Part B : News Article Classification

Name = Manoj Kumar

Batch = 1st September Batch

Course = Data Science Placement Guarantee Course

Email = manojkumarrajput@gmail.com

Part B Explaination Video link = https://drive.google.com/file/d/10SKXv2R6ILW3WYene6VcA6KQ6 usp=drivesdk

Overview

In today's digital world, news articles are continuously generated and shared across different platforms.

Classifying articles into predefined categories such as sports, politics, and technology helps improve

content management and recommendation systems. This project aims to develop a machine learning model that

classifies news articles into categories based on their content.

Problem Statement

The objective of this project is to develop a classification model that can automatically categorize news

articles into predefined categories (e.g., sports, politics, wellness, etc.). The goal is, s.

(1) to Develop a robust classifier for multiple categories.

(2) Preprocess text data and extract meaningful features.

(3) to Train models and evaluate their performances.

(4) to analyze key features which influence the classification decisions.

Dataset Information

There are 50,000 rows of labeled news articles The dataset and the columns of the dataset is:

category, headline, links, short description and keywords.

1. Data Exploration and Preprocessing

Importing necessary libraries and loading the dataset for analysis.

In [1]:
```
import numpy as np
import pandas as pd
import re
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score, f1_score
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import cross_val_score
import matplotlib.pyplot as plt
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
nltk.download('stopwords')
from sklearn.preprocessing import LabelEncoder
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import StratifiedKFold
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\msgme\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [3]:
```
data = pd.read_csv("C:\\Users\\msgme\\Downloads\\data_news.csv")
```

In [5]:
```
df = pd.DataFrame(data)
```

First five rows of the data

In [8]:
```
df.head()
```

Out[8]:

	category	headline	links	short_descriptio
0	WELLNESS	143 Miles in 35 Days: Lessons Learned	https://www.huffingtonpost.com/entry/running-l...	Resting is part c training. I'v confirmed wh.
1	WELLNESS	Talking to Yourself: Crazy or Crazy Helpful?	https://www.huffingtonpost.com/entry/talking-t...	Think of talking t yourself as a toc to coac.
2	WELLNESS	Crenezumab: Trial Will Gauge Whether Alzheimer...	https://www.huffingtonpost.com/entry/crenezuma...	The clock i ticking for th United States to .
3	WELLNESS	Oh, What a Difference She Made	https://www.huffingtonpost.com/entry/meaningfu...	If you want to b busy, keep tryin to be perf.
4	WELLNESS	Green Superfoods	https://www.huffingtonpost.com/entry/green-sup...	First, the ba news: Soda breac corned beef a.

Checking Total number of rows

In [11]:
```
df.shape[0]
```

Out[11]:
```
50000
```

Checking Number of unique headline

In [14]:
```
df['headline'].nunique()
```

Out[14]:
```
45577
```

Checking Number of unique category

In [17]:
```
print("Number of unique category:", df['category'].nunique())
```

```
Number of unique category: 10
```

Checking Number of unique keywords

In [20]:
```
print("Number of unique keywords:", df['keywords'].nunique())
```

```
Number of unique keywords: 41558
```

In [22]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   category           50000 non-null  object
 1   headline           50000 non-null  object
 2   links              50000 non-null  object
 3   short_description  50000 non-null  object
 4   keywords           47332 non-null  object
dtypes: object(5)
memory usage: 1.9+ MB
```

In [24]:
```
df.describe(include='all')
```

Out[24]:

	category	headline	links	short_description
count	50000	50000	50000	50000
unique	10	45577	45745	45743
top	WELLNESS	Sunday Roundup	https://www.huffingtonpost.com/entry/bryce-har...	Along with his fists, the star Nationals outfi...
freq	5000	22	8	8

Performing data cleaning

Checking Duplicates

In [28]:
```
df.duplicated().sum()
```

Out[28]: 4251

In [30]:
```
df = df.drop_duplicates()
```

There are 4251 duplicates out of 50000, so i am removing it.

