Name - Ravi kumar

Reg - 2020PGCACA72

Assign_7

In [ ]:

```
```

question 1 ¶

In [147]:

```
import numpy as np
import pandas as pd
df=pd.read_csv('spambase.csv')
df
```

Out[147]:

	0	0.64	0.64.1	0.1	0.32	0.2	0.3	0.4	0.5	0.6	...	0.40	0.41	0.42	0.778
0	0.21	0.28	0.50	0.0	0.14	0.28	0.21	0.07	0.00	0.94	...	0.000	0.132	0.0	0.372
1	0.06	0.00	0.71	0.0	1.23	0.19	0.19	0.12	0.64	0.25	...	0.010	0.143	0.0	0.276
2	0.00	0.00	0.00	0.0	0.63	0.00	0.31	0.63	0.31	0.63	...	0.000	0.137	0.0	0.137
3	0.00	0.00	0.00	0.0	0.63	0.00	0.31	0.63	0.31	0.63	...	0.000	0.135	0.0	0.135
4	0.00	0.00	0.00	0.0	1.85	0.00	0.00	1.85	0.00	0.00	...	0.000	0.223	0.0	0.000
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4595	0.31	0.00	0.62	0.0	0.00	0.31	0.00	0.00	0.00	0.00	...	0.000	0.232	0.0	0.000
4596	0.00	0.00	0.00	0.0	0.00	0.00	0.00	0.00	0.00	0.00	...	0.000	0.000	0.0	0.353
4597	0.30	0.00	0.30	0.0	0.00	0.00	0.00	0.00	0.00	0.00	...	0.102	0.718	0.0	0.000
4598	0.96	0.00	0.00	0.0	0.32	0.00	0.00	0.00	0.00	0.00	...	0.000	0.057	0.0	0.000
4599	0.00	0.00	0.65	0.0	0.00	0.00	0.00	0.00	0.00	0.00	...	0.000	0.000	0.0	0.125

4600 rows × 58 columns

In [21]:

```
df.head()
```

Out[21]:

	0	0.64	0.64.1	0.1	0.32	0.2	0.3	0.4	0.5	0.6	...	0.40	0.41	0.42	0.778	0.43
0	0.21	0.28	0.50	0.0	0.14	0.28	0.21	0.07	0.00	0.94	...	0.00	0.132	0.0	0.372	0.180
1	0.06	0.00	0.71	0.0	1.23	0.19	0.19	0.12	0.64	0.25	...	0.01	0.143	0.0	0.276	0.184
2	0.00	0.00	0.00	0.0	0.63	0.00	0.31	0.63	0.31	0.63	...	0.00	0.137	0.0	0.137	0.000
3	0.00	0.00	0.00	0.0	0.63	0.00	0.31	0.63	0.31	0.63	...	0.00	0.135	0.0	0.135	0.000
4	0.00	0.00	0.00	0.0	1.85	0.00	0.00	1.85	0.00	0.00	...	0.00	0.223	0.0	0.000	0.000

5 rows × 58 columns

In [22]:

```
df.tail()
```

Out[22]:

	0	0.64	0.64.1	0.1	0.32	0.2	0.3	0.4	0.5	0.6	...	0.40	0.41	0.42	0.778	0.43
4595	0.31	0.0	0.62	0.0	0.00	0.31	0.0	0.0	0.0	0.0	...	0.000	0.232	0.0	0.000	0.0
4596	0.00	0.0	0.00	0.0	0.00	0.00	0.0	0.0	0.0	0.0	...	0.000	0.000	0.0	0.353	0.0
4597	0.30	0.0	0.30	0.0	0.00	0.00	0.0	0.0	0.0	0.0	...	0.102	0.718	0.0	0.000	0.0
4598	0.96	0.0	0.00	0.0	0.32	0.00	0.0	0.0	0.0	0.0	...	0.000	0.057	0.0	0.000	0.0
4599	0.00	0.0	0.65	0.0	0.00	0.00	0.0	0.0	0.0	0.0	...	0.000	0.000	0.0	0.125	0.0

5 rows × 58 columns

In [23]:

```
df.shape
```

Out[23]:

```
(4600, 58)
```

In [24]:

```
[row, col] = df.shape
Data = df.iloc[0 : row, 0 : (col - 1)]
Label = df.iloc[0 : row, (col - 1)]
Data
```

Out[24]:

	0	0.64	0.64.1	0.1	0.32	0.2	0.3	0.4	0.5	0.6	...	0.39	0.40	0.41	0.42	0
0	0.21	0.28	0.50	0.0	0.14	0.28	0.21	0.07	0.00	0.94	...	0.0	0.000	0.132	0.0	0
1	0.06	0.00	0.71	0.0	1.23	0.19	0.19	0.12	0.64	0.25	...	0.0	0.010	0.143	0.0	0
2	0.00	0.00	0.00	0.0	0.63	0.00	0.31	0.63	0.31	0.63	...	0.0	0.000	0.137	0.0	0
3	0.00	0.00	0.00	0.0	0.63	0.00	0.31	0.63	0.31	0.63	...	0.0	0.000	0.135	0.0	0
4	0.00	0.00	0.00	0.0	1.85	0.00	0.00	1.85	0.00	0.00	...	0.0	0.000	0.223	0.0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
4595	0.31	0.00	0.62	0.0	0.00	0.31	0.00	0.00	0.00	0.00	...	0.0	0.000	0.232	0.0	0
4596	0.00	0.00	0.00	0.0	0.00	0.00	0.00	0.00	0.00	0.00	...	0.0	0.000	0.000	0.0	0
4597	0.30	0.00	0.30	0.0	0.00	0.00	0.00	0.00	0.00	0.00	...	0.0	0.102	0.718	0.0	0
4598	0.96	0.00	0.00	0.0	0.32	0.00	0.00	0.00	0.00	0.00	...	0.0	0.000	0.057	0.0	0
4599	0.00	0.00	0.65	0.0	0.00	0.00	0.00	0.00	0.00	0.00	...	0.0	0.000	0.000	0.0	0

4600 rows × 57 columns

In [25]:

```
Label
```

Out[25]:

```
0       1
1       1
2       1
3       1
4       1
       ..
4595    0
4596    0
4597    0
4598    0
4599    0
Name: 1, Length: 4600, dtype: int64
```

In [26]:

```
Label.value_counts()
```

Out[26]:

```
0    2788
1    1812
Name: 1, dtype: int64
```

knn algorithm

In [27]:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
import numpy as np
X = Data
y = Label
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)
knn = KNeighborsClassifier(n_neighbors = 4)
knn.fit(X_train, y_train)
print(knn.score(X_test, y_test))
```

```
0.7869565217391304
```

In [ ]:

```
,
```

In [151]:

```
knn = KNeighborsClassifier(n_neighbors = 2)
knn.fit(X_train, y_train)
print(knn.score(X_test, y_test))
```

```
0.7956521739130434
```

In [29]:

```
knn = KNeighborsClassifier(n_neighbors = 5)
knn.fit(X_train, y_train)
print(knn.score(X_test, y_test))
```

```
0.7923913043478261
```

In [30]:

```
knn = KNeighborsClassifier(n_neighbors = 10)
knn.fit(X_train, y_train)
print(knn.score(X_test, y_test))
```

```
0.7793478260869565
```

In [31]:

```
knn = KNeighborsClassifier(n_neighbors = 20)
knn.fit(X_train, y_train)
print(knn.score(X_test, y_test))
```

```
0.7565217391304347
```

In [ ]:

```

```

In [32]:

```
knn = KNeighborsClassifier(n_neighbors = 2)
knn.fit(X_train, y_train)
print(knn.score(X_test, y_test))
```

```
0.7902173913043479
```

precision in Knn

In [152]:

```
#precision in knn
from sklearn.metrics import precision_score
y_pred=k_means.predict(X_test)
precision_score(y_test,y_pred,average=None,zero_division=1)[0]
```

Out[152]:

```
0.8662790697674418
```

Recall in knn

In [153]:

```
#Recall_score in knn

from sklearn.metrics import recall_score
recall_score(y_test,y_pred,average='macro',zero_division=1)
```

Out[153]:

```
0.8976099945681695
```

F1_score

In [140]:

```
#F1_score

from sklearn.metrics import f1_score
f1_score(y_test,k_means.predict(X_test),average='weighted')
```

Out[140]:

```
0.3916741629185408
```

In [ ]:

In [ ]:

In [ ]:

In [142]:

```
###Decision_Tree
```

In [ ]:

In [101]:

