Problem Statement

The Gurugram-based company 'FlipItNews' aims to revolutionize the way Indians perceive finance, business, and capital market investment, by giving it a boost through artificial intelligence (AI) and machine learning (ML). They're on a mission to reinvent financial literacy for Indians, where financial awareness is driven by smart information discovery and engagement with peers. Through their smart content discovery and contextual engagement, the company is simplifying business, finance, and investment for millennials and first-time investors

Objective:

The goal of this project is to use a bunch of news articles extracted from the companies' internal database and categorize them into several categories like politics, technology, sports, business and entertainment based on their content. Use natural language processing and create & compare at least three different models.

Attribute Information:

- Article
- Category

The features names are themselves pretty self-explanatory

Concepts Tested:

- Natural Language Processing
- Text Processing
    - Stopwords, Tokenization, Lemmatization
    - Bag of Words, TF-IDF
- Multi-class Classification

1. Importing the libraries & Reading the data file
2. Exploring the dataset
    - Shape of the dataset
    - News articles per category
3. Processing the Textual Data i.e. the news articles
    - Removing the non-letters
    - Tokenizing the text
    - Removing stopwords
    - Lemmatization
4. Encoding and Transforming the data
    - Encoding the target variable
    - Bag of Words
    - TF-IDF
    - Train-Test Split
5. Model Training & Evaluation
    - Simple Approach
        1. Naive Bayes
    - Functionalized Code (Optional)
        1. Decision Tree
        2. Nearest Neighbors
        3. Random Forest

1. Importing the libraries & Reading the data file

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
# To ignore all warnings
import warnings

# For reading & manipulating the data
import pandas as pd
import numpy as np

# For visualizing the data
import matplotlib.pyplot as plt
import seaborn as sns


warnings.simplefilter('ignore')


# To use Natural Language Processing
import nltk

# For tokenization
from nltk.tokenize import word_tokenize
nltk.download('punkt')
```

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

Reading the data file

```
#Dataset
df = pd.read_csv("/content/drive/MyDrive/flipitnews-data.csv")
df.head(6)
```

	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...
5	Politics	howard hits back at mongrel jibe michael howar...

2. Exploring the dataset

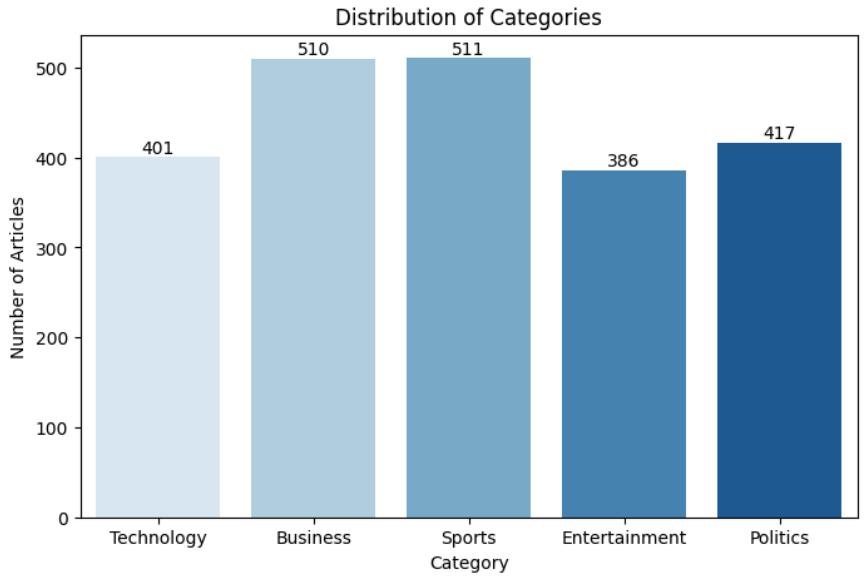
Shape of the dataset

```
print("Shape of the dataset ")
print("Number of rows is {}".format(df.shape[0]))
print("Columns are ", df.columns[0],"and" ,df.columns[1])
```

