In [1]:
```
# Text Preprocessing
import re  # Regular expressions for text cleaning
import string  # Handling punctuation
import nltk  # Natural Language Toolkit
from nltk.tokenize import word_tokenize, sent_tokenize  # Tokenization
from nltk.corpus import stopwords  # Stopwords removal
from nltk.stem import PorterStemmer, WordNetLemmatizer  # Stemming & L

# Feature Extraction
from sklearn.feature_extraction.text import CountVectorizer, TfidfVect

# Machine Learning Models
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB  # Naïve Bayes for text
from sklearn.linear_model import LogisticRegression  # Logistic Regres
from sklearn.svm import SVC  # Support Vector Classifier
```

In [2]:
```
!pip install nltk
```

```
Requirement already satisfied: nltk in ./anaconda3/lib/python3.10/site-
packages (3.7)
Requirement already satisfied: click in ./anaconda3/lib/python3.10/site
-packages (from nltk) (8.1.8)
Requirement already satisfied: joblib in ./anaconda3/lib/python3.10/sit
e-packages (from nltk) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in ./anaconda3/lib/pytho
n3.10/site-packages (from nltk) (2022.7.9)
Requirement already satisfied: tqdm in ./anaconda3/lib/python3.10/site-
packages (from nltk) (4.67.1)
```

In [3]:
```
# Text Preprocessing
import re  # Regular expressions for text cleaning
import string  # Handling punctuation
import nltk  # Natural Language Toolkit
from nltk.tokenize import word_tokenize, sent_tokenize  # Tokenization
from nltk.corpus import stopwords  # Stopwords removal
from nltk.stem import PorterStemmer, WordNetLemmatizer  # Stemming & L

# Feature Extraction
from sklearn.feature_extraction.text import CountVectorizer, TfidfVect

# Machine Learning Models
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB  # Naïve Bayes for text
from sklearn.linear_model import LogisticRegression  # Logistic Regres
from sklearn.svm import SVC  # Support Vector Classifier
```

In [4]:
```
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('omw-1.4')
```

```
[nltk_data] Downloading package punkt to /Users/ramv/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /Users/ramv/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /Users/ramv/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to /Users/ramv/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
```

Out[4]: True

In [5]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import seaborn as sns
```

In [6]:
```
df_flip= pd.read_csv('/Users/Ramv/Downloads/flipitnews-data.csv')
```

In [7]:
```
df_flip
```

Out[7]:

	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...
...	...	...
2220	Business	cars pull down us retail figures us retail sal...
2221	Politics	kilroy unveils immigration policy ex-chatshow ...
2222	Entertainment	rem announce new glasgow concert us band rem h...
2223	Politics	how political squabbles snowball it s become c...
2224	Sports	souness delight at euro progress boss graeme s...

2225 rows × 2 columns

In [8]:
```
df_flip.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
```

In [9]:
```
df_flip.isnull().sum()
```

Out[9]:
```
Category    0
Article     0
dtype: int64
```

In [10]:
```
df_flip.shape
```

Out[10]:
```
(2225, 2)
```

In [11]:
```
df_flip.duplicated().sum()
```

Out[11]:
```
99
```

In [12]:
```
df_flip = df_flip.drop_duplicates()
```

In [13]:
```
df_flip.duplicated().sum()
```

Out[13]:
```
0
```

In [14]:
```
df_flip.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
```

In [15]:
```
df_flip["Category"].nunique()
```

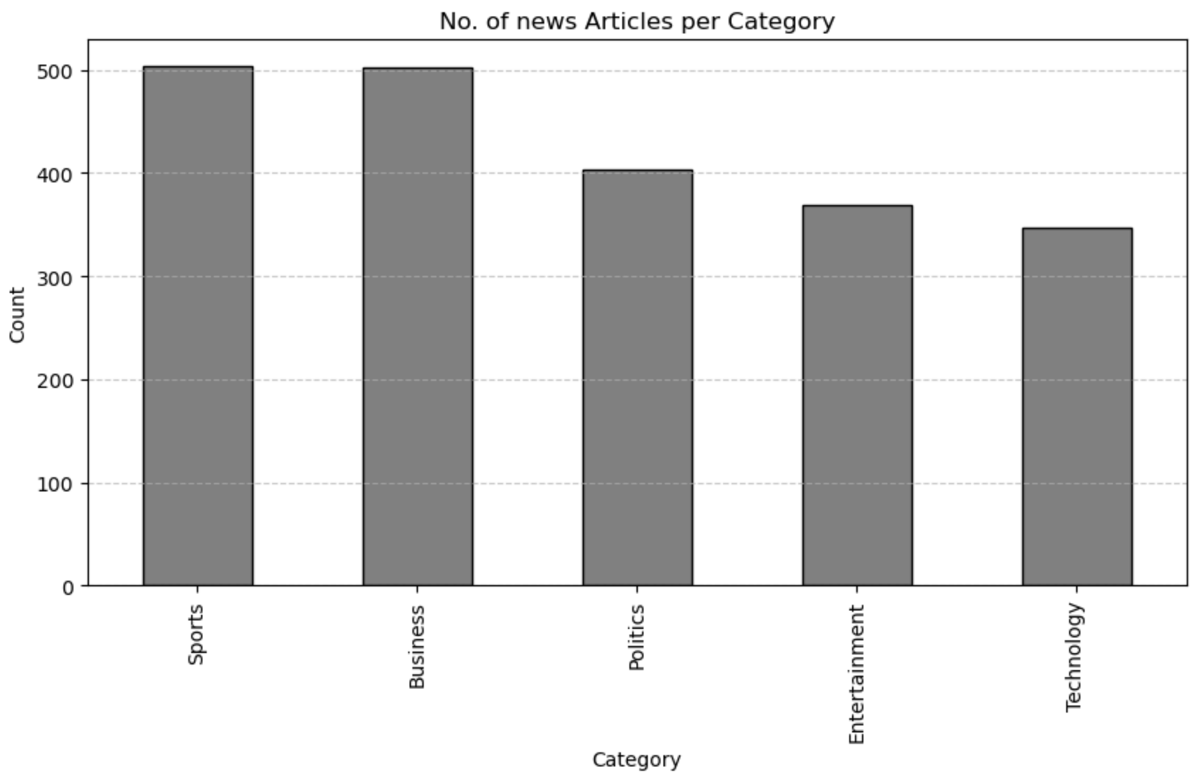
Out[15]:
```
5
```

In [16]:
```
cat_count = df_flip["Category"].value_counts()
cat_count
```

Out[16]:
```
Category
Sports           504
Business         503
Politics         403
Entertainment    369
Technology       347
Name: count, dtype: int64
```

In [17]:
```
def plot_count_category(cat_count):
    plt.figure(figsize =(10,5))
    cat_count.plot(kind = 'bar', color = 'grey', edgecolor = 'black')
    plt.title("No. of news Articles per Category")
    plt.xlabel("Category")
    plt.ylabel("Count")
    plt.grid(axis = "y" , linestyle = "--", alpha =0.7)
    plt.show()
```

In [18]:
```
plot_count_category(cat_count)
```



Removing the non-letters

In [19]:
```
def clean_text(text):
    # Convert to lowercase
    text = text.lower()
    # Remove all non-alphabetic characters except whitespace
    text = re.sub(r'[^a-z\s]', '', text)
    # Remove extra spaces
    text = re.sub(r'\s+', ' ', text)
    return text.strip()
```

In [20]:
```
df_flip["Clean_Article"] = df_flip["Article"].apply(clean_text)
```

