Practical 7 Naïve Bayes

import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
from sklearn.preprocessing import StandardScaler  
from imblearn.over\_sampling import RandomOverSampler

cols = ["fLength", "fWidth", "fSize", "fConc", "fConc1", "fAsym", "fM3Long", "fM3Trans", "fAlpha", "fDist", "class"]  
df = pd.read\_csv("magic04.data", names=cols)  
df.head()

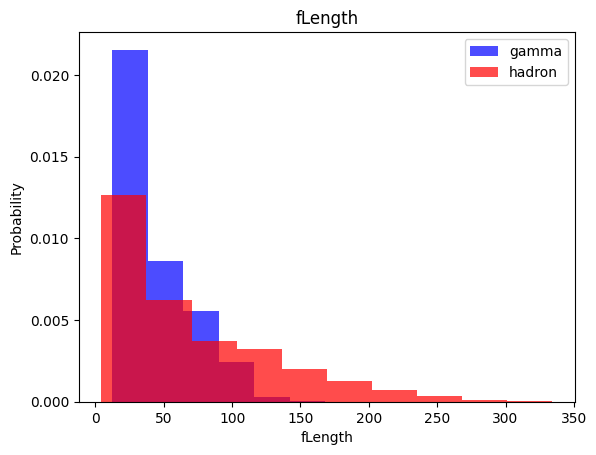
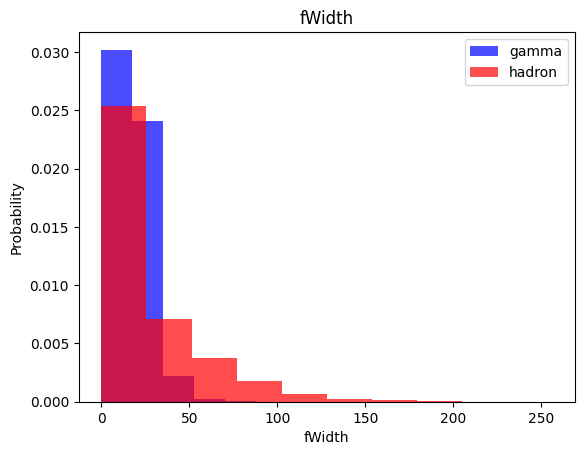
fLength fWidth fSize fConc fConc1 fAsym fM3Long fM3Trans \  
0 28.7967 16.0021 2.6449 0.3918 0.1982 27.7004 22.0110 -8.2027   
1 31.6036 11.7235 2.5185 0.5303 0.3773 26.2722 23.8238 -9.9574   
2 162.0520 136.0310 4.0612 0.0374 0.0187 116.7410 -64.8580 -45.2160   
3 23.8172 9.5728 2.3385 0.6147 0.3922 27.2107 -6.4633 -7.1513   
4 75.1362 30.9205 3.1611 0.3168 0.1832 -5.5277 28.5525 21.8393   
  
 fAlpha fDist class   
0 40.0920 81.8828 g   
1 6.3609 205.2610 g   
2 76.9600 256.7880 g   
3 10.4490 116.7370 g   
4 4.6480 356.4620 g

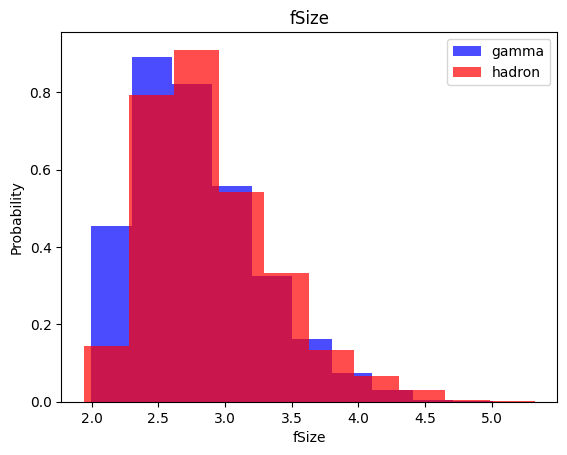
df["class"] = (df["class"] == "g").astype(int)

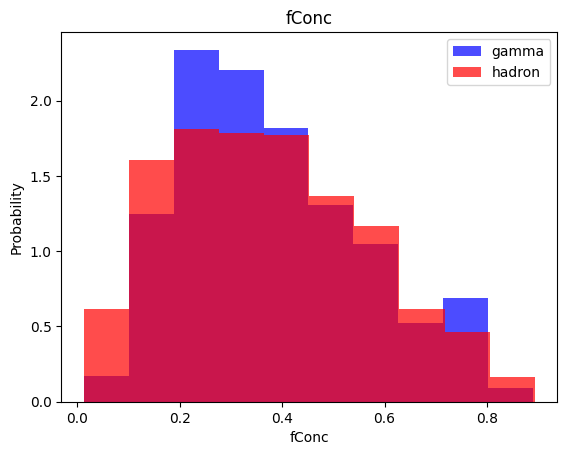
df.head()

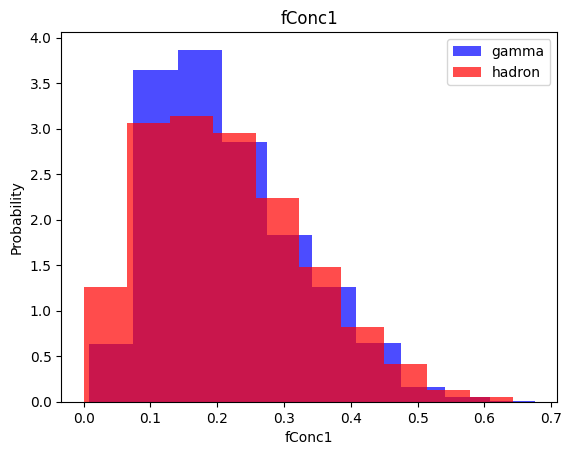
fLength fWidth fSize fConc fConc1 fAsym fM3Long fM3Trans \  
0 28.7967 16.0021 2.6449 0.3918 0.1982 27.7004 22.0110 -8.2027   
1 31.6036 11.7235 2.5185 0.5303 0.3773 26.2722 23.8238 -9.9574   
2 162.0520 136.0310 4.0612 0.0374 0.0187 116.7410 -64.8580 -45.2160   
3 23.8172 9.5728 2.3385 0.6147 0.3922 27.2107 -6.4633 -7.1513   
4 75.1362 30.9205 3.1611 0.3168 0.1832 -5.5277 28.5525 21.8393   
  
 fAlpha fDist class   
0 40.0920 81.8828 1   
1 6.3609 205.2610 1   
2 76.9600 256.7880 1   
3 10.4490 116.7370 1   
4 4.6480 356.4620 1

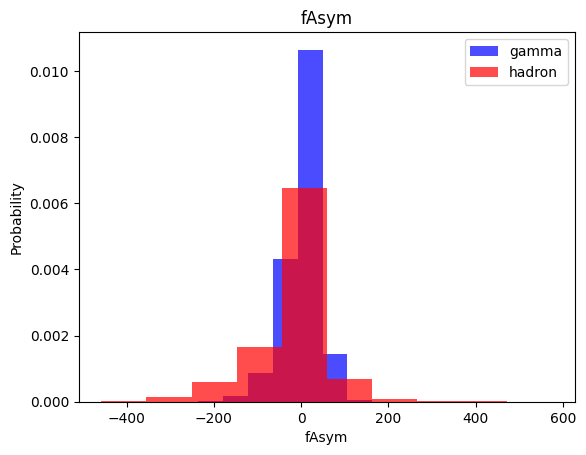
for label in cols[:-1]:  
 plt.hist(df[df["class"]==1][label], color='blue', label='gamma', alpha=0.7, density=True)  
 plt.hist(df[df["class"]==0][label], color='red', label='hadron', alpha=0.7, density=True)  
 plt.title(label)  
 plt.ylabel("Probability")  
 plt.xlabel(label)  
 plt.legend()  
 plt.show()

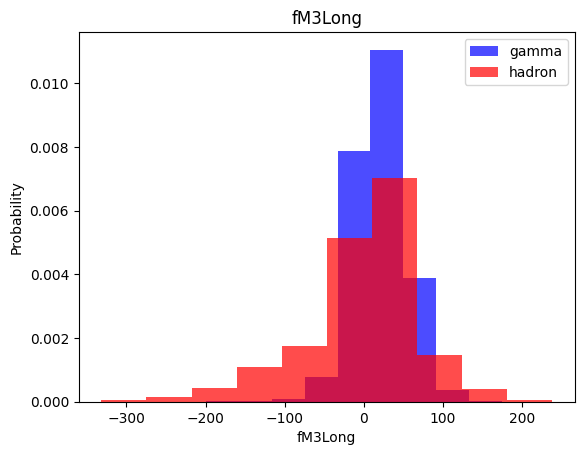


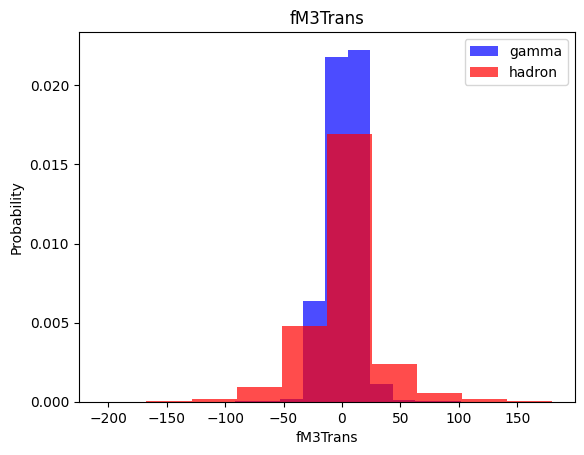


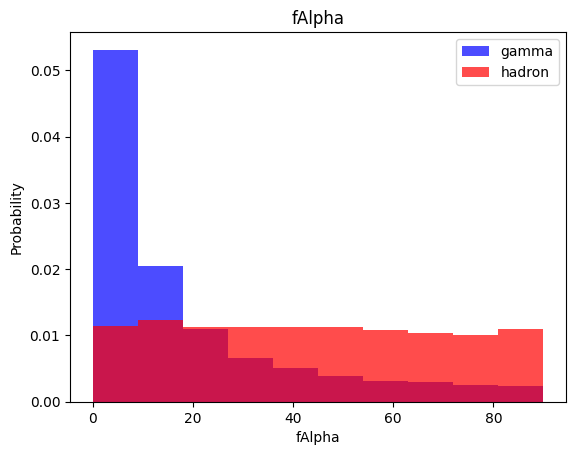


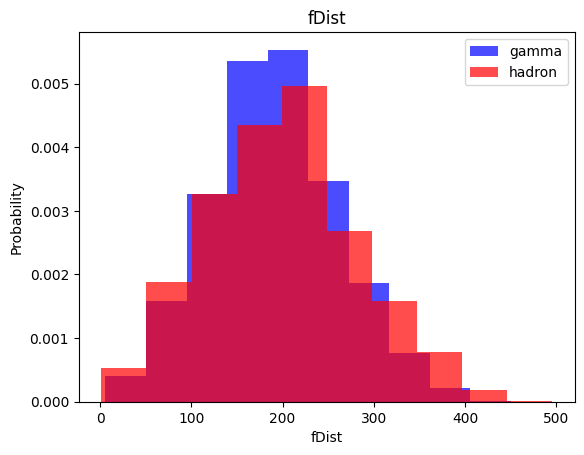












train, valid, test = np.split(df.sample(frac=1), [int(0.6\*len(df)), int(0.8\*len(df))])

c:\Users\91885\AppData\Local\Programs\Python\Python311\Lib\site-packages\numpy\core\fromnumeric.py:57: FutureWarning: 'DataFrame.swapaxes' is deprecated and will be removed in a future version. Please use 'DataFrame.transpose' instead.  
 return bound(\*args, \*\*kwds)

def scale\_dataset(dataframe, oversample=False):  
 X = dataframe[dataframe.columns[:-1]].values  
 y = dataframe[dataframe.columns[-1]].values  
  
 scaler = StandardScaler()  
 X = scaler.fit\_transform(X)  
  
 if oversample:  
 ros = RandomOverSampler()  
 X, y = ros.fit\_resample(X, y)  
  
 data = np.hstack((X, np.reshape(y, (-1, 1))))  
  
 return data, X, y

train, X\_train, y\_train = scale\_dataset(train, oversample=True)  
valid, X\_valid, y\_valid = scale\_dataset(valid, oversample=False)  
test, X\_test, y\_test = scale\_dataset(test, oversample=False)

from sklearn.metrics import classification\_report

from sklearn.naive\_bayes import GaussianNB

nb\_model = GaussianNB()  
nb\_model = nb\_model.fit(X\_train, y\_train)

y\_pred = nb\_model.predict(X\_test)  
print(classification\_report(y\_test, y\_pred))

precision recall f1-score support  
  
 0 0.72 0.39 0.51 1364  
 1 0.73 0.92 0.81 2440  
  
 accuracy 0.73 3804  
 macro avg 0.73 0.65 0.66 3804  
weighted avg 0.73 0.73 0.70 3804