In [1]:

```
import numpy as np
import pandas as pd
from sklearn import tree
from sklearn import datasets
from sklearn.tree import export_text
from sklearn.naive_bayes import GaussianNB
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import load_iris, load_breast_cancer
from sklearn.metrics import accuracy_score, classification_report
from sklearn.model_selection import train_test_split, cross_val_score
```

Iris

In [2]:

```
X, y = load_iris(return_X_y=True)
print(X)
y
```

```
 [4.5 2.3 1.3 0.3]
 [4.4 3.2 1.3 0.2]
 [5.  3.5 1.6 0.6]
 [5.1 3.8 1.9 0.4]
 [4.8 3.  1.4 0.3]
 [5.1 3.8 1.6 0.2]
 [4.6 3.2 1.4 0.2]
 [5.3 3.7 1.5 0.2]
 [5.  3.3 1.4 0.2]
 [7.  3.2 4.7 1.4]
 [6.4 3.2 4.5 1.5]
 [6.9 3.1 4.9 1.5]
 [5.5 2.3 4.  1.3]
 [6.5 2.8 4.6 1.5]
 [5.7 2.8 4.5 1.3]
 [6.3 3.3 4.7 1.6]
 [4.9 2.4 3.3 1. ]
 [6.6 2.9 4.6 1.3]
 [5.2 2.7 3.9 1.4]
 [5.  2.  3.5 1. ]
```

In [3]:

```
clf = tree.DecisionTreeClassifier()
clf = clf.fit(X,y)
```

In [4]:

```
iris = load_iris()
decisionTree = tree.DecisionTreeClassifier(random_state=0)
#decisionTree = tree.DecisionTreeClassifier(max_depth=3, random_state=0)
decisionTree = decisionTree.fit(iris.data,iris.target)
treeP = export_text(decisionTree, feature_names=iris['feature_names'])
print(treeP)
```

```
|--- petal width (cm) <= 0.80
|   |--- class: 0
|--- petal width (cm) >  0.80
|   |--- petal width (cm) <= 1.75
|   |   |--- petal length (cm) <= 4.95
|   |   |   |--- petal width (cm) <= 1.65
|   |   |   |   |--- class: 1
|   |   |   |--- petal width (cm) >  1.65
|   |   |   |   |--- class: 2
|   |   |--- petal length (cm) >  4.95
|   |   |   |--- petal width (cm) <= 1.55
|   |   |   |   |--- class: 2
|   |   |   |--- petal width (cm) >  1.55
|   |   |   |   |--- petal length (cm) <= 5.45
|   |   |   |   |   |--- class: 1
|   |   |   |   |--- petal length (cm) >  5.45
|   |   |   |   |   |--- class: 2
|   |--- petal width (cm) >  1.75
|   |   |--- petal length (cm) <= 4.85
|   |   |   |--- sepal width (cm) <= 3.10
|   |   |   |   |--- class: 2
|   |   |   |--- sepal width (cm) >  3.10
|   |   |   |   |--- class: 1
|   |   |--- petal length (cm) >  4.85
|   |   |   |--- class: 2
```

In [5]:

```
y_pred = decisionTree.predict(X)
acc = accuracy_score(y_pred, y)
cla = classification_report(y_pred, y)
print(acc)
print(cla)
```

```
1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        50
           1       1.00      1.00      1.00        50
           2       1.00      1.00      1.00        50

    accuracy                           1.00       150
   macro avg       1.00      1.00      1.00       150
weighted avg       1.00      1.00      1.00       150
```

In [6]:

```
X.shape, y.shape
```

Out[6]:

```
((150, 4), (150,))
```

In [7]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, randon
print (X_train.shape,y_train.shape)
print(X_test.shape,y_test.shape)
```

```
(105, 4) (105,)
(45, 4) (45,)
```

In [8]:
```
#clf = tree.DecisionTreeClassifier(max_depth=2)
clf = tree.DecisionTreeClassifier()
clf = clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
acc = accuracy_score(y_pred,y_test)
cla = classification_report(y_pred,y_test)
print(acc)
print(cla)
```

