In [4]:
```
import pandas as pd
import numpy as np
import re
import string
from sklearn.model_selection import train_test_split

file_path = 'data_news.csv'
df = pd.read_csv(file_path)

df['text'] = df['headline'].fillna('') + ' ' + df['short_description'].fillna(''

df = df[['category', 'text']]

df.dropna(subset=['category', 'text'], inplace=True)
df = df[df['text'].str.strip() != '']


def clean_text(text):
    text = text.lower()
    text = re.sub(r'\[.*?\]', '', text)
    text = re.sub(r'http\S+|www\S+|https\S+', '', text, flags=re.MULTILINE)
    text = re.sub(r'\@w+|\#','', text)
    text = re.sub(r'[%s]' % re.escape(string.punctuation), '', text)
    text = re.sub(r'\n', ' ', text)
    text = re.sub(r'\w*\d\w*', '', text)
    return text

df['clean_text'] = df['text'].apply(clean_text)

print("✅ Cleaned and preprocessed data (first 5 rows):")
df.head()
```

✅ Cleaned and preprocessed data (first 5 rows):

Out[4]:

	category	text	clean_text
0	WELLNESS	143 Miles in 35 Days: Lessons Learned Resting ...	miles in days lessons learned resting is par...
1	WELLNESS	Talking to Yourself: Crazy or Crazy Helpful? T...	talking to yourself crazy or crazy helpful thi...
2	WELLNESS	Crenezumab: Trial Will Gauge Whether Alzheimer...	crenezumab trial will gauge whether alzheimers...
3	WELLNESS	Oh, What a Difference She Made If you want to ...	oh what a difference she made if you want to b...
4	WELLNESS	Green Superfoods First, the bad news: Soda bre...	green superfoods first the bad news soda bread...

In [20]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
import pandas as pd

tfidf = TfidfVectorizer(max_features=500)
tfidf_features = tfidf.fit_transform(df['clean_text'])
tfidf_df = pd.DataFrame(tfidf_features.toarray(), columns=tfidf.get_feature_name

df['word_count'] = df['clean_text'].apply(lambda x: len(x.split()))
```

```
df['char_count'] = df['clean_text'].apply(len)

df['avg_word_len'] = df['char_count'] / df['word_count']

final_df = pd.concat([df[['category', 'clean_text', 'word_count', 'char_count',

print(" Final feature-engineered dataframe:")
print(final_df.head())
```

```
 Final feature-engineered dataframe:
   category                                         clean_text  word_count  \
0  WELLNESS  miles in days lessons learned resting is part ...          54
1  WELLNESS  talking to yourself crazy or crazy helpful thi...          46
2  WELLNESS  crenezumab trial will gauge whether alzheimers...          37
3  WELLNESS  oh what a difference she made if you want to b...          28
4  WELLNESS  green superfoods first the bad news soda bread...          26

   char_count  avg_word_len  about  according  across  actually  after  ...  \
0         299      5.537037    0.0        0.0     0.0       0.0    0.0  ...
1         255      5.543478    0.0        0.0     0.0       0.0    0.0  ...
2         202      5.459459    0.0        0.0     0.0       0.0    0.0  ...
3         131      4.678571    0.0        0.0     0.0       0.0    0.0  ...
4         135      5.192308    0.0        0.0     0.0       0.0    0.0  ...

   years  yet  york       you  youll  young      your  youre  yourself  youve
0    0.0  0.0   0.0  0.000000    0.0    0.0  0.000000    0.0   0.00000    0.0
1    0.0  0.0   0.0  0.157298    0.0    0.0  0.179213    0.0   0.78322    0.0
2    0.0  0.0   0.0  0.000000    0.0    0.0  0.000000    0.0   0.00000    0.0
3    0.0  0.0   0.0  0.258583    0.0    0.0  0.000000    0.0   0.00000    0.0
4    0.0  0.0   0.0  0.000000    0.0    0.0  0.000000    0.0   0.00000    0.0

[5 rows x 505 columns]
```

In [22]:
```
import pandas as pd
import re
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import LinearSVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

df = pd.read_csv("data_news.csv")

df['text'] = df['headline'].astype(str) + ' ' + df['short_description'].astype(s

df = df[['category', 'text']].dropna()

def clean_text(text):
    text = text.lower()
    text = re.sub(r'[^a-zA-Z\s]', '', text)
    text = re.sub(r'\s+', ' ', text).strip()
    return text

df['clean_text'] = df['text'].apply(clean_text)

le = LabelEncoder()
```

```
df['label'] = le.fit_transform(df['category'])

tfidf = TfidfVectorizer(max_features=5000)
X = tfidf.fit_transform(df['clean_text'])
y = df['label']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_

lr = LogisticRegression(max_iter=1000)
lr.fit(X_train, y_train)
pred_lr = lr.predict(X_test)
print("📘 Logistic Regression Accuracy:", round(accuracy_score(y_test, pred_lr)

nb = MultinomialNB()
nb.fit(X_train, y_train)
pred_nb = nb.predict(X_test)
print("📗 Naive Bayes Accuracy:", round(accuracy_score(y_test, pred_nb), 4))

svm = LinearSVC()
svm.fit(X_train, y_train)
pred_svm = svm.predict(X_test)
print("📕 SVM Accuracy:", round(accuracy_score(y_test, pred_svm), 4))

rf = RandomForestClassifier()
rf.fit(X_train, y_train)
pred_rf = rf.predict(X_test)
print("📙 Random Forest Accuracy:", round(accuracy_score(y_test, pred_rf), 4))
```

```
📘 Logistic Regression Accuracy: 0.7905
📗 Naive Bayes Accuracy: 0.7702
📕 SVM Accuracy: 0.7932
📙 Random Forest Accuracy: 0.7031
```

In [18]:

```
%whos
```

```
Variable                 Type                      Data/Info
------------------------------------------------------------
LabelEncoder             type                      <class 'sklearn.preproces<...>
ing._label.LabelEncoder'>
LinearSVC                type                      <class 'sklearn.svm._classes.L
inearSVC'>
LogisticRegression       type                      <class 'sklearn.linear_mo<...>
stic.LogisticRegression'>
MultinomialNB            ABCMeta                   <class 'sklearn.naive_bayes.Mu
ltinomialNB'>
NamespaceMagics          MetaHasTraits             <class 'IPython.core.magi<...>
mespace.NamespaceMagics'>
RandomForestClassifier   ABCMeta                   <class 'sklearn.ensemble.<...
>.RandomForestClassifier'>
SVC                      ABCMeta                   <class 'sklearn.svm._classes.S
VC'>
StandardScaler           type                      <class 'sklearn.preproces<...>
ng._data.StandardScaler'>
TfidfVectorizer          type                      <class 'sklearn.feature_e<...>
on.text.TfidfVectorizer'>
WordNetLemmatizer        type                      <class 'nltk.stem.wordnet.Word
NetLemmatizer'>
X                        csr_matrix                  (0, 2809)  0.15176461712<...
> 4029) 0.6760012758179277
X_test                   csr_matrix                  (0, 2307)  0.07256016422<...
> 2076) 0.1982033946938762
X_train                  csr_matrix                  (0, 3053)  0.05686856527<...
> 3420) 0.4042425188854908
acc                      float                     0.6088
accuracy_score           function                  <function accuracy_score at 0x
000002789623ADE0>
classification_report    function                  <function classification_<...>
rt at 0x00000278962542C0>
clean_text               function                  <function clean_text at 0x0000
0278E57E37E0>
confusion_matrix         function                  <function confusion_matrix at
0x000002789623AF20>
df                       DataFrame                        category         <...>
n[50000 rows x 4 columns]
file_path                str                       data_news.csv
final_df                 DataFrame                        category         <...>
50000 rows x 505 columns]
get_ipython              function                  <function get_ipython at 0x000
00278832600E0>
json                     module                    <module 'json' from 'C:\\<...>
\Lib\\json\\__init__.py'>
le                       LabelEncoder              LabelEncoder()
lr                       LogisticRegression        LogisticRegression(max_iter=10
00)
lr_model                 LogisticRegression        LogisticRegression(max_iter=10
00)
lr_preds                 ndarray                   10000: 10000 elems, type `obje
ct`, 80000 bytes
model                    RandomForestClassifier    RandomForestClassifier()
models                   dict                      n=4
name                     str                       Random Forest
nb                       MultinomialNB             MultinomialNB()
nb_model                 MultinomialNB             MultinomialNB()
nb_preds                 ndarray                   10000: 10000 elems, type `<U14
`, 560000 bytes (546.875 kb)
```

```
nltk                     module                   <module 'nltk' from 'C:\\<...>
ages\\nltk\\__init__.py'>
np                       module                   <module 'numpy' from 'C:\<...>
ges\\numpy\\__init__.py'>
pd                       module                   <module 'pandas' from 'C:<...>
es\\pandas\\__init__.py'>
plt                      module                   <module 'matplotlib.pyplo<...>
\\matplotlib\\pyplot.py'>
pred_lr                  ndarray                  10000: 10000 elems, type `int3
2`, 40000 bytes
pred_nb                  ndarray                  10000: 10000 elems, type `int3
2`, 40000 bytes
pred_rf                  ndarray                  10000: 10000 elems, type `int3
2`, 40000 bytes
pred_svm                 ndarray                  10000: 10000 elems, type `int3
2`, 40000 bytes
re                       module                   <module 're' from 'C:\\Us<...>
3\\Lib\\re\\__init__.py'>
rf                       RandomForestClassifier   RandomForestClassifier()
rf_model                 RandomForestClassifier   RandomForestClassifier()
rf_preds                 ndarray                  10000: 10000 elems, type `obje
ct`, 80000 bytes
sns                      module                   <module 'seaborn' from 'C<...>
s\\seaborn\\__init__.py'>
stopwords                LazyCorpusLoader         <WordListCorpusReader in <...>
pwords' (not loaded yet)>
string                   module                   <module 'string' from 'C:<...>
aconda3\\Lib\\string.py'>
svm                      LinearSVC                LinearSVC()
svm_model                LinearSVC                LinearSVC()
svm_preds                ndarray                  10000: 10000 elems, type `obje
ct`, 80000 bytes
sys                      module                   <module 'sys' (built-in)>
tfidf                    TfidfVectorizer          TfidfVectorizer(max_features=5
000)
tfidf_df                 DataFrame                       about  according  <...>
50000 rows x 500 columns]
tfidf_features           csr_matrix                 (0, 193)    0.067115601168
<...>, 198)     0.4603556655663005
train_test_split         function                 <function train_test_split at
0x00000278962AE7A0>
y                        Series                   0        8\n1        8\n2<...>
ngth: 50000, dtype: int32
y_pred                   ndarray                  10000: 10000 elems, type `obje
ct`, 80000 bytes
y_test                   Series                   33553    2\n9427     4\n1<...>
ngth: 10000, dtype: int32
y_train                  Series                   39087    9\n30893    2\n4<...>
ngth: 40000, dtype: int32
```

In [19]:
```python
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

models = {
    "📘 Logistic Regression": lr,
    "📗 Naive Bayes": nb,
    "📕 SVM": svm,
    "📙 Random Forest": rf
}
```

```python
for name, model in models.items():
    print(f"\n{name} Evaluation:")

    y_pred = model.predict(X_test)

    print("\n🧾 Classification Report:")
    print(classification_report(y_test, y_pred, target_names=le.classes_))

    cm = confusion_matrix(y_test, y_pred)
    plt.figure(figsize=(8, 6))
    sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
                xticklabels=le.classes_,
                yticklabels=le.classes_)
    plt.title(f"{name} - Confusion Matrix")
    plt.xlabel("Predicted")
    plt.ylabel("True")
    plt.xticks(rotation=45)
    plt.yticks(rotation=45)
    plt.tight_layout()
    plt.show()
```

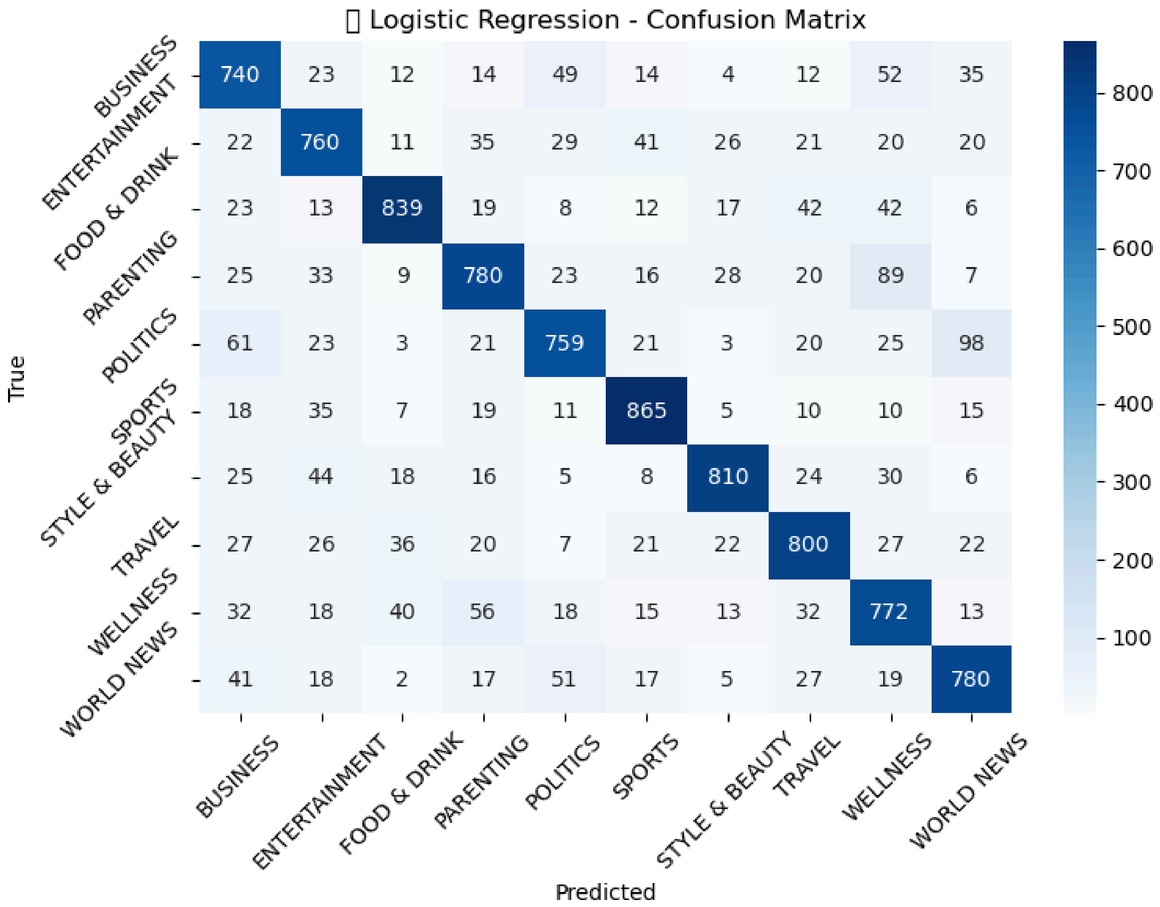
```
📘 Logistic Regression Evaluation:

🧾 Classification Report:
               precision    recall  f1-score   support

     BUSINESS       0.73      0.77      0.75       955
ENTERTAINMENT       0.77      0.77      0.77       985
 FOOD & DRINK       0.86      0.82      0.84      1021
    PARENTING       0.78      0.76      0.77      1030
     POLITICS       0.79      0.73      0.76      1034
       SPORTS       0.84      0.87      0.85       995
STYLE & BEAUTY      0.87      0.82      0.84       986
       TRAVEL       0.79      0.79      0.79      1008
     WELLNESS       0.71      0.77      0.74      1009
   WORLD NEWS       0.78      0.80      0.79       977

     accuracy                           0.79     10000
    macro avg       0.79      0.79      0.79     10000
 weighted avg       0.79      0.79      0.79     10000
```

```
C:\Users\jayde\AppData\Local\Temp\ipykernel_29552\2589437259.py:35: UserWarning:
Glyph 128216 (\N{BLUE BOOK}) missing from font(s) DejaVu Sans.
  plt.tight_layout()
C:\Users\jayde\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:170: UserWa
rning: Glyph 128216 (\N{BLUE BOOK}) missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
```



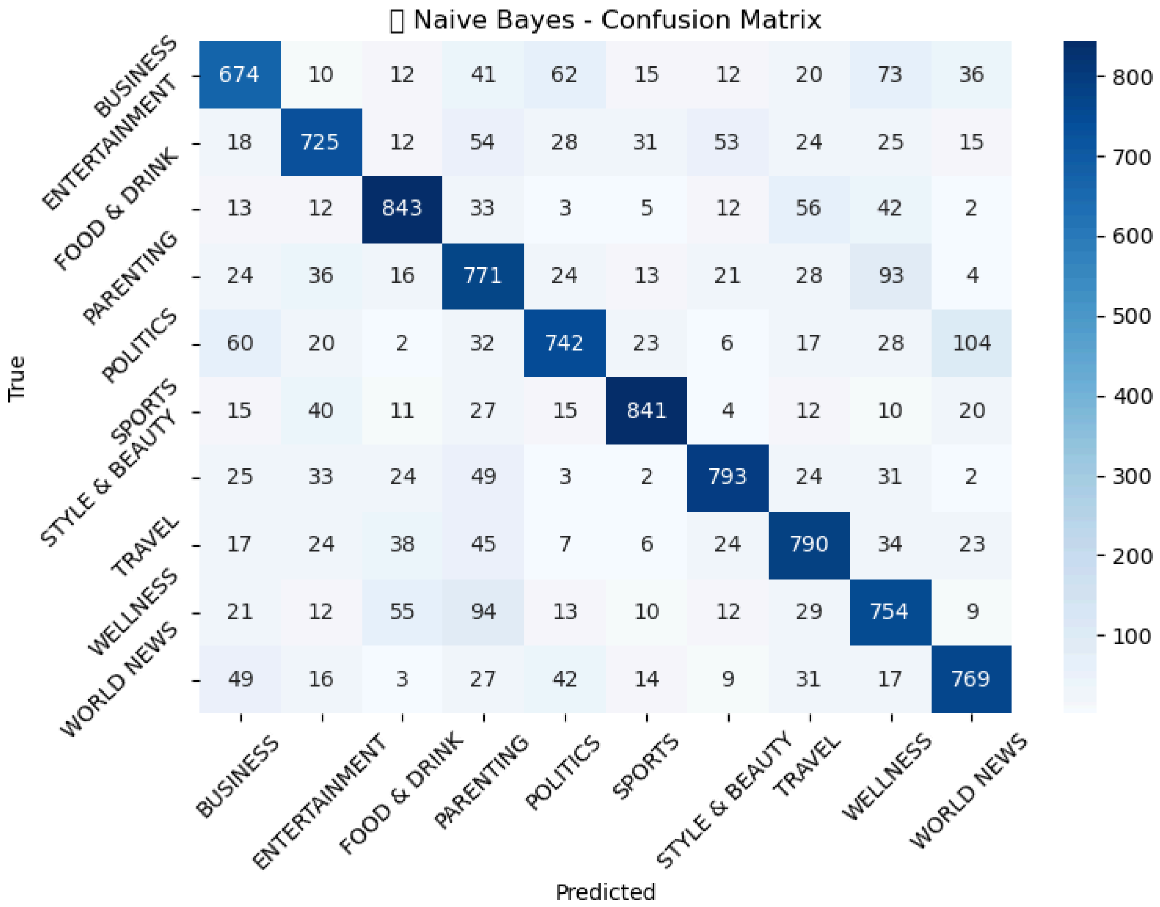

```
📗 Naive Bayes Evaluation:

🧾 Classification Report:
                precision    recall  f1-score   support

      BUSINESS       0.74      0.71      0.72       955
 ENTERTAINMENT       0.78      0.74      0.76       985
  FOOD & DRINK       0.83      0.83      0.83      1021
     PARENTING       0.66      0.75      0.70      1030
      POLITICS       0.79      0.72      0.75      1034
        SPORTS       0.88      0.85      0.86       995
STYLE & BEAUTY       0.84      0.80      0.82       986
        TRAVEL       0.77      0.78      0.77      1008
      WELLNESS       0.68      0.75      0.71      1009
    WORLD NEWS       0.78      0.79      0.78       977

      accuracy                           0.77     10000
     macro avg       0.77      0.77      0.77     10000
  weighted avg       0.77      0.77      0.77     10000
```

```
C:\Users\jayde\AppData\Local\Temp\ipykernel_29552\2589437259.py:35: UserWarning:
Glyph 128215 (\N{GREEN BOOK}) missing from font(s) DejaVu Sans.
  plt.tight_layout()
C:\Users\jayde\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:170: UserWa
rning: Glyph 128215 (\N{GREEN BOOK}) missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
```



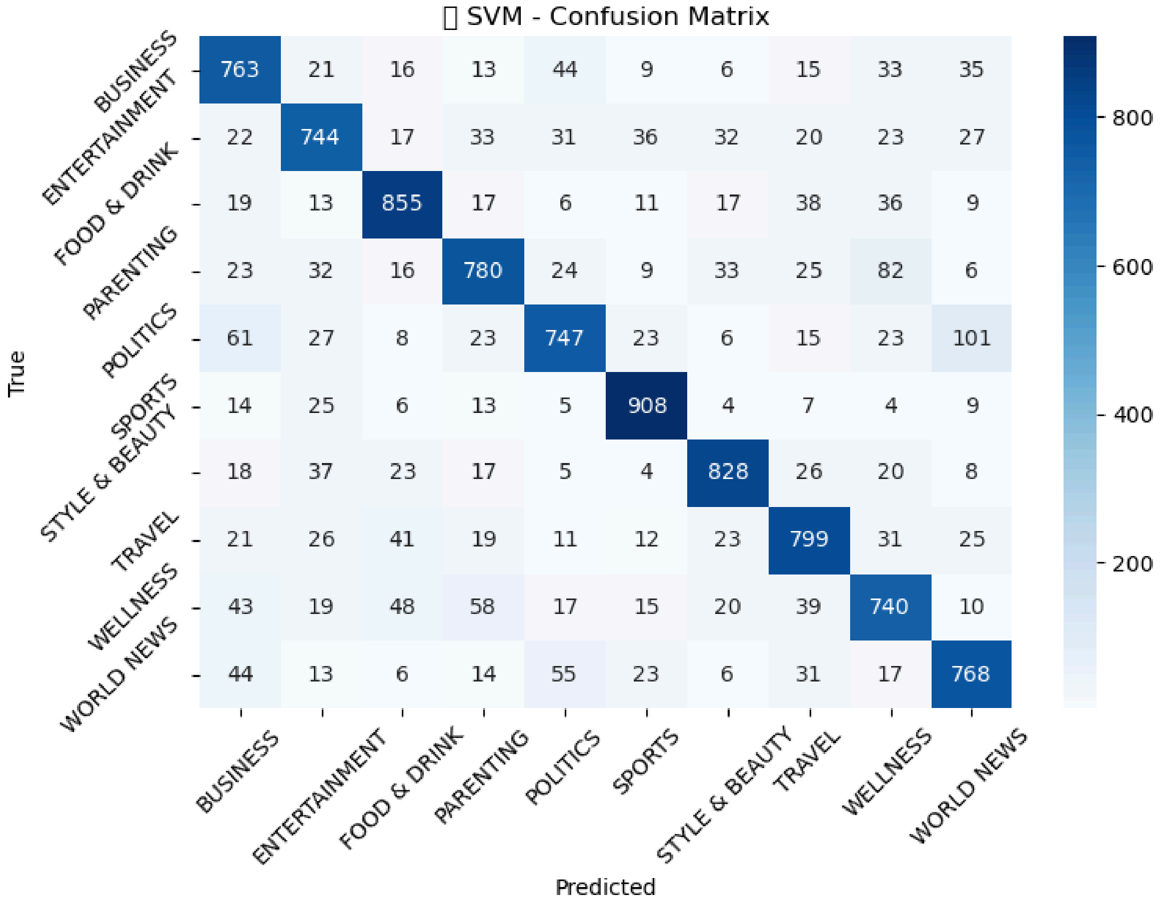

```
📕 SVM Evaluation:

🧾 Classification Report:
                precision    recall  f1-score   support

      BUSINESS       0.74      0.80      0.77       955
 ENTERTAINMENT       0.78      0.76      0.77       985
  FOOD & DRINK       0.83      0.84      0.83      1021
     PARENTING       0.79      0.76      0.77      1030
      POLITICS       0.79      0.72      0.75      1034
        SPORTS       0.86      0.91      0.89       995
STYLE & BEAUTY       0.85      0.84      0.84       986
        TRAVEL       0.79      0.79      0.79      1008
      WELLNESS       0.73      0.73      0.73      1009
    WORLD NEWS       0.77      0.79      0.78       977

      accuracy                           0.79     10000
     macro avg       0.79      0.79      0.79     10000
  weighted avg       0.79      0.79      0.79     10000
```

```
C:\Users\jayde\AppData\Local\Temp\ipykernel_29552\2589437259.py:35: UserWarning:
Glyph 128213 (\N{CLOSED BOOK}) missing from font(s) DejaVu Sans.
  plt.tight_layout()
C:\Users\jayde\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:170: UserWa
rning: Glyph 128213 (\N{CLOSED BOOK}) missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
```



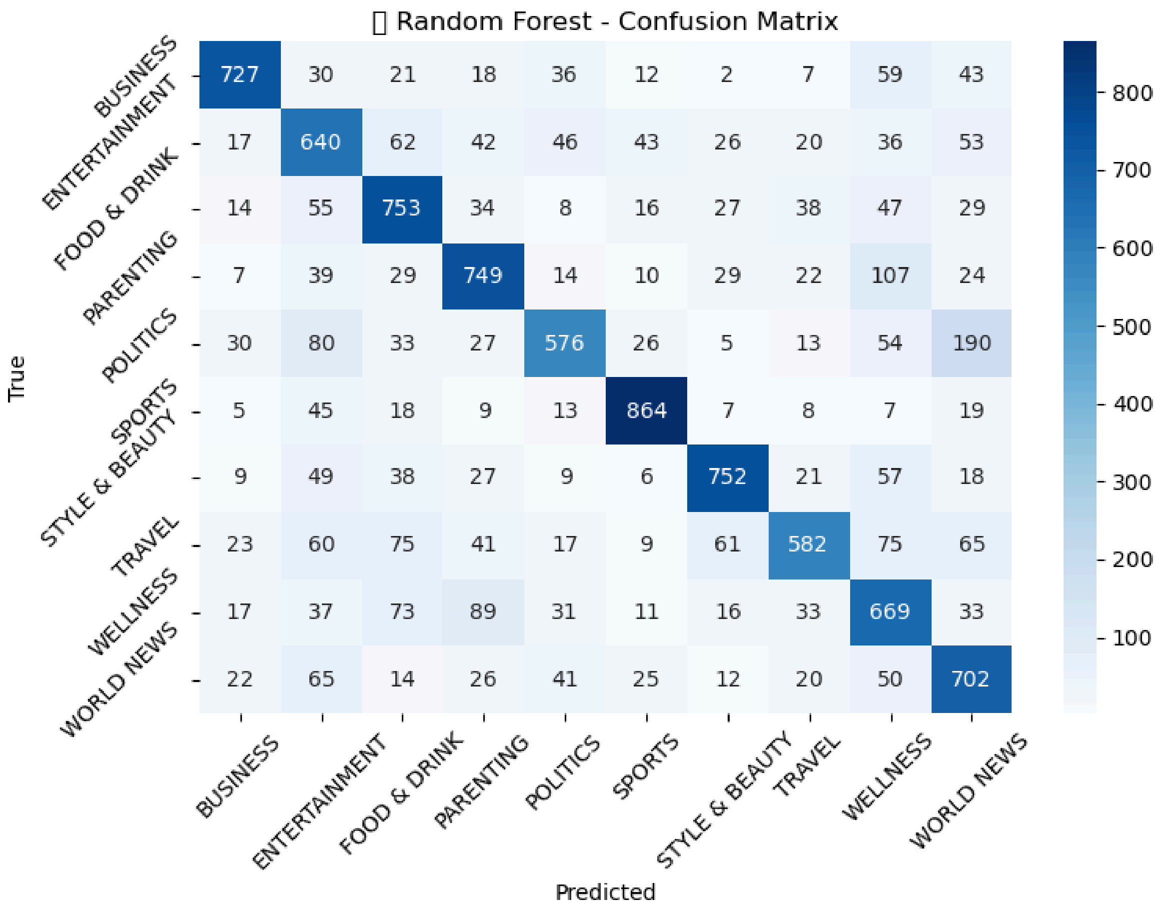

```
📙 Random Forest Evaluation:

🧾 Classification Report:
                precision    recall  f1-score   support

      BUSINESS       0.83      0.76      0.80       955
 ENTERTAINMENT       0.58      0.65      0.61       985
  FOOD & DRINK       0.67      0.74      0.70      1021
     PARENTING       0.71      0.73      0.72      1030
      POLITICS       0.73      0.56      0.63      1034
        SPORTS       0.85      0.87      0.86       995
STYLE & BEAUTY       0.80      0.76      0.78       986
        TRAVEL       0.76      0.58      0.66      1008
      WELLNESS       0.58      0.66      0.62      1009
    WORLD NEWS       0.60      0.72      0.65       977

      accuracy                           0.70     10000
     macro avg       0.71      0.70      0.70     10000
  weighted avg       0.71      0.70      0.70     10000
```

```
C:\Users\jayde\AppData\Local\Temp\ipykernel_29552\2589437259.py:35: UserWarning:
Glyph 128217 (\N{ORANGE BOOK}) missing from font(s) DejaVu Sans.
  plt.tight_layout()
C:\Users\jayde\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:170: UserWa
rning: Glyph 128217 (\N{ORANGE BOOK}) missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
```



News Article Classification – Final Report

Objective

In this project, I aimed to build a machine learning model that can automatically classify news articles into different categories such as politics, sports, food, technology, and more. Since we are surrounded by an overwhelming amount of digital content today, I wanted to explore how machine learning could help automate the process of organizing news content effectively.

Dataset Overview

I worked with a dataset of **50,000 news articles**, which included the following columns:

- `title`
- `short_description`
- `category` (target label)
- `link`

I combined the `title` and `short_description` columns to create a more meaningful text input for each article. This helped improve the quality of the features used for classification.

1.Text Preprocessing

To clean the text data before feeding it into any model, I applied the following preprocessing steps:

- Converted all text to lowercase
- Removed punctuation, numbers, and special characters
- Removed stopwords like "is", "the", "in", etc.
- Used **tokenization** and **lemmatization** with the help of NLTK

2.Feature Engineering

To represent the text numerically for model training, I extracted the following features:

- **TF-IDF Vectorization**: I used the top 5000 most relevant words based on TF-IDF scores
- **Text Length Features**:
  - Total word count
  - Character count
  - Average word length

By combining both semantic (TF-IDF) and structural (length-based) features, I tried to give my models a better understanding of the articles.

3.Model Development

I trained and evaluated four different machine learning models to identify which one could classify the articles most accurately:

Model	Accuracy	Macro F1-Score	My Observations
Logistic Regression	0.7905	~0.79	Performed well across most categories
Naive Bayes	0.7702	~0.77	Very fast, good baseline, but slightly less accurate
Support Vector Machine	0.7932	~0.79	Best performance overall
Random Forest Classifier	0.7014	~0.70	Not as effective, possibly due to sparse features

I used accuracy, F1-score, and confusion matrices to evaluate each model's performance.

4.Model Evaluation

The **Support Vector Machine (SVM)** gave me the best results, with an accuracy of **79.32%**, followed closely by **Logistic Regression**. Both models performed particularly well in categories like:

- **Sports**
- **Food & Drink**
- **Style & Beauty**

However, I noticed that categories like **Parenting**, **Travel**, and **Wellness** were more difficult to classify accurately. I think this could be due to overlapping keywords and fewer training examples in those categories.

Conclusion

This project taught me a lot about how natural language processing and classical machine learning techniques can be used together to solve real-world problems. I learned that:

- Cleaning the text thoroughly before modeling is crucial
- TF-IDF is very effective for text classification tasks
- SVM and Logistic Regression work surprisingly well for high-dimensional sparse data

In [ ]:

```

```