Name: Phan Bá Đại Phúc

Student Code: B1910688

NAÏVE BAYES

1) Given dataset:

Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

- How Naïve Bayes predicts the class for 4 examples as follows:

Outlook	**Temp**	**Humidity**	**Windy**	**Play**
Overcast	Cool	High	False	?
Rainy	Cool	High	False	?
Sunny	Hot	Normal	False	?
???	Hot	Normal	False	?

Training dataset weather **[Outlook, temp, humidity, windy]** → Play

Outlook			**Temp**			**Humidity**			**Windy**			**Play**	
	Yes	**No**		**Yes**	**No**		**Yes**	**No**		**Yes**	**No**	**Yes**	**No**
Overcast	4	0	Cool	3	1	High	3	4	TRUE	3	3	9	5
Rainy	3	2	Hot	2	2	Normal	6	1	FALSE	6	2		
Sunny	2	3	Mild	4	2								
Overcast	4/9	0	Cool	1/3	1/5	High	1/3	4/5	TRUE	1/3	3/5	9/14	5/14
Rainy	3/9	2/5	Hot	2/9	2/5	Normal	2/3	1/5	FALSE	2/3	2/5		
Sunny	2/9	3/5	Mild	4/9	2/5								

Outlook	Temp	Humidity	Windy	Play
Overcast	Cool	High	FALSE	
Rainy	Cool	High	FALSE	
Sunny	Hot	Normal	FALSE	
???	Hot	Normal	FALSE	

For **[Outlook, Temp, Humidity, Windy]** = **[Overcast, Cool, High, False]**

We calculate the likelihood of two class "Yes" and "No":

For "Yes" = $\frac{4}{9} \times \frac{1}{3} \times \frac{1}{3} \times \frac{2}{3} \times \frac{9}{14} = \mathbf{\frac{4}{189}} \approx 0.0212$

For "No" = $\frac{0+1}{5+3} \times \frac{1}{5} \times \frac{4}{5} \times \frac{2}{5} \times \frac{5}{14} = \mathbf{\frac{1}{350}} \approx 0.0029$

Because Likelihood of Yes > Likelihood of No (**0.0212** > **0.0029**), the answer of class "Play" is "Yes".

For **[Outlook, Temp, Humidity, Windy]** = **[Rainy, Cool, High, False]**

We calculate the likelihood of two class "Yes" and "No":

For "Yes" = $\frac{3}{9} \times \frac{1}{3} \times \frac{1}{3} \times \frac{2}{3} \times \frac{9}{14} = \mathbf{\frac{1}{63}} \approx 0.0159$

For "No" = $\frac{2}{5} \times \frac{1}{5} \times \frac{4}{5} \times \frac{2}{5} \times \frac{5}{14} = \mathbf{\frac{8}{875}} \approx 0.0091$

Because Likelihood of Yes > Likelihood of No (**0.0159** > **0.0091**), the answer of class "Play" is "Yes".

For **[Outlook, Temp, Humidity, Windy]** = **[Sunny, Hot, Normal, False]**

We calculate the likelihood of two class "Yes" and "No":

For "Yes" = $\frac{2}{9} \times \frac{2}{9} \times \frac{2}{3} \times \frac{2}{3} \times \frac{9}{14} = \mathbf{\frac{8}{567}} \approx 0.014$

For "No" = $\frac{3}{5} \times \frac{2}{5} \times \frac{1}{5} \times \frac{2}{5} \times \frac{5}{14} = \mathbf{\frac{6}{875}} \approx 0.007$

Because Likelihood of Yes > Likelihood of No (**0.014 > 0.007**), the answer of class "Play" is "Yes".

For **[Outlook, Temp, Humidity, Windy]** = **[Null, Hot, Normal, False]**

We calculate the likelihood of two class "Yes" and "No":

$$\text{For "Yes"} = \frac{2}{9} \times \frac{2}{3} \times \frac{2}{3} \times \frac{9}{14} = \boldsymbol{\frac{4}{63}} \approx 0.0635$$

$$\text{For "No"} = \frac{2}{5} \times \frac{1}{5} \times \frac{2}{5} \times \frac{5}{14} = \boldsymbol{\frac{2}{175}} \approx 0.0114$$

Because Likelihood of Yes > Likelihood of No (**0.0635 > 0.0114**), the answer of class "Play" is "Yes".

