Imports

In [65]:
```
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
import numpy as np
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import LabelEncoder
```

In [2]:
```
# nltk.download('stopwords')
# nltk.download('punkt')
# nltk.download('wordnet')
```

Loading Data

In [3]:
```
df = pd.read_csv('flipitnews-data.csv')
```

In [4]:
```
df.shape
```

Out[4]:
```
(2225, 2)
```

In [5]:
```
df.head()
```

Out[5]:

	Category	Article
0	Technology	tv future in the hands of viewers with home th...
1	Business	worldcom boss left books alone former worldc...
2	Sports	tigers wary of farrell gamble leicester say ...
3	Sports	yeading face newcastle in fa cup premiership s...
4	Entertainment	ocean s twelve raids box office ocean s twelve...

In [6]:
```
round(df['Category'].value_counts(1, dropna=False) * 100)
```

Out[6]:
```
Sports           23.0
Business         23.0
Politics         19.0
Technology       18.0
Entertainment    17.0
Name: Category, dtype: float64
```

In [7]:
```
df['Article'].str.len().describe()
```

Out[7]:
```
count     2225.00000
mean      2262.93618
std       1364.10253
min        501.00000
25%       1446.00000
50%       1965.00000
75%       2802.00000
max      25483.00000
Name: Article, dtype: float64
```

Processing Text Data

In [8]:
```
def pre_process(x):
    stop_words = set(stopwords.words('english'))
    wnl = WordNetLemmatizer()

    x = x.lower()
    x = word_tokenize(x)
    x = [wnl.lemmatize(word) for word in x if word not in stop_words if word.isalph

    return " ".join(x)
```

In [9]:
```
df['CleanedArticle'] = df['Article'].apply(lambda x : pre_process(x))
```

In [10]:
```
df.head()
```

Out[10]:

	Category	Article	CleanedArticle
0	Technology	tv future in the hands of viewers with home th...	tv future hand viewer home theatre system plas...
1	Business	worldcom boss left books alone former worldc...	worldcom bos left book alone former worldcom b...
2	Sports	tigers wary of farrell gamble leicester say ...	tiger wary farrell gamble leicester say rushed...
3	Sports	yeading face newcastle in fa cup premiership s...	yeading face newcastle fa cup premiership side...
4	Entertainment	ocean s twelve raids box office ocean s twelve...	ocean twelve raid box office ocean twelve crim...

Encoding

In [11]:
```
label_enc = LabelEncoder()
df['CategoryEncoded'] = label_enc.fit_transform(df['Category'])
```

Train Test Split

In [12]:
```
X_train, X_test, y_train, y_test = train_test_split(df['CleanedArticle'], df['Categ
```

In [13]:
```
X_train.shape, X_test.shape
```

Out[13]:
```
((1668,), (557,))
```

Vectorizing

In [37]:
```
cv = CountVectorizer(max_df=0.8, min_df=5, ngram_range=(1,2))
tf_idf = TfidfVectorizer(max_df=0.8, min_df=5, ngram_range=(1,2))
```

In [38]:
```
X_train_cv = pd.DataFrame(cv.fit_transform(X_train).todense())
X_train_cv.columns = cv.get_feature_names()

X_test_cv = pd.DataFrame(cv.transform(X_test).todense())
X_test_cv.columns = cv.get_feature_names()
```

In [39]:
```
X_train_cv.shape, X_test_cv.shape
```

Out[39]:
```
((1668, 10721), (557, 10721))
```

In [40]:
```
X_train_cv.head()
```

Out[40]:

	aaa	aaron	abandoned	abandoning	abbott	abc	ability	able	able access	able get	...	yukos said	yush
0	0	0	0	0	0	0	0	0	0	0	...	0	
1	0	0	0	0	0	0	0	0	0	0	...	0	
2	0	0	0	0	0	0	0	0	0	0	...	0	
3	0	0	0	0	0	0	0	0	0	0	...	0	
4	0	0	0	0	0	0	0	0	0	0	...	0	

5 rows × 10721 columns

In [41]:
```
(X_train_cv == 0).mean().mean() * 100, (X_test_cv == 0).mean().mean() * 100
```

Out[41]:
```
(98.60523296687713, 98.6536934759663)
```

In [42]:
```
X_train_tf_idf = pd.DataFrame(tf_idf.fit_transform(X_train).todense())
X_train_tf_idf.columns = tf_idf.get_feature_names()
```

```
X_test_tf_idf = pd.DataFrame(tf_idf.transform(X_test).todense())
X_test_tf_idf.columns = tf_idf.get_feature_names()
```

