ENGR 421/DASC 521: Introduction to Machine Learning
Homework 1: Naive Bayes Classifier
Deadline: March 11, 2024, 11:59 PM

In this homework, you will implement a naive Bayes classifier using Python. Here are the steps you need to follow:

1. Read Section 5.7 from the textbook.

2. You are given a multivariate classification data set, which contains 93925 nucleotide sequences of length 7. These sequences are from two distinct classes, namely, 1 (a splice site) and 2 (not a splice site), where we have 4208 splice sites. You are provided with two data files:

   a. `hw01_data_points.csv`: nucleotide sequences,

   b. `hw01_class_labels.csv`: corresponding class labels.

3. Divide the data set into two parts by assigning the first 50000 sequences to the training set and the remaining 44727 sequences to the test set. (10 points)

```
X_train, y_train, X_test, y_test = train_test_split(X, y)
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(50000, 7)
(50000,)
(43925, 7)
(43925,)
```

4. Calculate the prior probability estimates $\widehat{\Pr}(y = 1)$ and $\widehat{\Pr}(y = 2)$ using the data points you assigned to the training set. (10 points)

```
class_priors = estimate_prior_probabilities(y_train)
print(class_priors)
```

```
[0.0452 0.9548]
```

Hint: You can use the following equation to calculate the prior probability estimates.

$$\widehat{\Pr}(y = c) = \frac{\sum_{i=1}^{N} 1(y_i = c)}{N} = \frac{N_c}{N}$$

Hint: Let us define the following probabilities.

$$p_{\mathrm{A}cd} = \text{probability of having adenine (A) for class } c \text{ at location } d,$$
$$p_{\mathrm{C}cd} = \text{probability of having cytosine (C) for class } c \text{ at location } d,$$
$$p_{\mathrm{G}cd} = \text{probability of having guanine (G) for class } c \text{ at location } d,$$
$$p_{\mathrm{T}cd} = \text{probability of having thymine (T) for class } c \text{ at location } d.$$

5. Calculate the model parameter estimates $\hat{p}_{\mathrm{A}11}, \ldots, \hat{p}_{\mathrm{A}17}$, $\hat{p}_{\mathrm{A}21}, \ldots, \hat{p}_{\mathrm{A}27}$, $\hat{p}_{\mathrm{C}11}, \ldots, \hat{p}_{\mathrm{C}17}$, $\hat{p}_{\mathrm{C}21}, \ldots, \hat{p}_{\mathrm{C}27}$, $\hat{p}_{\mathrm{G}11}, \ldots, \hat{p}_{\mathrm{G}17}$, $\hat{p}_{\mathrm{G}21}, \ldots, \hat{p}_{\mathrm{G}27}$, $\hat{p}_{\mathrm{T}11}, \ldots, \hat{p}_{\mathrm{T}17}$, $\hat{p}_{\mathrm{T}21}, \ldots, \hat{p}_{\mathrm{T}27}$ using the data points you assigned to the training set. (20 points)

```
pAcd, pCcd, pGcd, pTcd = estimate_nucleotide_probabilities(X_train, y_train)
print(pAcd)
print(pCcd)
print(pGcd)
print(pTcd)
```

```
[[0.32345133 0.64424779 0.09424779 0.56681416 0.68938053 0.0800885
  0.16371681]
 [0.25450356 0.2728739  0.30117302 0.19522413 0.25473398 0.23990364
  0.26292417]]
[[0.37433628 0.12123894 0.02964602 0.03230088 0.08053097 0.05221239
  0.16725664]
 [0.21979472 0.2340176  0.06514453 0.20605362 0.21214914 0.24767491
  0.21723921]]
[[0.18716814 0.1199115  0.81814159 0.37477876 0.11371681 0.79513274
  0.19690265]
 [0.24805195 0.24434437 0.2676791  0.33636364 0.22869711 0.23104315
  0.23016339]]
[[0.11504425 0.11460177 0.0579646  0.02610619 0.11637168 0.07256637
  0.47212389]
 [0.27764977 0.24876414 0.36600335 0.26235861 0.30441977 0.2813783
  0.28967323]]
```

Hint: You can use the following equations to calculate the model parameter estimates.

$$\hat{p}_{\mathrm{A}cd} = \frac{\sum_{i=1}^{N} 1(x_{id} = \mathrm{A}) 1(y_i = c)}{N_c} \qquad \hat{p}_{\mathrm{C}cd} = \frac{\sum_{i=1}^{N} 1(x_{id} = \mathrm{C}) 1(y_i = c)}{N_c}$$

$$\hat{p}_{\mathrm{G}cd} = \frac{\sum_{i=1}^{N} 1(x_{id} = \mathrm{G}) 1(y_i = c)}{N_c} \qquad \hat{p}_{\mathrm{T}cd} = \frac{\sum_{i=1}^{N} 1(x_{id} = \mathrm{T}) 1(y_i = c)}{N_c}$$

6. Calculate the score values for the data points in your training and test sets using the estimated parameters. (40 points)

```
scores_train = calculate_score_values(X_train, pAcd, pCcd, pGcd, pTcd, class_priors
print(scores_train)

scores_test = calculate_score_values(X_test, pAcd, pCcd, pGcd, pTcd, class_priors)
print(scores_test)

[[-15.84302871  -9.3433522 ]
 [-16.73815388  -8.90270491]
 [-17.12201508  -8.87059123]
 ...
 [-12.64747986  -9.3077559 ]
 [-18.14584244  -9.54820083]
 [-14.67706629  -9.45002167]]
[[-17.02244643  -9.95385128]
 [-13.10176993  -9.42908242]
 [-12.61507938  -9.54180473]
 ...
 [-12.86990369  -9.21875465]
 [-12.90250138  -9.38540918]
 [-16.43266951  -9.37467256]]
```

Hint: You can use the following equation to calculate the score values.

$$
\begin{aligned}
g_c(\boldsymbol{x}) &= \log\left[\prod_{d=1}^{D}\hat{p}(x_d|y=c)\right] + \log\widehat{\Pr}(y=c) \\
&= \log\left[\prod_{d=1}^{D}\left(\hat{p}_{\mathrm{A}cd}^{1(x_d=\mathrm{A})}\hat{p}_{\mathrm{C}cd}^{1(x_d=\mathrm{C})}\hat{p}_{\mathrm{G}cd}^{1(x_d=\mathrm{G})}\hat{p}_{\mathrm{T}cd}^{1(x_d=\mathrm{T})}\right)\right] + \log\widehat{\Pr}(y=c)
\end{aligned}
$$

---

7. Calculate the confusion matrices for the data points in your training and test sets using the calculated score values. (20 points)

```
confusion_train = calculate_confusion_matrix(y_train, scores_train)
print(confusion_train)

confusion_test = calculate_confusion_matrix(y_test, scores_test)
print(confusion_test)

[[ 1066   484]
 [ 1194 47256]]
[[  891   416]
 [ 1057 41561]]
```

What to submit: You need to submit your source code in a single file (`.py` file). You are provided with a template file named as `0099999.py`, where `99999` should be replaced with your 5-digit student number. You are allowed to change the template file between the following lines.

```
# your implementation starts below

# your implementation ends above
```

How to submit: Submit the file you edited to Blackboard by following the exact style mentioned. Submissions that do not follow these guidelines will not be graded.

Late submission policy: Late submissions will not be graded.

Cheating policy: Very similar submissions will not be graded.