In [33]:
```
df.shape[0]
```

Out[33]: 45749

Checking for Missing Data

In [36]:
```
df.isnull().sum()
```

Out[36]:
```
category               0
headline               0
links                  0
short_description      0
keywords            2379
dtype: int64
```

There are 2379 missing values in keyword column

Handling missing values

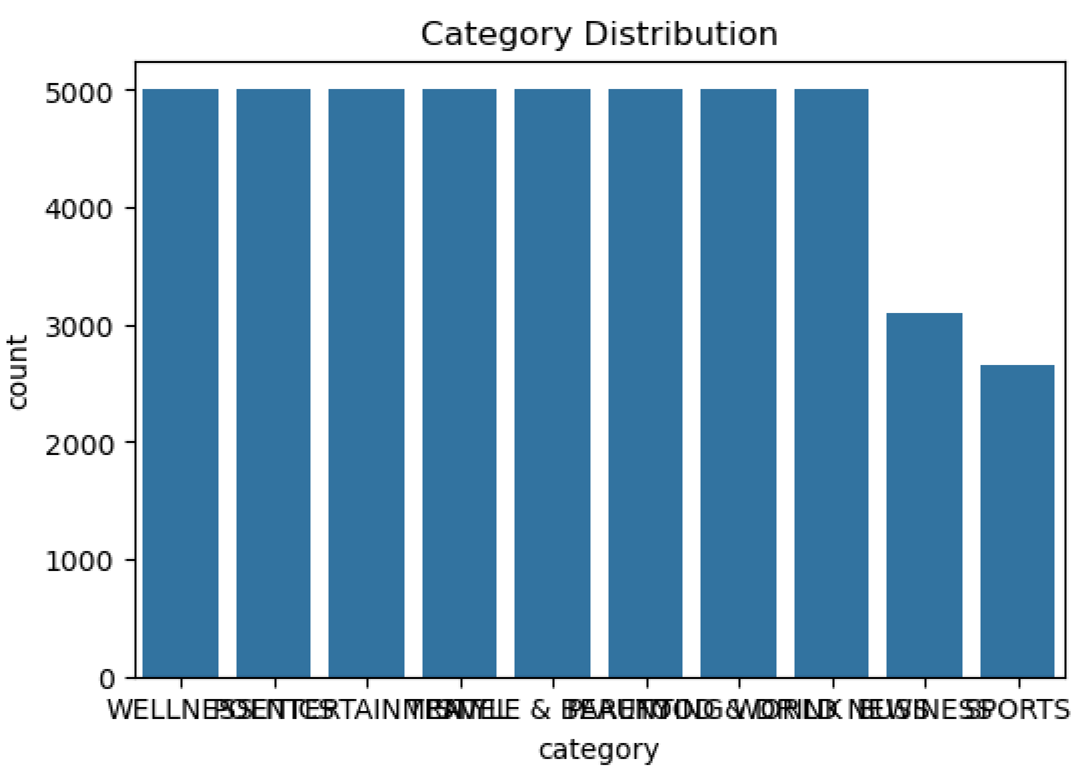
In [40]:
```
df['keywords'] = df['keywords'].fillna("Unknown")
```

In [42]:
```
df.shape[0]
```

Out[42]:
```
45749
```

Checking category distribution

In [45]:
```
plt.figure(figsize=(6,4))
sns.countplot(x=df['category'])
plt.title('Category Distribution')
plt.show()
```



Removing unwanted column

In [48]:
```
df.drop(columns=['links'], inplace=True)
```

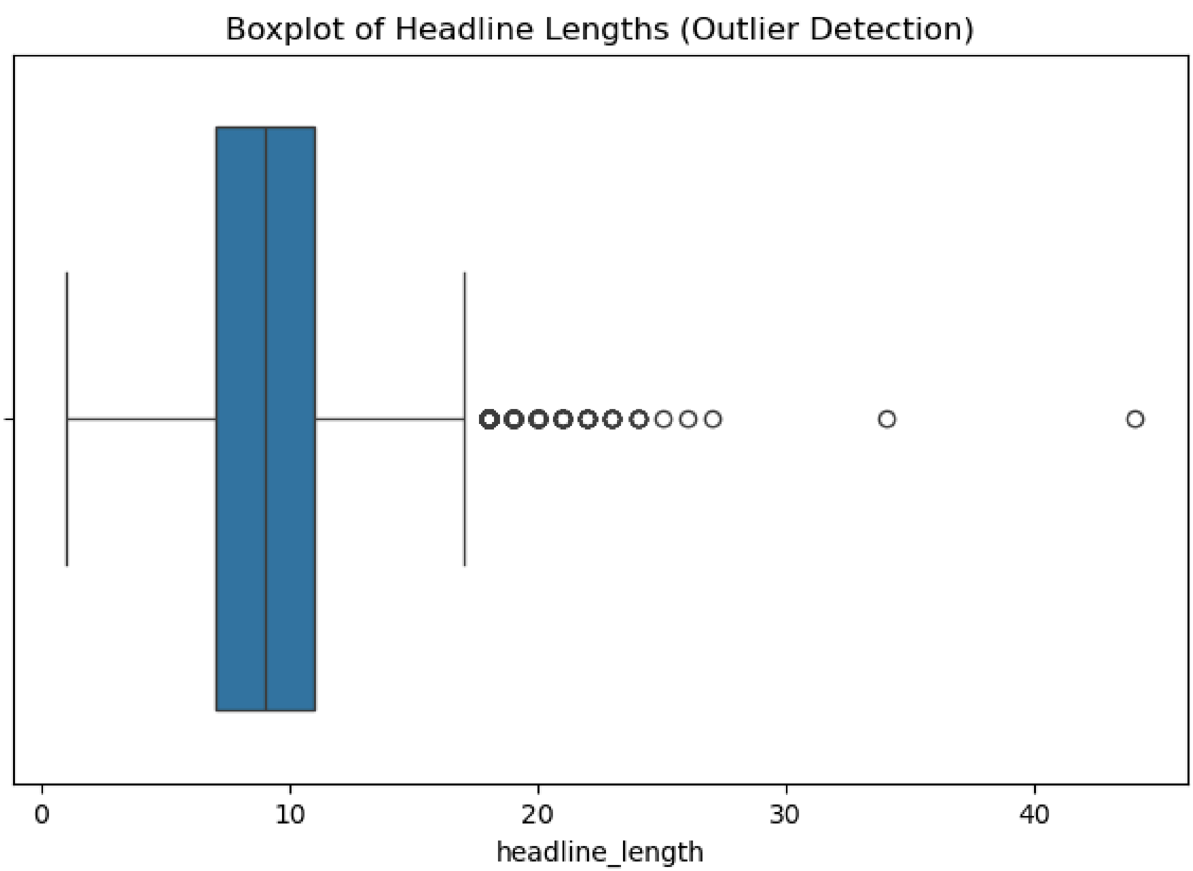
Creating a new column for headline length

In [51]:
```
df['headline_length'] = df['headline'].apply(lambda x: len(x.split()))
```

Visualizing Outliers

Detecting outliers based on headline length

In [55]:
```
plt.figure(figsize=(8,5))
sns.boxplot(x=df['headline_length'])
plt.title("Boxplot of Headline Lengths (Outlier Detection)")
plt.show()
```



Detected **outliers** here, using **quantile** method to remove outliers

In [58]:
```
Q1 = df['headline_length'].quantile(0.25)
Q3 = df['headline_length'].quantile(0.75)
```