So the answer is:

Outlook	Temp	Humidity	Windy	Play
Overcast	Cool	High	FALSE	Yes
Rainy	Cool	High	FALSE	Yes
Sunny	Hot	Normal	FALSE	Yes
???	Hot	Normal	FALSE	Yes

2)

Outlook			Temp			Humidity				Windy				Play	
	yes	no		yes	no		yes	no			yes	no		yes	no
sunny	2	3		83	85		86	85		FALSE	6	2		9	5
overcast	4	0		70	80		96	90		TRUE	3	3			
rainy	3	2		68	65		80	70							
				64	72		65	95							
				69	71		70	91							
				75			80								
				75			70								
				72			90								
				81			75								
sunny	2/9	0.6	mean	73	74.6	mean	79.1	86.2	mean	FALSE	2/3	0.4		0.643	0.357
overcast	4/9	0	std. dev.	6.2	7.9	std. dev.	10.2	9.7	std. dev.	TRUE	1/3	0.6			
rainy	1/3	0.4													

Naïve Bayes predicts the 4 examples:

outlook	Temp	Humidity	Windy	Play
Overcast	66	80	FALSE	
Rainy	73	90	FALSE	

Sunny	80	85	FALSE	
?	90	85	?	

For **[Outlook, Temp, Humidity, Windy]** = **[Overcast, 66, 80, False]**

First, we have to calculate f(tempurature=66|yes), f(humidity=80|yes), f(tempurature=66|no) and f(humidity=80|no)

f(tempurature=66|yes) = $\frac{1}{\sqrt{2\pi}6.2}e^{-\frac{(66-73)^2}{2*6.2^2}}$ = 0.0340

f(humidity=80|yes) = $\frac{1}{\sqrt{2\pi}10.2}e^{-\frac{(80-79.1)^2}{2*10.2^2}}$ = 0.0390

f(tempurature=66|no) = $\frac{1}{\sqrt{2\pi}7.9}e^{-\frac{(66-74.6)^2}{2*7.9^2}}$ = 0.0279

f(humidity=80|no) = $\frac{1}{\sqrt{2\pi}9.7}e^{-\frac{(80-86.2)^2}{2*9.7^2}}$ = 0.0335

Next, we calculate the likelihood of two class “Yes” and “No”:

For “Yes” = $\frac{4}{9}$ x 0.0340 x 0.0390 x $\frac{2}{3}$ x $\frac{9}{14}$ = 0.000252

For “No” = $\frac{0+1}{5+3}$ x 0.0279 x 0.0335 x 0.4 x $\frac{5}{14}$= 0.0000167

Because Likelihood of Yes > Likelihood of No (**0.000252 > 0.0000167**), the answer of class “Play” is “Yes”.

For **[Outlook, Temp, Humidity, Windy]** = **[Rainy, 73, 90, False]**

First, we have to calculate f(tempurature=73|yes), f(humidity=90|yes), f(tempurature=73|no) and f(humidity=90|no)

f(tempurature=73|yes) = $\frac{1}{\sqrt{2\pi}6.2}e^{-\frac{(73-73)^2}{2*6.2^2}}$ = 0.0644

f(humidity=90|yes) = $\frac{1}{\sqrt{2\pi}10.2}e^{-\frac{(90-79.1)^2}{2*10.2^2}}$ = 0.0221 f(tempurature=73|no) = $\frac{1}{\sqrt{2\pi}7.9}e^{-\frac{(73-74.6)^2}{2*7.9^2}}$ = 0.0495

f(humidity=90|no) = $\frac{1}{\sqrt{2\pi}9.7}e^{-\frac{(90-86.2)^2}{2*9.7^2}}$ = 0.0381

Next, we calculate the likelihood of two class “Yes” and “No”:

For “Yes” = $\frac{1}{3}$ x 0.0644 x 0.0221 x $\frac{2}{3}$ x $\frac{9}{14}$ = 0.000210

For “No” = 0.4 x 0.0495 x 0.0381 x 0.4 x $\frac{5}{14}$ = 0.000108

Because Likelihood of Yes > Likelihood of No (**0.000210 > 0.000108**), the answer of class “Play” is “Yes”.

For **[Outlook, Temp, Humidity, Windy]** = **[Sunny, 80, 85, False]**

First, we have to calculate f(tempurature=80|yes), f(humidity=85|yes), f(tempurature=80|no) and f(humidity=85|no)

f(tempurature=80|yes) = $\frac{1}{\sqrt{2\pi}6.2}e^{-\frac{(80-73)^2}{2*6.2^2}}$ = 0.0340

f(humidity=85|yes) = $\frac{1}{\sqrt{2\pi}10.2}e^{-\frac{(85-79.1)^2}{2*10.2^2}}$ = 0.0331

f(tempurature=80|no) = $\frac{1}{\sqrt{2\pi}7.9}e^{-\frac{(80-74.6)^2}{2*7.9^2}}$ = 0.0400

f(humidity=85|no) = $\frac{1}{\sqrt{2\pi}9.7}e^{-\frac{(85-86.2)^2}{2*9.7^2}}$ = 0.0408

Next, we calculate the likelihood of two class “Yes” and “No”:

For “Yes” = $\frac{2}{9}$ x 0.0340 x 0.0331 x $\frac{9}{14}$ x $\frac{2}{3}$ = 0.000107

For “No” = 0.6 x 0.0400 x 0.0408 x $\frac{5}{14}$ x 0.4 = 0.000140

Because Likelihood of Yes > Likelihood of No (**0.000107** < **0.000140**), the answer of class “Play” is “No”.

For **[Outlook, Temp, Humidity, Windy]** = **[Null, 90, 85, False]**

First, we have to calculate f(tempurature=90|yes), f(humidity=85|yes), f(tempurature=90|no) and f(humidity=85|no)

f(tempurature=90|yes) = $\frac{1}{\sqrt{2\pi}6.2} e^{-\frac{(90-73)^2}{2*6.2^2}}$ = 0.00150

f(humidity=85|yes) = $\frac{1}{\sqrt{2\pi}10.2} e^{-\frac{(85-79.1)^2}{2*10.2^2}}$ = 0.0331

f(tempurature=90|no) = $\frac{1}{\sqrt{2\pi}7.9} e^{-\frac{(90-74.6)^2}{2*7.9^2}}$ = 0.0076

f(humidity=85|no) = $\frac{1}{\sqrt{2\pi}9.7} e^{-\frac{(85-86.2)^2}{2*9.7^2}}$ = 0.0408

Next, we calculate the likelihood of two class "Yes" and "No":

For "Yes" = 0.00150 x 0.0331 x $\frac{9}{14}$ x $\frac{2}{3}$ = 0.0000213

For "No" = 0.0076 x 0.0408 x $\frac{5}{14}$ x 0.4 = 0.0000443

Because Likelihood of Yes > Likelihood of No (**0.0000213** < **0.0000443**), the answer of class "Play" is "No".

So, the answer is :

outlook	Temp	Humidity	Windy	Play
Overcast	66	80	FALSE	Yes
Rainy	73	90	FALSE	Yes
Sunny	80	85	FALSE	No
?	90	85	?	No

3) Implement the program using GassianNB in scikit-learn library. The program requires 2 parameters:

- Trainset
- Testset

The program reports the classification results (accuracy, confusion matrix) for 5 datasets:

- Iris (.trn: trainset, .tst: tests)
- Optics (.trn: trainset, .tst: tests)
- Letter (.trn: trainset, .tst: tests)
- Leukemia (.trn: trainset, .tst: tests)
- Fp (.trn: trainset, .tst: tests)

Code used for **import libraries** and **read files**:

```
import numpy as np
import csv
from csv import reader
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score

#Funciton for read file
def load_csv(filename):
    dataset = list()
    with open(filename, 'r') as file:
        csv_reader = csv.reader(file)
        for row in csv_reader:
            if not row:
                continue
            dataset.append(row)
    return dataset

```

Code for implementing Naïve Bayes:

```python
def NaiveBayes(filename1, filename2):
    print("Implement ", filename1, ": ")
    # This is my code to read file
    trainset = load_csv(filename1)
    testset = load_csv(filename2)
    temp1 = [i[0:-1] for i in trainset]
    yTrain = np.array(list(map(int, [i[-1] for i in trainset])))
    temp2 = [i[0:-1] for i in testset]
    yTest = np.array(list(map(int, [i[-1] for i in testset])))
    xTrain = np.array([list(map(float, i)) for i in temp1])
    xTest = np.array([list(map(float, i)) for i in temp2])

    #Code for implement naive Bayes
    model = GaussianNB()
    model.fit(xTrain,yTrain)
    y_predicted = model.predict(xTest)
    score = accuracy_score(yTest, y_predicted)*100

    #Code for printing the result
    print("Accuracy: ", score, "%")
    print("Confusion Matrix: \n" , metrics.confusion_matrix(yTest,y_predicted))
    print("Classification Report: \n" , metrics.classification_report(yTest,y_predicted))
```

Calling the **NaiveBayes** functions:

```python
NaiveBayes('iris.trn', 'iris.tst')
NaiveBayes('let.trn', 'let.tst')
NaiveBayes('opt.trn', 'opt.tst')
NaiveBayes('fp.trn', 'fp.tst')
NaiveBayes('ALLAML.trn', 'ALLAML.tst')
```

