```python
# Step 1: Import Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
import string

from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import LinearSVC
from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score

import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer

nltk.download('stopwords')
nltk.download('wordnet')
```

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

```
True
```

```python
# Load dataset
df = pd.read_csv('/content/data_news.csv')

# Show first few rows
df.head()
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 50000,\n  \"fields\":
[\n    {\n      \"column\": \"category\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 10,\n
\"samples\": [\n          \"BUSINESS\",\n          \"POLITICS\",\n
\"PARENTING\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"headline\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 45577,\n
\"samples\": [\n          \"These Artists Tried 'Erasing' Parts Of The
U.S.-Mexico Border Fence\",\n          \"Bugaboo Stroller Recall:
Thousands Of Cameleon3 Strollers Recalled Due To Fall Hazard\",\n
\"Airline Employee's Singing Tribute To Veteran Will Give You
Chills\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"links\",\n      \"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 45745,\n        \"samples\": [\n
```

```
\"https://www.huffingtonpost.com/entry/tommy-tuberville-dan-patrick-
texas-tech_us_5bb6c738e4b097869fd2c796\",\n
\"https://www.huffingtonpost.com/entry/relaxing-in-
texas_us_5b9bdd83e4b03a1dcc7ad455\",\n
\"https://www.huffingtonpost.com/entry/the-bachelor-top-hotel-
deals_us_5b9b79cee4b03a1dcc77e70c\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"short_description\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 45743,\n        \"samples\": [\n
\"Check out the two covers below. Do you think the pic was one worth
repeating? IO Donna: Jennifer Lawrence (you might have\",\n
\"Oh you individually wrapped your tacos in twine? They look
delicious!\",\n          \"The drama is called \\\"Perfect
Citizen.\\\"\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"keywords\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 41558,\n
\"samples\": [\n          \"typhoon-chan-hom-china\",\n          \"mj-
ryan-gps-guide\",\n          \"white-house-takes-big-step-towards-
halting-arctic-drilling\"\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df"}
```


```python
# Combine text fields (headline + short_description + keywords)
df['text'] = df['headline'].fillna('') + ' ' +
df['short_description'].fillna('') + ' ' + df['keywords'].fillna('')

# Drop unused columns
df = df[['category', 'text']]

# Drop missing or empty values
df.dropna(inplace=True)
df = df[df['text'].str.strip() != '']

# Text Cleaning Function
def clean_text(text):
    text = text.lower()
    text = re.sub(r'\[.*?\]', '', text)
    text = re.sub(r'https?://\S+|www\.\S+', '', text)
    text = re.sub(r'<.*?>+', '', 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

# Apply cleaning
df['clean_text'] = df['text'].apply(clean_text)
```

```python
# Stopwords and Lemmatization
stop = stopwords.words('english')
lemmatizer = WordNetLemmatizer()

def preprocess(text):
    words = text.split()
    words = [lemmatizer.lemmatize(word) for word in words if word not
in stop]
    return ' '.join(words)

df['processed_text'] = df['clean_text'].apply(preprocess)

# TF-IDF Vectorization
tfidf = TfidfVectorizer(max_features=5000)
X = tfidf.fit_transform(df['processed_text'])

# Target
y = df['category']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Models
models = {
    "Logistic Regression": LogisticRegression(max_iter=1000),
    "Naive Bayes": MultinomialNB(),
    "SVM": LinearSVC()
}

# Training and Evaluation
for name, model in models.items():
    print(f"\n{name}")
    model.fit(X_train, y_train)
    preds = model.predict(X_test)
    print("Accuracy:", accuracy_score(y_test, preds))
    print("Classification Report:\n", classification_report(y_test,
preds))
```

```
Logistic Regression
Accuracy: 0.7987
Classification Report:
                  precision    recall  f1-score   support

      BUSINESS       0.73      0.78      0.76       955
 ENTERTAINMENT       0.77      0.78      0.77       985
  FOOD & DRINK       0.85      0.82      0.84      1021
     PARENTING       0.78      0.76      0.77      1030
      POLITICS       0.79      0.74      0.77      1034
        SPORTS       0.86      0.89      0.88       995
```

```
STYLE & BEAUTY       0.86      0.85      0.85       986
        TRAVEL       0.83      0.80      0.82      1008
      WELLNESS       0.73      0.76      0.74      1009
    WORLD NEWS       0.79      0.81      0.80       977

      accuracy                           0.80     10000
     macro avg       0.80      0.80      0.80     10000
  weighted avg       0.80      0.80      0.80     10000


Naive Bayes
Accuracy: 0.7831
Classification Report:
                precision    recall  f1-score   support

      BUSINESS       0.71      0.73      0.72       955
 ENTERTAINMENT       0.80      0.74      0.77       985
  FOOD & DRINK       0.82      0.85      0.84      1021
     PARENTING       0.69      0.74      0.72      1030
      POLITICS       0.80      0.73      0.76      1034
        SPORTS       0.87      0.86      0.87       995
STYLE & BEAUTY       0.85      0.84      0.85       986
        TRAVEL       0.79      0.81      0.80      1008
      WELLNESS       0.71      0.72      0.72      1009
    WORLD NEWS       0.79      0.81      0.80       977

      accuracy                           0.78     10000
     macro avg       0.78      0.78      0.78     10000
  weighted avg       0.78      0.78      0.78     10000


SVM
Accuracy: 0.7914
Classification Report:
                precision    recall  f1-score   support

      BUSINESS       0.74      0.80      0.77       955
 ENTERTAINMENT       0.78      0.75      0.77       985
  FOOD & DRINK       0.83      0.83      0.83      1021
     PARENTING       0.76      0.75      0.76      1030
      POLITICS       0.78      0.73      0.75      1034
        SPORTS       0.86      0.91      0.89       995
STYLE & BEAUTY       0.84      0.85      0.84       986
        TRAVEL       0.81      0.78      0.80      1008
      WELLNESS       0.73      0.72      0.72      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


# Save trained models
trained_models = {
    "Logistic Regression": models["Logistic Regression"],
    "Naive Bayes": models["Naive Bayes"],
    "SVM": models["SVM"]
}

# Prediction Function
def predict_category(user_input, chosen_model):
    # Clean and preprocess input
    cleaned = clean_text(user_input)
    processed = preprocess(cleaned)

    # Transform with TF-IDF
    vectorized = tfidf.transform([processed])

    # Predict
    model = trained_models.get(chosen_model)
    if model:
        prediction = model.predict(vectorized)
        print(f"\nPredicted Category: {prediction[0]}")
    else:
        print("Invalid model choice.")

# Single input from user
print("\nEnter a news article to classify:")
user_text = input("Enter text (headline/description/keywords): ")

print("\nChoose a model for prediction:")
print("1. Logistic Regression")
print("2. Naive Bayes")
print("3. SVM")

choice = input("Enter 1, 2, or 3: ")

model_map = {
    "1": "Logistic Regression",
    "2": "Naive Bayes",
    "3": "SVM"
}

selected_model = model_map.get(choice)

if selected_model:
    predict_category(user_text, selected_model)
else:
    print("Invalid model selection.")
```

```
Enter a news article to classify:
Enter text (headline/description/keywords): Amazon reports record-
breaking quarterly revenue driven by strong e-commerce sales and
growth in its cloud computing division, AWS. Analysts project
continued expansion as the company invests heavily in logistics and AI
infrastructure.

Choose a model for prediction:
1. Logistic Regression
2. Naive Bayes
3. SVM
Enter 1, 2, or 3: 3

Predicted Category: BUSINESS
```