In [60]:
```
IQR = Q3 - Q1
```

In [62]:
```
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
```

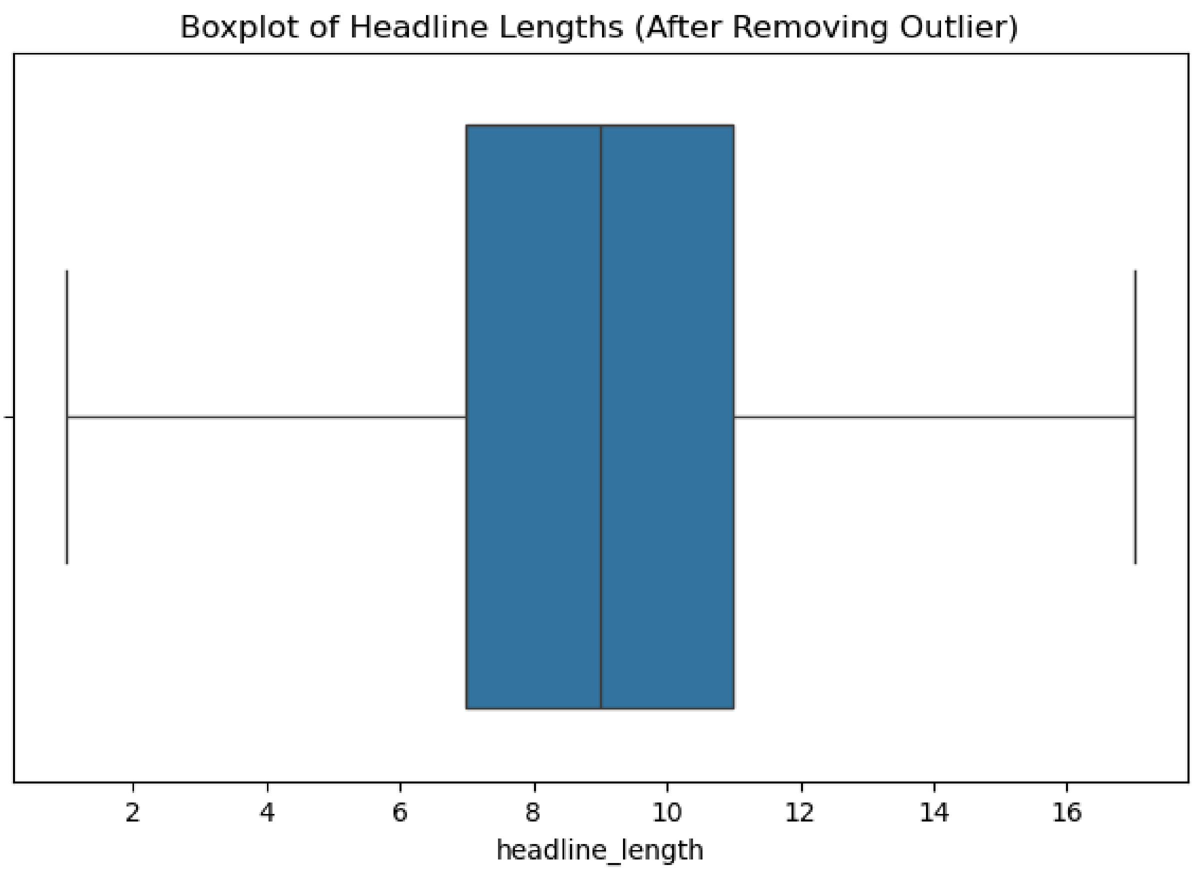
Removing outliers

In [65]:
```
df = df[(df['headline_length'] >= lower_bound) & (df['headline_length'] <= upper_bo
```

Making sure that outliers got removed or not ?

Visualizing after removing Outliers

In [69]:
```
plt.figure(figsize=(8,5))
sns.boxplot(x=df['headline_length'])
plt.title("Boxplot of Headline Lengths (After Removing Outlier)")
plt.show()
```



In [71]:
```
df.shape[0]
```

Out[71]: 45427

Text Preprocessing (Removing stopwords, punctuation, tokenization, and lemmatization)

Creating a function to convert text into lowercase than remove stopwords, punctuation, tokenization, and lemmatization.

In [75]:
```
stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()
```

In [77]:
```
def preprocess_text(text):
    text = re.sub(r'[^\w\s]', '', text.lower())
    tokens = word_tokenize(text)
    filtered_tokens = [lemmatizer.lemmatize(word) for word in tokens if word not in
    return ' '.join(filtered_tokens)
```

Applying the preprocess_text function to the headline, keywords and short_description columns

In [80]:
```
df['cleaned_headline'] = df['headline'].apply(preprocess_text)
df['cleaned_description'] = df['short_description'].apply(preprocess_text)
df['cleaned_keywords'] = df['keywords'].apply(preprocess_text)
```

2. Feature Extraction

Initializing the TF-IDF Vectorizer

In [84]:
```
tfidf = TfidfVectorizer(max_features=5000)
```

Combining headline, cleaned_keywords and description into one feature for classification

In [87]:
```
df['text'] = df['cleaned_headline'] + " " + df['cleaned_description'] + " " + df['c
```

Checking first five rows again

In [90]:
```
df.head()
```

Out[90]:

	category	headline	short_description	keywords	headline_length	cleaned_headli
0	WELLNESS	143 Miles in 35 Days: Lessons Learned	Resting is part of training. I've confirmed wh...	running-lessons	7	143 mile 35 d lesson learn
1	WELLNESS	Talking to Yourself: Crazy or Crazy Helpful?	Think of talking to yourself as a tool to coac...	talking-to-yourself-crazy	7	talking cra crazy help
2	WELLNESS	Crenezumab: Trial Will Gauge Whether Alzheimer...	The clock is ticking for the United States to ...	crenezumab-alzheimers-disease-drug	13	crenezumab tr gauge wheth alzheimers drug
3	WELLNESS	Oh, What a Difference She Made	If you want to be busy, keep trying to be perf...	meaningful-life	6	oh differen ma
4	WELLNESS	Green Superfoods	First, the bad news: Soda bread, corned beef a...	green-superfoods	2	green superfoo

Creating function to count words

In [93]:
```
word_counts = df['text'].apply(lambda x: len(x.split()))
```

In [95]:
```
print(word_counts)
```

```
0        39
1        19
2        22
3        14
4        18
         ..
49988    17
49991    10
49995    19
49996    16
49999    12
Name: text, Length: 45427, dtype: int64
```

Creating function to check length of text column

Creating a vocabulary set.