Result:

Iris:

```
Implement  iris.trn :
Accuracy:  92.0 %
Confusion Matrix:
 [[17  0  0]
 [ 0 15  0]
 [ 0  4 14]]
Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00        17
           1       0.79      1.00      0.88        15
           2       1.00      0.78      0.88        18

    accuracy                           0.92        50
   macro avg       0.93      0.93      0.92        50
weighted avg       0.94      0.92      0.92        50
```

Letter:

```
Implement  let.trn :
Accuracy:  63.15631563156315 %
Confusion Matrix:
 [[235   0   0   1   0   0   0   3   0   0   2   0   6   2   0   0   5   2
   10   0   1   0   1   3   3   0]
 [  0 163   0  11   0   1   0   2  31   2   1   0   3   0   1   0   1  21
    0   0   0   0   2   1   0   0]
 [  0   0 162   0   4   0  16   0   0   0  22   0   5   0   5   0   6   1
    2   1   2   0   0   0   0   0]
 [  1  20   0 192   0   1   0   1  15  11   4   0   3   1  16   0   0   6
    5   0   0   0   0   1   0   0]
 [  0   3   5   2  90   0  45   0  16   0  13   0   0   0   0   0  17   1
   11   4   1   0   0  45   1   8]
 [  0  11   0  11   0 206   6   2   1   0   0   0   0   3   0   9   3   2
    2   7   0   0   3   0   3   0]
 [  3   7  41   2   0   2 143   1   4   0   7   0   4   0   4   0  17   7
    9   0   0   0  12   0   0   0]
 [  1  11   0  17   0   5   4  72   1   0  11   0   5   2  41   0   0  18
    1   0   5   0   5  26   5   0]
 [  0   9   0  15   4   8   0   0 196  15   0   1   0   0   0   3   3   0
   11   1   0   0   0   1   0   2]
 [  0   5   0  10   0   4   0   0  13 185   0   0   0   0   2   2   1   2
   13   0   0   0   0   2   0   0]
 [  1   8   1   8  23   0   7   2   2   0 117   0   5   3   0   0   1  28
    0   3   4   0   0  29   1   0]
```

```
[  0   5   0   0   4   0   8   0   0  14  16 207   0   0   0   0   9   2
   2   0   0   0   0   3   0   0]
[  7   5   0   0   0   0   0   2   0   0   5   0 212   0   1   0   1   1
   0   0   1   0  10   0   0   0]
[  2   3   0   8   0   0   0  24   1   0   2   0   8 172  15   1   0   7
   0   0   5  10  14   0   0   0]
[  3   3   1   8   0   0  12   4  12   0   7   0   9   4 155   1   5   9
   0   0   0   0   4   0   0   0]
[  0   3   0  13   0  21   6   2   0   0   0   0   0   2   1 197   2   0
   1   0   0   1  13   0   4   0]
[  6   5   0   3   0   0   5   0   4   0   1   2   3   0  62   0 155   8
  22   0   0   0   2   1   1   0]
[  1  27   0  20   0   0   0   8   4   7   8   0   6   1   7   0   3 177
   0   0   0   0   1   1   0   0]
[ 16  34   1   7   5   5   4   2  21   1   2   2   0   0   0   1   7   6
  77   8   1   1   0  39   1  23]
[  0   0   0   1   2  14   7   1   0   0   8   0   1   0   0   0   1   2
   5 176   0   7   0   7  11   1]
[  0   0   5   2   0   0   4   8   0   0  14   1  18   5  12   0   1   0
   0   0 199   1   3   1   0   0]
[  0   6   0   0   0   3   2   1   0   0   0   0   3   1   0   6   0   1
   1   0   0 192  18   0   4   0]
[  0   9   0   0   0   0   0   1   0   0   0   0   9   4   4   1   0   0
   0   0   0  24 189   0   0   0]
```

```
[  0  16   0   5   4   0   0   0  19   2  10   2   0   0  56   0   1   1
   8   7   8   0   0 111   2   9]
[  0   0   0   3   0  18   0   0   0   0   0   0   4   1   1   2   8   0
  10  51   3  61   8   0  82   0]
[  3   0   0   2   7   3   0   0  26   6   4   5   0   0   0   0   1   5
  36   6   0   0   0   6   1 148]]
```

