LAB - 09 - SPOT - NAIVE BAYES AND KNN

1. NAIVE BAYES ALGORITHM

```
import numpy as np
import pandas as pd
from math import sqrt, exp, pi
from sklearn.model_selection import train_test_split
from sklearn import metrics

data = pd.read_csv('iris.data.csv')
print('training data set sample')
print(data.sample(5))
```

[2] ✓ 0.1s Python

```
training data set sample
     sepal_length  sepal_width  petal_length  petal_width       Class_name
133           6.3          2.8           5.1          1.5   Iris-virginica
50            7.0          3.2           4.7          1.4  Iris-versicolor
29            4.7          3.2           1.6          0.2      Iris-setosa
3             4.6          3.1           1.5          0.2      Iris-setosa
92            5.8          2.6           4.0          1.2  Iris-versicolor
```

```
def labels_to_num(data):
    # convert strings to numbers
    data.replace(['Iris-versicolor','Iris-setosa','Iris-virginica'], [0,1,2], inplace=True)

print('\nreplaced labels with numbers')
labels_to_num(data)
print(data.sample(5))

# split data by class
def split_by_class(data):
    # split data by class
    classes = dict()
    for row in data:
        if(row[-1] not in classes.keys()):
            classes[row[-1]] = list()

        classes[row[-1]].append(row)

    return classes
```

[4] ✓ 0.7s Python

```
replaced labels with numbers
     sepal_length  sepal_width  petal_length  petal_width  Class_name
9             4.9          3.1           1.5          0.1           1
11            4.8          3.4           1.6          0.2           1
89            5.5          2.5           4.0          1.3           0
119           6.0          2.2           5.0          1.5           2
42            4.4          3.2           1.3          0.2           1
```

```
# get mean, std and size
def get_info(data):
    # get data
    info = [(np.mean(col), np.std(col), len(col)) for col in zip(*data)]
    del info[-1] # remove target coloumn
    return info

# get mean, std and size by class
def get_info_by_class(data):
    classData = split_by_class(data)
    info = dict()
    for classVal, rows in classData.items():
        info[classVal] = get_info(rows)
    return info
```

[10] ✓ 0.1s Python

```python
def calc_prob(x, mean, stdev):
    exponent = exp(-((x-mean)**2 / (2 * stdev**2 )))
    return (1 / (sqrt(2 * pi) * stdev)) * exponent

def predict(info, dataset):
    outProbs = list()

    for data in dataset:
        total_rows = sum([info[label][0][2] for label in info])
        probs = dict()

        for classVal, class_info in info.items():
            probs[classVal] = info[classVal][0][2]/float(total_rows)
            for i in range(len(class_info)):
                mean, stdev, _ = class_info[i]
                probs[classVal] *= calc_prob(data[i], mean, stdev)

        outProbs.append(probs)

    preds = list()
    for prob in outProbs:
        preds.append(max(prob, key=prob.get))
    return preds
```

[11] ✓ 0.1s Python

```python
# split training and testing data
X_train, X_test = train_test_split(data,test_size=0.2)

info = get_info_by_class(X_train.values)
y_test = X_test.iloc[:,-1].values
X_test = X_test.iloc[:,0:-1]
preds = predict(info, X_test.values)

print('accuracy', metrics.accuracy_score(y_test,preds))
print('confusion matrix')
print(metrics.confusion_matrix(y_test, preds))
print('classification report')
print(metrics.classification_report(y_test, preds))
```

[12] ✓ 0.1s Python

```
accuracy 0.9
confusion matrix
[[ 7  0  2]
 [ 0 11  0]
 [ 1  0  9]]
classification report
              precision    recall  f1-score   support

           0       0.88      0.78      0.82         9
           1       1.00      1.00      1.00        11
           2       0.82      0.90      0.86        10

    accuracy                           0.90        30
   macro avg       0.90      0.89      0.89        30
weighted avg       0.90      0.90      0.90        30
```

2. KNN CLASSIFIER

```
import numpy as nm
import matplotlib.pyplot as mtp
from sklearn.preprocessing import LabelEncoder
import math
import pandas as pd

le = LabelEncoder()
df= pd.read_csv('iris.data.csv')

data_set = df.apply(le.fit_transform)
x= data_set.iloc[:, :-1].values
y= data_set.iloc[:, -1].values

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.25)

from sklearn.preprocessing import StandardScaler
st_x= StandardScaler()
x_train= st_x.fit_transform(x_train)
x_test= st_x.transform(x_test)
```

[34] ✓ 0.1s Python

```
from sklearn.neighbors import KNeighborsClassifier
classifier= KNeighborsClassifier(n_neighbors=3, metric='minkowski', p=2 )
classifier.fit(x_train, y_train)
```

[39] ✓ 0.7s Python

```
KNeighborsClassifier(n_neighbors=3)
```

```
y_pred= classifier.predict(x_test)
```

[40] ✓ 0.9s Python

```
from sklearn.metrics import confusion_matrix
cm= confusion_matrix(y_test, y_pred)
```

[42] ✓ 0.6s Python

```
cm
```

[43] ✓ 0.2s Python

```
array([[15,  0,  0],
       [ 0, 11,  0],
       [ 0,  3,  9]], dtype=int64)
```

```
from sklearn.metrics import classification_report
matrix = classification_report(y_test,y_pred,labels=[0,1])
print('\nClassification report : \n',matrix)
```

[44] ✓ 0.1s Python

```
Classification report :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        15
           1       0.79      1.00      0.88        11

   micro avg       0.90      1.00      0.95        26
   macro avg       0.89      1.00      0.94        26
weighted avg       0.91      1.00      0.95        26
```