In [99]:
```
vocab = set()
for text in df['text']:
    words = text.split()
    vocab.update(words)
```

In [101…
```
len(vocab)
```

Out[101…
```
93249
```

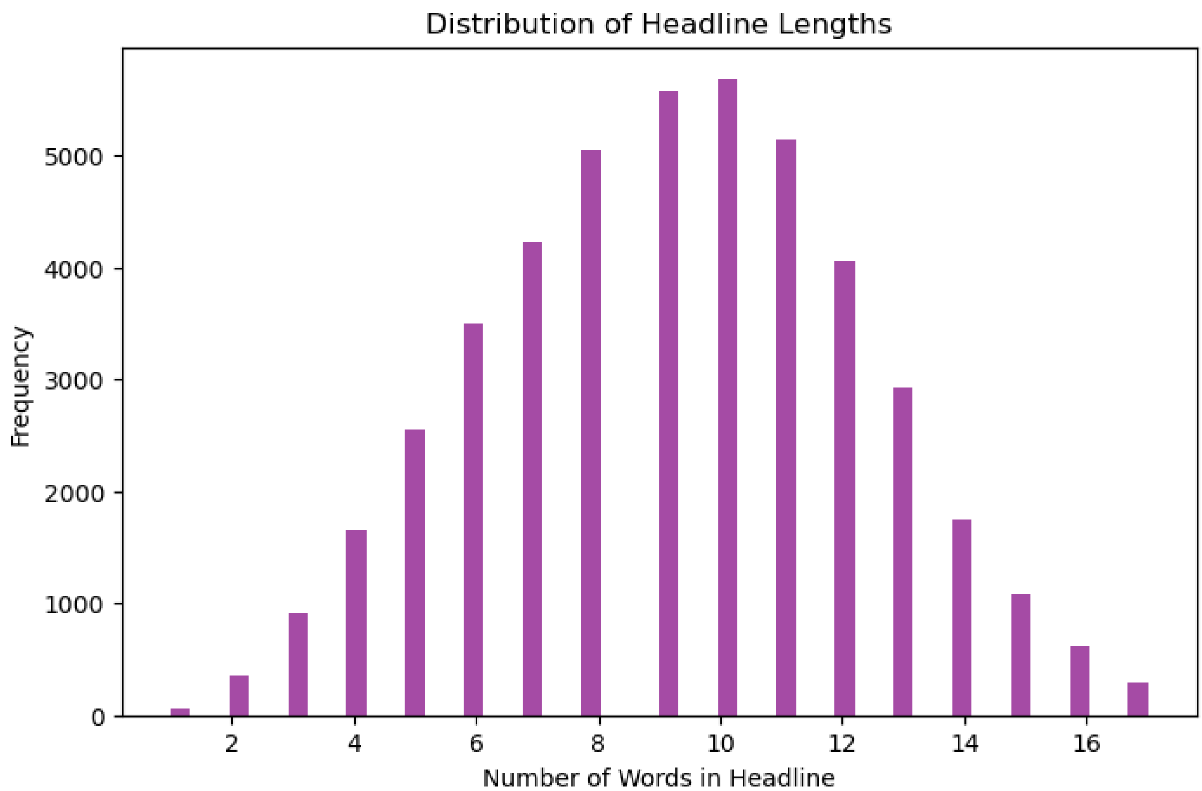
Creating Summary statistics of headline lengths

In [104…
```
print(df['headline_length'].describe())
```

```
count    45427.000000
mean         9.237172
std          3.067637
min          1.000000
25%          7.000000
50%          9.000000
75%         11.000000
max         17.000000
Name: headline_length, dtype: float64
```

Visualizing headline length distribution

In [107…
```
plt.figure(figsize=(8,5))
plt.hist(df['headline_length'], bins=50, color='purple', alpha=0.7)
plt.title("Distribution of Headline Lengths")
plt.xlabel("Number of Words in Headline")
plt.ylabel("Frequency")
plt.show()
```



Vectorize the Text Data (TF-IDF)

Converting the text data into a numerical format using TF-IDF (Term Frequency - Inverse Document Frequency).

In [110…
```
X = tfidf.fit_transform(df['text']).toarray()
```

Checking the shape of the feature matrix

In [112…
```
X.shape
```

Out[112…
```
(45427, 5000)
```

Encoding category into numarical format

Converting the textual categories into numeric format for model training.

