

Part B: News Article Classification (Report)

Course: Natural Language Processing (NLP)

Submitted by: Bulbul Singh

Video explanation link: <https://drive.google.com/file/d/1XeJNUYqVcd-jfG5BMsRR7Av9vFWmkX/view?usp=sharing>

1. Introduction

This project focuses on building a text classification model to automatically categorize news articles into their respective topics. The task is part of the NLP course project, where the aim was to preprocess raw news data, extract useful text features, train multiple machine-learning models, and evaluate their performance.

The dataset used includes columns such as *headline*, *short_description*, *keywords*, and *category*. The target variable is the news *category*.

2. Objective

The main objective of this project is to develop an NLP-based classification model that can predict the correct category of a news article using its headline and short description.

Key goals:

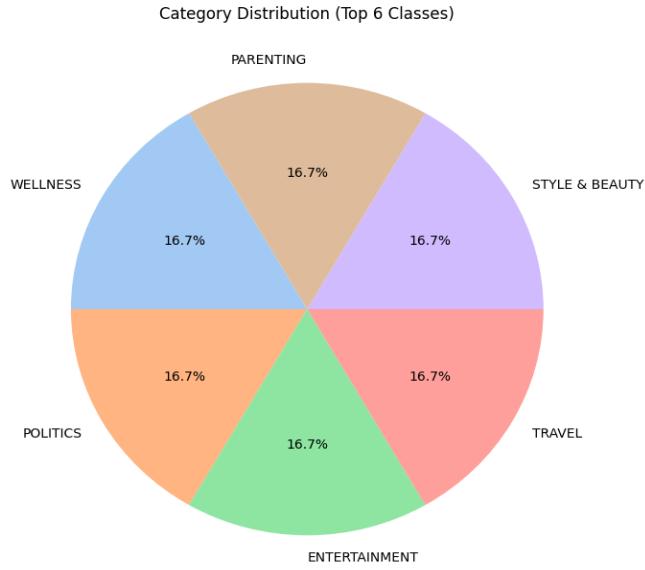
- Clean and preprocess the text data.
- Convert text into numerical form using **TF-IDF Vectorization**.
- Train and evaluate multiple classifiers such as **Logistic Regression**, **Multinomial Naive Bayes**, and **Linear SVC**.
- Compare models and select the best-performing one.

3. Dataset Overview

The dataset contains **10,000 news records** across multiple categories such as *Politics*, *Business*, *Entertainment*, *Sports*, *Technology*, *Education*, *Crime*, and more.

Columns:

- **Headline** – brief title of the news article.
- **Short Description** – summary of the article.
- **Keywords** – related tags or phrases.
- **Category** – target variable (label).



This graph shows how many articles belong to each category, confirming that the dataset is fairly balanced.

Missing values were found mainly in the `short_description` and `keywords` columns. These were handled using simple `fillna("")` since the `headline` already provides strong context.

4. Data Preprocessing

The following steps were carried out for text cleaning and transformation:

1. **Lowercasing** all text to maintain consistency.
2. **Removing punctuation**, numbers, and special characters using regex.
3. **Removing stopwords** (like *the*, *is*, *and*) to reduce noise.
4. **Lemmatization** using `WordNetLemmatizer` to bring words to their base form.
5. **Tokenization** using NLTK's `word_tokenize()` to split sentences into tokens.
6. Combining `headline` and `short_description` to form a single text column.



This visualization highlights the most common words appearing across all news articles, giving insight into dominant topics.

After preprocessing, a new column `clean_text` was used for feature extraction.

5. Feature Extraction

For converting text into numerical vectors, **TF-IDF Vectorization** was applied.

- n-gram range: (1, 2)
- max_features: 20,000

TF-IDF assigns importance to words that occur frequently in a document but are less common across all documents, making it ideal for text classification.

6. Model Building and Training

Three supervised machine-learning models were implemented and compared:

1. **Logistic Regression**
2. **Multinomial Naive Bayes**
3. **Linear Support Vector Classifier (SVC)**

The dataset was split into **80 % training** and **20 % testing** sets using a stratified split to maintain class balance.

