DSC-530: Predictive Modeling

Week 4 Assignment: Predictive Modeling Using Naïve Bayes Classification

Author: Jonathan Ibifubara Pollyn

College of Science, Engineering and Technology, Grand Canyon University

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import random
import math

from scipy import stats
from scipy.special import factorial

import pandas as pd
import numpy as np
```

In [2]:
```
class NaiveBayes_Classifier:

    def __init__(self, XNV, yNV):



        self.XNV, self.yNV =XNV, yNV

        self.ND = len(self.XNV)

        self.dimd = len(self.XNV[0])

        self.attrbt = [[] for _ in range(self.dimd)]

        self.output_result = {}

        self.datadt = []


        for i in range(len(self.XNV)):
            for j in range(self.dimd):

                if not self.XNV[i][j] in self.attrbt[j]:
                    self.attrbt[j].append(self.XNV[i][j])


            if not self.yNV[i] in self.output_result.keys():
                self.output_result[self.yNV[i]] = 1

            else:
                self.output_result[self.yNV[i]] += 1

            self.datadt.append([self.XNV[i], self.yNV[i]])



    def classifiers(self, entrydata):

        Solution = None
        MaxResu = -1

        for yNV in self.output_result.keys():

            probable = self.output_result[yNV]/self.ND

            for i in range(self.dimd):
                val = [xd for xd in self.datadt if xd[0][i] == entrydata[i] and xd[1] == yNV]
                nd = len(val)
                probable *= nd/self.ND


            if probable > MaxResu:
                MaxResu = probable
                Solution = yNV

        return Solution
```

In [4]:
```
#Importing the training and test data
framingham_train = pd.read_csv('C:/School/DSC-530/DataSets/framingham_nb_training.csv')
framingham_test = pd.read_csv('C:/School/DSC-530/DataSets/framingham_nb_test.csv')
```

In [5]:
```
print('\n------Contingency tables Educ and Death------')
data_crosstab_Educ_Death =pd.crosstab(index=framingham_train['Educ'], columns=framingham_train['Death'])
print(data_crosstab_Educ_Death)

print('\n-----Contingency tables Sex and Death--------')
data_crosstab_SeXDeath =pd.crosstab(index=framingham_train['Sex'], columns=framingham_train['Death'])
print(data_crosstab_SeXDeath)
```

```
------Contingency tables Educ and Death------
Death    0    1
Educ
1      173  287
2      146  135
3       84   80
4       47   48

-----Contingency tables Sex and Death--------
Death    0    1
Sex
1      184  308
2      266  242
```

In [7]:
```
XS = framingham_train.iloc[:, [0, 1]].values
YS = framingham_train.iloc[:, -1].values


print('\n1.The probability a randomly selected person is alive or is dead.\n')
dfLive = framingham_train

td=dfLive['Death'].sum()

sd = (dfLive['Death']==0).sum()

p=round(sd/td, 1)

print('Propbability of death : ' + str(p))


sa = (dfLive['Death']==1).sum()

p=round(sa/td, 1)

print('Propbability of live : ' + str(p))
dfLive = dfLive.sample(n=2)
print(dfLive)
```

```
1.The probability a randomly selected person is alive or is dead.

Propbability of death : 0.8
Propbability of live : 1.0
     Sex  Educ  Death
241    2     2      0
494    2     4      1
```

In [12]:
```
print('\n2.     The probability a randomly selected person is a male.\n')
dfMale = pd.read_csv('C:/School/DSC-530/DataSets/framingham_nb_training.csv', usecols = ['Sex']).isin([1])
td=dfMale['Sex'].sum()

sd = (dfMale['Sex']==1).sum()

p=round(sd/td, 1)

print('Propbability of Male : ' + str(p))


dfMale = dfMale.sample()
print(dfMale)
```

```
2.      The probability a randomly selected person is a male.

Propbability of Male : 1.0
      Sex
798  True
```

In [13]:
```
print('\n3.     The probability a randomly selected person has an Educ value of 3.\n')
dfEduc = pd.read_csv('C:/School/DSC-530/DataSets/framingham_nb_training.csv', usecols = ['Educ']).isin([3])
dfEduc = dfEduc.sample()
print(dfEduc)
tedu=dfLive['Educ'].sum()

sedu = (dfLive['Educ']==3).sum()

pedu=round(sedu/tedu, 1)

print('Propbability of Educ value of 3 : ' + str(pedu))
```

```
3.      The probability a randomly selected person has an Educ value of 3.