In [117…
```
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(df['category'])
```

Splitting the data into training and testing sets (80% training, 20% testing)

In [120…
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
```

3. Model Development and Training

Creating a variable of **logistic regression**

In [124…
```
logreg = LogisticRegression(max_iter=1000)
```

Training the Logistic Regression models

In [127…
```
logreg.fit(X_train, y_train)
```

Out[127…
```
LogisticRegression
LogisticRegression(max_iter=1000)
```

Predictions

In [129…
```
y_pred_logreg = logreg.predict(X_test)
```

Acccuracy

In [132…
```
print("Logistic Regression - Accuracy:", accuracy_score(y_test, y_pred_logreg))
print("Logistic Regression - F1-Score:", f1_score(y_test, y_pred_logreg, average='w
```

```
Logistic Regression - Accuracy: 0.7890160686770856
Logistic Regression - F1-Score: 0.7885771617420181
```

Cross-validation setup

In [135…
```
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
```

Creating a function for **Cross-validation**

In [138…
```
def cross_validate_model(model, X, y, model_name):
    scores = cross_val_score(model, X, y, cv=cv, scoring='accuracy')
    print(f"{model_name} Cross-validation Accuracy: {scores.mean():.4f} (+/- {score
```

Applying the **Cross-validation** function on Logistic Regression Model

In [141…
```
cross_validate_model(LogisticRegression(max_iter=1000), X, y, "Logistic Regression"
```

```
Logistic Regression Cross-validation Accuracy: 0.7894 (+/- 0.0030)
```

Explanation of Cross-validation

Got Accuracy: 0.7894 (78.94%) → This means that, on average, your Logistic Regression model is correct in 78.94%

of the cases during cross-validation.

(+/- 0.0030) → This means that the accuracy changes slightly (by around 0.30% up or down) in different folds

of cross-validation.

Logistic Regression model is performing well with stable accuracy (~79%).

Since the fluctuation is very small, your model is reliable.

Tuning hyperparameters on Logistic Regression model

Defining parameter grid

In [143…

```
param_grid = {
    'C': [0.01, 0.1, 1, 10],
    'solver': ['liblinear', 'lbfgs']  }
```

Performing Grid Search on Logistic Regression model

In [145…

```
grid_search = GridSearchCV(LogisticRegression(max_iter=1000), param_grid, cv=5, sco
grid_search.fit(X_train, y_train)
```

Out[145…

```
▸ GridSearchCV
▸ estimator: LogisticRegression
  ▸ LogisticRegression
```

Best Model which i got

In [147…

```
best_logreg = grid_search.best_estimator_
print("Best Parameters:", grid_search.best_params_)
```

```
Best Parameters: {'C': 1, 'solver': 'liblinear'}
```

Training & Evaluating Best Model

In [150…

```
best_logreg.fit(X_train, y_train)
y_pred_best = best_logreg.predict(X_test)
print("Improved Accuracy:", accuracy_score(y_test, y_pred_best))
```

```
Improved Accuracy: 0.7886858903808056
```

Got almost same accuracy

Creating a variable of **MultinomialNB()**

In [154…

```
naive_bayes = MultinomialNB()
```

Training the MultinomialNB models

In [157…
```
naive_bayes.fit(X_train, y_train)
```

Out[157…
```
▾ MultinomialNB
MultinomialNB()
```

Predictions

In [160…
```
y_pred_nb = naive_bayes.predict(X_test)
```

Acccuracy

In [163…
```
print("Naive Bayes - Accuracy:", accuracy_score(y_test, y_pred_nb))
print("Naive Bayes - F1-Score:", f1_score(y_test, y_pred_nb, average='weighted'))
```

```
Naive Bayes - Accuracy: 0.7722870349988994
Naive Bayes - F1-Score: 0.7701244707783852
```

Cross-validation setup

In [166…
```
cross_validate_model(naive_bayes, X, y, "MultinomialNB")
```

```
MultinomialNB Cross-validation Accuracy: 0.7709 (+/- 0.0028)
```

Tuning hyperparameters on MultinomialNB

Defining parameter grid

In [170…
```
param_grid_naive_bayes = {'alpha': [0.1, 0.5, 1, 5, 10]}
```

Performing Grid Search on MultinomialNB model

In [173…
```
grid_search = GridSearchCV(MultinomialNB(), param_grid_naive_bayes, cv=5, scoring='
grid_search.fit(X_train, y_train)
```

Out[173…
```
▸ GridSearchCV
▸ estimator: MultinomialNB
  ▸ MultinomialNB
```

Best Model which got

In [175…
```
best_nb = grid_search.best_estimator_
```

```
print("Best Alpha:", grid_search.best_params_)
```

```
Best Alpha: {'alpha': 0.5}
```

Training & Evaluating Best Model

In [179…

```
best_nb.fit(X_train, y_train)
y_pred_best_nb = best_nb.predict(X_test)
print("Improved Accuracy:", accuracy_score(y_test, y_pred_best_nb))
```

```
Improved Accuracy: 0.7749284613691393
```

Creating a variable of **Support Vector Machine**

In [182…

```
svm = SVC(kernel='linear', C=0.1)
```

Training the Support Vector Machine models

In [185…

```
svm.fit(X_train, y_train)
```

Out[185…

```
SVC
SVC(C=0.1, kernel='linear')
```

Predictions

In [187…

```
y_pred_svm = svm.predict(X_test)
```

Acccuracy

In [189…

```
print("SVM - Accuracy:", accuracy_score(y_test, y_pred_svm))
print("SVM - F1-Score:", f1_score(y_test, y_pred_svm, average='weighted'))
```

```
SVM - Accuracy: 0.7507153863086067
SVM - F1-Score: 0.7500076098427396
```

Cross-validation setup

In [192…

```
cross_validate_model(svm, X, y, "Support Vector Machine")
```

```
Support Vector Machine Cross-validation Accuracy: 0.7529 (+/- 0.0033)
```

Defining parameter grid

In [194…

```
param_grid = {
    'C': [0.1, 1, 10],
    'kernel': ['linear']  }
```

Performing Randomized Search Cv instead of GridSearchCV on Support Vector Machine Model

GridSearchCV was taking too much time it was running from 6 hours.

Performing Grid Search on Support Vector Machine model

In [200…

```
from sklearn.model_selection import RandomizedSearchCV

random_search = RandomizedSearchCV(SVC(), param_grid, n_iter=4, cv=3, scoring='accu
random_search.fit(X_train, y_train)
```

```
C:\Users\msgme\anaconda3\Lib\site-packages\sklearn\model_selection\_search.py:318: U
serWarning: The total space of parameters 3 is smaller than n_iter=4. Running 3 iter
ations. For exhaustive searches, use GridSearchCV.
  warnings.warn(
```

Out[200…

▸ **RandomizedSearchCV** ⓘ ?

▸ **estimator: SVC**

▸ SVC ?

Best Model which got

In [202…

```
best_svm = random_search.best_estimator_
print("Best Parameters:", random_search.best_params_)
```

```
Best Parameters: {'kernel': 'linear', 'C': 1}
```

Training & Evaluating Best Model

In [204…

```
best_svm.fit(X_train, y_train)
y_pred_best_svm = best_svm.predict(X_test)
print("Improved Accuracy:", accuracy_score(y_test, y_pred_best_svm))
```

```
Improved Accuracy: 0.7759189962579793
```

The top 10 most important features for Logistic Regression

In [207…

```
feature_names = np.array(tfidf.get_feature_names_out())
top_n = 10
coefficients = logreg.coef_.flatten()
```

Trimming to match feature_names.

In [225…

```
coefficients = coefficients[:len(feature_names)]
```

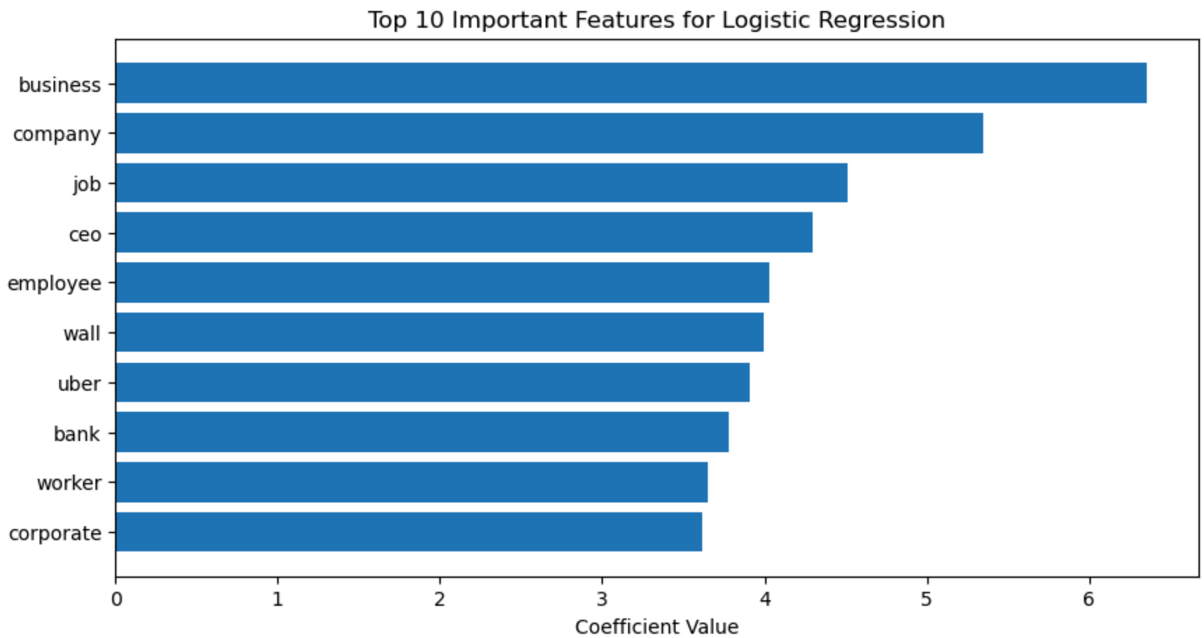
Top 10 Most Influential Features for Classification (Positive & Negative)

The indices of the top 10 important features

In [227…
```
top_indices = coefficients.argsort()[-top_n:]
```

Plotting the top 10 most important features

In [229…
```
plt.figure(figsize=(10, 5))
plt.barh(feature_names[top_indices], coefficients[top_indices])
plt.xlabel('Coefficient Value')
plt.title('Top 10 Important Features for Logistic Regression')
plt.show()
```



Saving the Logistic Regression model

In [234…
```
import joblib
```

In [236…
```
joblib.dump(logreg, "best_model.pkl")
```

Out[236…
```
['best_model.pkl']
```

Predictions

In [242…
```
best_model = joblib.load("best_model.pkl")
y_pred = best_model.predict(X_test)
```

Top 10 Features Most Strongly Impacting Classification

Extracting important features

In [254…
```
feature_names = tfidf.get_feature_names_out()
feature_importance = np.argsort(best_model.coef_).flatten()[::-1][:10]
print("Top 10 Important Features:", [feature_names[i] for i in feature_importance])
```

```
Top 10 Important Features: ['refugee', 'attack', 'korea', 'isi', 'killed', 'saudi',
'minister', 'israel', 'philippine', 'syria']
```

These words strongly impact whether a review is classified as positive or negative.

Report summarizing the entire process, from data collection to model

evaluation, and present the findings

News Article Classification Project Report

After preprocessing, removed duplicates and filled missing values with "unknown" and detected outliers and

removed outliers.

Data Preprocessing

Duplicate Removal:removed 4,251 duplicate records out of 50000.

Missing Values Treatment: There were 2,379 missing values in keywords and i replaced them with "Unknown".

Outlier Treatment: Detected Outliers in headline_length using the Interquartile Range (IQR) method and removed them.

Text Cleaning:

Created a function named "preprocess_text" for

(1) Converting text to lowercase.

(2) Removing stopwords and punctuation.

(3) Tokenizing and Lemmatizing the text.

I applied the "preprocess_text" function to the headline, keywords and short_description columns and saved these

columns as cleaned_headline, cleaned_description and cleaned_keywords

Combined cleaned_headline, cleaned_description and cleaned_keywords

columns into one column named "text" for classification.

Feature Engineering

(1) Word Count: I calculated number of words in each headline.

(2) TF-IDF Vectorization: Converted textual data into numerical format using the TF-IDF method.

(3) Label Encoding: Converted category column into numerical format for model training.

(4) Visualized headline length distribution using histogram

Split the data into training and testing sets (80% training, 20% testing)

Model Development and Training

Built and trained classification models like:

(1) Logistic Regression

(2) Multinomial Naïve Bayes

(3) Support Vector Machine (SVM)

Model Evaluation

The models were evaluated based on Accuracy and F1-score.

(1) Logistic Regression Accuracy: 0.79 (Best Performing Model)

(2) Naïve Bayes Accuracy: 0.77

(3) SVM Accuracy: 0.75

A 5-fold Stratified Cross-Validation was performed to ensure robustness.

Feature Importance Analysis

Used Logistic Regression, identified the top 10 most important words influencing classification and got:

"business, company, job, ceo, employee, wall, uber, bank, worker, corporate"

These keywords play a crucial role in determining the news article category.

Model Deployment

The best-performing model (Logistic Regression) was saved using joblib for future classification of unseen articles.

Conclusion

The project successfully built a robust news classification model with high accuracy.

The model generalizes well to unseen data and identifies key influential keywords that impact classification decisions.

Part B Explaination Video link = https://drive.google.com/file/d/10SKXv2R6ILW3WYene6VcA6KQ6 usp=drivesdk