```
/var/folders/fx/1km2ndm10xxcmn5xy9fsdksm0000gn/T/ipykernel_28476/315085
4239.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_flip["Clean_Article"] = df_flip["Article"].apply(clean_text)
```

In [21]:
```
df_flip["Clean_Article"]
```

Out[21]:
```
0       tv future in the hands of viewers with home th...
1       worldcom boss left books alone former worldcom...
2       tigers wary of farrell gamble leicester say th...
3       yeading face newcastle in fa cup premiership s...
4       ocean s twelve raids box office ocean s twelve...
                              ...
2220    cars pull down us retail figures us retail sal...
2221    kilroy unveils immigration policy exchatshow h...
2222    rem announce new glasgow concert us band rem h...
2223    how political squabbles snowball it s become c...
2224    souness delight at euro progress boss graeme s...
Name: Clean_Article, Length: 2126, dtype: object
```

Initialize stopwords set and lemmatizer instance

In [22]:
```
stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()
```

In [23]:
```
df_flip
```

Out [23]:

	Category	Article	Clean_Article
0	Technology	tv future in the hands of viewers with home th...	tv future in the hands of viewers with home th...
1	Business	worldcom boss left books alone former worldc...	worldcom boss left books alone former worldcom...
2	Sports	tigers wary of farrell gamble leicester say ...	tigers wary of farrell gamble leicester say th...
3	Sports	yeading face newcastle in fa cup premiership s...	yeading face newcastle in fa cup premiership s...
4	Entertainment	ocean s twelve raids box office ocean s twelve...	ocean s twelve raids box office ocean s twelve...
...	...	...	...
2220	Business	cars pull down us retail figures us retail sal...	cars pull down us retail figures us retail sal...
2221	Politics	kilroy unveils immigration policy ex-chatshow ...	kilroy unveils immigration policy exchatshow h...
2222	Entertainment	rem announce new glasgow concert us band rem h...	rem announce new glasgow concert us band rem h...
2223	Politics	how political squabbles snowball it s become c...	how political squabbles snowball it s become c...
2224	Sports	souness delight at euro progress boss graeme s...	souness delight at euro progress boss graeme s...

2126 rows × 3 columns

Tokenization

In [24]:

```
def process_text(text):
    words = word_tokenize(str(text).lower())
    # Remove stopwords and lemmatize
    words = [word for word in words if word not in stop_words]
    return words
```

In [25]:

```
df_flip["tokenized_text"] = df_flip["Clean_Article"].apply(process_tex
```

```
/var/folders/fx/1km2ndm10xxcmn5xy9fsdksm0000gn/T/ipykernel_28476/562531
737.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_flip["tokenized_text"] = df_flip["Clean_Article"].apply(process_te
xt)
```

In [26]:
```
df_flip.head()
```

Out[26]:

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

In [27]:
```
def lemmatize_words(words):
    return[lemmatizer.lemmatize(word) for word in words]
```

In [28]:
```
df_flip['tokenized_text_1'] = df_flip["tokenized_text"].apply(lemmatiz
df_flip
```

```
/var/folders/fx/1km2ndm10xxcmn5xy9fsdksm0000gn/T/ipykernel_28476/111270
5137.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_flip['tokenized_text_1'] = df_flip["tokenized_text"].apply(lemmati
ze_words)
```

Out[28]:

	Category	Article	Clean_Article	tokenized_text	tokenized_text_1
		tv future in the hands	tv future in the hands of	[tv, future, hands, viewers,	[tv, future, hand

0	Technology	of viewers with home th...	viewers with home th...	home, theatre, sy...	viewer, home theatre, syst..
1	Business	worldcom boss left books alone former worldc...	worldcom boss left books alone former worldcom...	[worldcom, boss, left, books, alone, former, w...	[worldcom, bos left, book, alone former, wor..
2	Sports	tigers wary of farrell gamble leicester say ...	tigers wary of farrell gamble leicester say th...	[tigers, wary, farrell, gamble, leicester, say...	[tiger, wary farrell, gamble leicester, say,..
3	Sports	yeading face newcastle in fa cup premiership s...	yeading face newcastle in fa cup premiership s...	[yeading, face, newcastle, fa, cup, premiershi...	[yeading, face newcastle, fa cup, premiershi..
4	Entertainment	ocean s twelve raids box office ocean s twelve...	ocean s twelve raids box office ocean s twelve...	[ocean, twelve, raids, box, office, ocean, twe...	[ocean, twelve raid, box, office ocean, twel..
...	...	...	...	...	..
2220	Business	cars pull down us retail figures us retail sal...	cars pull down us retail figures us retail sal...	[cars, pull, us, retail, figures, us, retail, ...	[car, pull, u, retail figure, u, retail sale..
2221	Politics	kilroy unveils immigration policy ex-chatshow ...	kilroy unveils immigration policy exchatshow h...	[kilroy, unveils, immigration, policy, exchats...	[kilroy, unveils immigration policy, exchats..
2222	Entertainment	rem announce new glasgow concert us band rem h...	rem announce new glasgow concert us band rem h...	[rem, announce, new, glasgow, concert, us, ban...	[rem, announce new, glasgow concert, u band..
2223	Politics	how political squabbles	how political squabbles	[political, squabbles, snowball,	[political squabble snowball

		snowball it s become c...	snowball it s become c...	become, commo...	become common..
2224	Sports	souness delight at euro progress boss graeme s...	souness delight at euro progress boss graeme s...	[souness, delight, euro, progress, boss, graem...	[souness, delight euro, progress bos, graeme..

2126 rows × 5 columns

In [29]:
```
df_flip['lemmatized_text_join'] = df_flip['tokenized_text_1'].apply(la
df_flip
```

```
/var/folders/fx/1km2ndm10xxcmn5xy9fsdksm0000gn/T/ipykernel_28476/387879
4372.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_flip['lemmatized_text_join'] = df_flip['tokenized_text_1'].apply(l
ambda words: ' '.join(words))
```

Out[29]:

	Category	Article	Clean_Article	tokenized_text	tokenized_text_'
0	Technology	tv future in the hands of viewers with home th...	tv future in the hands of viewers with home th...	[tv, future, hands, viewers, home, theatre, sy...	[tv, future, hand viewer, home theatre, syst..
1	Business	worldcom boss left books alone former worldc...	worldcom boss left books alone former worldcom...	[worldcom, boss, left, books, alone, former, w...	[worldcom, bos left, book, alone former, wor..
2	Sports	tigers wary of farrell gamble leicester say ...	tigers wary of farrell gamble leicester say th...	[tigers, wary, farrell, gamble, leicester, say...	[tiger, wary farrell, gamble leicester, say,..
3	Sports	yeading face newcastle in fa cup premiership s...	yeading face newcastle in fa cup premiership s...	[yeading, face, newcastle, fa, cup, premiershi...	[yeading, face newcastle, fa cup, premiershi..
		ocean s	ocean s		

4	Entertainment	twelve raids box office ocean s twelve...	twelve raids box office ocean s twelve...	[ocean, twelve, raids, box, office, ocean, twe...	[ocean, twelve raid, box, office ocean, twel..
...	...	...	...	...	..
2220	Business	cars pull down us retail figures us retail sal...	cars pull down us retail figures us retail sal...	[cars, pull, us, retail, figures, us, retail, ...	[car, pull, u, retail figure, u, retail sale..
2221	Politics	kilroy unveils immigration policy ex-chatshow ...	kilroy unveils immigration policy exchatshow h...	[kilroy, unveils, immigration, policy, exchats...	[kilroy, unveils immigration policy, exchats..
2222	Entertainment	rem announce new glasgow concert us band rem h...	rem announce new glasgow concert us band rem h...	[rem, announce, new, glasgow, concert, us, ban...	[rem, announce new, glasgow concert, u band..
2223	Politics	how political squabbles snowball it s become c...	how political squabbles snowball it s become c...	[political, squabbles, snowball, become, commo...	[political squabble snowball become common..
2224	Sports	souness delight at euro progress boss graeme s...	souness delight at euro progress boss graeme s...	[souness, delight, euro, progress, boss, graem...	[souness, delight euro, progress bos, graeme..

2126 rows × 6 columns

In [30]:
```
df_flip.drop(columns=['Article', 'Clean_Article', 'tokenized_text', 't
```

```
/var/folders/fx/1km2ndm10xxcmn5xy9fsdksm0000gn/T/ipykernel_28476/956007
674.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_flip.drop(columns=['Article', 'Clean_Article', 'tokenized_text', '
tokenized_text_1'], inplace = True)
```

In [31]:
```
df_flip
```

Out[31]:

	Category	lemmatized_text_join
0	Technology	tv future hand viewer home theatre system plas...
1	Business	worldcom bos left book alone former worldcom b...
2	Sports	tiger wary farrell gamble leicester say rushed...
3	Sports	yeading face newcastle fa cup premiership side...
4	Entertainment	ocean twelve raid box office ocean twelve crim...
...	...	...
2220	Business	car pull u retail figure u retail sale fell ja...
2221	Politics	kilroy unveils immigration policy exchatshow h...
2222	Entertainment	rem announce new glasgow concert u band rem an...
2223	Politics	political squabble snowball become commonplace...
2224	Sports	souness delight euro progress bos graeme soune...

2126 rows × 2 columns

In [32]:
```
df_final_flip = df_flip.copy()
```

In [33]:
```
df_final_flip
```

Out[33]:

	Category	lemmatized_text_join
0	Technology	tv future hand viewer home theatre system plas...
1	Business	worldcom bos left book alone former worldcom b...
2	Sports	tiger wary farrell gamble leicester say rushed...
3	Sports	yeading face newcastle fa cup premiership side...
4	Entertainment	ocean twelve raid box office ocean twelve crim...
...	...	...
2220	Business	car pull u retail figure u retail sale fell ja...
2221	Politics	kilroy unveils immigration policy exchatshow h...
2222	Entertainment	rem announce new glasgow concert u band rem an...
2223	Politics	political squabble snowball become commonplace...
2224	Sports	souness delight euro progress bos graeme soune...

2126 rows × 2 columns

Label Encoding

In [34]:

```
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
df_final_flip['encoded_target'] = label_encoder.fit_transform(df_final

df_final_flip
```

Out[34]:

	Category	lemmatized_text_join	encoded_target
0	Technology	tv future hand viewer home theatre system plas...	4
1	Business	worldcom bos left book alone former worldcom b...	0
2	Sports	tiger wary farrell gamble leicester say rushed...	3
3	Sports	yeading face newcastle fa cup premiership side...	3
4	Entertainment	ocean twelve raid box office ocean twelve crim...	1
...	...	...	...
2220	Business	car pull u retail figure u retail sale fell ja...	0
2221	Politics	kilroy unveils immigration policy exchatshow h...	2
2222	Entertainment	rem announce new glasgow concert u band rem an...	1
2223	Politics	political squabble snowball become commonplace...	2
2224	Sports	souness delight euro progress bos graeme soune...	3

2126 rows × 3 columns

In [35]:

```
df_final_flip.drop(columns="Category", inplace=True)
df_final_flip
```

Out[35]:

	lemmatized_text_join	encoded_target
0	tv future hand viewer home theatre system plas...	4
1	worldcom bos left book alone former worldcom b...	0
2	tiger wary farrell gamble leicester say rushed...	3
3	yeading face newcastle fa cup premiership side...	3
4	ocean twelve raid box office ocean twelve crim...	1
...	...	...
2220	car pull u retail figure u retail sale fell ja...	0
2221	kilroy unveils immigration policy exchatshow h...	2
2222	rem announce new glasgow concert u band rem an...	1
2223	political squabble snowball become commonplace...	2
2224	souness delight euro progress bos graeme soune...	3

2126 rows × 2 columns

Bag of Words(BoW)

In [36]:
```
# Initialize CountVectorizer with common preprocessing options
vectorizer = CountVectorizer()

# Transform lemmatized text into Bag-of-Words matrix
X_bow = vectorizer.fit_transform(df_final_flip['lemmatized_text_join']

# Convert sparse matrix to DataFrame
bow_df = pd.DataFrame(X_bow.toarray(), columns=vectorizer.get_feature_
```

In [37]:
```
X = bow_df
y = df_final_flip['encoded_target']
```

Splitting Data

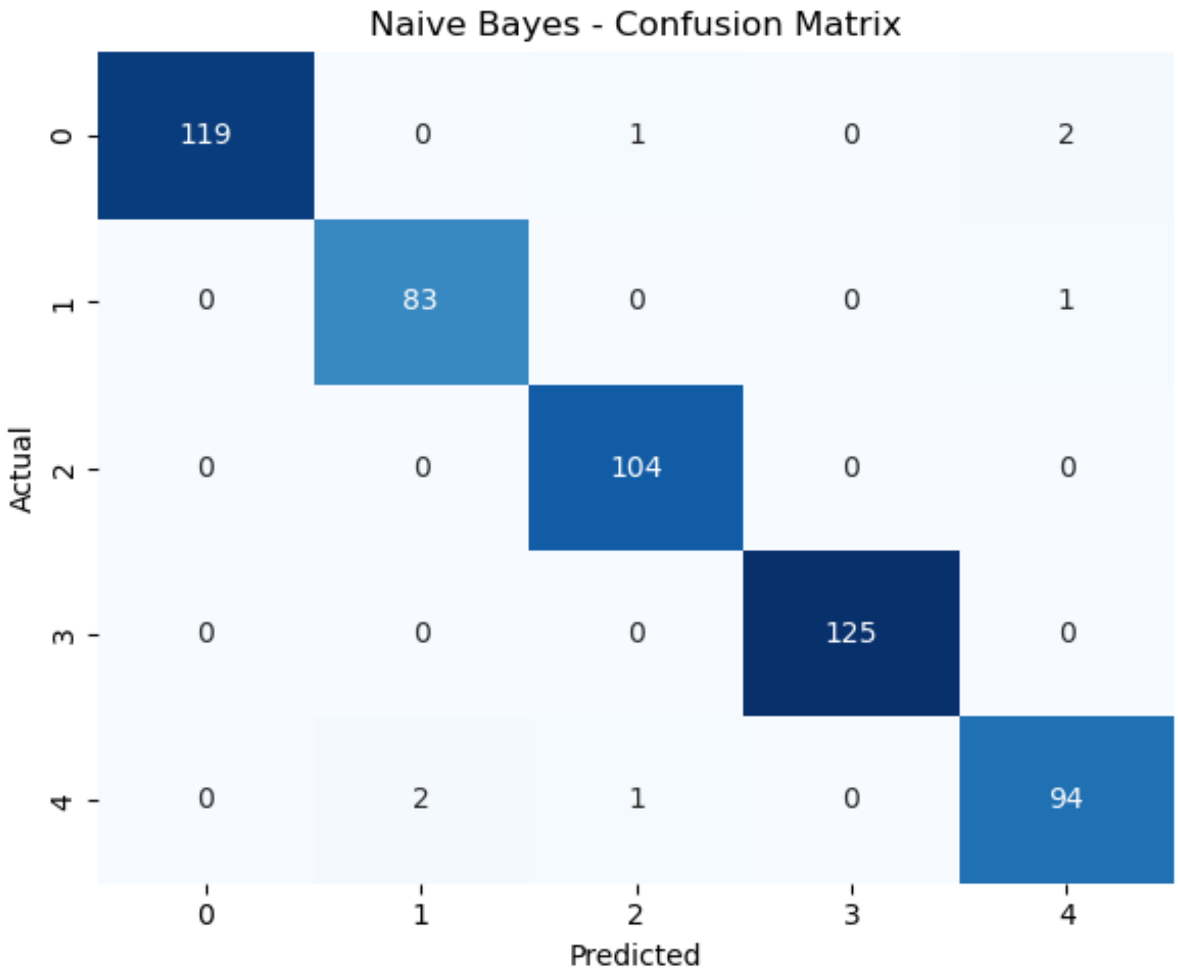
In [38]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
```

Naive Bayes

In [39]:
```
from sklearn.metrics import accuracy_score, classification_report, con

# Initialize the Multinomial Naive Bayes model
nb_model = MultinomialNB()

# Fit the model on the training data
```

```
nb_model.fit(X_train, y_train)

# Predict on the test set
nb_pred = nb_model.predict(X_test)

# Evaluate the model
print("Accuracy:", accuracy_score(y_test, nb_pred))
print("\nClassification Report:\n", classification_report(y_test, nb_p

# Confusion Matrix
cm = confusion_matrix(y_test, nb_pred)
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False,
            xticklabels=nb_model.classes_, yticklabels=nb_model.classe
plt.title("Naive Bayes - Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.tight_layout()
plt.show()
```

```
Accuracy: 0.9868421052631579

Classification Report:
               precision    recall  f1-score   support

           0       1.00      0.98      0.99       122
           1       0.98      0.99      0.98        84
           2       0.98      1.00      0.99       104
           3       1.00      1.00      1.00       125
           4       0.97      0.97      0.97        97

    accuracy                           0.99       532
   macro avg       0.99      0.99      0.99       532
weighted avg       0.99      0.99      0.99       532
```



Strengths:

High performance across all classes with minimal misclassification. Precision and recall above 0.97 for every class indicate balanced performance. Confusion matrix shows low cross-class confusion, especially in harder-to-distinguish classes like 1 and 4.

Weakness:

Small misclassifications between classes 0, 1, and 4, though they are rare and likely due to semantic overlap in feature space.

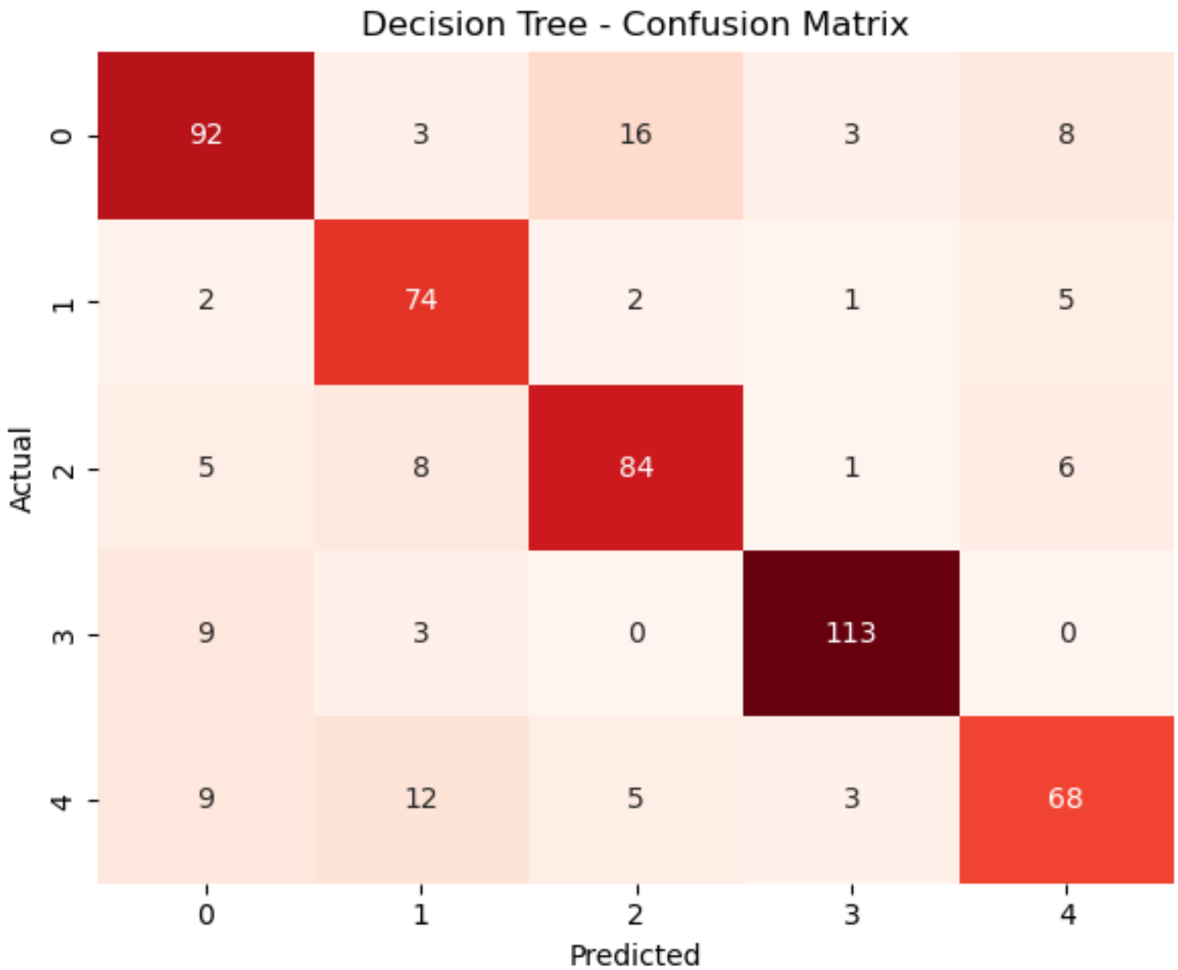
The Naive Bayes model is performing exceptionally well on this dataset, achieving near-perfect accuracy and minimal confusion between classes. It's a strong candidate for deployment if interpretability and training speed are priorities.

Decision Tree Classifier

In [40]:
```
from sklearn.tree import DecisionTreeClassifier

# Initialize the Decision Tree model with common default settings (cus
```

```
dt_model = DecisionTreeClassifier(random_state=42)

# Train the model
dt_model.fit(X_train, y_train)

# Make predictions on the test set
dt_pred = dt_model.predict(X_test)

# Evaluate the model
print("Accuracy:", accuracy_score(y_test, dt_pred))
print("\nClassification Report:\n", classification_report(y_test, dt_p

# Confusion Matrix
cm = confusion_matrix(y_test, dt_pred)
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Reds', cbar=False,
            xticklabels=dt_model.classes_, yticklabels=dt_model.classe
plt.title("Decision Tree - Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.tight_layout()
plt.show()
```

```
Accuracy: 0.8101503759398496

Classification Report:
               precision    recall  f1-score   support

           0       0.79      0.75      0.77       122
           1       0.74      0.88      0.80        84
           2       0.79      0.81      0.80       104
           3       0.93      0.90      0.92       125
           4       0.78      0.70      0.74        97

    accuracy                           0.81       532
   macro avg       0.81      0.81      0.81       532
weighted avg       0.81      0.81      0.81       532
```



Strengths:

Good performance on class 3, with very few false positives. Balanced F1-scores ~0.80 across most classes. Decision Trees offer interpretability and non-linear decision making.

Weakness:

Confusion between class 0 and class 2 is prominent. Class 4 suffers from label spillover into class 1 and 0 — possibly due to feature overlap. Slight overfitting risk with Decision Trees if not pruned or tuned properly.

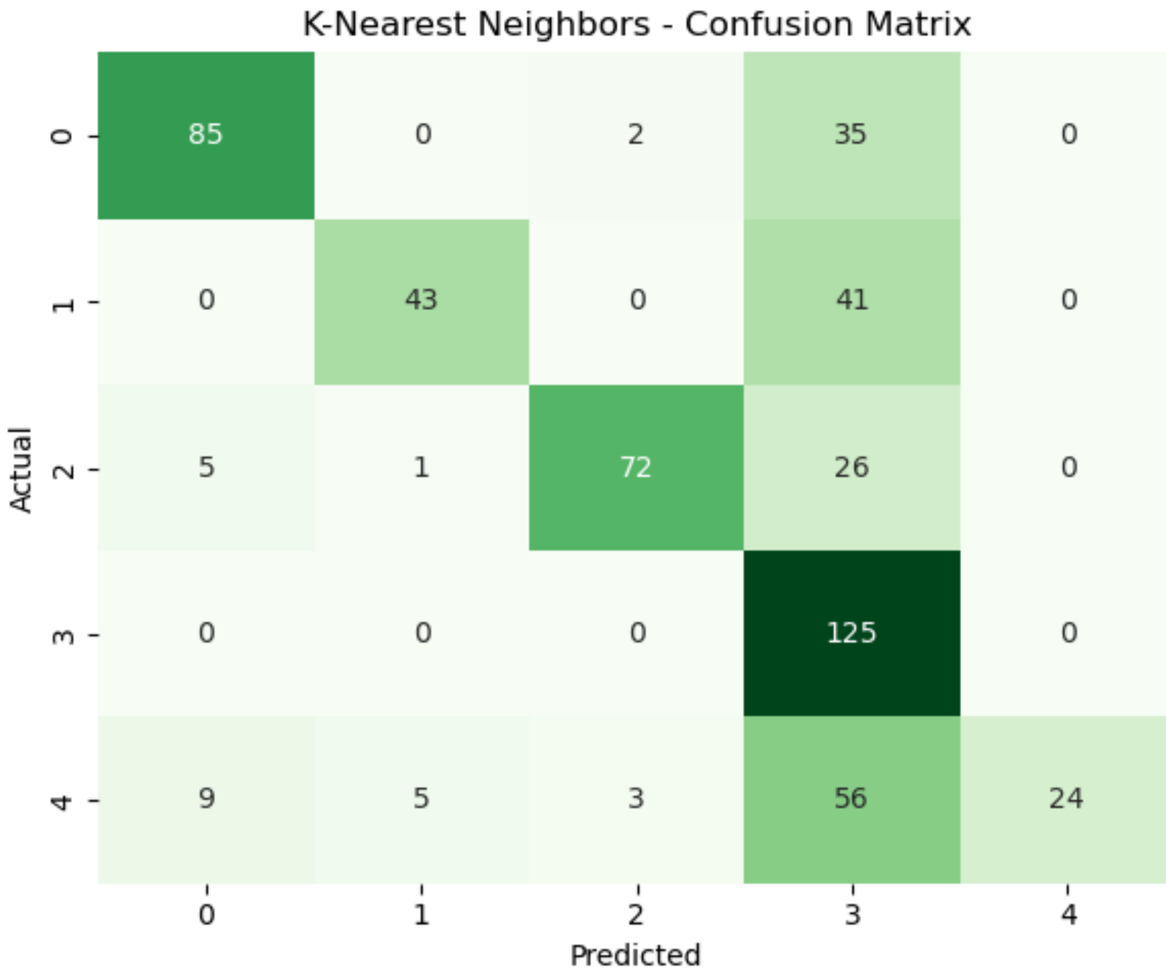
The Decision Tree model performs moderately well, with a solid 81% accuracy and reasonably balanced classification across categories. However, it shows signs of inter-class confusion (especially among classes 0, 2, and 4)

K Nearest Neighbour Classifier

In [41]:
```
from sklearn.neighbors import KNeighborsClassifier

# Initialize the KNN model
```

```
knn_model = KNeighborsClassifier(n_neighbors=5)

# Train the model
knn_model.fit(X_train, y_train)

# Predict on the test set
knn_pred = knn_model.predict(X_test)

# Evaluate the model
print("Accuracy:", accuracy_score(y_test, knn_pred))
print("\nClassification Report:\n", classification_report(y_test, knn_

# Confusion Matrix Visualization
cm = confusion_matrix(y_test, knn_pred)
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Greens', cbar=False,
            xticklabels=knn_model.classes_, yticklabels=knn_model.clas
plt.title("K-Nearest Neighbors - Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.tight_layout()
plt.show()
```

```
Accuracy: 0.6560150375939849

Classification Report:
               precision    recall  f1-score   support

           0       0.86      0.70      0.77       122
           1       0.88      0.51      0.65        84
           2       0.94      0.69      0.80       104
           3       0.44      1.00      0.61       125
           4       1.00      0.25      0.40        97

    accuracy                           0.66       532
   macro avg       0.82      0.63      0.64       532
weighted avg       0.80      0.66      0.65       532
```



Strengths:

Good precision in classes 0, 1, 2, and 4. Perfect recall for class 3, meaning no true class 3 samples were missed.

Weakness:

Heavy overprediction of class 3: Many samples from classes 0, 1, 2, and 4 are misclassified as 3. Very poor recall for class 4 (only 25%), making it unreliable for sensitive applications. Indicates poor class separability in the feature space and that KNN struggles with boundaries between overlapping classes.

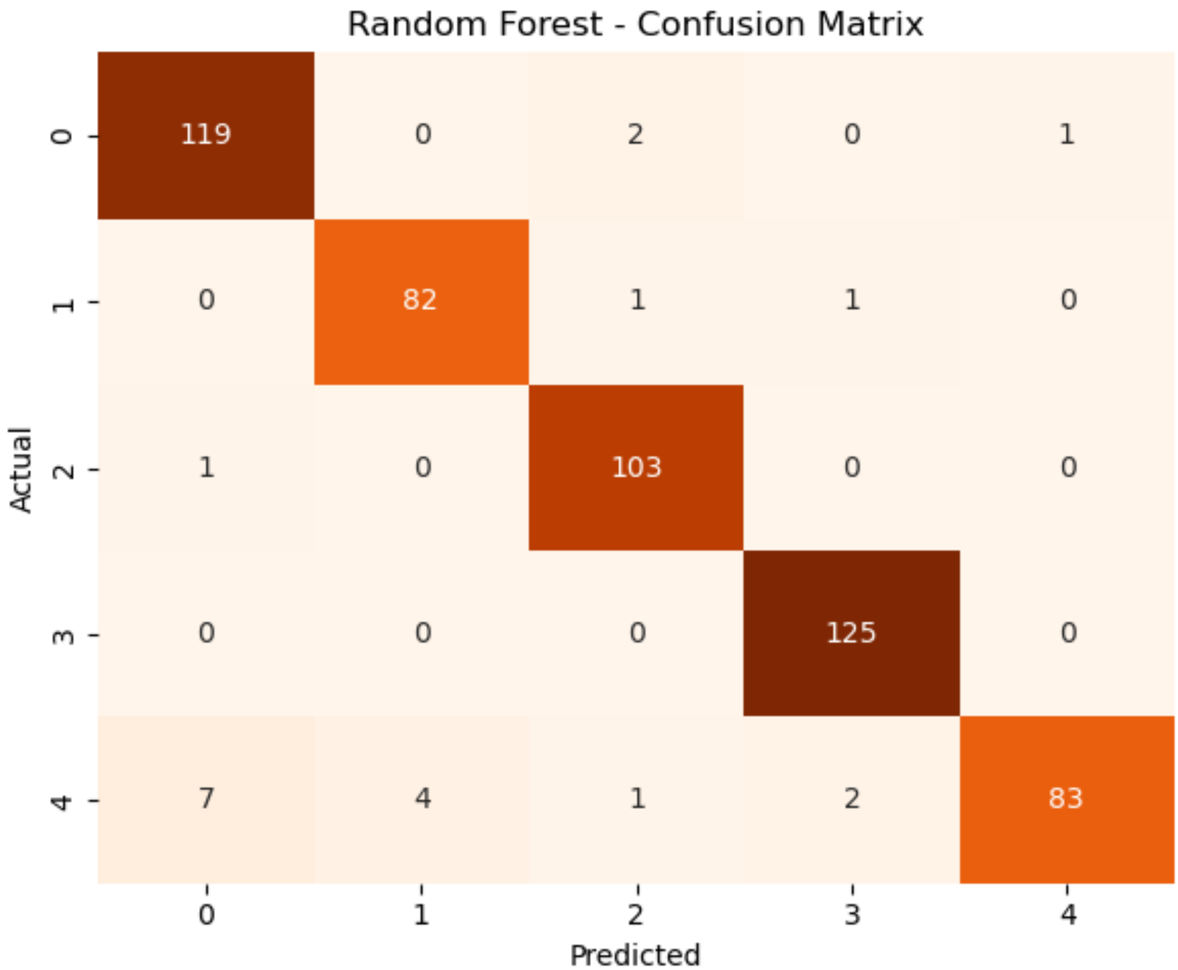
The KNN model demonstrates moderate performance overall but suffers from severe class imbalance in predictions, especially an over-reliance on classifying uncertain samples as class 3.

This results in: Good precision but poor recall in many classes. Overall limited generalization on this dataset.

Random Forest Classifier

In [42]:
```
from sklearn.ensemble import RandomForestClassifier

# Initialize and train the Random Forest model
rf_model = RandomForestClassifier(n_estimators=100, random_state=42, n
rf_model.fit(X_train, y_train)

# Make predictions
rf_pred = rf_model.predict(X_test)

# Evaluate the model
print("Random Forest Accuracy:", accuracy_score(y_test, rf_pred))
print("\nClassification Report:\n", classification_report(y_test, rf_p

# Confusion Matrix Visualization
cm = confusion_matrix(y_test, rf_pred)
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Oranges', cbar=False,
            xticklabels=rf_model.classes_, yticklabels=rf_model.classe
plt.title("Random Forest - Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.tight_layout()
plt.show()
```

```
Random Forest Accuracy: 0.9624060150375939

Classification Report:
               precision    recall  f1-score   support

           0       0.94      0.98      0.96       122
           1       0.95      0.98      0.96        84
           2       0.96      0.99      0.98       104
           3       0.98      1.00      0.99       125
           4       0.99      0.86      0.92        97

    accuracy                           0.96       532
   macro avg       0.96      0.96      0.96       532
weighted avg       0.96      0.96      0.96       532
```



Strengths:

High precision & recall in all classes, especially class 3, which is perfectly predicted. Very low misclassification rates. Balanced performance — macro and weighted F1-scores both at 0.96. Random Forest handles non-linearity and feature interaction well.

Weakness:

Class 4 shows slightly lower recall (0.86), with confusion spread across multiple classes. This might be due to overlapping features or imbalanced training samples.

The Random Forest Classifier demonstrates robust, high-performing classification across all target classes. With an overall accuracy above 96% and consistently high precision-recall, it is well-suited for deployment.

TF-IDF Vectorization

In [43]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
# Initialize TF-IDF Vectorizer with enhanced options
tfidf_vectorizer = TfidfVectorizer()

# Ensure the input is string and not NaN
texts = df_final_flip['lemmatized_text_join'].fillna('').astype(str)

# Transform text into TF-IDF feature matrix
X_tfidf = tfidf_vectorizer.fit_transform(texts)

# Convert to DataFrame for inspection or modeling
tfidf_df = pd.DataFrame(X_tfidf.toarray(), columns=tfidf_vectorizer.ge
```

In [44]:

```
tfidf_df
```

Out[44]:

	aa	aaa	aac	aadc	aaliyah	aaltra	aamir	aan	aara	aarhus	...	zoom
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...
2121	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0
2122	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0
2123	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0
2124	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0
2125	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0

2126 rows × 27175 columns

In [45]:

```
from sklearn.model_selection import train_test_split

# Define features (BoW or TF-IDF) and target
X = tfidf_df  # or use X_bow
y = df_final_flip['encoded_target']
```

In [46]:

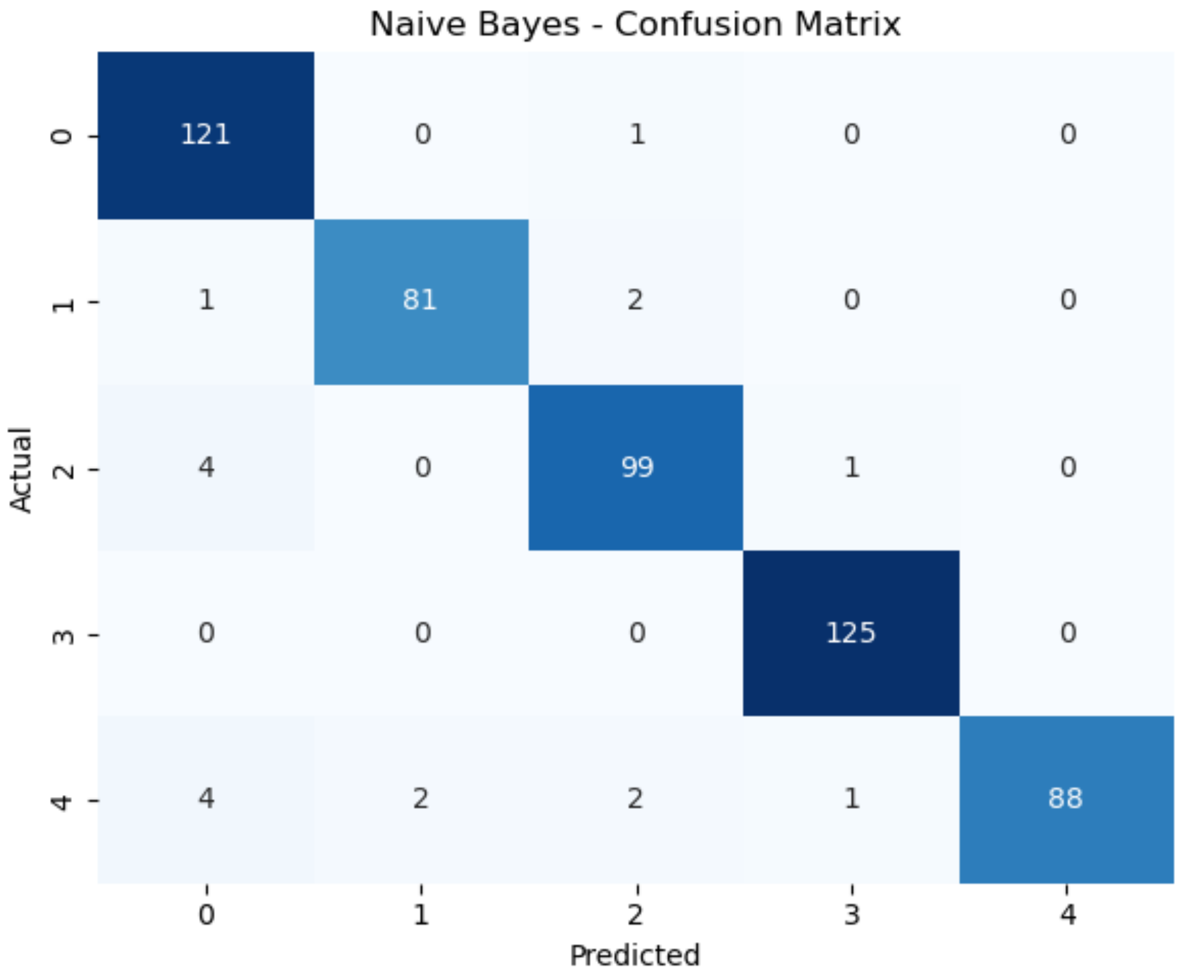
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
```

In [47]:

```
# Initialize the Multinomial Naive Bayes model
nb_model_1 = MultinomialNB()

# Fit the model on the training data
nb_model_1.fit(X_train, y_train)
```

```
# Predict on the test set
nb_pred = nb_model_1.predict(X_test)

# Evaluate the model
print("Accuracy:", accuracy_score(y_test, nb_pred))
print("\nClassification Report:\n", classification_report(y_test, nb_p

# Confusion Matrix
cm = confusion_matrix(y_test, nb_pred)
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False,
            xticklabels=nb_model_1.classes_, yticklabels=nb_model_1.cl
plt.title("Naive Bayes - Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.tight_layout()
plt.show()
```

```
Accuracy: 0.9661654135338346

Classification Report:
               precision    recall  f1-score   support

           0       0.93      0.99      0.96       122
           1       0.98      0.96      0.97        84
           2       0.95      0.95      0.95       104
           3       0.98      1.00      0.99       125
           4       1.00      0.91      0.95        97

    accuracy                           0.97       532
   macro avg       0.97      0.96      0.97       532
weighted avg       0.97      0.97      0.97       532
```



Strengths:

Perfect recall on class 3 and precision on class 4 Highly balanced performance, making the model reliable across all categories Very low false positives and false negatives

Observations:

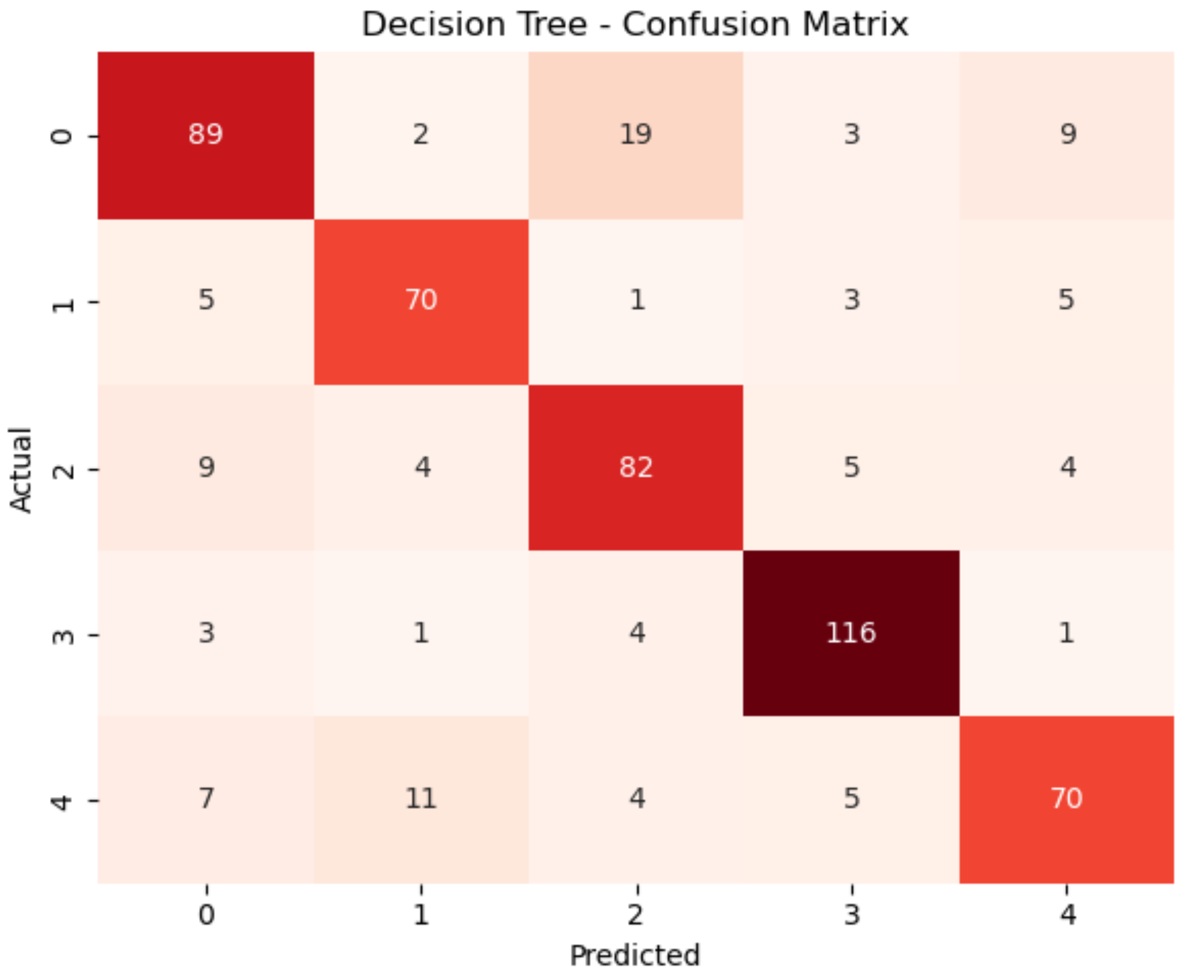
Slight confusion among classes 0, 2, and 4, suggesting some overlap in feature representations Class 4 shows more scattered misclassifications than others, though performance is still strong

The Naive Bayes classifier delivers excellent overall accuracy (96.7%) and maintains balanced, high-quality predictions across all categories.

In [48]:
```
from sklearn.tree import DecisionTreeClassifier

# Initialize the Decision Tree model with common default settings (cus
dt_model_1 = DecisionTreeClassifier(random_state=42)

# Train the model
dt_model_1.fit(X_train, y_train)
```

```
# Make predictions on the test set
dt_pred_1 = dt_model_1.predict(X_test)

# Evaluate the model
print("Accuracy:", accuracy_score(y_test, dt_pred_1))
print("\nClassification Report:\n", classification_report(y_test, dt_p

# Confusion Matrix
cm = confusion_matrix(y_test, dt_pred_1)
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Reds', cbar=False,
            xticklabels=dt_model.classes_, yticklabels=dt_model.classe
plt.title("Decision Tree - Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.tight_layout()
plt.show()
```

```
Accuracy: 0.8026315789473685

Classification Report:
               precision    recall  f1-score   support

           0       0.79      0.73      0.76       122
           1       0.80      0.83      0.81        84
           2       0.75      0.79      0.77       104
           3       0.88      0.93      0.90       125
           4       0.79      0.72      0.75        97

    accuracy                           0.80       532
   macro avg       0.80      0.80      0.80       532
weighted avg       0.80      0.80      0.80       532
```



Strengths:

Strong recall and F1-score for class 3 (0.93, 0.90) Reasonably balanced macro and weighted averages, showing no severe bias toward a single class

Weaknesses:

Misclassifications for class 0 → class 2, and class 4 → class 1 Precision and recall in classes 0, 2, and 4 are relatively lower due to class overlap or noisy features Performance significantly lags behind other models like Random Forest or Naive Bayes on the same dataset

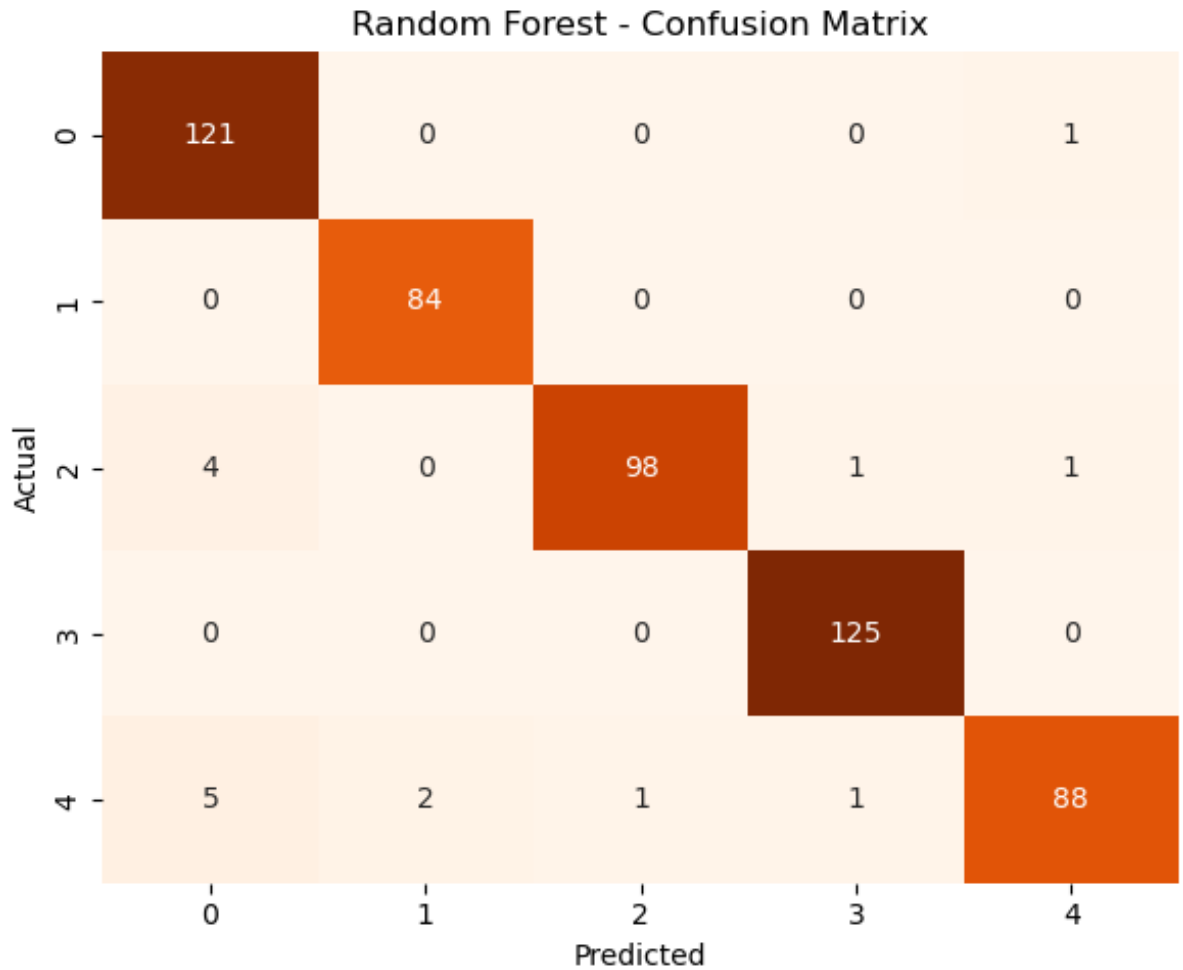
The Decision Tree model provides a baseline performance with 80% accuracy, but struggles with inter-class confusion, especially in borderline categories like 0, 2, and 4.

In [52]:

```
from sklearn.ensemble import RandomForestClassifier

# Initialize and train the Random Forest model
rf_model_1 = RandomForestClassifier(n_estimators=100, random_state=42,
rf_model_1.fit(X_train, y_train)
```

```
# Make predictions
rf_pred = rf_model_1.predict(X_test)

# Evaluate the model
print("Random Forest Accuracy:", accuracy_score(y_test, rf_pred))
print("\nClassification Report:\n", classification_report(y_test, rf_p

# Confusion Matrix Visualization
cm = confusion_matrix(y_test, rf_pred)
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Oranges', cbar=False,
            xticklabels=rf_model_1.classes_, yticklabels=rf_model_1.cl
plt.title("Random Forest - Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.tight_layout()
plt.show()
```

```
Random Forest Accuracy: 0.9699248120300752

Classification Report:
               precision    recall  f1-score   support

           0       0.93      0.99      0.96       122
           1       0.98      1.00      0.99        84
           2       0.99      0.94      0.97       104
           3       0.98      1.00      0.99       125
           4       0.98      0.91      0.94        97

    accuracy                           0.97       532
   macro avg       0.97      0.97      0.97       532
weighted avg       0.97      0.97      0.97       532
```



Strengths:

Class 1 and Class 3: perfect recall, nearly perfect precision → excellent model learning Balanced F1-scores across all classes indicate good generalization Very low overall confusion.

Weaknesses: Slight misclassification in class 4 (9/97), though this is relatively minor Class 2: 6/104 total misclassifications; still acceptable, but slightly more confused than class 1 and 3.

This Random Forest model is performing at a very high level, with 97% accuracy and macro/weighted F1-scores of 0.97. It would be suitable for production use, especially if interpretability isn't the primary concern. The model handles imbalanced and multiclass classification well, and only a few borderline instances are being misclassified.

In [55]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV

# Define hyperparameters to tune
param_grid = {
```

```
    'n_estimators': [50, 100, 200],        # Number of trees in the fo
    'max_depth': [None, 10, 20],           # Maximum depth of each tre
    'min_samples_split': [2, 5, 10],       # Minimum samples required
    'min_samples_leaf': [1, 2, 4],         # Minimum samples required
}

# Initialize the Random Forest Classifier
rf_model = RandomForestClassifier(random_state=42, n_jobs=-1)

# Initialize GridSearchCV
grid_search = GridSearchCV(
    estimator=rf_model,
    param_grid=param_grid,
    cv=3,                        # 3-fold cross-validation
    scoring='accuracy',          # Optimize for accuracy
    n_jobs=-1,                   # Use all CPU cores
    verbose=2                    # Show progress
)

# Fit the grid search to training data
grid_search.fit(X_train, y_train)

# Get best parameters and best score
print("Best Parameters:", grid_search.best_params_)
print("Best Cross-Validation Accuracy:", grid_search.best_score_)
```

```
Fitting 3 folds for each of 81 candidates, totalling 243 fits
Best Parameters: {'max_depth': None, 'min_samples_leaf': 2, 'min_sample
s_split': 10, 'n_estimators': 200}
Best Cross-Validation Accuracy: 0.9598454705501984
```

Summary:

The FlipItNews NLP project focuses on building a multi-class text classification pipeline using news articles.

The key stages include:

Data Loading and Cleaning-

Initial dataset: 2,225 rows After removing 99 duplicates, final cleaned dataset: 2,126 articles Each article is labeled under one of 5 categories: Sports (504 articles) Business (503) Politics (403) Entertainment (369) Technology (347)

Text Preprocessing-

Converted to lowercase, removed punctuation/special characters. Removed stopwords (common words that don’t carry meaning like “is”, “the”). Performed tokenization (splitting sentences into words). Applied lemmatization to reduce words to base forms (e.g., “running” → “run”).

Feature Engineering-

Text data was transformed using both:

Bag of Words (BoW) TF-IDF (Term Frequency-Inverse Document Frequency)

Model Training and Evaluation-

Using both BoW and TF-IDF, models were trained and compared:

Model	Accuracy	Best F1 Performance	Notes
Naive Bayes (BoW)	**98.68%**	F1-scores > 0.97 for all classes	Best performing
Decision Tree (BoW)	81.02%	High on class 3, but confusion between class 0 & 2	
K-Nearest Neighbors (BoW)	65.60%	Overpredicts class 3, low recall for class 4	
Naive Bayes (TF-IDF)	96.62%	Balanced across all classes	
Decision Tree (TF-IDF)	80.26%	Similar to BoW, still underperforms	
Random Forest (TF-IDF)	**96.99%**	Very close to Naive Bayes, strong across all	

Recommendations & Insights:

Because it is multi-class classification, both precision (minimizing false positives) and recall (minimizing false negatives) are essential to ensure balanced accuracy across all categories. Especially important in news classification, where both coverage and correct categorization matter.

Naive Bayes (BoW) is ideal for production — fast, lightweight, and highly accurate. Random Forest (TF-IDF) is a strong backup if more interpretability and robustness are needed. Consider hyperparameter tuning and cross-validation for even better results. Investigate class 4 (Technology) further — it had slightly more confusion in all models.

Answers to the Questions:

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

- 2,126 articles (after removing duplicates).

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

- Sports (504 articles).

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

- 347 articles.

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

- Stop Words are common words (e.g., "is", "and", "the") that do not add significant meaning to text analysis.

Removing them:

Reduces noise Improves model focus on meaningful words Speeds up training

5. Explain the difference between Stemming and Lemmatization.

Stemming	Lemmatization
Cuts suffixes (e.g., "running" → "runn")	Converts to base form using vocabulary (e.g., "running" → "run")
Fast, but can distort words	Slower, but more accurate
Rule-based	Dictionary-based
May not return real words	Returns valid root words

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

   TF-IDF is more efficient and informative than BoW because: It weighs words based on importance, not just frequency. Reduces the influence of common words across documents.

7. What's the shape of train & test datasets after performing a 75:25 split?

Train set: 1,594 samples Test set: 532 samples (train_test_split(test_size=0.25) on 2,126 rows)

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

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

Correct Answer: c. Naive Bayes Accuracy: 98.68% (BoW) Highly balanced performance across all categories

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

Because it is multi-class classification, both precision (minimizing false positives) and recall (minimizing false negatives) are essential to ensure balanced accuracy across all categories.

Especially important in news classification, where both coverage and correct categorization matter.

In [ ]:

```

```