```
0.9777777777777777
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        16
           1       0.94      1.00      0.97        17
           2       1.00      0.92      0.96        12

    accuracy                           0.98        45
   macro avg       0.98      0.97      0.98        45
weighted avg       0.98      0.98      0.98        45
```

In [9]:
```
#scores= cross_val_score(clf, X, y, cv = 5, scoring= 'precision_macro')
#scores= cross_val_score(clf, X, y, cv = 5, scoring= 'accuracy')
scores= cross_val_score(clf, X, y, cv = 5, scoring= 'recall_macro')
print(scores)
print('%0.2f accuracy with std of %0.2f' % (scores.mean(), scores.std()))
```

```
[0.96666667 0.96666667 0.9        1.         1.        ]
0.97 accuracy with std of 0.04
```

In [10]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, randon
gnb = GaussianNB()
y_pred = gnb.fit(X_train, y_train).predict(X_test)
```

In [11]:
```
acc = accuracy_score(y_pred,y_test)
cla = classification_report(y_pred,y_test)
print(acc)
print(cla)
```

```
1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        16
           1       1.00      1.00      1.00        18
           2       1.00      1.00      1.00        11

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```

In [12]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random
rfc = RandomForestClassifier(max_depth=2, random_state=0)
y_pred = rfc.fit(X_train, y_train).predict(X_test)
```

In [13]:
```
acc = accuracy_score(y_pred,y_test)
cla = classification_report(y_pred,y_test)
print(acc)
print(cla)
```

```
0.9777777777777777
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        16
           1       0.94      1.00      0.97        17
           2       1.00      0.92      0.96        12

    accuracy                           0.98        45
   macro avg       0.98      0.97      0.98        45
weighted avg       0.98      0.98      0.98        45
```

Breast Cancer

In [14]:
```
X_cancer, y_cancer = load_breast_cancer(return_X_y=True)
print(X_cancer)
y_cancer
```

```
[[1.799e+01 1.038e+01 1.228e+02 ... 2.654e-01 4.601e-01 1.189e-01]
 [2.057e+01 1.777e+01 1.329e+02 ... 1.860e-01 2.750e-01 8.902e-02]
 [1.969e+01 2.125e+01 1.300e+02 ... 2.430e-01 3.613e-01 8.758e-02]
 ...
 [1.660e+01 2.808e+01 1.083e+02 ... 1.418e-01 2.218e-01 7.820e-02]
 [2.060e+01 2.933e+01 1.401e+02 ... 2.650e-01 4.087e-01 1.240e-01]
 [7.760e+00 2.454e+01 4.792e+01 ... 0.000e+00 2.871e-01 7.039e-02]]
```

Out[14]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0,
       1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0,
       1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0,
       0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1,
       1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0,
       0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0,
       0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0,
       0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0,
       1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1,
       1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1,
       1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
       1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
       1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1,
       1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1])
```

In [15]:
```
clf_bc = tree.DecisionTreeClassifier()
clf_bc = clf_bc.fit(X_cancer,y_cancer)
```

In [16]:

```
bc = load_breast_cancer()
decisionTree_bc = tree.DecisionTreeClassifier(random_state=0)
#decisionTree_bc = tree.DecisionTreeClassifier(max_depth=3, random_state=0)
decisionTree_bc = decisionTree_bc.fit(X_cancer, y_cancer)
treeP_bc = export_text(decisionTree_bc, feature_names=['mean radius', 'mean textu
        'mean smoothness', 'mean compactness', 'mean concavity',
        'mean concave points', 'mean symmetry', 'mean fractal dimension',
        'radius error', 'texture error', 'perimeter error', 'area error',
        'smoothness error', 'compactness error', 'concavity error',
        'concave points error', 'symmetry error',
        'fractal dimension error', 'worst radius', 'worst texture',
        'worst perimeter', 'worst area', 'worst smoothness',
        'worst compactness', 'worst concavity', 'worst concave points',
        'worst symmetry', 'worst fractal dimension'])
print(treeP_bc)
```