      Educ
383  False
Propbability of Educ value of 3 : 0.0
```

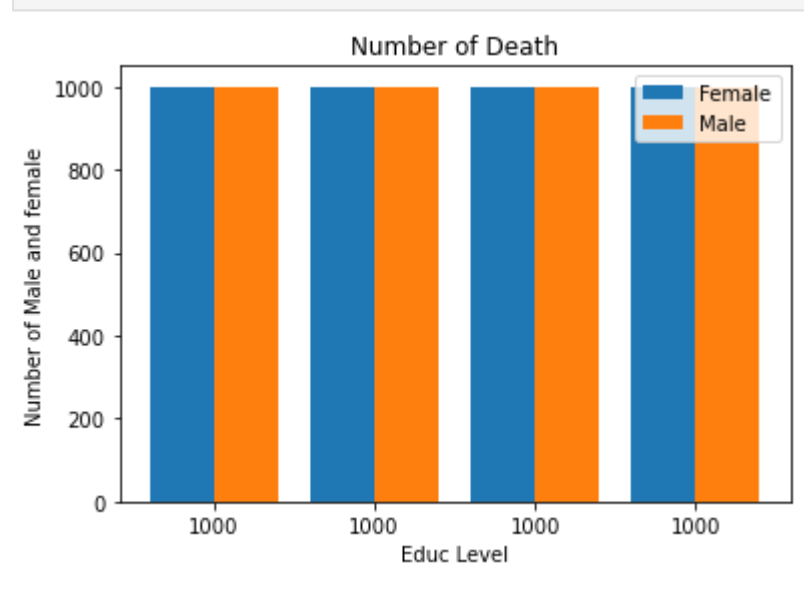
In [14]:
```
print('\n4.----------------------- The probabilities that a dead person is male with education level 1, and tha
dfdpm = pd.read_csv('C:/School/DSC-530/DataSets/framingham_nb_training.csv')

td=dfdpm['Death'].sum()
ts=dfdpm['Sex'].sum()
tE=dfdpm['Educ'].sum()

t=td+ts+tE

sd = (dfdpm['Death']==0).sum()
sx = (dfdpm['Sex']==2).sum()
sE = (dfdpm['Educ']==1).sum()

s=sd+ sx + sE


p=round(s/t, 1)

print('The probabilities that a dead person is male with education level 1 : ' + str(p))




sd = (dfdpm['Death']==1).sum()
sx = (dfdpm['Sex']==2).sum()
sE = (dfdpm['Educ']==1).sum()

s=sd+ sx + sE


pl=round(s/t, 1)

print('The probabilities living person is male with education level 1 : ' + str(pl))
```

```
4.----------------------- The probabilities that a dead person is male with education level 1, and that a livin
g person is male with education level 1.--------------------

The probabilities that a dead person is male with education level 1 : 0.4
The probabilities living person is male with education level 1 : 0.4
```

In [15]:
```
print('\n4.----------------------- The probabilities that a living person is female with education level 2, and
dfdpm = pd.read_csv('C:/School/DSC-530/DataSets/framingham_nb_training.csv')

td=dfdpm['Death'].sum()
ts=dfdpm['Sex'].sum()
tE=dfdpm['Educ'].sum()

t=td+ts+tE

sd = (dfdpm['Death']==0).sum()
sx = (dfdpm['Sex']==1).sum()
sE = (dfdpm['Educ']==2).sum()

s=sd+ sx + sE


p=round(s/t, 1)

print('The The probabilities that a living person is female with education level 2: ' + str(p))




sd = (dfdpm['Death']==1).sum()
sx = (dfdpm['Sex']==1).sum()
sE = (dfdpm['Educ']==2).sum()

s=sd+ sx + sE


pl=round(s/t, 1)

