Naive bayes

In [ ]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn
import warnings
fil_data=pd.read_excel('filtered_data.xlsx')
```

In [ ]:

```
import sklearn
from sklearn import model_selection
X_train, X_test, Y_train, Y_test = sklearn.model_selection.train_test_split(fil_data[
                                                                            fil_data[
```

In [ ]:

```
#preprocessing the data
from nltk.stem import PorterStemmer, WordNetLemmatizer
import sklearn.model_selection
import re
from nltk.corpus import stopwords
import numpy as np
import sklearn
import nltk
X_train, X_test, Y_train, Y_test = sklearn.model_selection.train_test_split(fil_data[
                                                                            fil_data[

X_train = np.array(X_train);
X_test = np.array(X_test);
Y_train = np.array(Y_train);
Y_test = np.array(Y_test);

procText_train = []
procText_test = []
number_train = len(X_train)
number_test = len(X_test)

lemmetizer = WordNetLemmatizer()
stemmer = PorterStemmer()
def get_words(headlines_list):
    headlines = headlines_list[0]
    headlines_only_letters = re.sub('[^a-zA-Z]', ' ', headlines)
    words = nltk.word_tokenize(headlines_only_letters.lower())
    stops = set(stopwords.words('english'))
    meaningful_words = [lemmetizer.lemmatize(w) for w in words if w not in stops]
    return ' '.join(meaningful_words )

for i in range(number_train):
    proctext = get_words(X_train[i]) #Processing the data and getting words with no s
    procText_train.append( proctext )
print("train words done")
for i in range(number_test):
    proctext = get_words(X_test[i]) #Processing the data and getting words with no sp
    procText_test.append( proctext )
print("test words done")
vectorize = sklearn.feature_extraction.text.TfidfVectorizer(analyzer = "word", max_fe
tfidwords_train = vectorize.fit_transform(procText_train)
X_train = tfidwords_train.toarray()

tfidwords_test = vectorize.transform(procText_test)
X_test = tfidwords_test.toarray()
print("vectorizer done")
```

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[i,:])
        y1_train.append(Y_train[i])
    elif Y_train[i]=="COMEDY":
        x2_train.append(X_train[i,:])
        y2_train.append(Y_train[i])
    elif Y_train[i]=="ENTERTAINMENT":
        x3_train.append(X_train[i,:])
        y3_train.append(Y_train[i])
    elif Y_train[i]=="FOOD & DRINK":
        x4_train.append(X_train[i,:])
        y4_train.append(Y_train[i])
    elif Y_train[i]=="HEALTHY LIVING":
        x5_train.append(X_train[i,:])
        y5_train.append(Y_train[i])
    elif Y_train[i]=="PARENTING":
        print(2)
        x6_train.append(X_train[i,:])
        y6_train.append(Y_train[i])
    elif Y_train[i]=="POLITICS":
        print(1)
        x7_train.append(X_train[i,:])
        y7_train.append(Y_train[i])
    elif Y_train[i]=="QUEER VOICES":
        x8_train.append(X_train[i,:])
        y8_train.append(Y_train[i])
    elif Y_train[i]=="SPORTS":
        x9_train.append(X_train[i,:])
        y9_train.append(Y_train[i])
    elif Y_train[i]=="STYLE & BEAUTY":
        x10_train.append(X_train[i,:])
        y10_train.append(Y_train[i])
    elif Y_train[i]=="TRAVEL":
        x11_train.append(X_train[i,:])
        y11_train.append(Y_train[i])
    elif Y_train[i]=="WELLNESS":
        x12_train.append(X_train[i,:])
        y12_train.append(Y_train[i])
```

In [ ]:

```
for i in range(Y_test.size):
    if Y_test[i]=="BUSINESS":
        x1_test.append(X_test[i,:])
        y1_test.append(Y_test[i])
    elif Y_test[i]=="COMEDY":
        x2_test.append(X_test[i,:])
        y2_test.append(Y_test[i])
    elif Y_test[i]=="ENTERTAINMENT":
        x3_test.append(X_test[i,:])
        y3_test.append(Y_test[i])
    elif Y_test[i]=="FOOD & DRINK":
        x4_test.append(X_test[i,:])
        y4_test.append(Y_test[i])
    elif Y_test[i]=="HEALTHY LIVING":
        x5_test.append(X_test[i,:])
        y5_test.append(Y_test[i])
    elif Y_test[i]=="PARENTING":
        x6_test.append(X_test[i,:])
        y6_test.append(Y_test[i])
    elif Y_test[i]=="POLITICS":
        x7_test.append(X_test[i,:])
        y7_test.append(Y_test[i])
    elif Y_test[i]=="QUEER VOICES":
        x8_test.append(X_test[i,:])
        y8_test.append(Y_test[i])
    elif Y_test[i]=="SPORTS":
        x9_test.append(X_test[i,:])
        y9_test.append(Y_test[i])
    elif Y_test[i]=="STYLE & BEAUTY":
        x10_test.append(X_test[i,:])
        y10_test.append(Y_test[i])
    elif Y_test[i]=="TRAVEL":
        x11_test.append(X_test[i,:])
        y11_test.append(Y_test[i])
    elif Y_test[i]=="WELLNESS":
        x12_test.append(X_test[i,:])
        y12_test.append(Y_test[i])
```

In [ ]:

```
total=len(x1_train)+len(x2_train)+len(x3_train)+len(x4_train)+len(x5_train)+len(x6_tr
print(total)
```

In [ ]:

```
print(len(x1_train))
print(len(x2_train))
print(len(x3_train))
print(len(x4_train))
print(len(x5_train))
print(len(x6_train))
print(len(x7_train))
print(len(x8_train))
print(len(x9_train))
print(len(x10_train))
print(len(x11_train))
print(len(x12_train))
```

In [ ]:

```
x1_train=np.array(x1_train)
x2_train=np.array(x2_train)
x3_train=np.array(x3_train)
x4_train=np.array(x4_train)
x5_train=np.array(x5_train)
x6_train=np.array(x6_train)
x7_train=np.array(x7_train)
x8_train=np.array(x8_train)
x9_train=np.array(x9_train)
x10_train=np.array(x10_train)
x11_train=np.array(x11_train)
x12_train=np.array(x12_train)
x1_test=np.array(x1_test)
x2_test=np.array(x2_test)
x3_test=np.array(x3_test)
x4_test=np.array(x4_test)
x5_test=np.array(x5_test)
x6_test=np.array(x6_test)
x7_test=np.array(x7_test)
x8_test=np.array(x8_test)
x9_test=np.array(x9_test)
x10_test=np.array(x10_test)
x11_test=np.array(x11_test)
x12_test=np.array(x12_test)
```

In [ ]:

```
count_words=np.zeros((10000))
for i in range(x1_train.shape[1]):
    for j in range(X_train.shape[0]):
        if X_train[j,i]!=0:
            count_words[i]+=1;
print(count_words)
```

In [ ]:

```
prob=np.zeros((12,10000))
for i in range(x1_train.shape[1]):
    count=0
    for j in range(x1_train.shape[0]):
        if x1_train[j,i]!=0:
            count=+1
    prob[0,i]=(count+1)/(100+count_words[i])
    count=0
    for j in range(x2_train.shape[0]):
        if x2_train[j,i]!=0:
            count=+1
    prob[1,i]=(count+1)/(100+count_words[i])
    count=0
    for j in range(x3_train.shape[0]):
        if x3_train[j,i]!=0:
            count=+1
    prob[2,i]=(count+1)/(100+count_words[i])
    count=0
    for j in range(x4_train.shape[0]):
        if x4_train[j,i]!=0:
            count=+1
    prob[3,i]=(count+1)/(100+count_words[i])
    count=0
    for j in range(x5_train.shape[0]):
        if x5_train[j,i]!=0:
            count=+1
    prob[4,i]=(count+1)/(100+count_words[i])
    count=0
    for j in range(x6_train.shape[0]):
        if x6_train[j,i]!=0:
            count=+1
    prob[5,i]=(count+1)/(100+count_words[i])
    count=0
    for j in range(x7_train.shape[0]):
        if x7_train[j,i]!=0:
            count=+1
    prob[6,i]=(count+1)/(100+count_words[i])
    count=0
    for j in range(x8_train.shape[0]):
        if x8_train[j,i]!=0:
            count=+1
    prob[7,i]=(count+1)/(100+count_words[i])
    count=0
    for j in range(x9_train.shape[0]):
        if x9_train[j,i]!=0:
            count=+1
    prob[8,i]=(count+1)/(100+count_words[i])
    count=0
    for j in range(x10_train.shape[0]):
        if x10_train[j,i]!=0:
            count=+1
    prob[9,i]=(count+1)/(100+count_words[i])
    count=0
    for j in range(x11_train.shape[0]):
        if x11_train[j,i]!=0:
            count=+1
    prob[10,i]=(count+1)/(100+count_words[i])
    count=0
    for j in range(x12_train.shape[0]):
        if x12_train[j,i]!=0:
            count=+1
    prob[11,i]=(count+1)/(100+count_words[i])

print(prob[:,0])
```

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 [ ]:

```
post_prob=np.ones((X_test.shape[0],12))
for i in range(X_test.shape[0]):
    for j in range(10000):
        for k in range(12):
            if X_test[i,j]!=0:
                post_prob[i,k]=post_prob[i,k]*prob[k,j]

```

In [ ]:

```
post_prob_prior=post_prob*prob_prior
```

In [ ]:

```
y_pred=np.zeros((X_test.shape[0]))
for i in range(X_test.shape[0]):
    try:
        y_pred[i]=(np.where(post_prob[i,:] == np.amax(post_prob[i,:]))[0])+1
    except:
        print(i)
        y_pred[i]=np.random.choice((np.where(post_prob[i,:] == np.amax(post_prob[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 [ ]:

```python
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 [ ]:

```python
from sklearn.metrics import accuracy_score
print(accuracy_score(Y_testno, np.asarray(y_pred)))
from sklearn.metrics import confusion_matrix
print(confusion_matrix(Y_testno, np.asarray(y_pred)))
```

KNN

In [ ]:

```
from scipy.spatial import distance
from sklearn.metrics import confusion_matrix
from sklearn.metrics import confusion_matrix

K={20}
#K={3}
count=0
err=np.zeros((6))
for k in K:
    print(k)
    y_pred=np.zeros((10000))
    for i in range(10000):
        neighbors=[]
        dist=np.zeros((20000,2))
        for j in range(20000):
            dist[j,0]=distance.euclidean(X_train[j,:], X_test[i,:])
            dist[j,1]=Y_trainno[j]
        dist=dist[dist[:,0].argsort()]
        for value in range(k):
            neighbors.append(dist[value,1])
        y_pred[i]=np.bincount(neighbors).argmax()
        if Y_testno[i]!=y_pred[i]:
            err[count]=err[count]+1
    print(confusion_matrix(Y_testno, y_pred))
    print(accuracy_score(Y_testno[0:10000], y_pred))
    count=count+1

```

AdaBoost, Linear SVC, Logistic regression, bagging and decision tree classifier

In [ ]:

```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.model_selection import cross_val_score
clf = AdaBoostClassifier(n_estimators=20)
#scores = cross_val_score(clf, X_train, Y_train, cv=5)

y_pred = clf.fit(X_train, Y_train).predict(X_test)

from sklearn.svm import LinearSVC
from sklearn.metrics import confusion_matrix

model = LinearSVC()
model.fit(X_train,Y_train)
Y_predict = model.predict(X_test)
accuracy = accuracy_score(Y_test,Y_predict)*100
print(format(accuracy, '.2f'))
print(confusion_matrix(Y_test,Y_predict))

from sklearn.linear_model import LogisticRegression
logistic_Regression = LogisticRegression()
logistic_Regression.fit(X_train,Y_train)
Y_predict = logistic_Regression.predict(X_test)
accuracy = accuracy_score(Y_test,Y_predict)*100
print(format(accuracy, '.2f'))
print(confusion_matrix(Y_test,Y_predict))

from sklearn.ensemble import BaggingClassifier
model = BaggingClassifier(random_state=0, n_estimators=10)
model.fit(X_train, Y_train)
prediction = model.predict(X_test)
print('Accuracy of bagged KNN is :',accuracy_score(prediction, Y_test))
print(confusion_matrix(Y_test,Y_predict))

from sklearn.tree import DecisionTreeClassifier

model = DecisionTreeClassifier()
model.fit(X_train, Y_train)
prediction_decision_tree = model.predict(X_test)
print('The accuracy of Decision Tree is', accuracy_score(prediction_decision_tree, Y_
print(confusion_matrix(Y_test,prediction_decision_tree))
```