
data
```

[6]:
```
       Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  \
0       1       5.867123           3.5            1.4      0.200000
1       2       4.900000           3.0            1.4      0.200000
2       3       4.700000           3.2            1.3      1.205369
3       4       5.867123           3.1            1.5      0.200000
4       5       5.000000           3.6            1.4      0.200000
..    ...            ...           ...            ...           ...
145   146       6.700000           3.0            5.2      2.300000
146   147       6.300000           2.5            5.0      1.900000
147   148       6.500000           3.0            5.2      2.000000
148   149       6.200000           3.4            5.4      2.300000
149   150       5.900000           3.0            5.1      1.800000

            Species
0       Iris-setosa
1       Iris-setosa
2       Iris-setosa
3       Iris-setosa
4       Iris-setosa
..              ...
145  Iris-virginica
```

```
146  Iris-virginica
147  Iris-virginica
148  Iris-virginica
149  Iris-virginica

[150 rows x 6 columns]
```

[7]:
```
data.isnull().sum() # you can see after data cleaning there is no missing value
```

[7]:
```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

9 Ques 2.5 Give your problem statement (describe what is your aim in implementing a data mining model for this dataset).

10 Ans 2.5 Given Sepal and Petal lengths and width predict the class of Iris.The dataset contains a set of 150 records under 5 attributes - Petal Length, Petal Width, Sepal Length, Sepal width and Class(Species).The data set consists of 50 samples from each of three species of Iris. Four features were measured from each sample: the length and the width of the sepals and petals, in centimeters.The aim is to classify iris flowers among three species (setosa, versicolor or virginica).

11 Ques 3. Divide your dataset into train and test set using the 80:20 method. Apply Decision tree and Naïve Bayesian classification algorithms on the dataset.

12 divide data set in 80:20

[8]:
```
y =  data.Species
y
```

[8]:
```
0         Iris-setosa
1         Iris-setosa
2         Iris-setosa
3         Iris-setosa
4         Iris-setosa
```

```
            …
145    Iris-virginica
146    Iris-virginica
147    Iris-virginica
148    Iris-virginica
149    Iris-virginica
Name: Species, Length: 150, dtype: object
```

[9]:
```
x=data.drop('Species',axis=1)
x
```

[9]:
```
       Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
0       1       5.867123           3.5            1.4      0.200000
1       2       4.900000           3.0            1.4      0.200000
2       3       4.700000           3.2            1.3      1.205369
3       4       5.867123           3.1            1.5      0.200000
4       5       5.000000           3.6            1.4      0.200000
..    …             …            …              …            …
145   146       6.700000           3.0            5.2      2.300000
146   147       6.300000           2.5            5.0      1.900000
147   148       6.500000           3.0            5.2      2.000000
148   149       6.200000           3.4            5.4      2.300000
149   150       5.900000           3.0            5.1      1.800000

[150 rows x 5 columns]
```

[10]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.
↪2,random_state=42)
```

[11]:
```
x_train
```

[11]:
```
       Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
22     23       4.600000           3.6       1.000000           0.2
15     16       5.700000           4.4       1.500000           0.4
65     66       6.700000           3.1       4.400000           1.4
11     12       5.867123           3.4       3.808844           0.2
42     43       4.400000           3.2       1.300000           0.2
..    …             …            …              …            …
71     72       6.100000           2.8       4.000000           1.3
106   107       4.900000           2.5       4.500000           1.7
14     15       5.800000           4.0       3.808844           0.2
92     93       5.800000           2.6       4.000000           1.2
102   103       7.100000           3.0       5.900000           2.1

[120 rows x 5 columns]
```

[12]:
```
x_test
```

[12]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
73	74	6.1	2.8	4.7	1.2
18	19	5.7	3.8	1.7	0.3
118	119	7.7	2.6	6.9	2.3
78	79	6.0	2.9	4.5	1.5
76	77	6.8	2.8	4.8	1.4
31	32	5.4	3.4	1.5	0.4
64	65	5.6	2.9	3.6	1.3
141	142	6.9	3.1	5.1	2.3
68	69	6.2	2.2	4.5	1.5
82	83	5.8	2.7	3.9	1.2
110	111	6.5	3.2	5.1	2.0
12	13	4.8	3.0	1.4	0.1
36	37	5.5	3.5	1.3	0.2
9	10	4.9	3.1	1.5	0.1
19	20	5.1	3.8	1.5	0.3
56	57	6.3	3.3	4.7	1.6
104	105	6.5	3.0	5.8	2.2
69	70	5.6	2.5	3.9	1.1
55	56	5.7	2.8	4.5	1.3
132	133	6.4	2.8	5.6	2.2
29	30	4.7	3.2	1.6	0.2
127	128	6.1	3.0	4.9	1.8
26	27	5.0	3.4	1.6	0.4
128	129	6.4	2.8	5.6	2.1
131	132	7.9	3.8	6.4	2.0
145	146	6.7	3.0	5.2	2.3
108	109	6.7	2.5	5.8	1.8
143	144	6.8	3.2	5.9	2.3
45	46	4.8	3.0	1.4	0.3
30	31	4.8	3.1	1.6	0.2

[13]:
```
y_train
```

[13]:
```
22         Iris-setosa
15         Iris-setosa
65     Iris-versicolor
11         Iris-setosa
42         Iris-setosa
              …
71     Iris-versicolor
106     Iris-virginica
14         Iris-setosa
92     Iris-versicolor
102     Iris-virginica
Name: Species, Length: 120, dtype: object
```

[14]:
```
y_test
```

[14]:
```
73     Iris-versicolor
18         Iris-setosa
118     Iris-virginica
78     Iris-versicolor
76     Iris-versicolor
31         Iris-setosa
64     Iris-versicolor
141     Iris-virginica
68     Iris-versicolor
82     Iris-versicolor
110     Iris-virginica
12         Iris-setosa
36         Iris-setosa
9          Iris-setosa
19         Iris-setosa
56     Iris-versicolor
104     Iris-virginica
69     Iris-versicolor
55     Iris-versicolor
132     Iris-virginica
29         Iris-setosa
127     Iris-virginica
26         Iris-setosa
128     Iris-virginica
131     Iris-virginica
145     Iris-virginica
108     Iris-virginica
143     Iris-virginica
45         Iris-setosa
30         Iris-setosa
Name: Species, dtype: object
```

13 Ques 3.1. Report the classification Accuracy, Error rate, Sensitivity, and Specificity. Display the confusion matrix.

14 Decision Tree

[15]:
```
dtree = DecisionTreeClassifier()
x,y = data.drop(['Id','Species'],axis=1), data['Species']
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.
 ↪3,random_state = 1)
dtree.fit(x_train,y_train)
prediction = dtree.predict(x_test)
cnf_matrix_dec_tree = confusion_matrix(y_test, prediction)
```

```
print('With Decision tree accuracy is: ',dtree.score(x_test,y_test))
print('Error rate: ',(1-dtree.score(x_test,y_test)))
print('Sensitivity:  When the actual value is positive, how often is the␣
 ↪prediction correct? ')
print('Also known as True Positive Rate or Recall TP / all positive all␣
 ↪positive = TP + FN')
print('\n')
print('Specificity: When the actual value is negative, how often is the␣
 ↪prediction correct?')
print("How 'specific' (or 'selective') is the classifier in predicting positive␣
 ↪instances? TN / all negative all negative = TN + FP")
print('\n')
print('Confusion matrix')
print('*'*20)
print(cnf_matrix_dec_tree)
print('*'*20)
```

```
With Decision tree accuracy is:  0.9555555555555556
Error rate:  0.0444444444444444
Sensitivity:  When the actual value is positive, how often is the prediction
correct?
Also known as True Positive Rate or Recall TP / all positive all positive = TP +
FN


Specificity: When the actual value is negative, how often is the prediction
correct?
How 'specific' (or 'selective') is the classifier in predicting positive
instances? TN / all negative all negative = TN + FP


