```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
import re

from sklearn.preprocessing import LabelEncoder

from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer

from sklearn.model_selection import train_test_split

from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report

from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier

np.random.seed(42)
```

```
%%capture
nltk.download('punkt')  # Download for tokenization
nltk.download('stopwords')  # Download stopwords
nltk.download('wordnet')  # Download for lemmatization
```

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

```
sns.set_style('white')
plt.rcParams['figure.figsize'] = (9, 9)
plt.rcParams['date.autoformatter.day'] = '%d-%b'
```

```
!gdown 1I3-pQFzbSufhpMrUKAROBLGULXcWiB9u
```

```
Downloading...
From: https://drive.google.com/uc?id=1I3-pQFzbSufhpMrUKAROBLGULXcWiB9u
To: /content/flipitnews-data.csv
100% 5.06M/5.06M [00:00<00:00, 76.7MB/s]
```

```
!ls
```

```
flipitnews-data.csv  sample_data
```

```
df= pd.read_csv('flipitnews-data.csv')
```

```
df.shape
```

```
(2225, 2)
```

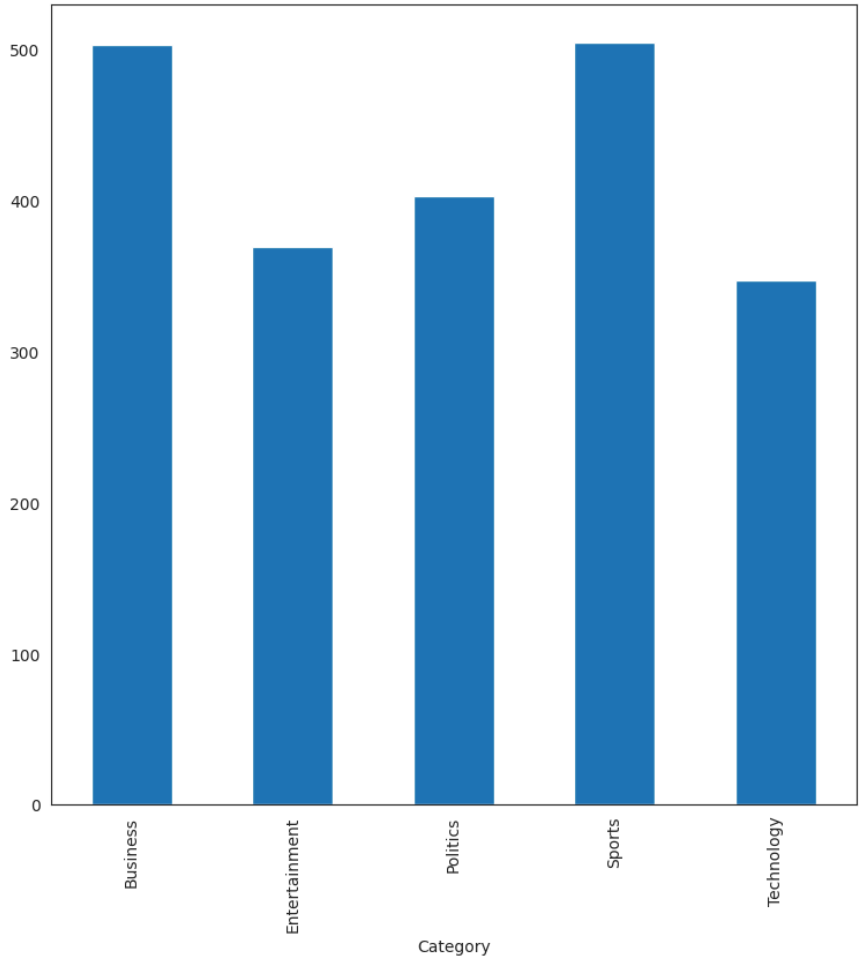
```
df.describe()
```

	Category	Article
count	2225	2225
unique	5	2126
top	Sports	kennedy questions trust of blair lib dem leade...
freq	511	2

```
df= df.drop_duplicates()
```

```
df.groupby('Category').size().plot(kind='bar')
```

```
<Axes: xlabel='Category'>
```




```
df.sample(5)
```

	Category	Article
283	Politics	plan to give elderly care control elderly and ...
2086	Business	beer giant swallows russian firm brewing giant...
1772	Sports	athens memories soar above lows well it s goo...
1008	Sports	corry backs skipper robinson england forward m...
2106	Business	us trade deficit widens sharply the gap betwee...

```
def preprocess_text(text):

    # Remove non-letters
    text = re.sub(r'[^a-zA-Z\s]', '', text)

    # Tokenize
    tokens = nltk.word_tokenize(text.lower())  # Lowercase for consistency

    # Remove stopwords
    stop_words = set(stopwords.words('english'))
    tokens = [word for word in tokens if word not in stop_words]

    # Lemmatize
    lemmatizer = WordNetLemmatizer()
    tokens = [lemmatizer.lemmatize(word) for word in tokens]

    # Join back into a string
    return ' '.join(tokens)


df['Processed_Articles'] = df['Article'].apply(preprocess_text)
```

```
sample = df.sample()


display(sample['Article'].values[0])
print('-'*100)
display(sample['Processed_Articles'].values[0])
```

```
'angry williams rejects criticism serena williams has angrily rejected claims
that she and sister venus are a declining force in tennis.  the sisters ended
last year without a grand slam title for the first time since 1998. but seren
a denied their challenge was fading  saying:  that s not fair - i m tired of
not saying anything.  we ve been practising hard. we ve had serious injuries.
i ve had surgery and after  i got to the wimbledon final. i don t know many w
ho have done that.  while serena is through to the australian open semi-final
s  venus went out in the fourth round  meaning she has not gone further than
the last eight in her last five grand slam appearances. but serena added:  ve
nus had a severe strain in her stomach. i actually had the same injury  but i
didn t tear it the way she did.  if i would have torn it  i wouldn t have be
```

```
le = LabelEncoder()
df['encoded_target'] = le.fit_transform(df['Category'])

df.sample(5)
```

	Category	Article	Processed_Articles	encoded_target
700	Politics	blunkett unveils policing plans people could b...	blunkett unveils policing plan people could gi...	2
1924	Politics	straw to attend auschwitz service foreign secr...	straw attend auschwitz service foreign secreta...	2
1984	Sports	llewellyn plans wales retirement	llewellyn plan wale retirement	3

```
def vectorize_articles(df, column_name, method='tfidf'):

    if method == 'tfidf':
        vectorizer = TfidfVectorizer()

    elif method == 'bow':
        vectorizer = CountVectorizer()
    else:
        raise ValueError("Invalid method. Choose between 'tfidf' or 'bow'.")

    return vectorizer.fit_transform(df[column_name])


bow_articles = vectorize_articles(df, 'Processed_Articles', method= 'bow')
tfidf_articles = vectorize_articles(df, 'Processed_Articles', method= 'tfidf')


X_train_bow, X_test_bow, y_train_bow, y_test_bow = train_test_split(
    bow_articles,
    df['encoded_target'],
    test_size=0.2,
    random_state=42,
    stratify= df['encoded_target']  # Stratify based on target variable
)


# Fit the Naive Bayes model
model = MultinomialNB()
model.fit(X_train_bow, y_train_bow)

# Make predictions on the test set
y_pred_bow = model.predict(X_test_bow)

# Evaluate the model
accuracy = accuracy_score(y_test_bow, y_pred_bow)
print("Accuracy:", accuracy)

print("Classification Report:")
print(classification_report(y_test_bow, y_pred_bow))
```

```
Accuracy: 0.9741784037558685
Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.96      0.96       101
           1       0.99      0.96      0.97        74
           2       0.99      0.98      0.98        81
           3       1.00      1.00      1.00       101
           4       0.93      0.97      0.95        69
```

```
          accuracy                           0.97       426
         macro avg       0.97      0.97      0.97       426
      weighted avg       0.97      0.97      0.97       426
```

```
X_train_tfidf, X_test_tfidf, y_train_tfidf, y_test_tfidf = train_test_split(
    tfidf_articles,
    df['encoded_target'],
    test_size=0.2,
    random_state=42,
    stratify= df['encoded_target']  # Stratify based on target variable
)


# Fit the Naive Bayes model for TF-IDF vectorised data
model = MultinomialNB()
model.fit(X_train_tfidf, y_train_tfidf)

# Make predictions on the test set
y_pred_tfidf = model.predict(X_test_tfidf)

# Evaluate the model
accuracy = accuracy_score(y_test_tfidf, y_pred_tfidf)
print("Accuracy:", accuracy)

print("Classification Report:")
print(classification_report(y_test_tfidf, y_pred_tfidf))
```

```
     Accuracy: 0.9530516431924883
     Classification Report:
                   precision    recall  f1-score   support

                0       0.89      1.00      0.94       101
                1       1.00      0.89      0.94        74
                2       0.94      0.98      0.96        81
                3       0.97      1.00      0.99       101
                4       1.00      0.86      0.92        69

            accuracy                           0.95       426
           macro avg       0.96      0.94      0.95       426
        weighted avg       0.96      0.95      0.95       426
```

```
def train_evaluate_model(model, X_train, X_test, y_train, y_test):

  # Fit the model
  model.fit(X_train, y_train)

  # Make predictions on the test set
  y_pred = model.predict(X_test)

  # Evaluate the model
  accuracy = accuracy_score(y_test, y_pred)
  classification_report_ = classification_report(y_test, y_pred)

  # Return evaluation metrics
  return {'accuracy': accuracy, 'classification_report': classification_report_}


decision_tree_bow = train_evaluate_model(DecisionTreeClassifier(), X_train_bow, X_test_bow, y_train_bow, y_test_bow)

accuracy_bow = decision_tree_bow['accuracy']
classification_report_bow = decision_tree_bow['classification_report']

print(accuracy_bow)
print(classification_report_bow)
```

```
     0.8380281690140845
                   precision    recall  f1-score   support

                0       0.79      0.91      0.84       101
                1       0.82      0.74      0.78        74
                2       0.86      0.81      0.84        81
                3       0.91      0.92      0.92       101
                4       0.81      0.74      0.77        69

            accuracy                           0.84       426
           macro avg       0.84      0.83      0.83       426
        weighted avg       0.84      0.84      0.84       426
```

```
decision_tree_tfidf = train_evaluate_model(DecisionTreeClassifier(), X_train_tfidf, X_test_tfidf, y_train_tfidf, y_test_tfic

accuracy_tfidf = decision_tree_tfidf['accuracy']
classification_report_tfidf = decision_tree_tfidf['classification_report']

print(accuracy_tfidf)
print(classification_report_tfidf)
```

```
0.8215962441314554
              precision    recall  f1-score   support

           0       0.79      0.88      0.84       101
           1       0.74      0.84      0.78        74
           2       0.91      0.75      0.82        81
           3       0.88      0.88      0.88       101
           4       0.79      0.71      0.75        69

    accuracy                           0.82       426
   macro avg       0.82      0.81      0.81       426
weighted avg       0.83      0.82      0.82       426
```

```
random_forest_bow = train_evaluate_model(RandomForestClassifier(), X_train_bow, X_test_bow, y_train_bow, y_test_bow)

accuracy_bow = random_forest_bow['accuracy']
classification_report_bow = random_forest_bow['classification_report']

print(accuracy_bow)
print(classification_report_bow)
```

```
0.9624413145539906
              precision    recall  f1-score   support

           0       0.88      1.00      0.94       101
           1       1.00      0.93      0.97        74
           2       1.00      0.95      0.97        81
           3       0.98      1.00      0.99       101
           4       1.00      0.90      0.95        69

    accuracy                           0.96       426
   macro avg       0.97      0.96      0.96       426
weighted avg       0.97      0.96      0.96       426
```

```
random_forest_tfidf = train_evaluate_model(RandomForestClassifier(), X_train_tfidf, X_test_tfidf, y_train_tfidf, y_test_tfic

accuracy_tfidf = random_forest_tfidf['accuracy']
classification_report_tfidf = random_forest_tfidf['classification_report']

print(accuracy_tfidf)
print(classification_report_tfidf)
```

```
0.960093896713615
              precision    recall  f1-score   support

           0       0.90      0.97      0.93       101
           1       1.00      0.96      0.98        74
           2       0.99      0.95      0.97        81
           3       0.97      1.00      0.99       101
           4       0.97      0.90      0.93        69

    accuracy                           0.96       426
   macro avg       0.97      0.96      0.96       426
weighted avg       0.96      0.96      0.96       426
```

```
knn_bow = train_evaluate_model(KNeighborsClassifier(), X_train_bow, X_test_bow, y_train_bow, y_test_bow)

accuracy_bow = knn_bow['accuracy']
classification_report_bow = knn_bow['classification_report']

print(accuracy_bow)
print(classification_report_bow)
```

```
0.6784037558685446
              precision    recall  f1-score   support

           0       0.86      0.66      0.75       101
           1       0.89      0.53      0.66        74
           2       0.89      0.67      0.76        81
           3       0.47      1.00      0.64       101
           4       1.00      0.41      0.58        69

    accuracy                           0.68       426
   macro avg       0.82      0.65      0.68       426
```

```
weighted avg       0.80      0.68      0.68       426
```

```
knn_tfidf = train_evaluate_model(KNeighborsClassifier(), X_train_tfidf, X_test_tfidf, y_train_tfidf, y_test_tfidf)

accuracy_tfidf = knn_tfidf['accuracy']
classification_report_tfidf = knn_tfidf['classification_report']

print(accuracy_tfidf)
print(classification_report_tfidf)
```

```
0.9530516431924883
              precision    recall  f1-score   support

           0       0.95      0.94      0.95       101
           1       0.96      0.91      0.93        74
           2       0.92      0.98      0.95        81
           3       0.99      0.99      0.99       101
           4       0.94      0.94      0.94        69

    accuracy                           0.95       426
   macro avg       0.95      0.95      0.95       426
weighted avg       0.95      0.95      0.95       426
```

Questionnaire:

How many news articles are present in the dataset that we have?

Ans. 2,126

```
df['Article'].nunique()
```

```
2126
```

Most of the news articles are from _____ category.

Ans. Sports

```
df['Category'].value_counts()
```

```
Category
Sports           504
Business         503
Politics         403
Entertainment    369
Technology       347
Name: count, dtype: int64
```

Only ___ no. of articles belong to the ‘Technology’ category.

Ans. 347

What are Stop Words and why should they be removed from the text data?

Ans. Stop words are common words in a language (like "the", "and", "is") that carry little meaning on their own.

They are removed for following reasons. Noise Reduction: Stop words can clutter text data, making it harder for models to focus on important keywords. Efficiency: Removing stop words reduces dataset size and speeds up processing. Improved Relevance: Focusing on meaningful words enhances the accuracy of text analysis and machine learning models.

Which of the techniques Bag of Words or TF-IDF is considered to be more efficient than the other?

Ans. Generally, TF-IDF is considered more efficient than Bag-of-Words for several reasons:

Reduced Vocabulary: TF-IDF assigns lower weights to common, less informative words, leading to a smaller vocabulary size. Faster Computations: Calculations with the reduced vocabulary tend to be faster. Better for Search: TF-IDF is highly effective for search-like tasks where you want to find documents most relevant to a query, even if those documents don't share the exact words.

What’s the shape of train & test data sets after performing a 75:25 split.

Train: 1994 Test: 532

```
X_train, X_test, y_train, y_test = train_test_split(
    tfidf_articles,
    df['encoded_target'],
    test_size=0.25,
    random_state=42,
    stratify= df['encoded_target']  # Stratify based on target variable
)
```

```
X_train.shape
X_test.shape
```

```
(532, 27175)
```

Which of the following is found to be the best performing model..

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

Ans. Naive Bayes

According to this particular use case, both precision and recall are equally important. (T/F)

Ans. True! For classifying news articles into categories, both precision and recall hold significant importance.

Precision: Ensures that when an article is classified into a category, it's highly likely to actually belong to that category. This prevents false positives and maintains the quality of your classification results.

Recall: Ensures that as many relevant articles as possible are correctly identified within each category. This minimizes false negatives, helping you capture a greater portion of the truly relevant articles.