Gaussian estimate conditional PDFs

In [ ]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn
import warnings
from sklearn import model_selection
import gensim


fil_data=pd.read_excel('filtered_data.xlsx')
X_train, X_test, Y_train, Y_test = model_selection.train_test_split(fil_data[['text']
                                                                               fil_data[
print(X_train.shape)
```

In [ ]:

```
from gensim.test.utils import common_texts
from gensim.models.doc2vec import Doc2Vec, TaggedDocument

documents = [TaggedDocument(doc, [i]) for i, doc in enumerate(common_texts)]
model = Doc2Vec(documents, vector_size=100, window=1, min_count=1, workers=4,dm=1)
```

In [ ]:

```
from gensim.test.utils import get_tmpfile

fname = get_tmpfile("my_doc2vec_model")

model.save(fname)
model = Doc2Vec.load(fname)  # you can continue training with the loaded model!
```

In [ ]:

```
X_train_no=np.zeros((87144,500))

for i in range(87144):
    X_train_no[i,:]= model.infer_vector(np.asarray(X_train)[i])
```

In [ ]:

```
Y_train = np.array(Y_train);
Y_test = np.array(Y_test);
```

In [ ]:

```
Y_trainno=np.zeros((Y_train.size))
for i in range(Y_train.size):
    if Y_train[i]=="BUSINESS":
        Y_trainno[i]=1
    elif Y_train[i]=="COMEDY":
        Y_trainno[i]=2
    elif Y_train[i]=="ENTERTAINMENT":
        Y_trainno[i]=3
    elif Y_train[i]=="FOOD & DRINK":
        Y_trainno[i]=4
    elif Y_train[i]=="HEALTHY LIVING":
        Y_trainno[i]=5
    elif Y_train[i]=="PARENTING":
        Y_trainno[i]=6
    elif Y_train[i]=="POLITICS":
        Y_trainno[i]=7
    elif Y_train[i]=="QUEER VOICES":
        Y_trainno[i]=8
    elif Y_train[i]=="SPORTS":
        Y_trainno[i]=9
    elif Y_train[i]=="STYLE & BEAUTY":
        Y_trainno[i]=10
    elif Y_train[i]=="TRAVEL":
        Y_trainno[i]=11
    elif Y_train[i]=="WELLNESS":
        Y_trainno[i]=12

```

In [ ]:

```
x1_train=[]
x2_train=[]
x3_train=[]
x4_train=[]
x5_train=[]
x6_train=[]
x7_train=[]
x8_train=[]
x9_train=[]
x10_train=[]
x11_train=[]
x12_train=[]
x1_test=[]
x2_test=[]
x3_test=[]
x4_test=[]
x5_test=[]
x6_test=[]
x7_test=[]
x8_test=[]
x9_test=[]
x10_test=[]
x11_test=[]
x12_test=[]
y1_train=[]
y2_train=[]
y3_train=[]
y4_train=[]
y5_train=[]
y6_train=[]
y7_train=[]
y8_train=[]
y9_train=[]
y10_train=[]
y11_train=[]
y12_train=[]
y1_test=[]
y2_test=[]
y3_test=[]
y4_test=[]
y5_test=[]
y6_test=[]
y7_test=[]
y8_test=[]
y9_test=[]
y10_test=[]
y11_test=[]
y12_test=[]

```

In [ ]:

```
for i in range(Y_train.size):
    if Y_train[i]=="BUSINESS":
        x1_train.append(X_train_no[i,:])
        y1_train.append(Y_trainno[i])
    elif Y_train[i]=="COMEDY":
        x2_train.append(X_train_no[i,:])
        y2_train.append(Y_trainno[i])
    elif Y_train[i]=="ENTERTAINMENT":
        x3_train.append(X_train_no[i,:])
        y3_train.append(Y_trainno[i])
    elif Y_train[i]=="FOOD & DRINK":
        x4_train.append(X_train_no[i,:])
        y4_train.append(Y_trainno[i])
    elif Y_train[i]=="HEALTHY LIVING":
        x5_train.append(X_train_no[i,:])
        y5_train.append(Y_trainno[i])
    elif Y_train[i]=="PARENTING":
        x6_train.append(X_train_no[i,:])
        y6_train.append(Y_trainno[i])
    elif Y_train[i]=="POLITICS":
        x7_train.append(X_train_no[i,:])
        y7_train.append(Y_trainno[i])
    elif Y_train[i]=="QUEER VOICES":
        x8_train.append(X_train_no[i,:])
        y8_train.append(Y_trainno[i])
    elif Y_train[i]=="SPORTS":
        x9_train.append(X_train_no[i,:])
        y9_train.append(Y_trainno[i])
    elif Y_train[i]=="STYLE & BEAUTY":
        x10_train.append(X_train_no[i,:])
        y10_train.append(Y_trainno[i])
    elif Y_train[i]=="TRAVEL":
        x11_train.append(X_train_no[i,:])
        y11_train.append(Y_trainno[i])
    elif Y_train[i]=="WELLNESS":
        x12_train.append(X_train_no[i,:])
        y12_train.append(Y_trainno[i])
```

In [ ]:

```
Y_testno=np.zeros((Y_test.size))
for i in range(Y_test.size):
    if Y_test[i]=="BUSINESS":
        Y_testno[i]=1
    elif Y_test[i]=="COMEDY":
        Y_testno[i]=2
    elif Y_test[i]=="ENTERTAINMENT":
        Y_testno[i]=3
    elif Y_test[i]=="FOOD & DRINK":
        Y_testno[i]=4
    elif Y_test[i]=="HEALTHY LIVING":
        Y_testno[i]=5
    elif Y_test[i]=="PARENTING":
        Y_testno[i]=6
    elif Y_test[i]=="POLITICS":
        Y_testno[i]=7
    elif Y_test[i]=="QUEER VOICES":
        Y_testno[i]=8
    elif Y_test[i]=="SPORTS":
        Y_testno[i]=9
    elif Y_test[i]=="STYLE & BEAUTY":
        Y_testno[i]=10
    elif Y_test[i]=="TRAVEL":
        Y_testno[i]=11
    elif Y_test[i]=="WELLNESS":
        Y_testno[i]=12

```

In [ ]:

```
total=len(x1_train)+len(x2_train)+len(x3_train)+len(x4_train)+len(x5_train)+len(x6_tr
print(total)
total=len(y1_train)+len(y2_train)+len(y3_train)+len(y4_train)+len(y5_train)+len(y6_tr
print(total)
```

In [ ]:

```
mu1_es=(1/len(x1_train))*(np.sum(x1_train,axis=0));
mu2_es=(1/len(x2_train))*(np.sum(x2_train,axis=0));
mu3_es=(1/len(x3_train))*(np.sum(x3_train,axis=0));
mu4_es=(1/len(x4_train))*(np.sum(x4_train,axis=0));
mu5_es=(1/len(x5_train))*(np.sum(x5_train,axis=0));
mu6_es=(1/len(x6_train))*(np.sum(x6_train,axis=0));
mu7_es=(1/len(x7_train))*(np.sum(x7_train,axis=0));
mu8_es=(1/len(x8_train))*(np.sum(x8_train,axis=0));
mu9_es=(1/len(x9_train))*(np.sum(x9_train,axis=0));
mu10_es=(1/len(x10_train))*(np.sum(x10_train,axis=0));
mu11_es=(1/len(x11_train))*(np.sum(x11_train,axis=0));
mu12_es=(1/len(x12_train))*(np.sum(x12_train,axis=0));

#Estimated variance
cov1_es=(1/len(x1_train))*np.dot((np.transpose(x1_train-mu1_es)),(x1_train-mu1_es));
cov2_es=(1/len(x2_train))*np.dot((np.transpose(x2_train-mu2_es)),(x2_train-mu2_es));
cov3_es=(1/len(x3_train))*np.dot((np.transpose(x3_train-mu3_es)),(x3_train-mu3_es));
cov4_es=(1/len(x4_train))*np.dot((np.transpose(x4_train-mu4_es)),(x4_train-mu4_es));
cov5_es=(1/len(x5_train))*np.dot((np.transpose(x5_train-mu5_es)),(x5_train-mu5_es));
cov6_es=(1/len(x6_train))*np.dot((np.transpose(x6_train-mu6_es)),(x6_train-mu6_es));
cov7_es=(1/len(x7_train))*np.dot((np.transpose(x7_train-mu7_es)),(x7_train-mu7_es));
cov8_es=(1/len(x8_train))*np.dot((np.transpose(x8_train-mu8_es)),(x8_train-mu8_es));
cov9_es=(1/len(x9_train))*np.dot((np.transpose(x9_train-mu9_es)),(x9_train-mu9_es));
cov10_es=(1/len(x10_train))*np.dot((np.transpose(x10_train-mu10_es)),(x10_train-mu10_
cov11_es=(1/len(x11_train))*np.dot((np.transpose(x11_train-mu11_es)),(x11_train-mu11_
cov12_es=(1/len(x12_train))*np.dot((np.transpose(x12_train-mu12_es)),(x12_train-mu12_

```

In [ ]:

```
X_test_no=np.zeros((42923,500))

for i in range(42923):
    X_test_no[i,:]= model.infer_vector(np.asarray(X_test)[i])
```

In [ ]:

```
prob_prior=np.zeros((12))
prob_prior[0]=len(x1_train)/total
prob_prior[1]=len(x2_train)/total
prob_prior[2]=len(x3_train)/total
prob_prior[3]=len(x4_train)/total
prob_prior[4]=len(x5_train)/total
prob_prior[5]=len(x6_train)/total
prob_prior[6]=len(x7_train)/total
prob_prior[7]=len(x8_train)/total
prob_prior[8]=len(x9_train)/total
prob_prior[9]=len(x10_train)/total
prob_prior[10]=len(x11_train)/total
prob_prior[11]=len(x12_train)/total
print(prob_prior)
```

In [ ]:

```
from sklearn.metrics import confusion_matrix,accuracy_score
from scipy.stats import multivariate_normal
y_pred=np.zeros((42923))
for i in range(42923):
    p1=multivariate_normal.pdf(X_test_no[i,:],mu1_es,cov1_es)*prob_prior[0]
    p2=multivariate_normal.pdf(X_test_no[i,:],mu2_es,cov2_es)*prob_prior[1]
    p3=multivariate_normal.pdf(X_test_no[i,:],mu3_es,cov3_es)*prob_prior[2]
    p4=multivariate_normal.pdf(X_test_no[i,:],mu4_es,cov4_es)*prob_prior[3]
    p5=multivariate_normal.pdf(X_test_no[i,:],mu5_es,cov5_es)*prob_prior[4]
    p6=multivariate_normal.pdf(X_test_no[i,:],mu6_es,cov6_es)*prob_prior[5]
    p7=multivariate_normal.pdf(X_test_no[i,:],mu7_es,cov7_es)*prob_prior[6]
    p8=multivariate_normal.pdf(X_test_no[i,:],mu8_es,cov8_es)*prob_prior[7]
    p9=multivariate_normal.pdf(X_test_no[i,:],mu9_es,cov9_es)*prob_prior[8]
    p10=multivariate_normal.pdf(X_test_no[i,:],mu10_es,cov10_es)*prob_prior[9]
    p11=multivariate_normal.pdf(X_test_no[i,:],mu11_es,cov11_es)*prob_prior[10]
    p12=multivariate_normal.pdf(X_test_no[i,:],mu12_es,cov12_es)*prob_prior[11]
    prob=np.column_stack((p1,p2,p3,p4,p5,p6,p7,p8,p9,p10,p11,p12))
    print(prob)
    try:
        y_pred[i]=(np.where(prob[0,:] == np.amax(prob[0,:]))[0])+1
    except:
        y_pred[i]=1
print(confusion_matrix(Y_testno, y_pred))
print(accuracy_score(Y_testno, y_pred))
```

In [ ]:

```
print(accuracy_score(Y_testno, y_pred))
```