In [43]:
```
X_train_tf_idf.shape, X_test_tf_idf.shape
```

Out[43]:
```
((1668, 10721), (557, 10721))
```

In [44]:
```
(X_train_tf_idf == 0).mean().mean() * 100, (X_test_tf_idf == 0).mean().mean() * 100
```

Out[44]:
```
(98.60523296687713, 98.6536934759663)
```

In [36]:
```
X_train_tf_idf.head()
```

Out[36]:

	aaa	abandoned	abc	ability	able	abroad	absence	absolute	absolutely	abuse	...	youn peopl
0	0.0	0.0	0.0	0.0	0.0	0.043807	0.0	0.0	0.0	0.0	...	0.
1	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	...	0.
2	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	...	0.
3	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	...	0.
4	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	...	0.

5 rows × 4965 columns

Modelling

In [61]:
```
def generate_modelling_report(model, X_train, y_train, X_test, y_test):
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
    print('--------------------------------')
    print("Classification Report:\n", classification_report(y_test, y_pred))
    return
```

In [70]:
```
for model in [MultinomialNB(), RandomForestClassifier(), DecisionTreeClassifier(),
    print('****************************************'*2)
    print('Model: ', model)
    print('\n\tUsing CountVectorizer........')
    generate_modelling_report(model, X_train_cv, y_train, X_test_cv, y_test)
    print('\n\tUsing TfidfVectorizer........')
    generate_modelling_report(model, X_train_tf_idf, y_train, X_test_tf_idf, y_test
```

```
********************************************************************************
Model:  MultinomialNB()

        Using CountVectorizer........
Confusion Matrix:
 [[121   2   3   0   2]
 [  1  93   1   0   2]
 [  0   0 104   0   0]
 [  0   0   0 128   0]
 [  1   0   0   0  99]]
---------------------------------
Classification Report:
               precision    recall  f1-score   support

           0       0.98      0.95      0.96       128
           1       0.98      0.96      0.97        97
           2       0.96      1.00      0.98       104
           3       1.00      1.00      1.00       128
           4       0.96      0.99      0.98       100

    accuracy                           0.98       557
   macro avg       0.98      0.98      0.98       557
weighted avg       0.98      0.98      0.98       557


        Using TfidfVectorizer........
Confusion Matrix:
 [[124   0   3   0   1]
 [  1  93   1   0   2]
 [  1   0 102   1   0]
 [  0   0   0 128   0]
 [  2   0   0   0  98]]
---------------------------------
Classification Report:
               precision    recall  f1-score   support

           0       0.97      0.97      0.97       128
           1       1.00      0.96      0.98        97
           2       0.96      0.98      0.97       104
           3       0.99      1.00      1.00       128
           4       0.97      0.98      0.98       100

    accuracy                           0.98       557
   macro avg       0.98      0.98      0.98       557
weighted avg       0.98      0.98      0.98       557

********************************************************************************
Model:  RandomForestClassifier()

        Using CountVectorizer........
Confusion Matrix:
 [[124   0   3   0   1]
 [  2  91   3   0   1]
 [  3   0  97   2   2]
 [  0   0   0 128   0]
 [  1   1   0   0  98]]
```

```
---------------------------------
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.97      0.96       128
           1       0.99      0.94      0.96        97
           2       0.94      0.93      0.94       104
           3       0.98      1.00      0.99       128
           4       0.96      0.98      0.97       100

    accuracy                           0.97       557
   macro avg       0.97      0.96      0.96       557
weighted avg       0.97      0.97      0.97       557


        Using TfidfVectorizer........
Confusion Matrix:
 [[123   0   4   0   1]
 [  3  92   1   0   1]
 [  1   0  98   3   2]
 [  1   0   0 127   0]
 [  3   1   0   1  95]]
---------------------------------
Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.96      0.95       128
           1       0.99      0.95      0.97        97
           2       0.95      0.94      0.95       104
           3       0.97      0.99      0.98       128
           4       0.96      0.95      0.95       100

    accuracy                           0.96       557
   macro avg       0.96      0.96      0.96       557
weighted avg       0.96      0.96      0.96       557

*********************************************************************************
Model:  DecisionTreeClassifier()

        Using CountVectorizer........
Confusion Matrix:
 [[104   2  11   4   7]
 [  4  84   1   6   2]
 [  6   3  85   6   4]
 [  1   2   0 123   2]
 [  5   3   2   3  87]]
---------------------------------
Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.81      0.84       128
           1       0.89      0.87      0.88        97
           2       0.86      0.82      0.84       104
           3       0.87      0.96      0.91       128
           4       0.85      0.87      0.86       100
```

```
    accuracy                           0.87       557
   macro avg       0.87      0.87      0.87       557
weighted avg       0.87      0.87      0.87       557


        Using TfidfVectorizer........
Confusion Matrix:
 [[102   3  12   6   5]
 [  3  84   2   6   2]
 [  4   3  88   5   4]
 [  1   2   1 122   2]
 [  6   4   2   2  86]]
---------------------------------
Classification Report:
               precision    recall  f1-score   support

           0       0.88      0.80      0.84       128
           1       0.88      0.87      0.87        97
           2       0.84      0.85      0.84       104
           3       0.87      0.95      0.91       128
           4       0.87      0.86      0.86       100

    accuracy                           0.87       557
   macro avg       0.87      0.86      0.86       557
weighted avg       0.87      0.87      0.86       557

********************************************************************************
Model:  KNeighborsClassifier()

        Using CountVectorizer........
Confusion Matrix:
 [[ 74   0   1  52   1]
 [  2  46   0  49   0]
 [  2   0  67  35   0]
 [  0   0   0 128   0]
 [  5   4   0  59  32]]
---------------------------------
Classification Report:
               precision    recall  f1-score   support

           0       0.89      0.58      0.70       128
           1       0.92      0.47      0.63        97
           2       0.99      0.64      0.78       104
           3       0.40      1.00      0.57       128
           4       0.97      0.32      0.48       100

    accuracy                           0.62       557
   macro avg       0.83      0.60      0.63       557
weighted avg       0.81      0.62      0.63       557


        Using TfidfVectorizer........
Confusion Matrix:
 [[119   0   5   1   3]
 [  1  91   1   0   4]
 [  2   2  99   1   0]
```

```
 [  2   0   0 126   0]
 [  2   1   3   0  94]]
---------------------------------
Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.93      0.94       128
           1       0.97      0.94      0.95        97
           2       0.92      0.95      0.93       104
           3       0.98      0.98      0.98       128
           4       0.93      0.94      0.94       100

    accuracy                           0.95       557
   macro avg       0.95      0.95      0.95       557
weighted avg       0.95      0.95      0.95       557
```