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

           0       0.84      0.86      0.85       274
           1       0.46      0.68      0.55       240
           2       0.75      0.72      0.73       226
           3       0.56      0.69      0.62       277
           4       0.63      0.34      0.44       262
           5       0.71      0.77      0.74       269
           6       0.53      0.54      0.54       263
           7       0.53      0.31      0.39       230
           8       0.54      0.73      0.62       269
           9       0.76      0.77      0.77       239
          10       0.46      0.48      0.47       243
          11       0.94      0.77      0.84       270
          12       0.70      0.87      0.77       245
          13       0.86      0.63      0.73       272
          14       0.40      0.65      0.50       237
          15       0.88      0.74      0.81       266
          16       0.62      0.55      0.59       280
          17       0.58      0.65      0.61       271
          18       0.34      0.29      0.31       264
          19       0.67      0.72      0.69       244
          20       0.87      0.73      0.79       274
          21       0.65      0.81      0.72       238
```
`

```
          22       0.66      0.78      0.72       241
          23       0.40      0.43      0.41       261
          24       0.69      0.33      0.44       252
          25       0.77      0.57      0.66       259

    accuracy                           0.63      6666
   macro avg       0.65      0.63      0.63      6666
weighted avg       0.65      0.63      0.63      6666
```

Opt:

```
Implement  opt.trn :
Accuracy:  78.63105175292154 %
Confusion Matrix:
 [[177   0   0   0   0   0   0   0   1   0]
 [  0 129  14   0   0   0   7   0  23   9]
 [  0   7 141   1   0   1   1   1  24   1]
 [  1   1   2 135   0   2   0   8  33   1]
 [ 10  31   0   0  93   2  12  26   7   0]
 [  2   1   4   5   0 131   1   5  31   2]
 [  2   1   0   0   0   0 175   0   3   0]
 [  0   0   1   0   0   0   0 176   1   1]
 [  0   6   0   0   0   1   0   1 165   1]
 [  5   4   1  21   0   0   0   6  52  91]]
```

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

           0       0.90      0.99      0.94       178
           1       0.72      0.71      0.71       182
           2       0.87      0.80      0.83       177
           3       0.83      0.74      0.78       183
           4       1.00      0.51      0.68       181
           5       0.96      0.72      0.82       182
           6       0.89      0.97      0.93       181
           7       0.79      0.98      0.88       179
           8       0.49      0.95      0.64       174
           9       0.86      0.51      0.64       180

    accuracy                           0.79      1797
   macro avg       0.83      0.79      0.79      1797
weighted avg       0.83      0.79      0.79      1797
```

Fp:

```
Implement  fp.trn :
Accuracy:  75.0 %
Confusion Matrix:
 [[29  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  4  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  4  8  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  7  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  9  0  0  0  0  0  0  0  0  0  0]
 [ 0  1  0  0  0 13  0  0  0  0  0  0  0  0  0]
 [ 0  3  0  0  0  0  7  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0 11  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  7  0  0  0  0  0  0]
 [ 0  0  0  2  0  0  0  0  0  2  0  0  0  0  0]
 [ 0  7  0  0  0  0  0  0  0  0  3  0  0  0  0]
 [ 0  2  0  0  0  0  0  0  0  0  0  7  0  0  0]
 [ 0  1  0  2  0  0  1  0  0  0  0  0  6  0  0]
 [ 0  2  0  3  1  0  1  0  0  0  0  0  0  3  0]
 [ 0  6  0  2  0  0  2  0  0  0  0  0  0  0  4]]
```

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

           1       1.00      1.00      1.00        29
           2       0.13      1.00      0.24         4
           3       1.00      0.67      0.80        12
           4       0.44      1.00      0.61         7
           5       0.90      1.00      0.95         9
           6       1.00      0.93      0.96        14
           7       0.64      0.70      0.67        10
           8       1.00      1.00      1.00        11
           9       1.00      1.00      1.00         7
          10       1.00      0.50      0.67         4
          11       1.00      0.30      0.46        10
          12       1.00      0.78      0.88         9
          13       1.00      0.60      0.75        10
          14       1.00      0.30      0.46        10
          15       1.00      0.29      0.44        14

    accuracy                           0.75       160
   macro avg       0.87      0.74      0.73       160
weighted avg       0.93      0.75      0.77       160
```

Leukemia

```
Implement  ALLAML.trn :
Accuracy:  91.17647058823529 %
Confusion Matrix:
 [[13  1]
 [ 2 18]]
Classification Report:
              precision    recall  f1-score   support

          -1       0.87      0.93      0.90        14
           1       0.95      0.90      0.92        20

    accuracy                           0.91        34
   macro avg       0.91      0.91      0.91        34
weighted avg       0.91      0.91      0.91        34
```