```
import spacy
import string
import warnings

import numpy as np
import pandas as pd

from pprint import pprint
from IPython.utils import io
from tqdm.notebook import tqdm
from gensim.models import Word2Vec

# Data Modeling
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report
from sklearn.metrics import confusion_matrix

#from langdetect import DetectorFactory, detect
from IPython.core.display import HTML, display
from IPython.display import Image
from spacy.lang.en.stop_words import STOP_WORDS

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
import plotly.express as px
# SetUp NLTK
!pip install --user -U nltk



warnings.filterwarnings('ignore')
tqdm.pandas()
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.11/dist-packages (3.9.1)
Requirement already satisfied: click in /usr/local/lib/python3.11/dist-packages (from nltk) (8.1.8)
Requirement already satisfied: joblib in /usr/local/lib/python3.11/dist-packages (from nltk) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.11/dist-packages (from nltk) (2024.11.6)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (from nltk) (4.67.1)
```

```
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]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
True
```

```
!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, 23.1MB/s]
```

```
from google.colab import files
uploaded = files.upload()
```

```
Choose Files  No file chosen    Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to
```

```
!gdown 1cMaGmxpJbOKV7_mok5XQ8AKqUwI12Cru
```

```
DATA = pd.read_csv("flipitnews-data.csv")
DATA.reset_index(inplace=True, drop=True)
print(DATA.columns)
DATA.head(2)
```

```
Index(['Category', 'Article'], dtype='object')
```

	Category	Article
0	Technology	tv future in the hands of viewers with home th...
1	Business	worldcom boss left books alone former worldc...

```
DATA.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2225 entries, 0 to 2224
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   Category  2225 non-null   object
 1   Article   2225 non-null   object
dtypes: object(2)
memory usage: 34.9+ KB
```

News_article per category

```
DATA["Category"].value_counts()
```

Category	count
Sports	511
Business	510
Politics	417
Technology	401
Entertainment	386

dtype: int64

Dataset Shape

```
DATA.shape
```

```
(2225, 2)
```

```
DATA.describe(include='all')
```

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

total unique categories are 5 and unique articles are 2126. Top category is "Sports"

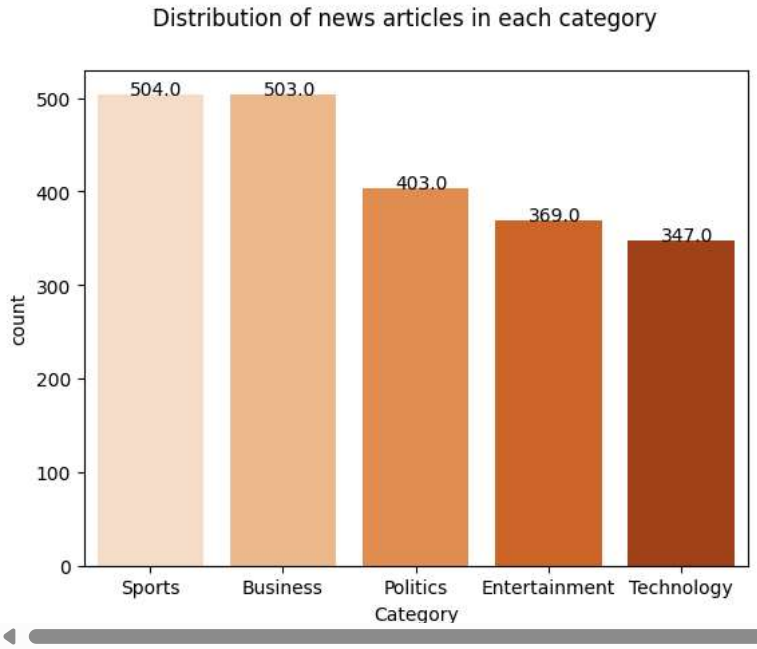
Drop Duplicates

```
DATA.drop_duplicates(['Article'], inplace=True)
DATA.dropna(inplace=True)
DATA.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 2126 entries, 0 to 2224
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   Category  2126 non-null   object
 1   Article   2126 non-null   object
dtypes: object(2)
memory usage: 49.8+ KB
```

Distribution of news articles in each category

```
# matplotlib for vizualization
import matplotlib.pyplot as plt
import seaborn as sns

#distribution of news articles in each category

ax=sns.countplot(x=DATA["Category"],palette="Oranges",order=DATA["Category"].value_counts().index)
for p in ax.patches:
   ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.25, p.get_height()+0.01))

plt.suptitle("Distribution of news articles in each category")
plt.show()
```



- The graph shows that maximum number of articles are from sports and business category.

```
DATA.Category.value_counts(normalize=True)*100
```

Category	proportion
Sports	23.706491
Business	23.659454
Politics	18.955786
Entertainment	17.356538
Technology	16.321731

dtype: float64

*the analysis clearly show that ,23% of the articles are from sports category and 23% article from business category. and these are the top two categories

Solving common preprocessing Problems in NLP.

Remove Non-Letters

```
import re
def remove_non_letters(text):
   return re.sub('[^a-zA-Z]', '', text)
```

```
#DATA["Article"]= DATA["Article"].progress_apply(remove_non_letters)
```

Removing Stop Words and punctuations

*Stop words are common words like "the," "and," "is," etc., that often don't carry significant meaning in text analysis. Removing them can help focus on the core content of the text.

```
import nltk
from nltk.corpus import stopwords

nltk.download('stopwords')

def remove_stopwords(text):
    stop_words = set(stopwords.words('english'))
    words = text.split()
    words = [word for word in words if word not in stop_words]
    return ' '.join(words)
```

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

```
punctuations = string.punctuation
stopwords = list(STOP_WORDS)


def pre_processor(sentence):
    mytokens = sentence.split(' ')
    mytokens = [word.lower() for word in mytokens if word not in stopwords and word not in punctuations]
    mytokens = " ".join([i for i in mytokens])
    return mytokens

DATA["Article"] = DATA["Article"].progress_apply(pre_processor)
DATA.head()
```

```
100%                                                2126/2126 [00:05<00:00, 641.81it/s]
```

	Category	Article
0	Technology	tv future hands viewers home theatre systems p...
1	Business	worldcom boss left books worldcom boss bernie ...
2	Sports	tigers wary farrell gamble leicester rushed ma...
3	Sports	yeading face newcastle fa cup premiership newc...
4	Entertainment	ocean s raids box office ocean s crime caper s...

Word Tokenization

*Word tokenization is the process of breaking down a text into individual words.

```
def word_tokenize(text):
    text = text.lower()
    text = re.sub(r'[^\w\s]', '', text)
    tokens = text.split()
    return tokens
```

Lemmatization

```
#import lemmetizer
import nltk
nltk.download('wordnet')
from nltk.stem import WordNetLemmatizer

# Create a WordNetLemmatizer object
lemmatizer = WordNetLemmatizer()

# Define a function to lemmatize a single word
def lemmatize_word(word):
  return lemmatizer.lemmatize(word)
```

```
# Apply the lemmatization function to the 'text' column
DATA['lemmatized_text'] = DATA['Article'].apply(lambda x: ' '.join([lemmatize_word(word) for word in x.split()]))
```

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

```
DATA.head(5)
```

	Category	Article	lemmatized_text
0	Technology	tv future hands viewers home theatre systems p...	tv future hand viewer home theatre system plas...
1	Business	worldcom boss left books worldcom boss bernie ...	worldcom bos left book worldcom bos bernie ebb...
2	Sports	tigers wary farrell gamble leicester rushed ma...	tiger wary farrell gamble leicester rushed mak...
3	Sports	yeading face newcastle fa cup premiership newc...	yeading face newcastle fa cup premiership newc...
4	Entertainment	ocean s raids box office ocean s crime caper s...	ocean s raid box office ocean s crime caper se...

Encoding and Transforming the data

1. Encoding the target variable

```
pip install category_encoders
```

```
Collecting category_encoders
  Downloading category_encoders-2.8.0-py3-none-any.whl.metadata (7.9 kB)
Requirement already satisfied: numpy>=1.14.0 in /usr/local/lib/python3.11/dist-packages (from category_encoders) (1.26.4)
Requirement already satisfied: pandas>=1.0.5 in /usr/local/lib/python3.11/dist-packages (from category_encoders) (2.2.2)
Requirement already satisfied: patsy>=0.5.1 in /usr/local/lib/python3.11/dist-packages (from category_encoders) (1.0.1)
Requirement already satisfied: scikit-learn>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from category_encoders) (1.6.0)
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.11/dist-packages (from category_encoders) (1.13.1)
Requirement already satisfied: statsmodels>=0.9.0 in /usr/local/lib/python3.11/dist-packages (from category_encoders) (0.14.4)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.0.5->category_encoc
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.0.5->category_encoders) (2024
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.0.5->category_encoders) (20
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn>=1.6.0->category_encoders
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn>=1.6.0->category_e
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.11/dist-packages (from statsmodels>=0.9.0->category_encoder
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas>=1.0.5->cate
Downloading category_encoders-2.8.0-py3-none-any.whl (85 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 85.7/85.7 kB 2.1 MB/s eta 0:00:00
Installing collected packages: category_encoders
Successfully installed category_encoders-2.8.0
```

```
import category_encoders as ce
encode=ce.OrdinalEncoder(cols=["Category"])
DATA["Category_id"]=encode.fit_transform(DATA[["Category"]])
DATA.head()
```

	Category	Article	lemmatized_text	Category_id
0	Technology	tv future hands viewers home theatre systems p...	tv future hand viewer home theatre system plas...	1
1	Business	worldcom boss left books worldcom boss bernie ...	worldcom bos left book worldcom bos bernie ebb...	2
2	Sports	tigers wary farrell gamble leicester rushed ma...	tiger wary farrell gamble leicester rushed mak...	3
3	Sports	yeading face newcastle fa cup premiership newc...	yeading face newcastle fa cup premiership newc...	3
4	Entertainment	ocean s raids box office ocean s crime caper s...	ocean s raid box office ocean s crime caper se...	4

Tokenization

```
pip install nltk
```

```
import nltk
from nltk.tokenize import word_tokenize ,sent_tokenize
nltk.download('punkt')

word_cnt,unique_word_cnt=0,0

#Corpus-----> the entire Document
```

```
corpus=DATA['Article'].str.cat(sep=', ')

print('Number of words in the entire corpus:',len(corpus))

#Find the letters used in Corpus
Unique_char=set(DATA['Article'].str.cat(sep=', '))
print('Unique letters used in corpus:',Unique_char)

#Vocabulary of all articles
Vocabulary = list(DATA['Article'].str.cat(sep=', '))
print('Vocabulary of all articles:',Vocabulary)

for i in set(word_tokenize(Vocabulary)):
  unique_word_cnt+=1
print('Number of words in the vocabulary:',unique_word_cnt)
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
Number of words in the entire corpus: 3215347
Unique letters used in corpus: {'4', '(', '8', 'l', ',', 'u', 'c', '1', 'b', '5', ']', 'f', 'd', '`', '[', '@', '&', 'g', 'r', 'w',
Vocabulary of all articles: ['t', 'v', ' ', 'f', 'u', 't', 'u', 'r', 'e', ' ', 'h', 'a', 'n', 'd', 's', ' ', 'v', 'i', 'e', 'w', 'e
---------------------------------------------------------------------------
LookupError                               Traceback (most recent call last)
<ipython-input-23-7a7b061fe2e9> in <cell line: 0>()
     18 print('Vocabulary of all articles:',Vocabulary)
     19
---> 20 for i in set(word_tokenize(Vocabulary)):
     21   unique_word_cnt+=1
     22 print('Number of words in the vocabulary:',unique_word_cnt)

                                 5 frames
/usr/local/lib/python3.11/dist-packages/nltk/data.py in find(resource_name, paths)
    577     sep = "*" * 70
    578     resource_not_found = f"\n{sep}\n{msg}\n{sep}\n"
--> 579     raise LookupError(resource_not_found)
    580
    581

LookupError:
**********************************************************************
  Resource punkt_tab not found.
  Please use the NLTK Downloader to obtain the resource:

  >>> import nltk
  >>> nltk.download('punkt_tab')

  For more information see: https://www.nltk.org/data.html

  Attempted to load tokenizers/punkt_tab/english/

  Searched in:
    - '/root/nltk_data'
    - '/usr/nltk_data'
    - '/usr/share/nltk_data'
    - '/usr/lib/nltk_data'
    - '/usr/share/nltk_data'
    - '/usr/local/share/nltk_data'
    - '/usr/lib/nltk_data'
    - '/usr/local/lib/nltk_data'
**********************************************************************
```

2. Bag Of Words

- It computes the frequencies (or) presence of a word in a document of a corpus
- It assigns equal priority to every word. It cannot detect stopwords

Start coding or generate with AI.

3. TF-IDF

- It computes the feature importances of a word in a document of a corpus
- It can detect and nullify the effect of stopwords on feature vector of a sentence

```
from sklearn.feature_extraction.text import TfidfVectorizer

def perform_tfidf(df, text_column):
  # Create a TfidfVectorizer object
```

```
  tfidf_vectorizer = TfidfVectorizer(stop_words='english')

  # Fit and transform the text data
  tfidf_matrix = tfidf_vectorizer.fit_transform(df[text_column])

  return tfidf_matrix, tfidf_vectorizer

tfidf_matrix, tfidf_vectorizer = perform_tfidf(DATA, 'lemmatized_text')

#Get the feature names (words)
feature_names = tfidf_vectorizer.get_feature_names_out()

# Create a DataFrame from the TF-IDF matrix
tfidf_DATA = pd.DataFrame(tfidf_matrix.toarray(), columns=feature_names)

# Concatenate the TF-IDF DataFrame with the original DataFrame
DATA_with_tfidf = pd.concat([DATA["Article"], tfidf_DATA], axis=1)

DATA_with_tfidf.head()
```

	Article	00	000	0001	000bn	000m	000s	000th	001	001and	...	zooms	zooropa	zornotza	zorro	zubair	zuluaga	zuric
0	tv future hands viewers home theatre systems p...	0.0	0.019570	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.
1	worldcom boss left books worldcom boss bernie ...	0.0	0.024161	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.

```
tf_idf=TfidfVectorizer()
X=tf_idf.fit_transform(DATA["lemmatized_text"]).toarray()
y=np.array(DATA["Category"].values)
```

4. Train-Test Split

```
X_train,X_val,y_train,y_val=train_test_split(X,y,test_size=0.25,shuffle=True,stratify=y)
```

```
print("Shape of X_train:", X_train.shape)
print("Shape of X_val:", X_val.shape)
print("Shape of y_train:", y_train.shape)
print("Shape of y_val:", y_val.shape)
```

```
Shape of X_train: (1594, 27938)
Shape of X_val: (532, 27938)
Shape of y_train: (1594,)
Shape of y_val: (532,)
```

Model Training & Evaluation

Simple Approach: Naive Bayes

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score,roc_auc_score

nb=MultinomialNB()
nb.fit(X_train,y_train)

nb.train=accuracy_score(y_train,nb.predict(X_train))
nb.test=accuracy_score(y_val,nb.predict(X_val))
print("Training Accuracy:",nb.train)
print("Test Accuracy:",nb.test)
y_pred=nb.predict(X_val)
y_pred_praba=nb.predict_proba(X_val)
```

```
roc_auc_score(y_val,y_pred_praba,multi_class='ovr')
```

```
Training Accuracy: 0.986198243412798
Test Accuracy: 0.9624060150375939
0.9987901198670883
```

```
from sklearn.metrics import precision_score,recall_score,f1_score
_
precision=precision_score(y_val,y_pred,average='weighted')
recall=recall_score(y_val,y_pred,average='weighted')
f1=f1_score(y_val,y_pred,average='weighted')


print("Precision:",precision)
print("Recall:",recall)
print("F1 Score:",f1)
```

```
Precision: 0.9641583700862123
Recall: 0.9624060150375939
F1 Score: 0.962596215366759
```

Functionalized Code

```
def conf_matrix(y_test, y_pred):
  from sklearn.metrics import confusion_matrix
  import seaborn as sns
  conf_mat = confusion_matrix(y_test, y_pred)

  sns.heatmap(conf_mat, annot=True, xticklabels=DATA['Category'].unique(), yticklabels=DATA['Category'].unique(), cmap="YlGnBu", fmt='g
```

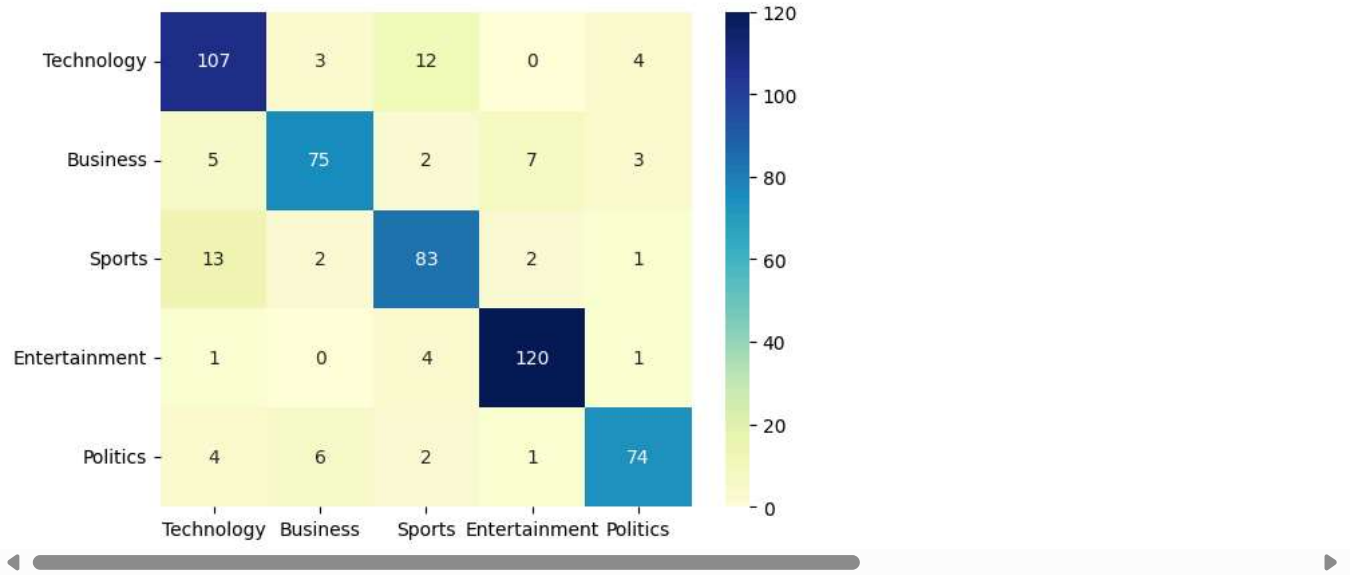
```
def model_train(obj):
  obj.fit(X_train, y_train) # Training the model
  y_pred = obj.predict(X_val) # Making predictions
  y_pred_proba = obj.predict_proba(X_val)
  return y_pred, y_pred_proba
```

```
def model_eval(obj, y_pred, y_pred_proba):
    print("-------------------------")

    # Calculating the train & test accuracy
    train_acc = accuracy_score(y_train, obj.predict(X_train))
    test_acc = accuracy_score(y_val, obj.predict(X_val))

    print("Train Accuracy: {:.3f}".format(train_acc))
    print("Test Accuracy: {:.3f}\n".format(test_acc))

    # Computing the ROC AUC score
    print("ROC AUC Score: {:.3f}\n".format(roc_auc_score(y_val, y_pred_proba, multi_class='ovr')))

    # Computing the precision, recall & f1 score
    precision = precision_score(y_val, y_pred, average='weighted')
    recall = recall_score(y_val, y_pred, average='weighted')
    f1 = f1_score(y_val, y_pred, average='weighted')

    print("Precision: {:.3f}".format(precision))
    print("Recall: {:.3f}".format(recall))
    print("F1 Score: {:.3f}".format(f1))

    print("-------------------------")
```

Decision Tree

```
dt = DecisionTreeClassifier()

y_pred_dt, y_pred_proba_dt = model_train(dt)

model_eval(dt, y_pred_dt, y_pred_proba_dt)

conf_matrix(y_val, y_pred_dt)
```

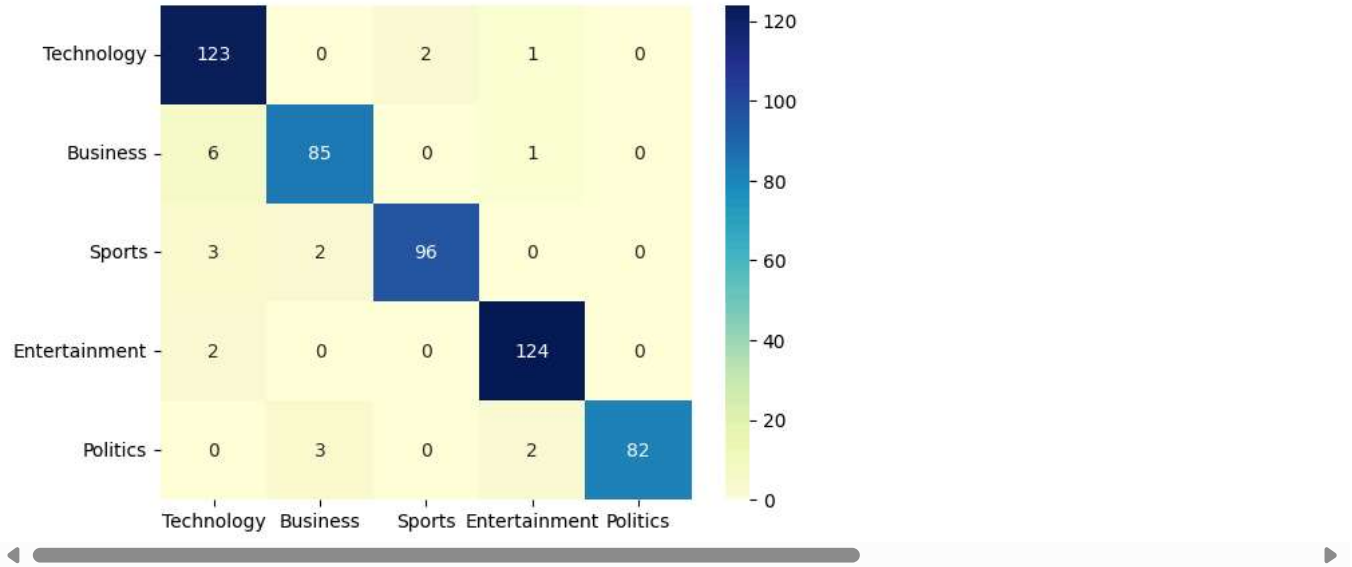
```
-------------------------
Train Accuracy: 1.000
Test Accuracy: 0.863

ROC AUC Score: 0.912

Precision: 0.863
Recall: 0.863
F1 Score: 0.863
-------------------------
```



K Nearest Neighbors

```
knn = RandomForestClassifier()

y_pred_knn, y_pred_proba_knn = model_train(knn)

model_eval(dt, y_pred_knn, y_pred_proba_knn)

conf_matrix(y_val, y_pred_knn)
```

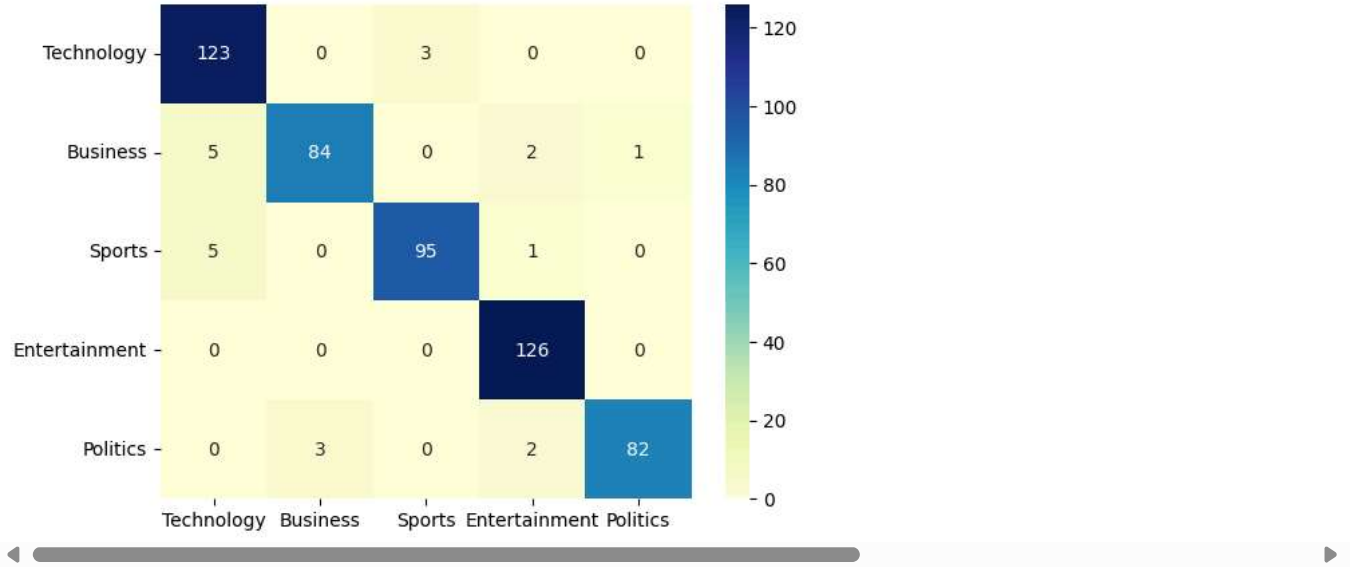
```
-------------------------
Train Accuracy: 1.000
Test Accuracy: 0.863

ROC AUC Score: 0.998

Precision: 0.960
Recall: 0.959
F1 Score: 0.959
-------------------------
```



Random Forest

```
rf = RandomForestClassifier()

y_pred_rf, y_pred_proba_rf = model_train(rf)

model_eval(dt, y_pred_rf, y_pred_proba_rf)

conf_matrix(y_val, y_pred_rf)
```

```
-------------------------
Train Accuracy: 1.000
Test Accuracy: 0.863

ROC AUC Score: 0.998

Precision: 0.959
Recall: 0.959
F1 Score: 0.959
-------------------------
```



Analysis with Bag of Words

```
from sklearn.feature_extraction.text import CountVectorizer

cv = CountVectorizer(max_features = 5000)

X = cv.fit_transform(DATA['Article']).toarray()
y = np.array(DATA['Category'].values)


X_train, X_val, y_train, y_val = train_test_split(X, y, test_size = 0.25, shuffle=True, stratify=y)
```

Multinomial NB

```
nb = MultinomialNB()

nb.fit(X_train, y_train)
```

```
MultinomialNB()
```

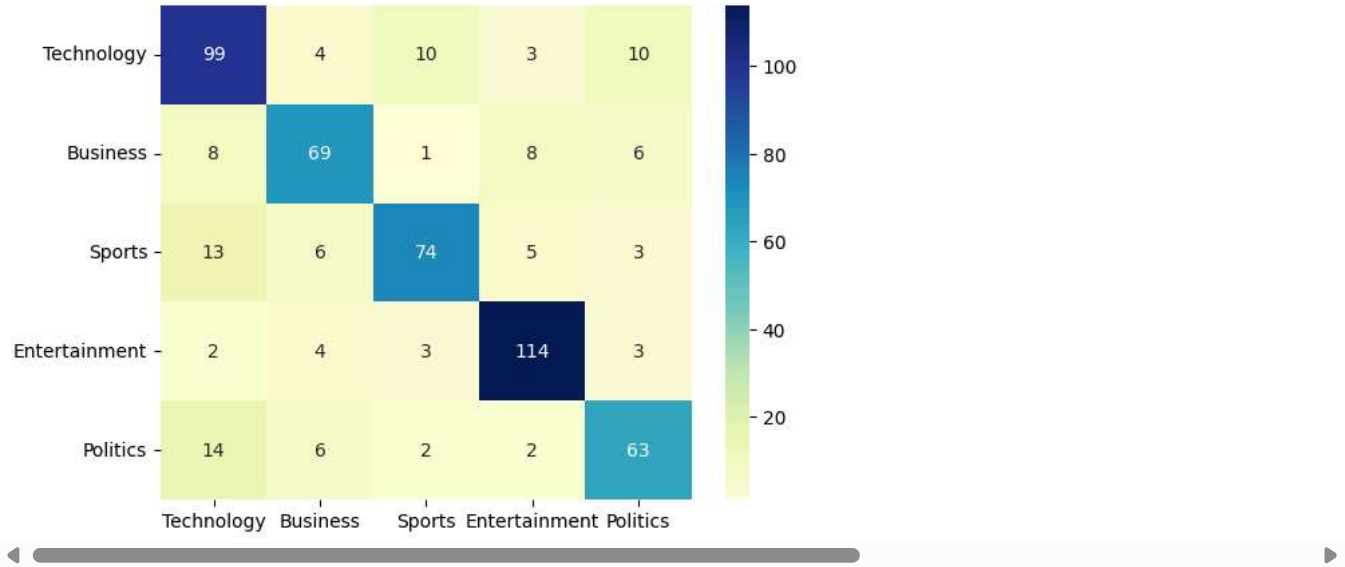
```
nb_train = accuracy_score(y_train, nb.predict(X_train))
nb_test = accuracy_score(y_val, nb.predict(X_val))


print("Train accuracy: ", nb_train)
print("Test Accuracy: ", nb_test)
```

```
Train accuracy:  0.986198243412798
Test Accuracy:  0.9624060150375939
```

Decision tree classiifer

```
dt = DecisionTreeClassifier()

y_pred_dt, y_pred_proba_dt = model_train(dt)

model_eval(dt, y_pred_dt, y_pred_proba_dt)
conf_matrix(y_val, y_pred_dt)
```

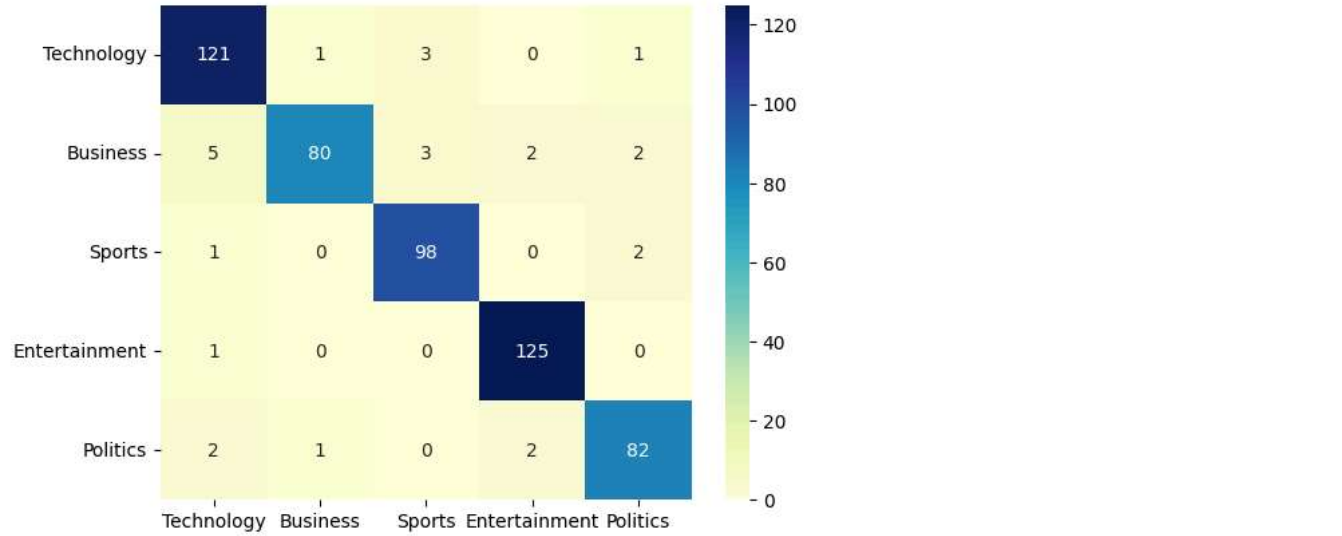
```
-------------------------
Train Accuracy: 1.000
Test Accuracy: 0.788

ROC AUC Score: 0.863

Precision: 0.788
Recall: 0.788
F1 Score: 0.787
-------------------------
```



K Nearest Neighbors

```
knn = RandomForestClassifier()

y_pred_knn, y_pred_proba_knn = model_train(knn)

model_eval(dt, y_pred_knn, y_pred_proba_knn)

conf_matrix(y_val, y_pred_knn)
```

```
-------------------------
Train Accuracy: 1.000
Test Accuracy: 0.797

ROC AUC Score: 0.997

Precision: 0.952
Recall: 0.951
F1 Score: 0.951
-------------------------
```



Start coding or generate with AI.

LSTM

```
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Embedding, GRU, SimpleRNN
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
```

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

	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...

```
max_features = 5000
maxlen = 100
embedding_size = 100
batch_size = 500
epochs = 10
```

```
def preprocess_text(DATA, text_column):
  DATA[text_column] = DATA[text_column].apply(lambda x: x.lower())
  return DATA

DATA = preprocess_text(DATA, 'Article')
```

```
tokenizer = Tokenizer(num_words = max_features)
tokenizer.fit_on_texts(DATA['Article'])

sequences = tokenizer.texts_to_sequences(DATA['Article'])
data = pad_sequences(sequences, maxlen = maxlen)


le = LabelEncoder()

labels = le.fit_transform(DATA['Category'])
labels = tf.keras.utils.to_categorical(labels)


labels
```

```
array([[0., 0., 0., 0., 1.],
       [1., 0., 0., 0., 0.],
       [0., 0., 0., 1., 0.],
       ...,
       [0., 1., 0., 0., 0.],
       [0., 0., 1., 0., 0.],
       [0., 0., 0., 1., 0.]])
```

```
#train test split
X_train, X_test, y_train, y_test = train_test_split(data, labels, test_size = 0.2, random_state=42)


def load_glove_embeddings(embedding_path, embedding_dim, tokenizer, max_features):

  embeddings_index = {}

  with open(embedding_path, 'r', encoding='utf8') as f:
    for line in f:
      values = line.split()
      word = values[0]
      coefs = np.asarray(values[1:], dtype='float32')
      embeddings_index[word] = coefs


  limited_word_index = {word:index for word, index, in tokenizer.word_index.items() if index < max_features}

  embedding_matrix =  np.zeros((min(max_features + 1, len(limited_word_index) + 1), embedding_dim))


  for word, i in limited_word_index.items():
    if i > max_features:
      continue

    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
      embedding_matrix[i] = embedding_vector

  return embedding_matrix
```

Start coding or generate with AI.

```
embedding_matrix = load_glove_embeddings('glove.6B.100d.txt', embedding_size, tokenizer, max_features)
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-75-f159f3706821> in <cell line: 0>()
----> 1 embedding_matrix = load_glove_embeddings('glove.6B.100d.txt', embedding_size, tokenizer, max_features)

<ipython-input-74-e3fabf746770> in load_glove_embeddings(embedding_path, embedding_dim, tokenizer, max_features)
      3   embeddings_index = {}
      4
----> 5   with open(embedding_path, 'r', encoding='utf8') as f:
      6     for line in f:
      7       values = line.split()

FileNotFoundError: [Errno 2] No such file or directory: 'glove.6B.100d.txt'
```

Questionnaire:

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

```
DATA["Article"].nunique()
```

```
2126
```

```
DATA["Category"].value_counts(normalize=True)*100
```

Category	proportion
Sports	22.966292
Business	22.921348
Politics	18.741573
Technology	18.022472
Entertainment	17.348315

dtype: float64

- Total News articles are 2126.
- And each category percentage in article count is Sports 22.96% Business 22.92% Politics 18.74% Technology 18.02% Entertainment 17.34%

2. Most of the news articles are from _____ category.

- Most of the news articles from sports category.

```
DATA["Category"].value_counts()
```

Category	count
Sports	511
Business	510
Politics	417
Technology	401
Entertainment	386

dtype: int64

3. Only ___ no. of articles belong to the 'Technology' category.

```
DATA["Article"][DATA["Category"]=="Technology"].nunique()
```

```
347
```

- 347 unique 'Technology' category data are avialable in dataset.

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

```
from nltk.corpus import stopwords
nltk.download('stopwords')
# Load English stopwords
stop_words = set(stopwords.words('english'))


def find_stopwords(text):
    words = text.lower().split()
    return [word for word in words if word in stop_words]

# Apply the function to the article column
DATA["Stopwords"] = DATA["Article"].apply(find_stopwords)
```

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

```
def get_unique_words_in_column(df, column_name):
  all_words = []
  for text in df[column_name]:
      for i in text:
        words = i.lower().split()  # Split into words and convert to lowercase
```

```
        all_words.extend(words)
    return set(all_words)

unique_words = get_unique_words_in_column(DATA, 'Stopwords')
pruniint(unique_words)
```

```
{'does', 'under', 'haven', 'itself', 'such', 'hasn', 'had', 'few', 'over', 'you', 'its', 'd', 'this', 'too', 'to', 'me', 'were', 'he
```

```
print("Stopwords count in the dataset are:",len(unique_words) )

print("Stopwords in the dataset are:", unique_words )
```

```
Stopwords count in the dataset are: 149
Stopwords in the dataset are: {'does', 'under', 'haven', 'itself', 'such', 'hasn', 'had', 'few', 'over', 'you', 'its', 'd', 'this',
```

5. **Explain the difference between Stemming and Lemmatization.** Answer: Stemming is the process of removing word endings to get to the base form of a word, while lemmatization is the process of reducing a word to its root form. Both are used to analyze the meaning of words