In [75]:

```
"""*********************************************************************************
Model:  MultinomialNB()

       Using CountVectorizer........
Confusion Matrix:
 [[121   2   3   0   2]
 [  1  93   1   0   2]
 [  0   0 104   0   0]
 [  0   0   0 128   0]
 [  1   0   0   0  99]]
---------------------------------
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.95      0.96       128
           1       0.98      0.96      0.97        97
           2       0.96      1.00      0.98       104
           3       1.00      1.00      1.00       128
           4       0.96      0.99      0.98       100

    accuracy                           0.98       557
   macro avg       0.98      0.98      0.98       557
weighted avg       0.98      0.98      0.98       557


       Using TfidfVectorizer........
Confusion Matrix:
 [[124   0   3   0   1]
 [  1  93   1   0   2]
 [  1   0 102   1   0]
 [  0   0   0 128   0]
 [  2   0   0   0  98]]
---------------------------------
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.97      0.97       128
           1       1.00      0.96      0.98        97
           2       0.96      0.98      0.97       104
```

```
           3       0.99      1.00      1.00       128
           4       0.97      0.98      0.98       100

    accuracy                           0.98       557
   macro avg       0.98      0.98      0.98       557
weighted avg       0.98      0.98      0.98       557

*********************************************************************************
Model:  RandomForestClassifier()

        Using CountVectorizer........
Confusion Matrix:
 [[124   0   3   0   1]
 [  2  91   3   0   1]
 [  3   0  97   2   2]
 [  0   0   0 128   0]
 [  1   1   0   0  98]]
--------------------------------
Classification Report:
               precision    recall  f1-score   support

           0       0.95      0.97      0.96       128
           1       0.99      0.94      0.96        97
           2       0.94      0.93      0.94       104
           3       0.98      1.00      0.99       128
           4       0.96      0.98      0.97       100

    accuracy                           0.97       557
   macro avg       0.97      0.96      0.96       557
weighted avg       0.97      0.97      0.97       557


        Using TfidfVectorizer........
Confusion Matrix:
 [[123   0   4   0   1]
 [  3  92   1   0   1]
 [  1   0  98   3   2]
 [  1   0   0 127   0]
 [  3   1   0   1  95]]
--------------------------------
Classification Report:
               precision    recall  f1-score   support

           0       0.94      0.96      0.95       128
           1       0.99      0.95      0.97        97
           2       0.95      0.94      0.95       104
           3       0.97      0.99      0.98       128
           4       0.96      0.95      0.95       100

    accuracy                           0.96       557
   macro avg       0.96      0.96      0.96       557
weighted avg       0.96      0.96      0.96       557

*********************************************************************************
Model:  DecisionTreeClassifier()
```

```
       Using CountVectorizer........
Confusion Matrix:
 [[104   2  11   4   7]
 [  4  84   1   6   2]
 [  6   3  85   6   4]
 [  1   2   0 123   2]
 [  5   3   2   3  87]]
---------------------------------
Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.81      0.84       128
           1       0.89      0.87      0.88        97
           2       0.86      0.82      0.84       104
           3       0.87      0.96      0.91       128
           4       0.85      0.87      0.86       100

    accuracy                           0.87       557
   macro avg       0.87      0.87      0.87       557
weighted avg       0.87      0.87      0.87       557


       Using TfidfVectorizer........
Confusion Matrix:
 [[102   3  12   6   5]
 [  3  84   2   6   2]
 [  4   3  88   5   4]
 [  1   2   1 122   2]
 [  6   4   2   2  86]]
---------------------------------
Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.80      0.84       128
           1       0.88      0.87      0.87        97
           2       0.84      0.85      0.84       104
           3       0.87      0.95      0.91       128
           4       0.87      0.86      0.86       100

    accuracy                           0.87       557
   macro avg       0.87      0.86      0.86       557
weighted avg       0.87      0.87      0.86       557

*********************************************************************************
Model:  KNeighborsClassifier()

       Using CountVectorizer........
Confusion Matrix:
 [[ 74   0   1  52   1]
 [  2  46   0  49   0]
 [  2   0  67  35   0]
 [  0   0   0 128   0]
 [  5   4   0  59  32]]
---------------------------------
Classification Report:
              precision    recall  f1-score   support
```

```
              0       0.89      0.58      0.70       128
              1       0.92      0.47      0.63        97
              2       0.99      0.64      0.78       104
              3       0.40      1.00      0.57       128
              4       0.97      0.32      0.48       100

    accuracy                           0.62       557
   macro avg       0.83      0.60      0.63       557
weighted avg       0.81      0.62      0.63       557


        Using TfidfVectorizer........
Confusion Matrix:
 [[119   0   5   1   3]
 [  1  91   1   0   4]
 [  2   2  99   1   0]
 [  2   0   0 126   0]
 [  2   1   3   0  94]]
---------------------------------
Classification Report:
               precision    recall  f1-score   support

           0       0.94      0.93      0.94       128
           1       0.97      0.94      0.95        97
           2       0.92      0.95      0.93       104
           3       0.98      0.98      0.98       128
           4       0.93      0.94      0.94       100

    accuracy                           0.95       557
   macro avg       0.95      0.95      0.95       557
weighted avg       0.95      0.95      0.95       557

"""
print()
```