```
|--- worst radius <= 16.80
|   |--- worst concave points <= 0.14
|   |   |--- radius error <= 1.05
|   |   |   |--- area error <= 38.60
|   |   |   |   |--- smoothness error <= 0.00
|   |   |   |   |   |--- worst concavity <= 0.19
|   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- worst concavity >  0.19
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |--- smoothness error >  0.00
|   |   |   |   |   |--- worst texture <= 33.27
|   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- worst texture >  33.27
|   |   |   |   |   |   |--- worst texture <= 33.56
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- worst texture >  33.56
|   |   |   |   |   |   |   |--- class: 1
|   |   |   |--- area error >  38.60
|   |   |   |   |--- area error <= 39.15
|   |   |   |   |   |--- class: 0
|   |   |   |   |--- area error >  39.15
|   |   |   |   |   |--- worst compactness <= 0.08
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- worst compactness >  0.08
|   |   |   |   |   |   |--- class: 1
|   |   |--- radius error >  1.05
|   |   |   |--- class: 0
|   |--- worst concave points >  0.14
|   |   |--- worst texture <= 25.67
|   |   |   |--- worst area <= 810.30
|   |   |   |   |--- mean smoothness <= 0.12
|   |   |   |   |   |--- class: 1
|   |   |   |   |--- mean smoothness >  0.12
|   |   |   |   |   |--- class: 0
|   |   |   |--- worst area >  810.30
|   |   |   |   |--- mean perimeter <= 92.79
|   |   |   |   |   |--- class: 0
|   |   |   |   |--- mean perimeter >  92.79
|   |   |   |   |   |--- class: 1
|   |   |--- worst texture >  25.67
|   |   |   |--- mean concave points <= 0.05
|   |   |   |   |--- worst texture <= 28.55
|   |   |   |   |   |--- class: 1
|   |   |   |   |--- worst texture >  28.55
|   |   |   |   |   |--- class: 0
|   |   |   |--- mean concave points >  0.05
|   |   |   |   |--- class: 0
|--- worst radius >  16.80
|   |--- worst texture <= 19.91
|   |   |--- compactness error <= 0.02
|   |   |   |--- class: 1
|   |   |--- compactness error >  0.02
|   |   |   |--- class: 0
|   |--- worst texture >  19.91
|   |   |--- worst smoothness <= 0.09
|   |   |   |--- class: 1
|   |   |--- worst smoothness >  0.09
|   |   |   |--- worst concavity <= 0.18
|   |   |   |   |--- mean concave points <= 0.04
|   |   |   |   |   |--- class: 0
|   |   |   |   |--- mean concave points >  0.04
|   |   |   |   |   |--- class: 1
|   |   |   |--- worst concavity >  0.18
|   |   |   |   |--- class: 0
```

In [17]:

```
y_pred = decisionTree_bc.predict(X_cancer)
acc = accuracy_score(y_pred, y_cancer)
cla = classification_report(y_pred, y_cancer)
print(acc)
print(cla)
```

```
1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       212
           1       1.00      1.00      1.00       357

    accuracy                           1.00       569
   macro avg       1.00      1.00      1.00       569
weighted avg       1.00      1.00      1.00       569
```

In [18]:

```
X_cancer.shape, y_cancer.shape
```

Out[18]:

```
((569, 30), (569,))
```

In [19]:

```
X_train, X_test, y_train, y_test = train_test_split(X_cancer, y_cancer, test_size
print (X_train.shape,y_train.shape)
print(X_test.shape,y_test.shape)
```

```
(398, 30) (398,)
(171, 30) (171,)
```

In [20]:
```
clf = tree.DecisionTreeClassifier()
clf = clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
acc = accuracy_score(y_pred,y_test)
cla = classification_report(y_pred,y_test)
print(acc)
print(cla)
```

```
0.9064327485380117
              precision    recall  f1-score   support

           0       0.94      0.83      0.88        71
           1       0.89      0.96      0.92       100

    accuracy                           0.91       171
   macro avg       0.91      0.90      0.90       171
weighted avg       0.91      0.91      0.91       171
```

In [21]:
```
#scores= cross_val_score(clf, X_cancer, y_cancer, cv = 5, scoring= 'precision_mac
#scores= cross_val_score(clf, X_cancer, y_cancer, cv = 5, scoring= 'accuracy')
scores= cross_val_score(clf, X_cancer, y_cancer, cv = 5, scoring= 'recall_macro')
print(scores)
print('%0.2f accuracy with std of %0.2f' % (scores.mean(), scores.std()))
```

```
[0.9158205  0.9207337  0.91269841 0.93154762 0.89872569]
0.92 accuracy with std of 0.01
```

In [22]:
```
X_train, X_test, y_train, y_test = train_test_split(X_cancer, y_cancer, test_size
gnb = GaussianNB()
y_pred = gnb.fit(X_train, y_train).predict(X_test)
```

In [23]:
```
acc = accuracy_score(y_pred,y_test)
cla = classification_report(y_pred,y_test)
print(acc)
print(cla)
```