```
Shape of the dataset
Number of rows is 2225
Columns are  Category and Article
```

News articles per category

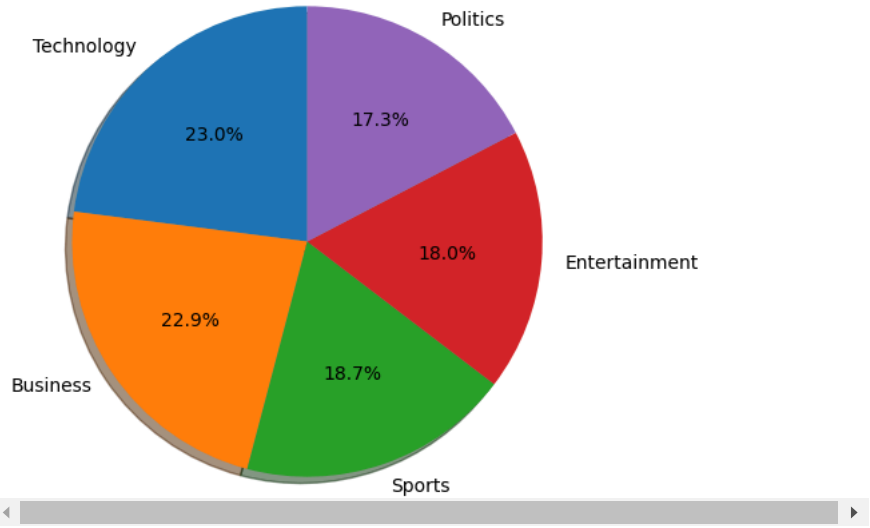
```
plt.figure(figsize=(8, 5))
ax = sns.countplot(x='Category', data=df, hue='Category', palette='Blues', legend=False)
for bar in ax.containers:
    ax.bar_label(bar)
ax.set_title('Distribution of Categories')
ax.set_xlabel('Category')
ax.set_ylabel('Number of Articles')
plt.show()
```




```
# Check tweets distribution
def pie_chart(dataframe):
  # Converting pd object to list of string
  label_types  = df.Category.unique().astype(str)
  # Count tweets for each label
  label_counts = df.Category.value_counts()
  print('Labels in the dataset: ', label_types)
  print(label_counts)
  labels = label_types
  # Sizes for each slide
  sizes = [count for count in label_counts]
  # Declare a figure with a custom size
  fig = plt.figure(figsize=(5, 5))
  # Declare pie chart, where the slices will be ordered and plotted counter-clockwise:
  plt.pie(sizes, labels=labels, autopct='%1.1f%%',
          shadow=True, startangle=90)
  # Equal aspect ratio ensures that pie is drawn as a circle.
  plt.axis('equal')
  # Display the chart
  plt.show()

pie_chart(df)
```

```
Labels in the dataset:  ['Technology' 'Business' 'Sports' 'Entertainment' 'Politics']
Category
Sports           511
Business         510
Politics         417
Technology       401
Entertainment    386
Name: count, dtype: int64
```



3. Processing the Textual Data i.e. the news articles

Removing the non-letters

```
df['Article'][0]
```

```
'tv future in the hands of viewers with home theatre systems  plasma high-definition
tvs  and digital video recorders moving into the living room  the way people watch t
v will be radically different in five years  time.  that is according to an expert p
anel which gathered at the annual consumer electronics show in las vegas to discuss
how these new technologies will impact one of our favourite pastimes. with the us le
ading the trend  programmes and other content will be delivered to viewers via home
networks  through cable  satellite  telecoms companies  and broadband service provid
```

```
import re
def remove_non_letters(text):
  letters_only = re.sub("[^a-zA-Z]", " ", text)
  return letters_only
df['Article'] = df['Article'].apply(remove_non_letters)
```

```
df['Article'][0]
```

```
'tv future in the hands of viewers with home theatre systems  plasma high definition
tvs  and digital video recorders moving into the living room  the way people watch t
v will be radically different in five years  time   that is according to an expert p
anel which gathered at the annual consumer electronics show in las vegas to discuss
how these new technologies will impact one of our favourite pastimes  with the us le
ading the trend  programmes and other content will be delivered to viewers via home
networks  through cable  satellite  telecoms companies  and broadband service provid
```

Tokenizing the text

```
df['Article'] = df['Article'].apply(word_tokenize)
```

Removing stopwords

```
from nltk.corpus import stopwords
nltk.download("stopwords")
stopwords = stopwords.words('english')
df['Article'] = df['Article'].apply(lambda x: [word for word in x if word not in stopwords])
```

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

Lemmatization

```
from nltk.stem import WordNetLemmatizer
nltk.download('wordnet')
lemmatizer = nltk.stem.WordNetLemmatizer()
df['Article'] = df['Article'].apply(lambda x: [lemmatizer.lemmatize(word) for word in x])

df.head()
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
```

	Category	Article
0	Technology	[tv, future, hand, viewer, home, theatre, syst...
1	Business	[worldcom, bos, left, book, alone, former, wor...
2	Sports	[tiger, wary, farrell, gamble, leicester, say,...
3	Sports	[yeading, face, newcastle, fa, cup, premiershi...
4	Entertainment	[ocean, twelve, raid, box, office, ocean, twel...

4. Encoding and Transforming the data

Encoding the target variable

Ordinal encoding uses a single column of integers to represent the classes. Encode 'Category', the target variable column using ordinal encoding technique

```
from sklearn.preprocessing import OrdinalEncoder
encoder = OrdinalEncoder()
df["Category"] = encoder.fit_transform(df[["Category"]])
```

```
df.head()
```

	Category	Article
0	4.0	[tv, future, hand, viewer, home, theatre, syst...
1	0.0	[worldcom, bos, left, book, alone, former, wor...
2	3.0	[tiger, wary, farrell, gamble, leicester, say,...
3	3.0	[yeading, face, newcastle, fa, cup, premiershi...
4	1.0	[ocean, twelve, raid, box, office, ocean, twel...

Bag of Words or TF-IDF

```
# prompt: Create an option for the user to choose between Bag of Words and TF-IDF techniques for vectorizing the data.
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer

bow_or_tfidf = str(input("Choose between 'bow' and 'tfidf' for vectorizing the data: "))
if bow_or_tfidf == 'bow':

  vectorizer = CountVectorizer(max_features=2000, ngram_range=(1, 3))
  X = vectorizer.fit_transform(df['Article'].apply(lambda x: " ".join(x))).toarray()
elif bow_or_tfidf == 'tfidf':

  vectorizer = TfidfVectorizer(max_features=2000, ngram_range=(1, 3))
  X = vectorizer.fit_transform(df['Article'].apply(lambda x: " ".join(x))).toarray()
else:
  print("Invalid input. Please choose between 'bow' and 'tfidf'.")
  exit()
```

```
Choose between 'bow' and 'tfidf' for vectorizing the data: bow
```

```
y = np.array(df['Category'].values)
```

Train-Test Split

```
# To perform train-test split
from sklearn.model_selection import train_test_split
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.25,shuffle=True, stratify=y,random_state=42)
```

```
print("No. of rows in train set is {}.".format(X_train.shape[0]))
print("No. of rows in test set is {}.".format(X_val.shape[0]))
```

```
No. of rows in train set is 1668.
No. of rows in test set is 557.
```

5. Model Training & Evaluation
    - Simple Approach
        1. Naive Bayes

```
# Training the model -
from sklearn.naive_bayes import MultinomialNB
nb = MultinomialNB()
nb.fit(X_train, y_train)
```

```
▾ MultinomialNB
MultinomialNB()
```

```
# Performace Metrics for evaluating the model
from sklearn.metrics import accuracy_score, roc_auc_score, f1_score, precision_score, recall_score
from sklearn.metrics import confusion_matrix, classification_report
```

```
# Calculating the train & test accuracy -
nb_train = accuracy_score(y_train, nb.predict(X_train))
nb_test = accuracy_score(y_val, nb.predict(X_val))

print("Train accuracy :{:.3f}".format(nb_train))
print("Test accuracy :{:.3f}".format(nb_test))
```

```
Train accuracy :0.984
Test accuracy :0.969
```

```
# Making predictions on the test set -
y_pred_nb = nb.predict(X_val)
y_pred_proba_nb = nb.predict_proba(X_val)
```

```
# Computing the ROC AUC score -
roc_auc = roc_auc_score(y_val, y_pred_proba_nb, multi_class='ovr')
print("ROC AUC Score: {:.3f}".format(roc_auc))
```

```
ROC AUC Score: 0.999
```

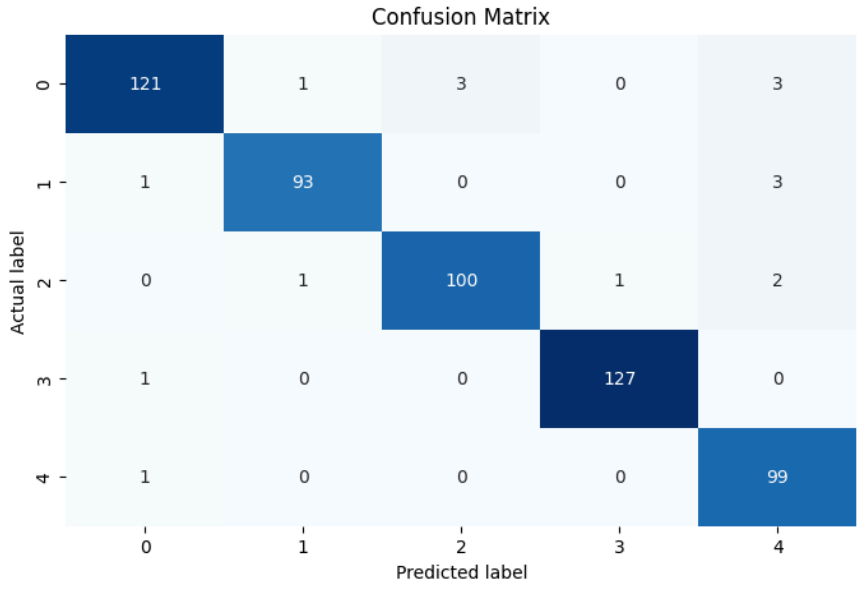
```
# Computing the precision, recall & f1 score -
precision = precision_score(y_val, y_pred_nb, average='weighted')
recall = recall_score(y_val, y_pred_nb, average='weighted')
f1 = f1_score(y_val, y_pred_nb, average='weighted')

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

```
Precision: 0.970
Recall: 0.969
F1 Score: 0.970
```

```
#Plotting the Confusion Matrix -
cm = confusion_matrix(y_val, y_pred_nb)

plt.figure(figsize = (8, 5))
sns.heatmap(cm, annot=True, fmt='d', cbar=False, cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted label')
plt.ylabel('Actual label')
plt.show()
```



```
#printing the Classification Report -
print(classification_report(y_val, y_pred_nb))
```

```
                  precision    recall  f1-score   support

             0.0       0.98      0.95      0.96       128
             1.0       0.98      0.96      0.97        97
             2.0       0.97      0.96      0.97       104
             3.0       0.99      0.99      0.99       128
             4.0       0.93      0.99      0.96       100

        accuracy                           0.97       557
       macro avg       0.97      0.97      0.97       557
    weighted avg       0.97      0.97      0.97       557
```

Functionalized Code (Optional)

```
#Model Training
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


##Model Evaluation
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}".format(test_acc))

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

  # 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("-------------------------")
  return train_acc,test_acc,roc_auc, precision,recall,f1


#Plotting the Confusion Matrix and #printing the Classification Report -
def confusion_matrix_plot(y_val,y_pred_nb):
  cm = confusion_matrix(y_val, y_pred_nb)
  plt.figure(figsize = (8, 5))
  sns.heatmap(cm, annot=True, fmt='d', cbar=False, cmap='Blues')
  plt.title('Confusion Matrix')
  plt.xlabel('Predicted label')
  plt.ylabel('Actual label')
  plt.show()
#printing the Classification Report -
  print(classification_report(y_val, y_pred_nb))
```

1. Decision Tree

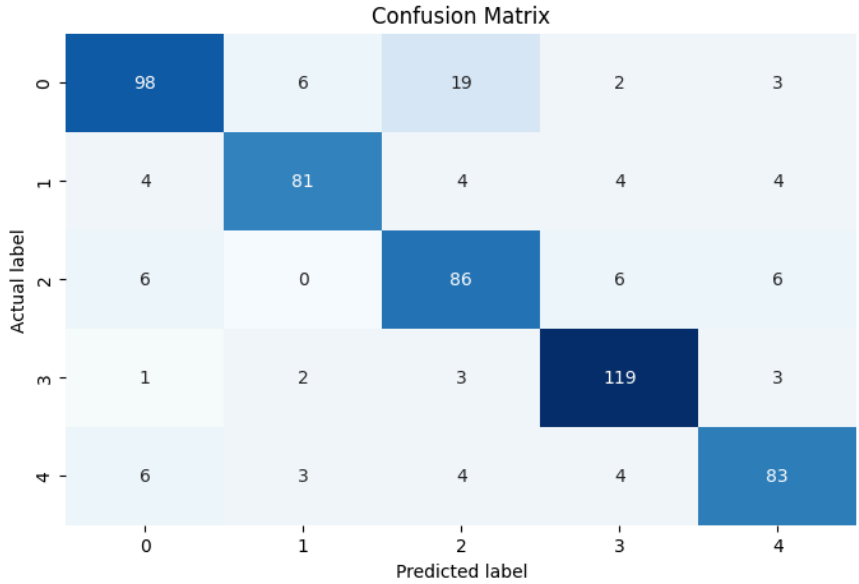
```
# Creating the model object -
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()

# Training the model -
y_pred_dt, y_pred_proba_dt = model_train(dt)

# Evaluatong the model -
dtrain_acc,dtest_acc,droc_auc, dprecision,drecall,df1 = model_eval(dt, y_pred_dt, y_pred_proba_dt)
```

```
-------------------------
Train Accuracy: 1.000
Test Accuracy: 0.838
ROC AUC Score: 0.899
Precision: 0.841
Recall: 0.838
F1 Score: 0.838
-------------------------
```

```
confusion_matrix_plot(y_val,y_pred_dt)
```



	precision	recall	f1-score	support
0.0	0.85	0.77	0.81	128
1.0	0.88	0.84	0.86	97
2.0	0.74	0.83	0.78	104
3.0	0.88	0.93	0.90	128
4.0	0.84	0.83	0.83	100
accuracy			0.84	557
macro avg	0.84	0.84	0.84	557
weighted avg	0.84	0.84	0.84	557

2. Nearest Neighbors

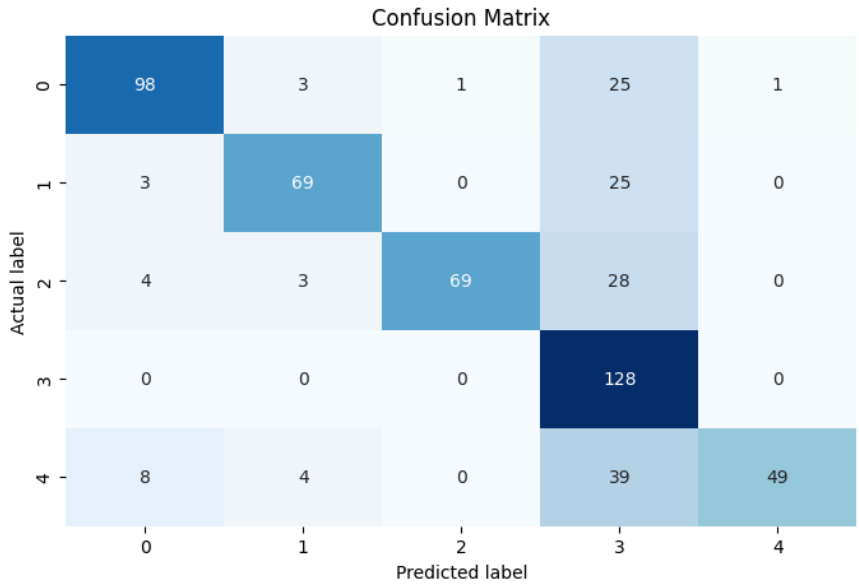
```
# Creating the model object -
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=5)

# Training the model -
y_pred_knn, y_pred_proba_knn = model_train(knn)

# Evaluatong the model -
ntrain_acc,ntest_acc,nroc_auc, nprecision,nrecall,nf1 = model_eval(knn, y_pred_knn, y_pred_proba_knn)
```

```
-------------------------
Train Accuracy: 0.802
Test Accuracy: 0.741
ROC AUC Score: 0.940
Precision: 0.831
Recall: 0.741
F1 Score: 0.747
-------------------------
```

```
confusion_matrix_plot(y_val,y_pred_knn)
```



	precision	recall	f1-score	support
0.0	0.87	0.77	0.81	128
1.0	0.87	0.71	0.78	97
2.0	0.99	0.66	0.79	104
3.0	0.52	1.00	0.69	128
4.0	0.98	0.49	0.65	100
accuracy			0.74	557
macro avg	0.85	0.73	0.75	557
weighted avg	0.83	0.74	0.75	557

3. Random Forest

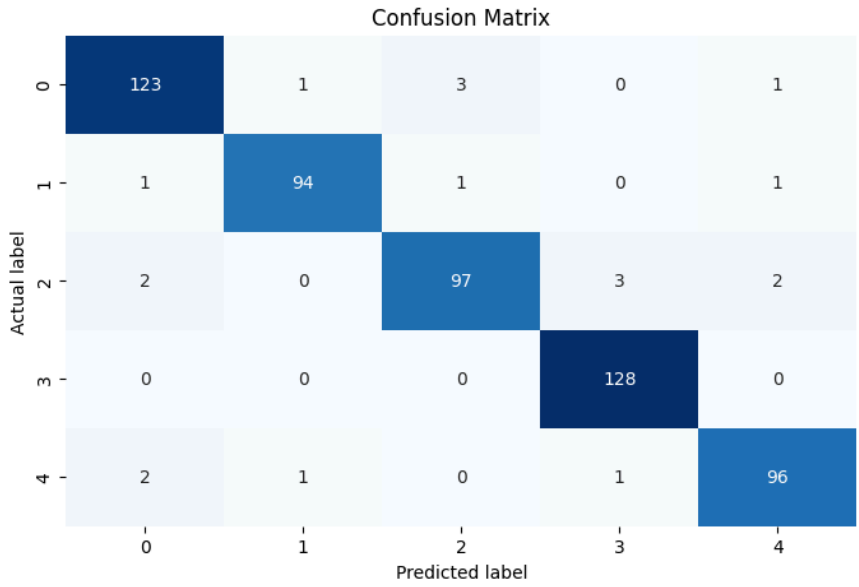
```
from sklearn.ensemble import RandomForestClassifier
# Creating the model object -
rf = RandomForestClassifier()

# Training the model -
y_pred_rf, y_pred_proba_rf = model_train(rf)

# Evaluatong the model -
rtrain_acc,rtest_acc,rroc_auc, rprecision,rrecall,rf1 = model_eval(rf, y_pred_rf, y_pred_proba_rf)
```

```
-------------------------
Train Accuracy: 1.000
Test Accuracy: 0.966
ROC AUC Score: 0.999
Precision: 0.966
Recall: 0.966
F1 Score: 0.966
-------------------------
```

```
confusion_matrix_plot(y_val,y_pred_rf)
```



	precision	recall	f1-score	support
0.0	0.96	0.96	0.96	128
1.0	0.98	0.97	0.97	97
2.0	0.96	0.93	0.95	104
3.0	0.97	1.00	0.98	128
4.0	0.96	0.96	0.96	100
accuracy			0.97	557
macro avg	0.97	0.96	0.97	557
weighted avg	0.97	0.97	0.97	557

```
# prompt: print train_acc ,test_acc,roc_auc, precision,recall,f1 of decision tree, Nearest Neighbors and Random Forest in a tabular form

print("Model \t\t    Train Accuracy  Test Accuracy     ROC AUC Score \t Precision \t Recall \t F1 Score")
print("-----------------------------------------------------------------------------------------------------------------")
print("Naive Bayes \t  |\t {:.3f} \t\t {:.3f} \t\t {:.3f} \t\t {:.3f} \t\t {:.3f} \t\t {:.3f}".format(nb_train,nb_test,roc_auc, precisic
print("Decision Tree \t  |\t {:.3f} \t\t {:.3f} \t\t {:.3f} \t\t {:.3f} \t\t {:.3f} \t\t {:.3f}".format(dtrain_acc,dtest_acc,droc_auc, c
print("Nearest Neighbors |\t {:.3f} \t\t {:.3f} \t\t {:.3f} \t\t {:.3f} \t\t {:.3f} \t\t {:.3f}".format(ntrain_acc,ntest_acc,nroc_auc, r
print("Random Forest \t  |\t {:.3f} \t\t {:.3f} \t\t {:.3f} \t\t {:.3f} \t\t {:.3f} \t\t {:.3f}".format(rtrain_acc,rtest_acc,rroc_auc, r
```

```
Model              Train Accuracy  Test Accuracy     ROC AUC Score      Precision       Recall          F1 Score
-----------------------------------------------------------------------------------------------------------------
Naive Bayes       |      0.984          0.969          0.999          0.970          0.969          0.970
Decision Tree     |      1.000          0.838          0.899          0.841          0.838          0.838
Nearest Neighbors |      0.802          0.741          0.940          0.831          0.741          0.747
Random Forest     |      1.000          0.966          0.999          0.966          0.966          0.966
```

Comparing all the four methods, the random forest method has better parameters (train and test accuracy, ROC AUC score, precision, recall, F1 score)

Model	Train Accuracy	Test Accuracy	ROC AUC Score	Precision	Recall	F1 Score
Naive Bayes	0.984	0.969	0.999	0.970	0.969	0.970
Decision Tree	1.000	0.838	0.899	0.841	0.838	0.838
Nearest Neighbors	0.802	0.741	0.940	0.831	0.741	0.747
Random Forest	1.000	0.966	0.999	0.966	0.966	0.966

Questionnaire:

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

- Solution : 2225

2. Most of the news articles are from the __category.

- Solution: Sports, Business, and Politics
- Sports 511
- Business 510
- Politics 417

- Technology 401
- Entertainment 386

3. Only ____ no. of articles belong to the ‘Technology’ category.
    - Solution : 401
4. What are Stop Words and why should they be removed from the text data?

- Solution : Stop words are common words in a language that carry little meaning on their own in a text analysis context. Examples in English include "the", "is", "a", "an", "in", "on", "to", etc. These words are important for grammatical structure but don't hold much weight for understanding the core content of a text.
- The benefit of removing Stop Words from the text data are focus on the content that truly matters, identify patterns and relationships between the important words, computational tasks become faster.

5. Explain the difference between Stemming and Lemmatization.

- Solution : Stemming and Lemmatization are both techniques used in Natural Language Processing (NLP) to normalize words into their base forms. However, they differ in their approach and outcome.
- Stemming:

  Simpler and faster: Stemming employs a set of rules to chop off prefixes or suffixes from words, aiming to get to a root form.

  Less accurate: This process can be aggressive and sometimes lead to incorrect or unrecognizable words. For instance, stemming "running" might result in "run" which is valid, but stemming "agrees" could lead to "agre" which isn't a real word.

  Doesn't consider context: Stemming focuses solely on the word itself, ignoring its grammatical role (part of speech) in the sentence.
- Lemmatization:

  More complex and slower: Lemmatization involves a deeper understanding of the language. It uses dictionaries and morphological analysis to map a word to its dictionary form, called a lemma.

  More accurate: By considering context and grammatical information, lemmatization ensures the output is a valid word in the language. For example, lemmatizing "running" would return "run" (the base verb form), and "agrees" would become "agree" (the base adjective form).

  Preserves meaning: Since lemmatization aims for the dictionary base form, it ensures the core meaning of the word is retained.

  Choosing between Stemming and Lemmatization:

    - If speed and simplicity are crucial, stemming can be a good initial approach.
    - If