Questionnaire

In [76]:
```
# How many news articles are present in the dataset that we have?
```

In [78]:
```
df.shape[0]
```

Out[78]:
```
2225
```

In [79]:
```
# Most of the news articles are from _____ category.
```

In [82]:
```
df['Category'].value_counts().idxmax()
```

Out[82]:
```
'Sports'
```

In [83]:
```
# Only ___ no. of articles belong to the 'Technology' category.
```

In [84]:
```
(df['Category'] == 'Technology').sum()
```

Out[84]:
```
401
```

In [85]:
```
# What are Stop Words and why should they be removed from the text data?
```

Stop words are common, uninformative words like "the," "and," and "in." They're removed from text data because they don't contribute much to meaning and can slow down processing.

In [86]:
```
# Explain the difference between Stemming and Lemmatization.
```

Both normalize words, but stemming shortens words to their base, even if it's not a real word ("chang" for "changes"). Lemmatization gets real word bases ("run" for "running").

In [87]:
```
# Which of the techniques Bag of Words or TF-IDF is considered to be more efficient
```

Bag of Words (BoW): Counts words in documents equally. TF-IDF: Considers word importance; slower due to IDF calculation. In practice, TF-IDF's slight efficiency difference isn't usually significant, and its weighting makes it more versatile for NLP tasks.

In [88]:
```
# What’s the shape of train & test data sets after performing a 75:25 split.
```

In [89]:
```
y_train.shape[0], y_test.shape[0]
```

Out[89]:
```
(1668, 557)
```

In [90]:
```
# Which of the following is found to be the best performing model..

# a. Random Forest b. Nearest Neighbors c. Naive Bayes
```

c. Naive Bayes Classifier

In [91]:
```
# According to this particular use case, both precision and recall are equally impo
```

True