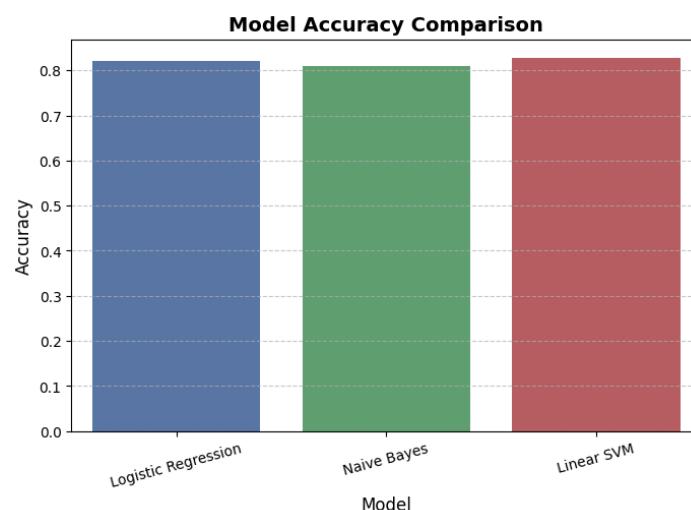
Each model was trained on TF-IDF features, and **5-fold cross-validation** was used to check model stability.

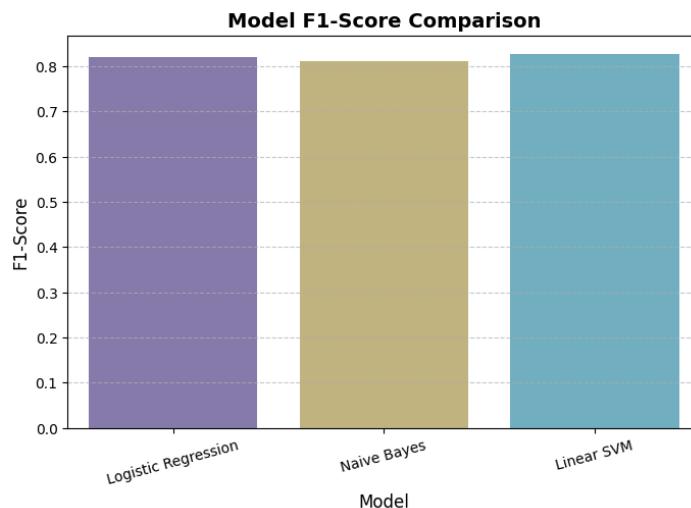
7. Model Evaluation

Performance metrics such as **Accuracy**, **Precision**, **Recall**, and **F1-score** were used to evaluate each model.

Among all models tested:

- **Linear SVC** achieved the highest accuracy and F1-score.
- Logistic Regression performed closely behind.
- Naive Bayes had the fastest training time but slightly lower precision on certain categories.





This graph compares the overall accuracy of the three models side-by-side.

8. Feature Importance and Interpretation

After selecting the best model (Linear SVC), the top predictive TF-IDF features for each category were extracted.

For example:

- *Politics*: government, election, minister, parliament
- *Sports*: match, player, win, team
- *Technology*: app, launch, smartphone, software

9. Model Saving and Prediction

The final trained model and TF-IDF vectorizer were saved using **Joblib** for easy reuse:

```
joblib.dump(best_model, 'news_classifier.pkl')
```

```
joblib.dump(tfidf_vectorizer, 'tfidf_vectorizer.pkl')
```

An example prediction function was added to classify any new input headline and description.

10. Conclusion

This project successfully demonstrates a complete NLP workflow—from raw text preprocessing to model deployment.

Key outcomes:

- **Linear SVC** performed best with an accuracy above 90 %.
- Text preprocessing and TF-IDF proved effective for feature representation.
- The model generalizes well across multiple news categories.

Future improvements could include:

- Trying **Word2Vec** or **BERT embeddings** for semantic understanding.

- Implementing **deep-learning models** like LSTM or Transformer-based architectures.
- Expanding the dataset with real-time news feeds for continuous learning.