```
0.9239766081871345
              precision    recall  f1-score   support

           0       0.90      0.89      0.90        64
           1       0.94      0.94      0.94       107

    accuracy                           0.92       171
   macro avg       0.92      0.92      0.92       171
weighted avg       0.92      0.92      0.92       171
```

In [24]:
```
X_train, X_test, y_train, y_test = train_test_split(X_cancer, y_cancer, test_size
rfc = RandomForestClassifier(max_depth=2, random_state=0)
y_pred = rfc.fit(X_train, y_train).predict(X_test)
```

In [25]:
```
acc = accuracy_score(y_pred,y_test)
cla = classification_report(y_pred,y_test)
print(acc)
print(cla)
```

```
0.935672514619883
              precision    recall  f1-score   support

           0       0.92      0.91      0.91        64
           1       0.94      0.95      0.95       107

    accuracy                           0.94       171
   macro avg       0.93      0.93      0.93       171
weighted avg       0.94      0.94      0.94       171
```

Adult Train

In [26]:
```
adult_train_df = pd.read_csv('adult.csv')
adult_train_df
```

Out[26]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black

In [27]:
```
lb_enc = LabelEncoder()
adult_train_df['age'] = lb_enc.fit_transform(adult_train_df['age'])
adult_train_df['workclass'] = lb_enc.fit_transform(adult_train_df['workclass'])
adult_train_df['fnlwgt'] = lb_enc.fit_transform(adult_train_df['fnlwgt'])
adult_train_df['education'] = lb_enc.fit_transform(adult_train_df['education'])
adult_train_df['education-num'] = lb_enc.fit_transform(adult_train_df['education-
adult_train_df['marital-status'] = lb_enc.fit_transform(adult_train_df['marital-s
adult_train_df['occupation'] = lb_enc.fit_transform(adult_train_df['occupation'])
adult_train_df['relationship'] = lb_enc.fit_transform(adult_train_df['relationshi
adult_train_df['race'] = lb_enc.fit_transform(adult_train_df['race'])
adult_train_df['sex'] = lb_enc.fit_transform(adult_train_df['sex'])
adult_train_df['capital-gain numeric'] = lb_enc.fit_transform(adult_train_df['cap
adult_train_df['capital-loss numeric'] = lb_enc.fit_transform(adult_train_df['cap
adult_train_df['hours-per-week numeric'] = lb_enc.fit_transform(adult_train_df['h
adult_train_df['native-coutry'] = lb_enc.fit_transform(adult_train_df['native-cou
adult_train_df['class'] = lb_enc.fit_transform(adult_train_df['class'])
adult_train_df.to_csv("adult_train_Ndf.csv")
```

In [28]:
```
adult_numeric_df = pd.read_csv("adult_train_Ndf.csv")
adult_numeric_df.drop("Unnamed: 0",axis=1)
#list(adult_numeric_df.columns)
```

Out[28]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex
0	22	7	2671	9	12	4	1	1	4	1
1	33	6	2926	9	12	2	4	0	4	1
2	21	4	14086	11	8	0	6	1	4	1
3	36	4	15336	1	6	2	6	0	2	1
4	11	4	19355	9	12	2	10	5	2	0
...	...	...	...	...	...	...	...	...	...	...
32556	10	4	16528	7	11	2	13	5	4	0
32557	23	4	8080	11	8	2	7	0	4	1
32558	41	4	7883	11	8	6	1	4	4	0
32559	5	4	12881	11	8	4	1	3	4	1
32560	35	5	17825	11	8	2	4	5	4	0

32561 rows × 15 columns

In [29]:
```
X_adt = adult_numeric_df.values[:,0:14]
y_adt = adult_numeric_df.values[:,14]
```

In [30]:
```
X_train, X_test, y_train, y_test = train_test_split(X_adt, y_adt, test_size = 0.3
print (X_train.shape,y_train.shape)
print(X_test.shape,y_test.shape)
```

```
(22792, 14) (22792,)
(9769, 14) (9769,)
```

In [31]:
```
clf_adt = DecisionTreeClassifier(random_state=0)
model = clf_adt.fit(X_adt, y_adt)
```

In [32]:
```
text_representation = tree.export_text(clf_adt)
print(text_representation)
```

