Exercice 1

auteurs : Costa Pedro, Delabays Louis, Guerne Jonathan

Partie A Histogramme

implémenter un classifieur basé sur Bayes en utilisant un histogramme pour éstimer la vraisemblance.

```
import numpy as np
import matplotlib.pyplot as plt
from math import pi, pow, exp
```

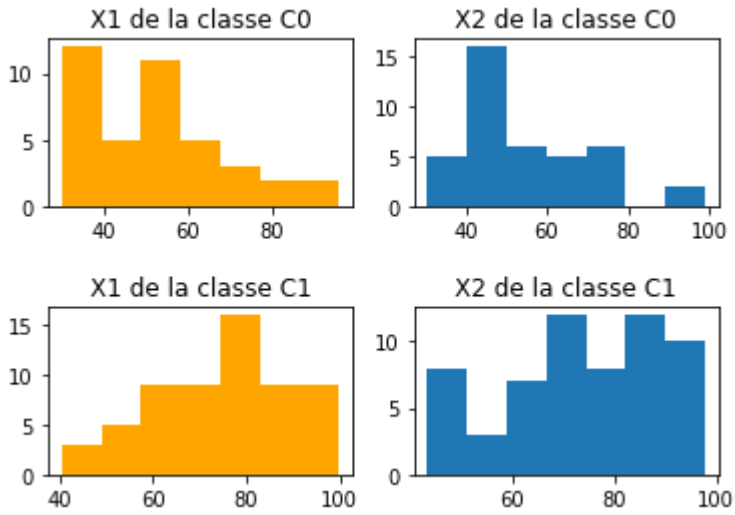
```
train_data = np.loadtxt('ex1-data-train.csv',delimiter=",")

nb_data = len(train_data)

print("Training set : {}".format(train_data.shape))

class_0 = train_data[train_data[:,2] == 0]
class_1 = train_data[train_data[:,2] == 1]


p_c0 = len(class_0)/nb_data
p_c1 = len(class_1)/nb_data

print("Prior class 0 P(C0) : {}".format(p_c0))
print("Prior class 1 P(C1) : {}".format(p_c1))
```

```
Training set : (100, 3)
Prior class 0 P(C0) : 0.4
Prior class 1 P(C1) : 0.6
```

Impression des histogrammes

```
f, axarr = plt.subplots(2,2)

axarr[0, 0].set_title('X1 de la classe C0')
axarr[0,0].hist(class_0[:,0], bins='auto', color='orange')

axarr[0, 1].set_title('X2 de la classe C0')
axarr[0,1].hist(class_0[:,1], bins='auto')

axarr[1, 0].set_title('X1 de la classe C1')
```

```
axarr[1,0].hist(class_1[:,0], bins='auto', color='orange')

axarr[1,1].set_title('X2 de la classe C1')
axarr[1,1].hist(class_1[:,1], bins='auto')

f.subplots_adjust(hspace=0.6)

plt.show()
```



Mesure de la vraisenblance

```
def likelihoodHist(x, histValues, edgeValues):
    """Return the likelihood of a given histogram"""
    nb_values = np.sum(histValues)

    # if the input data is outside of the given edges
    if x < edgeValues[0] or x > edgeValues[-1]:
        return 0

    # if the input is on the top edge border
    if x == edgeValues[-1] :
        return histValues[-1]/nb_values

    sub_edges = edgeValues[x >= edgeValues]
    index = len(sub_edges)-1
    return histValues[index]/nb_values

hist,edges = np.histogram(class_0[:,0],bins='auto')

print(np.sum([likelihoodHist(i,hist,edges) for i in edges[:-1]]))
```

```
1.0
```

Implémentation de la classification avec Bayes

```
test_data = np.loadtxt('ex1-data-test.csv',delimiter=",")

test_data_x1 = test_data[:,0]
test_data_x2 = test_data[:,1]
test_data_xs = test_data[:,:2]

test_data_y = test_data[:,2]

print("Testing set : {}".format(test_data.shape))

preds = {}
```

```
Testing set : (100, 3)
```

