MACHINE LEARNING – 2CS501

PRACTICAL 5

Name: Bhanderi Mihir

Roll No.: 19BCE023

Batch No.: A-1

1) Gaussian Naïve Bayes

Code:

```
from sklearn import datasets, metrics
from sklearn.naive_bayes import GaussianNB

X, y = datasets.load_iris(return_X_y=True)
X_train = X[(range(0,150,2)),:]
y_train = y[(range(0,150,2))]
X_test = X[(range(1,150,2)),:]
y_test = y[(range(1,150,2))]

print("------------------------- Gaussian --------------------------")
clf = GaussianNB()
clf.fit(X_train,y_train)
params = clf.get_params(deep = True)
print(params)
predictions = clf.predict(X_test)
print(predictions)

print("Accuracy : ", metrics.accuracy_score(y_test, predictions, normalize = True))

print("Report : \n", metrics.classification_report(y_test, predictions))
print("Confusion Matrix : \n", metrics.confusion_matrix(y_test, predictions))

"""

------------------------- Gaussian -------------------------
{'priors': None, 'var_smoothing': 1e-09}
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1
 1 2 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 1 2 2 2 2 2 2 2
 2]
Accuracy :  0.96
Report :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        25
           1       0.92      0.96      0.94        25
           2       0.96      0.92      0.94        25

    accuracy                           0.96        75
   macro avg       0.96      0.96      0.96        75
weighted avg       0.96      0.96      0.96        75
```

```
Confusion Matrix :
 [[25  0  0]
 [ 0 24  1]
 [ 0  2 23]]

"""
```

2) KNN

Code:

```
from sklearn import datasets, metrics,neighbors
from sklearn.model_selection import GridSearchCV


X, y = datasets.load_iris(return_X_y=True)
X_train = X[(range(0,150,2)),:]
y_train = y[(range(0,150,2))]
X_test = X[(range(1,150,2)),:]
y_test = y[(range(1,150,2))]

kc=neighbors.KNeighborsClassifier()

gs=GridSearchCV(kc,{'n_neighbors':[1,4,7,10,13],'weights':['uniform','distance']},cv=3)

gs.fit(X_train,y_train)

gs.best_params_

predictions = gs.predict(X_test)


print("------------------------- KNN -------------------------")
print("\nPredictions:\n",predictions)
print("\nAccuracy:",metrics.accuracy_score(y_test, predictions, normalize=True))
print("\nClassification report :\n",metrics.classification_report(y_test, predictions))
print("\n Confusion matrix :\n",metrics.confusion_matrix(y_test, predictions))

"""

------------------------- KNN -------------------------

Predictions:
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 2 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 1 2 2 2 1 2 2 1 2 2 2 2 2 2 2
 2]

Accuracy: 0.9466666666666667

Classification report :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        25
           1       0.89      0.96      0.92        25
           2       0.96      0.88      0.92        25

    accuracy                           0.95        75
   macro avg       0.95      0.95      0.95        75
weighted avg       0.95      0.95      0.95        75


 Confusion matrix :
 [[25  0  0]
 [ 0 24  1]
 [ 0  3 22]]

"""
```

3) Bernoulli Naïve Bayes

Code:

```
import pandas as pd
from sklearn import metrics
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import BernoulliNB

print("------------------------- Bernoulli --------------------------")
sms = pd.read_csv("spam.csv",delimiter=",",encoding = "ISO-8859-1")
print(sms.keys())
sms = sms.drop(['Unnamed: 2','Unnamed: 3','Unnamed: 4'], axis = 1)
print(sms.head(5))

sms['label_num'] = sms.v1.map({'ham':0, 'spam':1})
print(sms.head())

X = sms.v2
y = sms.label_num
print(X.shape)
print(y.shape)

X_train = X[0:4179]
X_test = X[4179:]
y_train = y[0:4179]
y_test = y[4179:]

vect = CountVectorizer()
vect.fit(X_train)

X_train_dtm = vect.transform(X_train)
X_test_dtm = vect.transform(X_test)

ber = BernoulliNB()
ber.fit(X_train_dtm, y_train)

y_pred_class = ber.predict(X_test_dtm)

print("Accuracy : ", metrics.accuracy_score(y_test, y_pred_class))
print("Confusion Matrix : \n", metrics.confusion_matrix(y_test, y_pred_class))

"""

------------------------- Bernoulli --------------------------
Index(['v1', 'v2', 'Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'], dtype='object')
     v1                                                 v2
0   ham  Go until jurong point, crazy.. Available only ...
1   ham                      Ok lar... Joking wif u oni...
2  spam  Free entry in 2 a wkly comp to win FA Cup fina...
3   ham  U dun say so early hor... U c already then say...
4   ham  Nah I don't think he goes to usf, he lives aro...
     v1                                                 v2  label_num
0   ham  Go until jurong point, crazy.. Available only ...          0
1   ham                      Ok lar... Joking wif u oni...          0
2  spam  Free entry in 2 a wkly comp to win FA Cup fina...          1
3   ham  U dun say so early hor... U c already then say...          0
4   ham  Nah I don't think he goes to usf, he lives aro...          0
(5572,)
(5572,)
Accuracy :  0.9755922469490309
```

```
Confusion Matrix :
 [[1210    1]
 [  33  149]]

"""
```

4) Multinomial Naïve Bayes

Code:

```
import pandas as pd
from sklearn import metrics
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB

print("------------------------- Multinomial --------------------------")
sms = pd.read_csv("spam.csv",delimiter=",",encoding = "ISO-8859-1")
print(sms.keys())
sms = sms.drop(['Unnamed: 2','Unnamed: 3','Unnamed: 4'], axis = 1)
print(sms.head(5))

sms['label_num'] = sms.v1.map({'ham':0, 'spam':1})
print(sms.head())

X = sms.v2
y = sms.label_num


X_train = X[0:4179]
X_test = X[4179:]
y_train = y[0:4179]
y_test = y[4179:]

vect = CountVectorizer()
vect.fit(X_train)

X_train_dtm = vect.transform(X_train)
X_test_dtm = vect.transform(X_test)

nb = MultinomialNB()
nb.fit(X_train_dtm, y_train)

y_pred_class = nb.predict(X_test_dtm)

print("Accuracy : ", metrics.accuracy_score(y_test, y_pred_class))
print("Confusion Matrix : \n", metrics.confusion_matrix(y_test, y_pred_class))

"""

------------------------- Multinomial --------------------------
Index(['v1', 'v2', 'Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'], dtype='object')
     v1                                                 v2
0   ham  Go until jurong point, crazy.. Available only ...
1   ham                      Ok lar... Joking wif u oni...
2  spam  Free entry in 2 a wkly comp to win FA Cup fina...
3   ham  U dun say so early hor... U c already then say...
4   ham  Nah I don't think he goes to usf, he lives aro...
     v1                                                 v2  label_num
0   ham  Go until jurong point, crazy.. Available only ...          0
1   ham                      Ok lar... Joking wif u oni...          0
2  spam  Free entry in 2 a wkly comp to win FA Cup fina...          1
3   ham  U dun say so early hor... U c already then say...          0
4   ham  Nah I don't think he goes to usf, he lives aro...          0

Accuracy :  0.9856424982053122
Confusion Matrix :
 [[1203    8]
```

```
 [  12  170]]

"""
```