```
|--- feature_9 <= 1.50
|   |--- feature_9 <= 0.50
|   |   |--- feature_3 <= 15892.00
|   |   |   |--- feature_0 <= 32337.00
|   |   |   |   |--- feature_11 <= 107.00
|   |   |   |   |   |--- feature_4 <= 1.50
|   |   |   |   |   |   |--- feature_0 <= 1444.50
|   |   |   |   |   |   |   |--- feature_7 <= 3.50
|   |   |   |   |   |   |   |   |--- class: 39
|   |   |   |   |   |   |   |--- feature_7 >  3.50
|   |   |   |   |   |   |   |   |--- class: 11
|   |   |   |   |   |   |--- feature_0 >  1444.50
|   |   |   |   |   |   |   |--- feature_13 <= 16.50
|   |   |   |   |   |   |   |   |--- class: 35
|   |   |   |   |   |   |   |--- feature_13 >  16.50
|   |   |   |   |   |   |   |   |--- class: 39
|   |   |   |   |   |--- feature_4 >  1.50
|   |   |   |   |   |   |--- feature_5 <= 9.50
|   |   |   |   |   |   |   |--- class: 39
|   |   |   |   |   |   |--- feature_5 >  9.50
```

In [33]:
```
X_tt = adult_train_df.values[:,0:14]
y_tt = adult_train_df.values[:,14]
```

In [34]:
```
X_train, X_test, y_train, y_test = train_test_split(X_tt, y_tt, test_size = 0.3,
print (X_train.shape,y_train.shape)
print(X_test.shape,y_test.shape)
```

```
(22792, 14) (22792,)
(9769, 14) (9769,)
```

In [35]:
```
clf = tree.DecisionTreeClassifier()
clf = clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
acc = accuracy_score(y_pred,y_test)
cla = classification_report(y_pred,y_test)
print(acc)
print(cla)
```

```
0.8115467294503019
              precision    recall  f1-score   support

           0       0.88      0.88      0.88      7414
           1       0.61      0.61      0.61      2355

    accuracy                           0.81      9769
   macro avg       0.74      0.74      0.74      9769
weighted avg       0.81      0.81      0.81      9769
```

In [36]:
```
#scores= cross_val_score(clf, X_tt, y_tt, cv = 5, scoring= 'precision_macro')
#scores= cross_val_score(clf, X_tt, y_tt, cv = 5, scoring= 'accuracy')
scores= cross_val_score(clf, X_tt, y_tt, cv = 5, scoring= 'recall_macro')
print(scores)
print('%0.2f accuracy with std of %0.2f' % (scores.mean(), scores.std()))
```

```
[0.74327671 0.73288071 0.74243156 0.74786899 0.737571  ]
0.74 accuracy with std of 0.01
```

In [37]:
```
X_train, X_test, y_train, y_test = train_test_split(X_tt, y_tt, test_size = 0.3,
gnb = GaussianNB()
y_pred = gnb.fit(X_train, y_train).predict(X_test)
```

In [38]:
```
acc = accuracy_score(y_pred,y_test)
cla = classification_report(y_pred,y_test)
print(acc)
print(cla)
```

```
0.8234210256935203
              precision    recall  f1-score   support

           0       0.94      0.85      0.89      8176
           1       0.47      0.70      0.56      1593

    accuracy                           0.82      9769
   macro avg       0.70      0.77      0.73      9769
weighted avg       0.86      0.82      0.84      9769
```

In [39]:
```
X_train, X_test, y_train, y_test = train_test_split(X_tt, y_tt, test_size = 0.3,
rfc = RandomForestClassifier(max_depth=2, random_state=0)
y_pred = rfc.fit(X_train, y_train).predict(X_test)
```

In [40]:

```
acc = accuracy_score(y_pred,y_test)
cla = classification_report(y_pred,y_test)
print(acc)
print(cla)
```

```
0.7931210973487562
              precision    recall  f1-score   support

           0       1.00      0.79      0.88      9418
           1       0.15      0.99      0.26       351

    accuracy                           0.79      9769
   macro avg       0.57      0.89      0.57      9769
weighted avg       0.97      0.79      0.86      9769
```

In [ ]: