#!/usr/bin/env python  
# coding: utf-8  
  
# In[1]:  
  
  
import pandas as pd  
import re  
from sklearn.utils import shuffle  
from nltk.tokenize import word\_tokenize  
import matplotlib.pyplot as plt  
from nltk.corpus import stopwords  
from nltk.stem import WordNetLemmatizer  
from sklearn.feature\_extraction.text import TfidfVectorizer  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.linear\_model import LogisticRegression  
from sklearn.ensemble import RandomForestClassifier,AdaBoostClassifier,GradientBoostingClassifier  
from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import classification,classification\_report,accuracy\_score  
from sklearn.preprocessing import StandardScaler  
from sklearn.pipeline import Pipeline, make\_pipeline  
lemaoi = WordNetLemmatizer()  
  
  
# In[190]:  
  
  
raw\_data = pd.read\_csv('train\_data.csv')  
raw\_data  
  
  
# In[191]:  
  
  
print(raw\_data['sentiment'].value\_counts())  
print('It is very clear that the data is not balanced to make it balanced we will 100 sample')  
  
  
# In[211]:  
  
  
raw\_data = shuffle(raw\_data)  
  
  
# In[137]:  
  
  
def under\_sample(raw):  
 posi = raw\_data.loc[raw\_data['sentiment']=='Positive'][:100]  
 Neu = raw\_data.loc[raw\_data['sentiment']=='Neutral'][:100]  
 Nega = raw\_data.loc[raw\_data['sentiment']=='Negative']  
 traning\_data = pd.concat([posi, Neu, Nega])  
 traning\_data.reset\_index(inplace=True)  
 traning\_data['sentiment'].replace({'Positive':1,'Negative':-1,'Neutral':0}, inplace=True)  
 return traning\_data  
  
  
# In[138]:  
  
  
def over\_sample(raw):  
 posi = raw\_data.loc[raw\_data['sentiment']=='Positive'][:1000]  
 Neu = raw\_data.loc[raw\_data['sentiment']=='Neutral']  
 Nega = raw\_data.loc[raw\_data['sentiment']=='Negative']  
 Neu\_new = pd.concat([Neu,Neu,Neu,Neu,Neu,Neu], axis=0)  
 Nega\_new = pd.concat([Nega,Nega,Nega,Nega,Nega,Nega,Nega,Nega,Nega,Nega])  
 traning\_data = pd.concat([posi, Neu\_new, Nega\_new])  
 traning\_data.reset\_index(inplace=True)  
 traning\_data['sentiment'].replace({'Positive':1,'Negative':-1,'Neutral':0}, inplace=True)  
 return traning\_data  
   
  
  
# In[212]:  
  
  
train\_frame = over\_sample(raw\_data)  
train\_frame.columns  
  
  
# In[213]:  
  
  
print('Clubbing Text and Title of Reviews of Amazon Dataset')  
train\_frame['data'] = train\_frame['reviews.text'] + ' '+ train\_frame['reviews.title']  
data\_to\_train = train\_frame.loc[:,['data','sentiment']]  
data\_to\_train  
  
  
# In[214]:  
  
  
data\_to\_train.dropna(inplace=True, axis=0)  
  
  
# In[215]:  
  
  
data\_to\_train.head(5)  
  
  
# In[216]:  
  
  
def pre\_processing(x):  
 x = x.lower()  
 x = re.sub('[^a-z0-9 ]','',x)  
 result = []  
 for i in x.split():  
 if i in stopwords.words('english'):  
 pass  
 else:  
 if len(i) <3:  
 pass  
 else:  
 result.append(lemaoi.lemmatize(i))  
 res = ' '.join(result)  
 res = re.sub('\d{2,}\w+','',res)  
 return res   
  
  
# In[217]:  
  
  
data\_to\_train['cleaned\_data'] = data\_to\_train['data'].apply(lambda x: pre\_processing(x))  
data\_to\_train  
  
  
# In[218]:  
  
  
vectorizer = TfidfVectorizer()  
vectors = vectorizer.fit\_transform(data\_to\_train['cleaned\_data'])  
feature\_names = vectorizer.get\_feature\_names()  
X = vectors.todense()  
Y = data\_to\_train['sentiment']  
  
  
# In[200]:  
  
  
pipeline\_linear = make\_pipeline(StandardScaler(),LogisticRegression(solver = 'lbfgs'))  
pipeline\_knn = make\_pipeline(StandardScaler(),KNeighborsClassifier(n\_neighbors=15))  
pipeline\_forest = make\_pipeline(StandardScaler(),RandomForestClassifier(n\_estimators=300))  
pipeline\_ada = make\_pipeline(StandardScaler(),AdaBoostClassifier(n\_estimators=150))  
pipeline\_gradient = make\_pipeline(StandardScaler(),GradientBoostingClassifier(n\_estimators=60,learning\_rate=1))  
  
  
# In[219]:  
  
  
x\_train, x\_test, y\_train, y\_test= train\_test\_split(X,Y)  
print(x\_train.shape)  
print(x\_test.shape)  
print(y\_train.shape)  
print(y\_test.shape)  
  
  
# In[220]:  
  
  
file\_obj = open('test\_results.txt','w')  
file\_obj.writelines('Over Sample')  
file\_obj.writelines('\n')  
  
  
# In[221]:  
  
  
for i in [pipeline\_forest,pipeline\_knn,pipeline\_linear,pipeline\_ada,pipeline\_gradient]:  
 file\_obj.writelines(str(i))  
 file\_obj.writelines('\n')  
 obj = i.fit(x\_train,y\_train)  
 y\_pred = obj.predict(x\_test)  
 file\_obj.writelines(classification\_report(y\_test,y\_pred))  
 print(classification\_report(y\_test,y\_pred))  
  
  
# In[222]:  
  
  
file\_obj.close()  
  
  
# In[180]:  
  
  
x\_test = pd.read\_csv('test\_data.csv')  
x\_test['data'] = x\_test['reviews.text'] + ' '+ x\_test['reviews.title']  
x\_test.dropna(axis=0, inplace=True)  
  
  
# In[181]:  
  
  
x\_test['cleaned'] = x\_test['data'].apply(lambda x :pre\_processing(x))  
  
  
# In[182]:  
  
  
vector = vectorizer.transform(x\_test['cleaned'])  
x\_test\_new = vector.todense()  
  
  
# In[183]:  
  
  
y\_pred\_test = pipeline\_forest.predict(x\_test\_new)  
  
  
# In[184]:  
  
  
x\_test['predicted\_under'] = y\_pred\_test  
  
  
# In[185]:  
  
  
x\_test.to\_excel('submiss\_under.xlsx')  
  
  
# In[ ]: