Machine Learning Assignment

Karn Arora (J078)

Exp 11:- Naive Bayes from Scratch

Importing Libraries

```
import numpy as np
import pandas as pd
from sklearn import datasets
from collections import Counter
```

```
iris = datasets.load_iris()
Species = iris.target
data = pd.DataFrame(np.c_[iris.data, Species.reshape((Species.shape[0],1))], columns = iri
data.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	Species
0	5.1	3.5	1.4	0.2	0.0
1	4.9	3.0	1.4	0.2	0.0
2	4.7	3.2	1.3	0.2	0.0
3	4.6	3.1	1.5	0.2	0.0
4	5.0	3.6	1.4	0.2	0.0

```
data['Species'].value_counts()
```

```
2.0    50
1.0    50
0.0    50
Name: Species, dtype: int64
```

Using Naive Bayes function

```
from sklearn.model_selection import train_test_split
train, test = train_test_split(data, test_size = 0.2, random_state = 0)
```

```
class NB():
    def __init__(self,train):
        self.train = train
        self.X_train = train.drop('Species', axis = 1)
```

```
            self.Y_train = train['Species']
            self.s = {}

        def fit(self):
            self.result = Counter(self.Y_train)
            for target in self.result.keys():
                for col in self.X_train.columns:
                    self.s[target,col,"mean"] = self.train[self.train['Species'] == target].me
                    self.s[target,col,"std"] = self.train[self.train['Species'] == target].std

            for i in self.result:
                self.result[i] = round(self.result[i]/len(self.X_train.index),8)

        def predict(self,X_test):
            count = 0
            prediction = []
            for i in X_test.index: #enters into a row-wise loop
                prob_index = {}
                for target in self.result: #enters into a loop for every value of target
                    prob = self.result[target]
                    for col in self.X_train:
                        a = 1/(((2*np.pi)**0.5)*self.s[target,col,"std"])
                        b = -((X_test[col][i] - self.s[target,col,"mean"])**2)
                        c = 2*(self.s[target,col,"std"]**2)
                        prob = prob * a * np.exp(b/c)
                    prob_index[target] = prob

                probability = 0
                for target in prob_index:
                    if prob_index[target] > probability:
                        pred = target
                        probability = prob_index[target]
                prediction.append(pred)

            return prediction


    clf = NB(train)
    clf.fit()


    Y_test = test['Species']
    X_test = test.drop('Species', axis = 1)
    predictions = clf.predict(X_test)


    from sklearn.metrics import accuracy_score
    accuracy_score(Y_test, predictions)
```

```
     0.9666666666666667
```

Using Naive Bayes in Scikit Learn

```
X = data.drop(['Species'],axis = 1)
y = data['Species']

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2,random_state=0)


from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
predictions1 = gnb.fit(X_train, y_train).predict(X_test)
accuracy_score(y_test, predictions1)
```

```
0.9666666666666667
```