Confusion matrix
********************
[[14  0  0]
 [ 0 17  1]
 [ 0  1 12]]
********************
```

[16]:
```
TP = cnf_matrix_dec_tree[1, 1]
TN = cnf_matrix_dec_tree[0, 0]
FP = cnf_matrix_dec_tree[0, 1]
FN = cnf_matrix_dec_tree[1, 0]
```

[17]:
```
sensitivity = TP / float(FN + TP)
print('sensitivity: =',sensitivity)
```

```
sensitivity: = 1.0
```

[18]:
```
specificity = TN / (TN + FP)
print('specificity: =',specificity)
```

```
specificity: = 1.0
```

15 Naive Bayes

[19]:
```
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
```

[20]:
```
nb = GaussianNB()
x,y = data.drop(['Id','Species'],axis=1), data['Species']
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.
 ↪2,random_state = 1)
nb.fit(x_train,y_train)
prediction = nb.predict(x_test)
cnf_matrix_gnb = confusion_matrix(y_test, prediction)

print('With NB accuracy is: ',nb.score(x_test,y_test))
print('Error rate: ',(1-nb.score(x_test,y_test)))
print('*'*20)
print(cnf_matrix_gnb)
print('*'*20)
```

```
With NB accuracy is:  0.9666666666666667
Error rate:  0.033333333333333326
********************
[[11  0  0]
 [ 0 12  1]
 [ 0  0  6]]
********************
```

[21]:
```
TP = cnf_matrix_gnb[1, 1]
TN = cnf_matrix_gnb[0, 0]
FP = cnf_matrix_gnb[0, 1]
FN = cnf_matrix_gnb[1, 0]
```

[22]:
```
sensitivity = TP / float(FN + TP)
print('sensitivity: =',sensitivity)
```

```
sensitivity: = 1.0
```

[23]:
```
specificity = TN / (TN + FP)
print('specificity: =',specificity)
```

```
specificity: = 1.0
```

16 3.2 Ans: No, iris dataset is balanced,because i have 3 types of data and total rows= 150(each type has 50 data points)

17 3.3 Ans: Decision Tree alogorithm is better than naive bayes algorithm..

18 Ques 4: For decision tree algorithm, try with three different attribute selection methods and report which performs better in your problem. after the modification its only based on criterion (entropy,gini).

[24]:
```
entropy_dtree = DecisionTreeClassifier(criterion="entropy")
entropy_x,entropy_y = data.drop(['Id','Species'],axis=1), data['Species']
x_train_entropy,x_test_entropy,y_train_entropy,y_test_entropy =␣
 ↪train_test_split(entropy_x,entropy_y,test_size = 0.2,random_state = 45)
elf = entropy_dtree.fit(x_train_entropy,y_train_entropy)
prediction = entropy_dtree.predict(x_test_entropy)
cnf_matrix_dec_tree_entropy = confusion_matrix(y_test_entropy, prediction)
print('With Decision tree accuracy is: ',entropy_dtree.
 ↪score(x_test_entropy,y_test_entropy))
print('Error rate: ',(1-entropy_dtree.score(x_test_entropy,y_test_entropy)))
```

```
With Decision tree accuracy is:  0.9333333333333333
Error rate:  0.06666666666666665
```

[25]:
```
gini_dtree = DecisionTreeClassifier(criterion="gini")
x_gini,y_gini = data.drop(['Id','Species'],axis=1), data['Species']
x_train_gini,x_test_gini,y_train_gini,y_test_gini =␣
 ↪train_test_split(x_gini,y_gini,test_size = 0.2,random_state = 45)
glf = gini_dtree.fit(x_train_gini,y_train_gini)
prediction = gini_dtree.predict(x_test_gini)
cnf_matrix_dec_tree_gini = confusion_matrix(y_test_gini, prediction)
print('With Decision tree accuracy is: ',gini_dtree.
 ↪score(x_test_gini,y_test_gini))
print('Error rate: ',(1-gini_dtree.score(x_test_gini,y_test_gini)))
```

```
With Decision tree accuracy is:  1.0
Error rate:  0.0
```

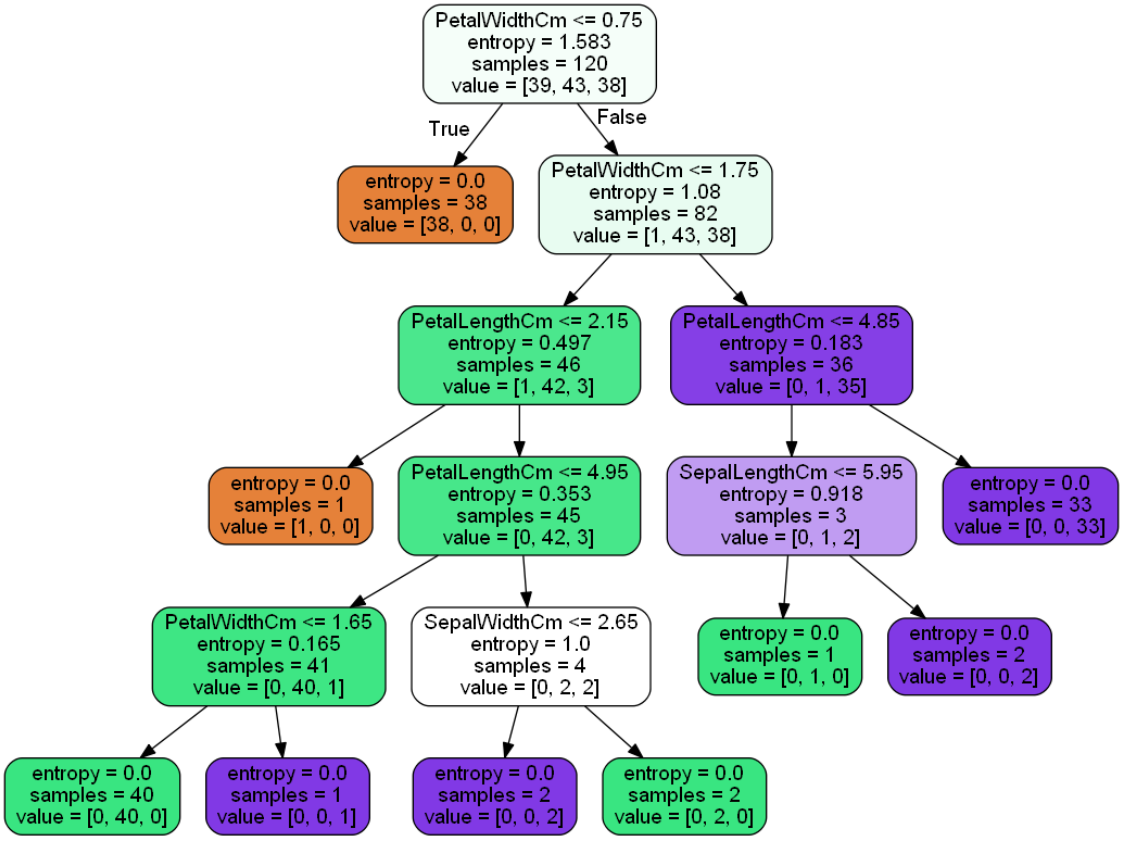
19 Ques 5: Plot the final tree obtained, with highest accuracy based on question 4, for your problem.

[38]:
```
from IPython.display import Image
from sklearn.externals.six import StringIO
from sklearn.tree import export_graphviz
import pydot

features = list(data.columns[1:5])
features
```

[38]:
```
['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']
```

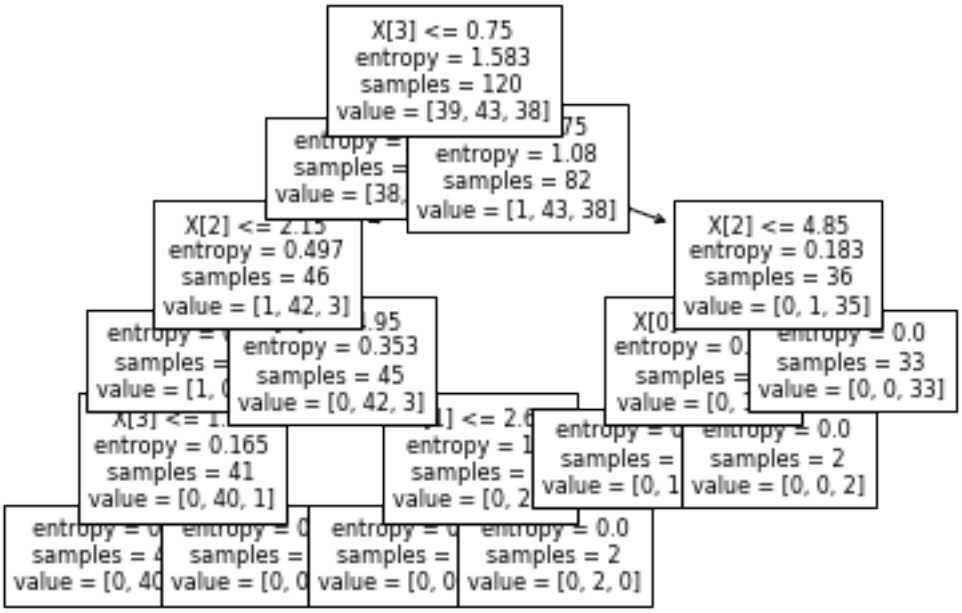
[39]:
```
dot_data = StringIO()
export_graphviz(elf,␣
 ↪out_file=dot_data,feature_names=features,filled=True,rounded=True)

graph = pydot.graph_from_dot_data(dot_data.getvalue())
Image(graph[0].create_png())
```

[39]:



[27]:
```
from sklearn import tree
tree.plot_tree(elf,fontsize=8)
```

[27]:
```
[Text(153.45000000000002, 199.32, 'X[3] <= 0.75\nentropy = 1.583\nsamples =
120\nvalue = [39, 43, 38]'),
 Text(125.55000000000001, 163.07999999999998, 'entropy = 0.0\nsamples =
38\nvalue = [38, 0, 0]'),
 Text(181.35000000000002, 163.07999999999998, 'X[3] <= 1.75\nentropy =
1.08\nsamples = 82\nvalue = [1, 43, 38]'),
 Text(83.7, 126.83999999999999, 'X[2] <= 2.15\nentropy = 0.497\nsamples =
46\nvalue = [1, 42, 3]'),
 Text(55.800000000000004, 90.6, 'entropy = 0.0\nsamples = 1\nvalue = [1, 0,
0]'),
 Text(111.60000000000001, 90.6, 'X[2] <= 4.95\nentropy = 0.353\nsamples =
45\nvalue = [0, 42, 3]'),
 Text(55.800000000000004, 54.359999999999985, 'X[3] <= 1.65\nentropy =
0.165\nsamples = 41\nvalue = [0, 40, 1]'),
 Text(27.900000000000002, 18.119999999999976, 'entropy = 0.0\nsamples =
40\nvalue = [0, 40, 0]'),
 Text(83.7, 18.119999999999976, 'entropy = 0.0\nsamples = 1\nvalue = [0, 0,
1]'),
 Text(167.4, 54.359999999999985, 'X[1] <= 2.65\nentropy = 1.0\nsamples =
4\nvalue = [0, 2, 2]'),
 Text(139.5, 18.119999999999976, 'entropy = 0.0\nsamples = 2\nvalue = [0, 0,
2]'),
 Text(195.3, 18.119999999999976, 'entropy = 0.0\nsamples = 2\nvalue = [0, 2,
0]'),
 Text(279.0, 126.83999999999999, 'X[2] <= 4.85\nentropy = 0.183\nsamples =
36\nvalue = [0, 1, 35]'),
 Text(251.10000000000002, 90.6, 'X[0] <= 5.95\nentropy = 0.918\nsamples =
3\nvalue = [0, 1, 2]'),
 Text(223.20000000000002, 54.359999999999985, 'entropy = 0.0\nsamples = 1\nvalue
= [0, 1, 0]'),
 Text(279.0, 54.359999999999985, 'entropy = 0.0\nsamples = 2\nvalue = [0, 0,
2]'),
 Text(306.90000000000003, 90.6, 'entropy = 0.0\nsamples = 33\nvalue = [0, 0,
33]')]
```



20 Ans 5: Entropy has highest Accuracy.above tree is final tree.