Uniquement les features X_1

```
# get histogram of both classes
hist_0,edges_0 = np.histogram(class_0[:,0],bins='auto')
hist_1,edges_1 = np.histogram(class_1[:,0],bins='auto')

preds['x1'] = [np.argmax([likelihoodHist(x,hist_0,edges_0) *
p_c0,likelihoodHist(x,hist_1,edges_1) * p_c1]) for x in test_data_x1]
```

Uniquement les features X_2

```
# get histogram of both classes
hist_0,edges_0 = np.histogram(class_0[:,1],bins='auto')
hist_1,edges_1 = np.histogram(class_1[:,1],bins='auto')

preds['x2'] = [np.argmax([likelihoodHist(x,hist_0,edges_0) *
p_c0,likelihoodHist(x,hist_1,edges_1) * p_c1]) for x in test_data_x2]
```

X_1 et X_2 avec hypothèse d'indépendances des features

```
# get histogram of both classes
hist_0x1,edges_0x1 = np.histogram(class_0[:,0],bins='auto')
hist_0x2,edges_0x2 = np.histogram(class_0[:,1],bins='auto')
```

```
hist_1x1,edges_1x1 = np.histogram(class_1[:,0],bins='auto')
hist_1x2,edges_1x2 = np.histogram(class_1[:,1],bins='auto')

preds['x1 and x2'] = [np.argmax([
    likelihoodHist(x,hist_0x1,edges_0x1)* p_c0 *
    likelihoodHist(y,hist_0x2,edges_0x2) * p_c0,
    likelihoodHist(x,hist_1x1,edges_1x1)* p_c1 *
    likelihoodHist(y,hist_1x2,edges_0x2) * p_c1])
                    for x,y in test_data_xs]
```

Calcul des performances

```
for key, pred in preds.items():

    performance = len(test_data_y[test_data_y == pred])/len(test_data_y)
    performance *= 100.0

    print("prediction using {} has {} % of accuracy".
          format(key,performance))
```

```
prediction using x1 has 70.0 % of accuracy
prediction using x2 has 73.0 % of accuracy
prediction using x1 and x2 has 81.0 % of accuracy
```

Partie B distribution gausienne univariée

```
# likelihood for the univariate gaussian
def likelihood_UG(x, mean, var):
    part1 = 1/pow(2 * pi * var,0.5)
    part2 = exp(-1*1/(2*var)*pow(x-mean,2))

    return part1 * part2

mean = np.mean(class_0[:,0])
var = np.var(class_0[:,0])

likelihood_UG(50,mean,var)
```

```
0.02286479268492325
```

```
print("Testing set : {}".format(test_data.shape))

preds = {}

mean_0x1 = np.mean(class_0[:,0])
var_0x1 = np.mean(class_0[:,0])

mean_0x2 = np.mean(class_0[:,1])
var_0x2 = np.mean(class_0[:,1])


mean_1x1 = np.mean(class_1[:,0])
var_1x1 = np.mean(class_1[:,0])

mean_1x2 = np.mean(class_1[:,1])
var_1x2 = np.mean(class_1[:,1])
```

```
Testing set : (100, 3)
```

```
preds['x1'] = [np.argmax(
    [likelihood_UG(x,mean_0x1,var_0x1) * p_c0,
     likelihood_UG(x,mean_1x1,var_1x1)* p_c1])
               for x in test_data_x1]
```

```
preds['x2'] = [np.argmax(
    [likelihood_UG(x,mean_0x2,var_0x2) * p_c0,
     likelihood_UG(x,mean_1x2,var_1x2) * p_c1])
               for x in test_data_x2]
```

```
preds['x1 and x2'] = [np.argmax(
    [likelihood_UG(x,mean_0x1,var_0x1)* p_c0 *
     likelihood_UG(y,mean_0x2,var_0x2) * p_c0,

     likelihood_UG(x,mean_1x1,var_1x1)* p_c1 *
     likelihood_UG(y,mean_1x2,var_1x2) * p_c1])
               for x,y in test_data_xs]
```

```
for key, pred in preds.items():

    performance = len(test_data_y[test_data_y == pred])/len(test_data_y)
    performance *= 100.0
```

```
    print("prediction using {} has {} % of accuracy".
          format(key,performance))
```

```
prediction using x1 has 70.0 % of accuracy
prediction using x2 has 78.0 % of accuracy
prediction using x1 and x2 has 86.0 % of accuracy
```