```
#Decision Tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import precision_score, recall_score, f1_score, accuracy_score
X_train, X_test, y_train, y_test = train_test_split(Data, Label, test_size = 0.1, random_st
dt = DecisionTreeClassifier(random_state = 0, max_depth = 2)
dt.fit(X_train, y_train)
y_pred = dt.predict(X_test)
```

In [102]:

```
print('Accuracy: %.4f' % accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.8435
```

In [89]:

```
from sklearn.metrics import precision_score
precision_score(y_test,k_means.predict(X_test),average=None)[0]
```

Out[89]:

```
0.4838709677419355
```

In [98]:

```
from sklearn.metrics import precision_score
precision_score(y_test,k_means.predict(X_test),average='macro')
```

Out[98]:

```
0.17885829030407344
```

In [97]:

```
from sklearn.metrics import precision_score
precision_score(y_test,k_means.predict(X_test),average='micro')
```

Out[97]:

```
0.4891304347826087
```

In [93]:

```
from sklearn.metrics import precision_score
precision_score(y_test,k_means.predict(X_test),average='weighted')
```

Out[93]:

```
0.6785413744740533
```

In [143]:

```
#precision in Decision Tree
from sklearn.metrics import precision_score
y_pred=k_means.predict(X_test)
precision_score(y_test,y_pred,average=None,zero_division=1)[0]
```

Out[143]:

```
0.8662790697674418
```

In [144]:

```
#Recall_score in Decision Tree

from sklearn.metrics import recall_score
recall_score(y_test,y_pred,average='macro',zero_division=1)
```

Out[144]:

```
0.8976099945681695
```

In [145]:

```
#F1_score in Decision Tree

from sklearn.metrics import f1_score
f1_score(y_test,k_means.predict(X_test),average='weighted')
```

Out[145]:

```
0.3916741629185408
```

In [ ]:

```

```

In [ ]:

```

```

In [146]:

```
                        #kmeans algorithm
```

In [73]:

```
from sklearn.model_selection import train_test_split
from sklearn.cluster import KMeans as k_means
from sklearn import cluster
X = Data
y = Label
X_train, X_test,y_train,y_test =train_test_split(X,y,test_size=0.20,random_state=70)
k_means = cluster.KMeans(n_clusters=2)
k_means.fit(X_train)
#print(k_means.labels_[:])
#print(y_train[:])

score = accuracy_score(y_test,k_means.predict(X_test))
print(score)
```

```
0.6489130434782608
```

In [76]:

```
k_means = cluster.KMeans(n_clusters=5)
k_means.fit(X_train)
#print(k_means.labels_[:])
#print(y_train[:])

score = accuracy_score(y_test,k_means.predict(X_test))
print(score)
```

```
0.6010869565217392
```

In [96]:

```
k_means = cluster.KMeans(n_clusters=10)
k_means.fit(X_train)
#print(k_means.labels_[:])
#print(y_train[:])

score = accuracy_score(y_test,k_means.predict(X_test))
print(score)
```

```
0.4891304347826087
```

In [113]:

```
k_means = cluster.KMeans(n_clusters=20)
k_means.fit(X_train)
#print(k_means.labels_[:])
#print(y_train[:])

score = accuracy_score(y_test,k_means.predict(X_test))
print(score)
```

```
0.3239130434782609
```

In [ ]:

```

```

In [88]:

```
from sklearn.metrics import precision_score
precision_score(y_test,k_means.predict(X_test),average=None)[0]
```

Out[88]:

```
0.4838709677419355
```

In [86]:

```
from sklearn.metrics import precision_score
precision_score(y_test,k_means.predict(X_test),average='macro')
```

Out[86]:

```
0.0872865275142315
```

In [85]:

```
from sklearn.metrics import precision_score
precision_score(y_test,k_means.predict(X_test),average='micro')
```

Out[85]:

```
0.03695652173913044
```

In [84]:

```
from sklearn.metrics import precision_score
precision_score(y_test,k_means.predict(X_test),average='weighted')
```

Out[84]:

```
0.6785413744740533
```

In [105]:

```
#Recall_score

from sklearn.metrics import recall_score
recall_score(y_test,y_pred,average='macro')
```

Out[105]:

```
0.8255389782092606
```

In [106]:

```
#F1_score

from sklearn.metrics import f1_score
f1_score(y_test,k_means.predict(X_test),average='weighted')
```

Out[106]:

```
0.4203055229142185
```

In [ ]:

In [ ]:

In [ ]: