Files ×

IMPORTING THE REQUIRED LIBRARIES

```
import copy
import numpy as np
import matplotlib.pyplot as plt
import re
import nltk
nltk.download('stopwords')
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_selection import VarianceThreshold
from imblearn.over_sampling import SMOTE
from sklearn.dummy import DummyClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import RandomForestClassifier
#from sklearn.metrics import accuracy_score
#from sklearn.model_selection import cross_val_score, KFold
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
import seaborn as sns
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

LOADING THE DATA FILE dsjVoxArticles.tsv file

```
titles = []
categories = []
with open('dsjVoxArticles.tsv','r') as tsv:
    count = 0;
    for line in tsv:
        a = line.strip().split('\t')[:3]
        if a[2] in ['Business & Finance', 'Health Care', 'Science & Health',
            title = a[0].lower()
            title = re.sub('\s\W',' ',title)
            title = re.sub('\W\s',' ',title)
            titles.append(title)
            categories.append(a[2])
```

CHECK FOR titles and categories IN THE DATASET

```
print("Titles-\n", "\n".join(titles[:5]))
print("\nCategories-\n", "\n".join(categories[:5]))
```

```
Titles-
 bitcoin is down 60 percent this year here's why i'm still optimistic.
9 charts that explain the history of global wealth
remember when legal marijuana was going to send crime skyrocketing?
obamacare succeeded for one simple reason it's horrible to be uninsurec
the best obamacare data comes from a home office in michigan

Categories-
 Business & Finance
Business & Finance
Criminal Justice
```

```
Health Care
Health Care
```

SPLITTING DATA

Split data into 3 parts - training, development and test. We will use training data to train out model and use development data to check and tune hyper parameters. And finally use test data to see how our model performs

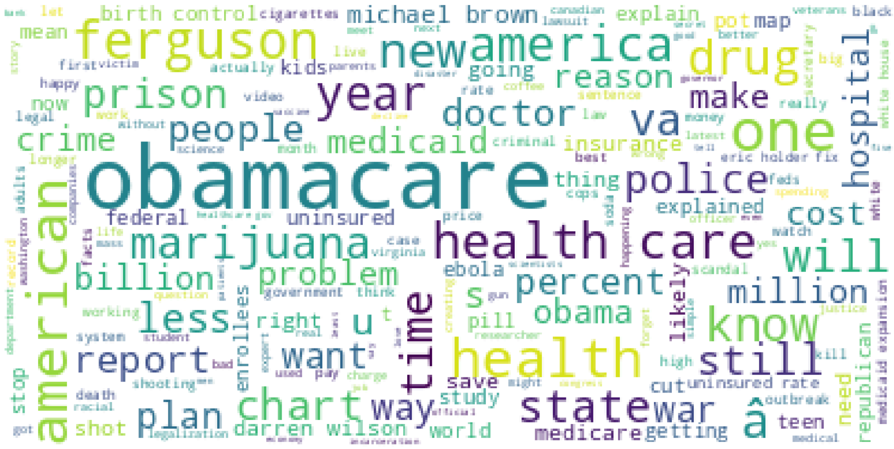
```
title_tr, title_te, category_tr, category_te = train_test_split(titles,cate
title_tr, title_de, category_tr, category_de = train_test_split(title_tr,ca
print("Training: ",len(title_tr))
print("Developement: ",len(title_de),)
print("Testing: ",len(title_te))
```

```
Training:  324
Developement:  108
Testing:  145
```

visualize the data using word cloud

```
from wordcloud import WordCloud
text = " ".join(title_tr)
wordcloud = WordCloud().generate(text)
plt.figure()
plt.subplots(figsize=(20,12))
wordcloud = WordCloud(
    background_color="white",
    max_words=len(text),
    max_font_size=40,
    relative_scaling=.5).generate(text)
plt.imshow(wordcloud)
plt.axis("off")
plt.show()
```

```
<Figure size 640x480 with 0 Axes>
```



DATA PREPROCESSING

```
tokenizer = nltk.tokenize.RegexpTokenizer(r"\w+")
stop_words = nltk.corpus.stopwords.words("english")
vectorizer = CountVectorizer(tokenizer=tokenizer.tokenize, stop_words=stop_

vectorizer.fit(iter(title_tr))
Xtr = vectorizer.transform(iter(title_tr))
Xde = vectorizer.transform(iter(title_de))
Xte = vectorizer.transform(iter(title_te))

encoder = LabelEncoder()
encoder.fit(category_tr)
Ytr = encoder.transform(category_tr)
Yde = encoder.transform(category_de)
Yte = encoder.transform(category_te)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/text
  warnings.warn(
```

```
reverse_vocabulary = {}
vocabulary = vectorizer.vocabulary_
for word in vocabulary:
    index = vocabulary[word]
    reverse_vocabulary[index] = word

vector = vectorizer.transform(iter(['Nasa scientists are good']))
indexes = vector.indices
for i in indexes:
    print (reverse_vocabulary[i])
```

```
good
scientists
```

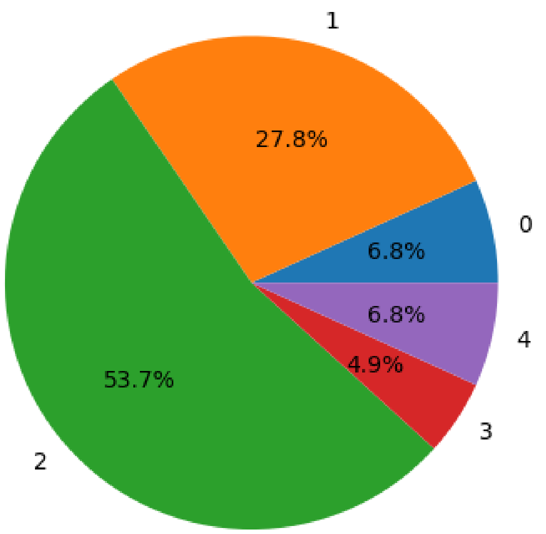
Feature Reduction We can check the variance of the feature and drop them based on a threshold

```
print("Number of features before reduction : ", Xtr.shape[1])
selection = VarianceThreshold(threshold=0.001)
Xtr_whole = copy.deepcopy(Xtr)
Ytr_whole = copy.deepcopy(Ytr)
selection.fit(Xtr)
Xtr = selection.transform(Xtr)
Xde = selection.transform(Xde)
Xte = selection.transform(Xte)
print("Number of features after reduction : ", Xtr.shape[1])
```

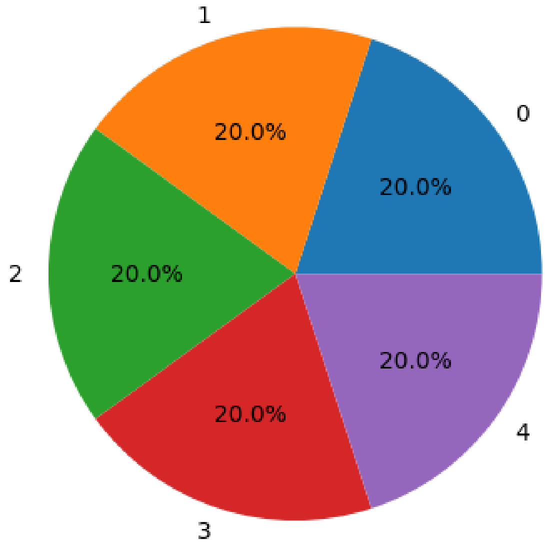
```
Number of features before reduction :  1175
Number of features after reduction :  1175
```

SAMPLING THE DATA

```
labels = list(set(Ytr))
counts = []
for label in labels:
    counts.append(np.count_nonzero(Ytr == label))
plt.pie(counts, labels=labels, autopct='%1.1f%%')
plt.show()
```




```
sm = SMOTE(random_state=42)
Xtr, Ytr = sm.fit_resample(Xtr, Ytr)
labels = list(set(Ytr))
counts = []
for label in labels:
    counts.append(np.count_nonzero(Ytr == label))
plt.pie(counts, labels=labels, autopct='%1.1f%%')
plt.show()
```



TRAINING MODELS

Baseline Model “stratified”: generates predictions by respecting the training set’s class distribution.

```
dc = DummyClassifier(strategy="stratified")
dc.fit(Xtr, Ytr)
pred = dc.predict(Xde)
print(classification_report(Yde, pred, target_names=encoder.classes_))
```

```
                    precision    recall  f1-score   support

  Business & Finance       0.07      0.25      0.11         4
    Criminal Justice       0.32      0.20      0.25        40
         Health Care       0.40      0.21      0.28        47
   Politics & Policy       0.07      0.20      0.11        10
    Science & Health       0.00      0.00      0.00         7

            accuracy                           0.19       108
           macro avg       0.17      0.17      0.15       108
        weighted avg       0.30      0.19      0.23       108
```

Multinomial Naive Bayesian

```
nb = MultinomialNB()
nb.fit(Xtr, Ytr)
pred = nb.predict(Xde)
print(classification_report(Yde, pred, target_names=encoder.classes_))
```

```
                    precision    recall  f1-score   support

  Business & Finance       0.17      0.25      0.20         4
    Criminal Justice       0.83      0.75      0.79        40
         Health Care       0.78      0.85      0.82        47
   Politics & Policy       0.20      0.10      0.13        10
```

```
   Science & Health       0.40      0.57      0.47         7

           accuracy                           0.70       108
          macro avg       0.48      0.50      0.48       108
       weighted avg       0.70      0.70      0.70       108
```

SUPPORT VECTOR MACHINE CLASSSIFIER

```
from sklearn.svm import SVC
svc = SVC()
svc.fit(Xtr, Ytr)
pred = svc.predict(Xde)
print(classification_report(Yde, pred, target_names=encoder.classes_))
```

```
                         precision    recall  f1-score   support

     Business & Finance       0.33      0.25      0.29         4
       Criminal Justice       0.95      0.47      0.63        40
            Health Care       0.58      0.96      0.73        47
      Politics & Policy       0.33      0.20      0.25        10
       Science & Health       1.00      0.29      0.44         7

               accuracy                           0.64       108
              macro avg       0.64      0.43      0.47       108
           weighted avg       0.71      0.64      0.61       108
```

Multilayered Perceptron

```
mlp = MLPClassifier(solver='adam', alpha=1e-5, hidden_layer_sizes=(100, 20)
mlp.fit(Xtr, Ytr)
pred = mlp.predict(Xde)
print(classification_report(Yde, pred, target_names=encoder.classes_))
```

```
                         precision    recall  f1-score   support

     Business & Finance       0.25      0.25      0.25         4
       Criminal Justice       0.76      0.78      0.77        40
            Health Care       0.88      0.79      0.83        47
      Politics & Policy       0.33      0.20      0.25        10
       Science & Health       0.27      0.57      0.36         7

               accuracy                           0.69       108
              macro avg       0.50      0.52      0.49       108
           weighted avg       0.72      0.69      0.70       108
```

Final Model: Multinomial Naive Bayesian

Multinomial Naive Bayesian works the best. Lets run NB on our test data and get the confusion matrix and its heat map.

PREDICT TEST DATA

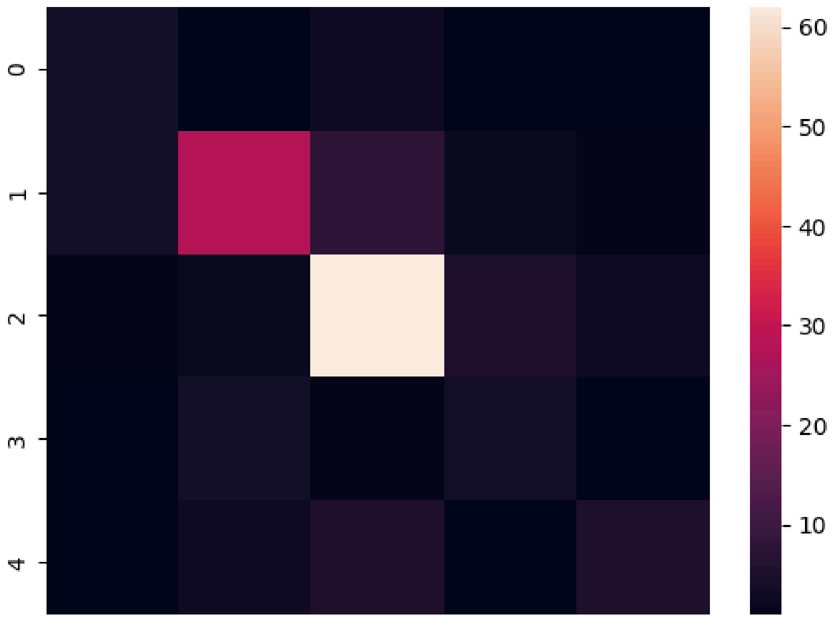
```
pred = nb.predict(Xte)
print(classification_report(Yte, pred, target_names=encoder.classes_))
sns.heatmap(confusion_matrix(Yte, pred))
```

```
                    precision    recall  f1-score   support

Business & Finance       0.44      0.57      0.50         7
  Criminal Justice       0.76      0.65      0.70        43
       Health Care       0.78      0.85      0.82        73
 Politics & Policy       0.36      0.44      0.40         9
  Science & Health       0.56      0.38      0.45        13

          accuracy                           0.71       145
         macro avg       0.58      0.58      0.57       145
      weighted avg       0.71      0.71      0.71       145

<Axes: >
```



Multinomial Naive Bayesian Explained

We will now try to understand why Naive Bayesian is getting good results. We will get all the coefficents of the features and then print the top 20 words based on its weight. As we can see all the words are closely related to the category, hence multinomial naive bayesian predcits correct label with good F1 score.

```
nb1 = MultinomialNB()
nb1.fit(Xtr_whole, Ytr_whole)
coefs = nb1.feature_log_prob_
target_names = encoder.classes_

for i in range(len(target_names)):
    words = []
    for j in coefs[i].argsort()[-20:]:
        words.append(reverse_vocabulary[j])
    print (target_names[i], '-', words, "\n")
```