print('The probabilities dead person is female with education level 2 : ' + str(pl))
```

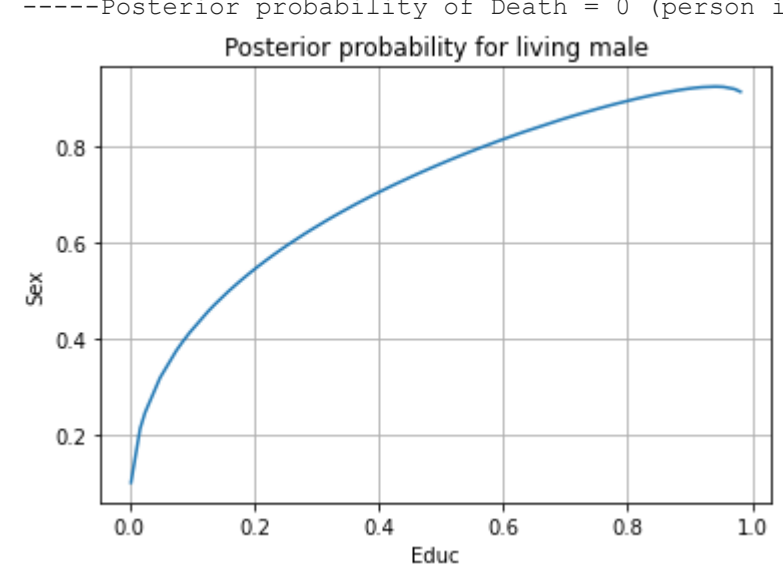
```
4.----------------------- The probabilities that a living person is female with education level 2, and that a d
ead person is female with education level 2.--------------------

The The probabilities that a living person is female with education level 2: 0.3
The probabilities dead person is female with education level 2 : 0.3
```

In [16]:
```
df = pd.read_csv('C:/School/DSC-530/DataSets/framingham_nb_training.csv')
X = [ ( df['Death']==1).count(),( df['Death']==0).count(),( df['Death']==1).count(),( df['Death']==0).count()]

YSex =[ (df['Sex']==1).count(),(df['Sex']==2).count(),(df['Sex']==1).count(),(df['Sex']==2).count()]
ZEduc =[(df['Educ']==1).count(),(df['Educ']==2).count(),(df['Educ']==3).count(),(df['Educ']==4).count()]
Xaxis = np.arange(len(X))

plt.bar(Xaxis - 0.2, YSex, 0.4, label = 'Female')
plt.bar(Xaxis + 0.2, ZEduc, 0.4, label = 'Male')

plt.xticks(Xaxis, X)
plt.xlabel("Educ Level")
plt.ylabel("Number of Male and female")
plt.title("Number of Death")
plt.legend()
plt.show()
```

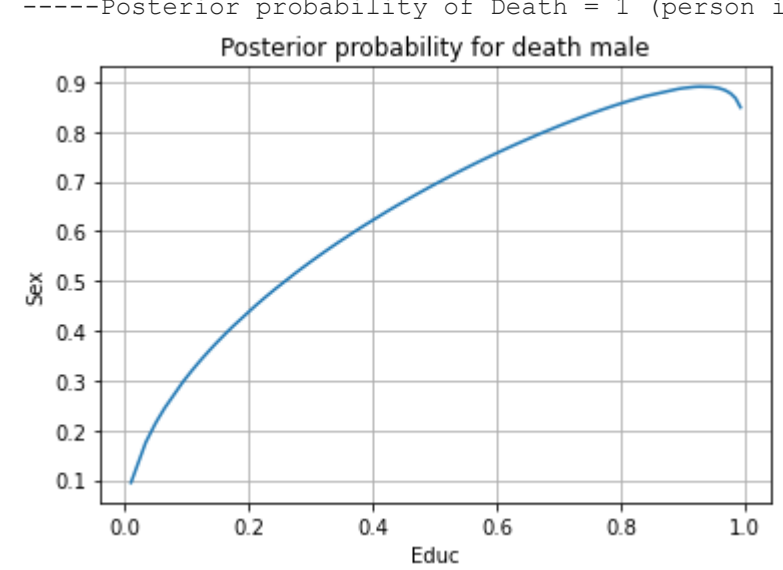


In [20]:
```
print('----------------------Posterior probability-----------------------------------')
Title=''
def likes(Educ, nd, xd):
    return (factorial(nd) / (factorial(xd) * factorial(nd - xd))) * (Educ ** xd) * ((1 - Educ) ** (nd - xd))

def Posterior_probability (priors, posterior, noccured, nevents):
    return pd.Series(map(lambda Educ: likes(Educ, nevents, noccured), priors))

def Model_Generation(nevents, pss):
    return np.random.binomial(nevents, pss)

def Run(dnoccured, dnevents, dndraws=100 ):
    dpriors = pd.Series(sorted(np.random.uniform(0, 1, size=dndraws)))
    dsimdata = [Model_Generation(dnevents ,pss) for pss in dpriors]
    dposteriorss = dpriors[list(map(lambda xd: xd == dnoccured, dsimdata))]
    Posteriorprobability = Posterior_probability (dpriors, dposteriorss, dnoccured, dnevents)


    fs, axs = plt.subplots(1)
    axs.plot(dpriors, Posteriorprobability)
    axs.set_xlabel("Educ")
    axs.set_ylabel("Sex")
    axs.grid()

    axs.set_title(Title)
    plt.show()
print('-----Posterior probability of Death = 0 (person is living) for a male with education level 1. -----')
df = pd.read_csv('C:/School/DSC-530/DataSets/framingham_nb_training.csv')
df=df[df['Death']==0]

esum=(df['Death']==0).sum()
e=(df['Educ']==1).sum()/esum
s=(df['Sex']==1).sum()/esum
Title="Posterior probability for living male"
Run( e,s)


print('-----Posterior probability of Death = 1 (person is dead) for a male with education level 1. -----')


def Model_Generation1(nevents1, pss1):
    return np.random.binomial(nevents1, int(pss1))

def Run1(dnoccured1, dnevents1, dndraws1=100 ):
    dpriors1 = pd.Series(sorted(np.random.uniform(0, 1, size=dndraws1)))
    dsimdata1 = [Model_Generation1(dnevents1 ,pss1) for pss1 in dpriors1]
    dposteriorss1 = dpriors1[list(map(lambda xd: xd == dnoccured1, dsimdata1))]
    Posteriorprobability1 = Posterior_probability (dpriors1, dposteriorss1, dnoccured1, dnevents1)


    fs, axs = plt.subplots(1)
    axs.plot(dpriors1, Posteriorprobability1)
    axs.set_xlabel("Educ")
    axs.set_ylabel("Sex")
    axs.grid()

    axs.set_title(Title)
    plt.show()


dfs = pd.read_csv('C:/School/DSC-530/DataSets/framingham_nb_training.csv')
dfs=dfs[dfs['Death']==1]

esum=(dfs['Death']==0).count()
e=(dfs['Educ']==1).sum()/esum
s=(dfs['Sex']==1).sum()/esum
Title="Posterior probability for death male"
Run1( e,s)

print('-------------------------------------------------------------------------------------')
print('-----Posterior probability of Death = 0 (person is living) for a female with education level 2. -----')
df = pd.read_csv('C:/School/DSC-530/DataSets/framingham_nb_training.csv')
df=df[df['Death']==0]

esum=(df['Death']==0).sum()
e=(df['Educ']==2).sum()/esum
s=(df['Sex']==2).sum()/esum
Title="Posterior probability for living female"
Run( e,s)

print('-------------------------------------------------------------------------------------')
print('-----Posterior probability of Death = 1 (person is dead) for a female with education level 2. -----')
df = pd.read_csv('C:/School/DSC-530/DataSets/framingham_nb_training.csv')
df=df[df['Death']==1]

esum=(df['Death']==1).sum()
e=(df['Educ']==2).sum()/esum
s=(df['Sex']==2).sum()/esum
Title="Posterior probability for Death female"
Run( e,s)

print('-------------------------------------------------------------------------------------')


from sklearn.model_selection import train_test_split
#X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 0)
Xtrain, Xtest, Ytrain, Ytest = train_test_split(XS, YS, test_size = 0.20, random_state = 0)

# Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
Xtrain = sc.fit_transform(Xtrain)
Xtest = sc.transform(Xtest)

#  Naive Bayes model
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(Xtrain, Ytrain)

# Predicting  results
Ypred = classifier.predict(Xtest)

from sklearn.metrics import confusion_matrix, accuracy_score
accuracy = accuracy_score(Ytest,Ypred)
print('Accuracy : ' +  str(accuracy))
confusionmatrix = confusion_matrix(Ytest, Ypred)
error_rate = 1 - accuracy
print('Error_rate : ' +  str(error_rate))



data = pd.read_csv('C:/School/DSC-530/DataSets/framingham_nb_training.csv')

yaxiz = list(map(lambda v: '1' if v == 1 else '0', data['Death'].values))


Xaxiz = data[['Death', 'Sex', 'Educ']].values


ytrain = yaxiz[:600]
yval = yaxiz[600:]
Xtrain = Xaxiz[:600]
Xval = Xaxiz[600:]



nbcNaiveBayes_Classifier = NaiveBayes_Classifier(Xtrain, ytrain)


total = len(yval)


living = 0
death = 0

for i in range(total):
    predict = nbcNaiveBayes_Classifier.classifiers(Xval[i])
#
    if yval[i] == predict:
        living += 1
    else:
        death += 1

#print('TOTAL :', total)

#print('ACCURACY:', living/total)



#Display result contingency table
data_crosstab_Educ_Death =pd.crosstab(index=data['Educ'], columns=data['Death'])
print(data_crosstab_Educ_Death)
```

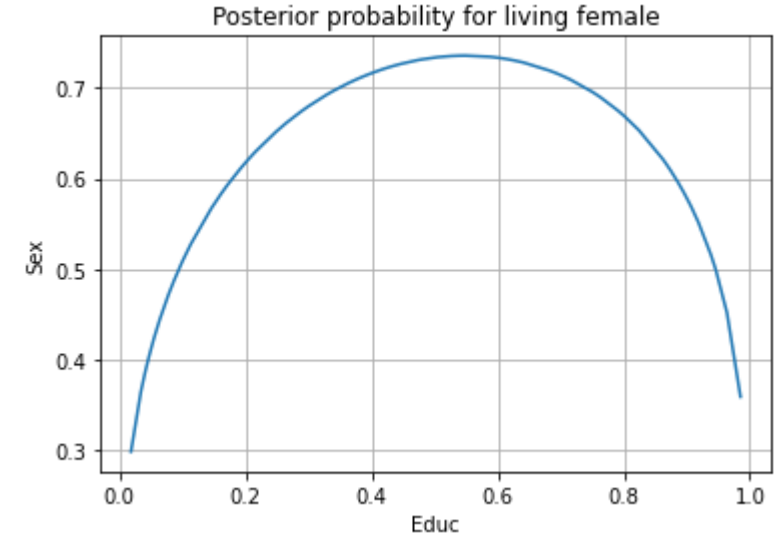
```
----------------------Posterior probability-----------------------------------
-----Posterior probability of Death = 0 (person is living) for a male with education level 1. -----
```



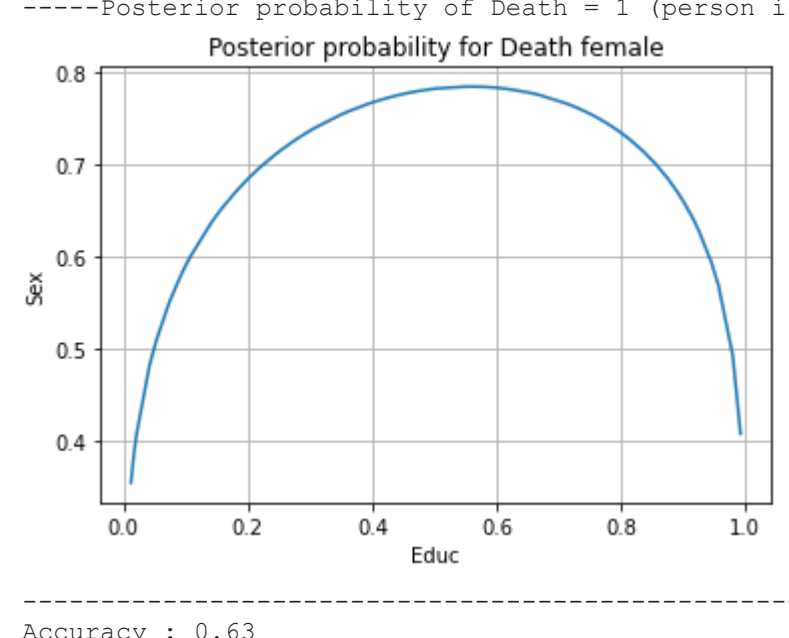

```
-----Posterior probability of Death = 1 (person is dead) for a male with education level 1. -----
```




```
-------------------------------------------------------------------------------------
-----Posterior probability of Death = 0 (person is living) for a female with education level 2. -----
```




```
-------------------------------------------------------------------------------------
-----Posterior probability of Death = 1 (person is dead) for a female with education level 2. -----
```




```
-------------------------------------------------------------------------------------
Accuracy : 0.63
Error_rate : 0.37
Death    0    1
Educ
1      173  287
2      146  135
3       84   80
4       47   48
```

In [ ]: