# VISVESVARAYATECHNOLOGICALUNIVERSITY

### JNANA SANGAMA, BELAGAVI-590018

**MINIPROJECT REPORT on**

**“ NEWS CLASSIFICATION USING NLP ”**

*Submitted in partial fulfillment for the award of degree of Bachelor Of Computer Science and Engineering (Data Science) during the year 2024-25*

##### Submitted by AISHWARYA D(4MH22CD003)

**MEGHANA A S (4MH22CD026)**

##### Under the Guidence of

**SHALINI M C**

Asst. Profesor

Dept. of CSE(DataScience),

MIT Mysore



##### 2024-25

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (DATASCIENCE), MAHARAJAINSTITUTE OF TECHNOLOGY MYSORE BELAWADI, SRIRANGAPATNA**

**TALUK, MANDYA-571477**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERIN (DATASCIENCE)**

### MAHARAJA INSTITUTE OF TECHNOLOGY MYSORE MANDYA-571477

**CERTIFICATE**

This is to certify that the miniproject report **entitled “-News Classification using NLP”** has been successfully carried out by **Aishwarya.D [4MH22CD003]**, **Meghana.A.S [4MH22CD026],** bonafide student of **Maharaja Institute of Technology Mysore** in partial fulfilment of requirements of**Degree of Bachelor of Engineering in Computer Science and Engineering (DataScience) ofVisveswaraya Technological University, Belagavi** during the academic year 2024-2025. It is certified that all corrections/suggestions indicated for the internal assessment have been incorporated in the report deposited in the department library.

Signature of Guide Signature of HOD

**Shalini M C Dr. Pushpa D**

Assistant Professor Professor & Head

Dept. of CSE(DS), MITM Dept. of CSE(DS), MITM

**ACKNOWLEDEGEMENT**

We sincerely owe our gratitude to all the people who helped and guide completing this mini project work.

We are thankful to **Dr. B.G. Naresh Kumar, Principal, Maharaja Institute of Technology Mysore**, for having supported us in our academic endeavors.

We are extremely thankful to **Dr. Pushpa D, Professor & Head, Department of Computer Science and Engineering (DataScience)** for his valuable support and his timely inquiries into the progress of the work.

We are obliged to all **teaching and non-teaching staff members** of the **Department of Computer Science and Engineering (DataScience)** for the valuable information provided by them in their respective field’s. We are grateful for their co-operation during the period of our mini project.

#### Aishwarya.D (4MH22CD003) Meghana.A.S (4MH22CD026)

**ABSTRACT**

In the era of information overload, accurately classifying news articles has become critical for combating misinformation and enhancing user experience. This project explores the development of an advanced Natural Language Processing (NLP)-based system for news classification, the system aims to deliver high precision and scalability in handling diverse datasets. News categorization is the process that humans are interested one from online news articles. But the readers have the difficulty if the news is not in a categorized order.

For news category classification, a new model is proposed.

By training on diverse datasets, it ensures adaptability to various languages and contexts, making it suitable for global applications. the project aims to empower users with tools to access reliable and organized information while contributing the advancements in multilingual and context aware NLP technologies.

The project "News Classification Using Natural Language Processing" focuses on automatically categorizing news articles into topics such as sports, business, politics, and more. By leveraging Natural Language Processing (NLP) techniques and machine learning algorithms, the system will classify news articles based on their content. The BBC News Classification dataset will be used for model training, showcasing the role of AI in improving news organization and retrieval.

**TABLE OF CONTENT**

1. INTRODUCTION 1
   1. [Overview 1](#_TOC_250014)
   2. [Problem Statement 2](#_TOC_250013)
   3. [The Solution 2](#_TOC_250012)
   4. [Proposed System 2-3](#_TOC_250011)
2. LITERATURE SURVEY 4
   1. [Survey Papers 4-7](#_TOC_250010)
   2. [Survey Findings 8](#_TOC_250009)
3. SOFTWARE REQUIREMENT SPECIFICATION 9
   1. [Topic 9](#_TOC_250008)
   2. [Functional Requirements 9-10](#_TOC_250007)
   3. [Non-Functional Requirements 10](#_TOC_250006)
   4. [System Design Constraints 10](#_TOC_250005)
   5. [Software And Hardware Specification 10-11](#_TOC_250004)
   6. [Limitations 11](#_TOC_250003)
   7. [Applications 11](#_TOC_250002)
4. IMPLEMENTATION 12
   1. [Methodology 12-14](#_TOC_250001)
   2. [Pseudo Code 15-33](#_TOC_250000)
5. SNAPSHOTS 34-35
6. CONCLUSION 36
7. REFERENCES 37

**CHAPTER 1**

# INTRODUCTION

## Overview

In the digital age, the volume of news published daily has made it challenging to manage and assess the authenticity and relevance of information. News classification, powered by Natural Language Processing (NLP), offers an efficient solution to categorize, organize, and analyze news content. This process involves using computational techniques to automatically assign news articles to predefined categories, such as politics, sports, entertainment, or technology.

With the rise of multilingual content and misinformation, the need for advanced and adaptable systems has grown significantly. Traditional classification methods often struggle with the complexities of natural language, such as nuances, context, and cross-lingual variations. To address these challenges, modern AI techniques, including deep learning and transformer- based models like BERT, have become integral to building powerful NLP systems. These models enable precise classification, even in multilingual scenarios, while supporting additional features like sentiment analysis and fake news detection.

This project focuses on developing an intelligent and scalable NLP-based news classification system. It leverages diverse datasets to train the system for global applicability, ensuring it performs well in multiple languages and domains. Additionally, the project incorporates fake news detection mechanisms to combat the growing issue of misinformation on digital platforms. By improving information retrieval and delivering well-organized content, this system aims to empower users with reliable and relevant news while addressing critical challenges in today’s media landscape.

## Problem statement

With the rapid increase in online news articles across various domains, manually categorizing these articles into topics is not only time-consuming but also prone to human error. Existing methods lack efficiency and scalability. This project addresses the challenge of developing an automated news classification system to accurately categorize news articles using advanced Natural Language Processing (NLP) and Machine Learning (ML) techniques.

## The Solution

Existing methods lack efficiency and scalability, To address these challenges, this project proposes the development of an advanced NLP-based news classification system. The system leverages state-of-the-art AI techniques, including deep learning and transformer models, to classify news articles into predefined categories with high precision. By training the system on diverse multilingual datasets, it ensures adaptability to various languages and contexts, making it suitable for global applications.

By addressing the complexities of multilingual content and combating misinformation, this project provides a comprehensive framework for tackling the critical challenges of the modern media landscape.

## Proposed System

The proposed system is a NLP-Based News Classification System that uses advanced techniques to classify news articles in multiple languages. It improves over existing systems by focusing on accuracy, efficiency, and multilingual support. The system works in the following steps:

1. Data Collection and Preprocessing:

Collect news articles in different languages from reliable sources.Detect the language of each article using automatic tools and clean the text by removing unnecessary symbols, special characters, and duplicates.

1. Model Design:

Used a pretrained multilingual model like CNN,RNN to process text in various languages. These models understand multiple languages, so there is no need to create a separate model for each one.

Fine-tune this model to make it perform better specifically for classifying news into categories like politics, sports, and entertainment.

1. Evaluation

Testing the system to check its accuracy in identifying the correct category of news articles.

Advantages Over Existing Systems

1. Better Performance: Uses state-of-the-art AI models for higher accuracy in news classification.
2. Data Efficiency: Works well even when there is less data available.
3. Scalable and Flexible: Easily adaptable to more languages or categories in the future.

**CHAPTER 2**

# LITERATURE SURVEY

## Survey Papers

### PAPER 1

**TITLE :** News Category Classification using Natural Language Processing Transformer

**AUTHORS :** Parvathavarthini S, Shreekanth M, Vignesh Kumar S

**YEAR OF PUBLICATION :** 2023(IEEE)

**EXPLANATION :** This work proposes a NLP transformer developed by the pre-trained model of BERT sequence classification. The provided method was used to classify the news into the categories. The suggested method was tested by giving the categorized news from the different articles. Based on the metrics calculated, this method is successful in categorising the given news and achieves high accuracy. However, significant drawbacks are present in suggested techniques, such as cannot handle the limited input sequences efficiently. This is due to the fine-tuning of the model.

### PAPER 2

**TITLE :** NLP based Text Summarization Techniques for News Articles

**AUTHORS :** B.Logesh Kumaran, N.V. Ravindran

**YEAR OF PUBLICATION :** 2024 (IEEE)

### EXPLANATION :

This paper contains implementation using the Streamlit library, offers a robust and interactive platform for accessing and summarizing news articles. The functionality is divided into distinct

features, including fetching news from Google News based on different categories such as trending news, favorite topics, and user-specified search queries. The user interface is thoughtfully designed with a three-column layout, featuring an image to enhance visual appeal. Error handling is seamlessly integrated to address potential issues during data retrieval and parsing, ensuring a smooth user experience. News articles are displayed with essential information, including titles, summaries, dates.

### PAPER 3

**TITLE :** Multilabel News Category Classification using Machine Learning

**AUTHORS :** Roshan Kumar Shah, Sahil Kumar, Shashank **YEAR OF PUBLICATION :**2023(IEEE) **EXPLANATION :**

In this paper, multilabel news classification was introduced by utilizing three PTAs called ”Binary Relevance, Label Powerset Classifier Chain, ”two TV called count vectorizer and TF-IDF,” and three machine learning (ML)models for classification called Logistic Regression(LR). In summary, it may be stated that the optimum result was obtained by using the support vector classifier (SVC) as a machine learning model, label powerset (LP) as a problem transformation technique, and TF-

### PAPER 4

**TITLE :** A Natural Language Processing System for Truth Detection and Text Summarization

**AUTHORS :** Rohith H P, Kavitha Sooda, Karunakara Rai B, Srinivas D B

**YEAR OF PUBICATION :** 2023(IEEE)

**EXPLANATION :** In the proposed model, the embedding feature vector value = 40, which is the target feature vector for the embedding layer. A single LSTM Layer with 100 nodes is employed. Since this is a binary classification challenge, a dense layer with one neuron and a sigmoid function is utilized. The dropout approach is employed to minimize overfitting, and the Adam optimizer is applied to tune the loss function. The current truth detection system detects truth only in a trained subject matter. To provide insight in another subject matter we need to train another model. To streamline this process it is better to add more subject matters to the overall corpus. The current model of summarization is limited to two-line extractive.

### PAPER 5

**TITLE :** News Classification using Natural Language Processing **AUTHORS** : K. Yasaswi, Vijaya Kumar Kambala, P. Sai Pavan, V. Jasmika **YEAR OF PUBICATION :**2022(IEEE)

**EXPLANATION:** This paper uses a natural language processing to automate the process of classification of news to genuine and fake along with relevant supervised machine learning algorithms. They have used a limited number of articles around 6335. Each figure represent how accurate the model predictions are based upon the metrics. As a result, from the above matrices and tables we can conclude that Passive Aggressive Classification is the most efficient for classifying small sized articles and produced very much correct result and LR is the next best method for this purpose with a very slight variation in producing accurate result.

### PAPER 6

**TITLE :** A Study on Natural Language Processing Classified News **AUTHORS** : Meng-Jin Wu, Tzu-Yuan Fu, Chia-Wei Lee, Yao-Chung Chang **YEAR OF PUBICATION :**2020(ICAN)

**EXPLANATION:** In this paper, we used the web crawler technology, Jieba word segmentation for training the computer. The experimental result demonstrated that the accuracy of the news classification is 97.43%. This will help journalist to reduce the consumption of Labor. In social networks, it will analyze which categories are the readers focus on. Then, it could recommend the news for users precisely.7

## Survey Findings

* Limitations of Traditional Models: Simple models like Logistic Regression and Naive Bayes struggle to capture complex language patterns and context, limiting their effectiveness.
* Role of Text Preprocessing: Techniques like tokenization, stemming, and TF-IDF improve data quality, which enhances model performance.
* Scalability Issues: Traditional models have difficulty handling large datasets or unstructured text, limiting their practical use for big data applications.
* Feature Engineering: Methods like TF-IDF are important for extracting relevant features that improve classification accuracy.
* Need for Advanced Models: Advanced models like BERT, CNN, and RNN are needed for handling complex and large-scale news classification tasks.

**CHAPTER 3**

# SOFTWARE REQUIREMENT SPECIFICATION

## Topic

A Software Requirements Specification (SRS) in the report is a comprehensive document that outlines the functional and non-functional requirements for the software system that we are developing. In the context of our news classification project using NLP, the SRS serves as a blueprint for the project, ensuring that all stakeholders have a clear understanding of the system's objectives, capabilities, and constraints.

**Purpose:** NLP-based news classification system (e.g., to classify news articles).

**Scope: .**The project focuses on news articles from the BBC News Classification dataset, allowing for diverse topics.

**.**It will include both traditional machine learning models and deep learning architectures to evaluate performance across different approaches.

**.**The project aims to achieve a classification accuracy of over 95%, making it suitable for real- world applications in automated news processing and information retrieval.

## Functional Requirements

Defining specific tasks the system should perform:

* + - Accept input news articles in multiple languages.
    - Perform preprocessing (e.g., tokenization, stopword removal).
    - Classify articles into categories (e.g., politics, sports, fake news).

## . 3.3 Non-Functional Requirements

* + - **Performance:** Specify speed and accuracy goals for the model.
    - **Scalability:** The ability to handle large datasets.
    - **Usability:** Ensure ease of use for end users.
    - **Security:** Protect against misuse or tampering.

## System Design Constraints

* + - Tools and frameworks (e.g., Python, numpy etc).
    - Dataset requirements for training and testing.
    - Integration with external APIs for multilingual support.

## Software And Hardware Specification

#### Software:

* + - Python (with libraries such as pandas, NumPy, sci-kit-learn)
    - Jupyter Notebook or any preferred Python IDE for development and testing
    - Data Visualization Libraries (Matplotlib, Seaborn) for presenting results effectively

#### Hardware:

A computer with sufficient RAM and processing power to handle machine learning tasks**.**

## Limitations:

1. **Ambiguity:** NLP models may misinterpret words or phrases with multiple meanings, leading to inaccurate classification.
2. **Complex Articles:** Handling long or multi-topic articles can be difficult for models with limited token processing capacity.
3. **Domain-Specific Language**: General NLP models may struggle with jargon or specialized terms in specific domains

## Applications

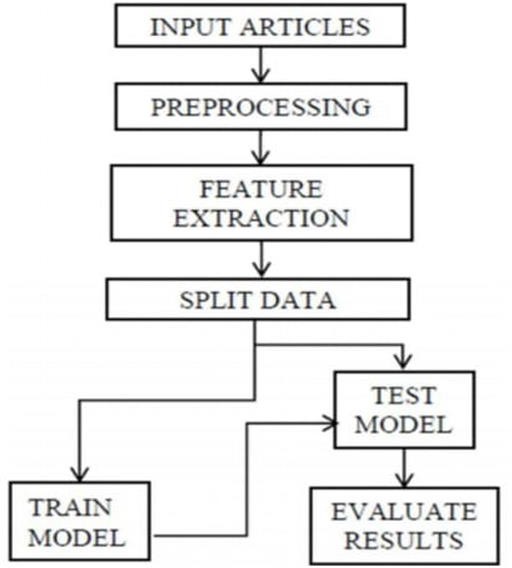
* + - Personalized news aggregation.
    - Automated categorization for news organizations.
    - Trend analysis and sentiment assessment.
    - Real-time monitoring and alerts for breaking news.
    - Integration with chatbots and educational tools.

**CHAPTER 4**

#### Methodology

# IMPLEMENTATION

The methodology of the project involves the following steps:



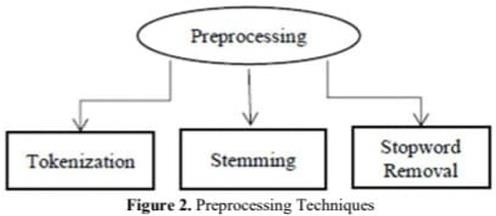
##### Fig: Methodology used in NLP

1. **Data Preprocessing:**

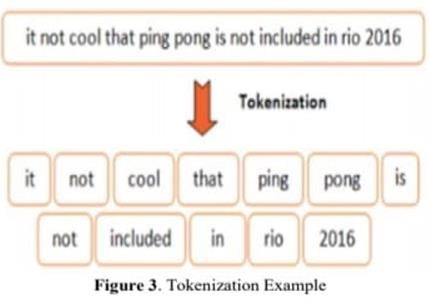
This step ensures the quality of the input data, which is crucial for effective modeling.

##### Cleaning Data:

Removing unwanted characters, HTML tags, and punctuation to produce clean text.



#### Tokenization :



* Dividing the text into smaller units (tokens), such as words, which helps in understanding the structure of the language.
* **Stop Word Removal:** Filtering out common words that carry little meaning, such as "the" and "is," to reduce noise in the dataset.
* **Stemming:** Reducing words to their root form (e.g., "running" to "run"), which helps in consolidating variations of the same word.
* **Vectorization:** Applying TF-IDF (Term Frequency-Inverse Document Frequency) to convert text data into numerical format suitable for model training.

#### Building ML Models:

* + **Convolutional Neural Networks (CNN):** Highly effective for image data, CNNs can also be applied to text data by capturing local patterns in the text.
  + **Recurrent Neural Networks(RNN):** RNNs are applied to understand the context and meaning of news articles, enabling them to classify content effectively.

1. **Model Evaluation:** After training, the models will be evaluated using performance metrics such as accuracy, precision, recall, and F1-score to ensure their effectiveness in real-world scenarios.

#### Pseudo Code

import pandas as pd import seaborn as sns import numpy as np import pickle

import torch

import torch.nn as nn import torch.optim as optim import re

import matplotlib.pyplot as plt import nltk

import joblib import requests

from wordcloud import WordCloud ,STOPWORDS

from sklearn.feature\_extraction.text import TfidfVectorizer from sklearn.model\_selection import train\_test\_split

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

from sklearn.metrics import accuracy\_score, classification\_report from torch.utils.data import DataLoader, TensorDataset

from sklearn.preprocessing import LabelEncoder from transformers import Trainer, TrainingArguments

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

from nltk.tokenize import word\_tokenize from bs4 import BeautifulSoup

!pip install torch scikit-learn pandas numpy

!pip install torch transformers scikit-learn

!pip install --user torch scikit-learn

!pip install transformers torch scikit-learn pandas

!pip install nltk

!pip install requests beautifulsoup4 nltk.download('wordnet') # For lemmatization nltk.download('stopwords') # For stopword removal nltk.download('punkt') # For tokenization

nltk.download('omw-1.4') # This will download the missing resource import warnings

warnings.filterwarnings("ignore”)

data = pd.read\_csv("BBC News.csv") data.head()

data['Category'].unique() data.shape

data.dtypes data.isnull().any()

data['News\_length'] = data['Text'].str.len() print(data['News\_length']) sns.histplot(data['News\_length']).set\_title('News length distribution');

nltk.download('wordnet') # For lemmatization (WordNet lexical database) nltk.download('stopwords') # For stopword removal nltk.download('punkt') # For tokenization (word tokenizer)

nltk.download('omw-1.4') # For additional resources required by lemmatizer

def clean\_text(text):

# Initialize NLTK tools

stop\_words = set(stopwords.words('english')) stemmer = PorterStemmer()

lemmatizer = WordNetLemmatizer()

# Convert text to lowercase text\_lower = text.lower() # Remove punctuation

text\_no\_punctuation = re.sub(r'[^\w\s]', '', text\_lower) # Tokenize the text

tokens = word\_tokenize(text\_no\_punctuation) # Remove stopwords

tokens\_no\_stopwords = [word for word in tokens if word not in stop\_words] # Apply stemming (optional)

tokens\_stemmed = [stemmer.stem(word) for word in tokens\_no\_stopwords] # Apply lemmatization

tokens\_lemmatized = [lemmatizer.lemmatize(word) for word in tokens\_stemmed] # Join tokens back into a string

clean\_text = ' '.join(tokens\_lemmatized) return clean\_text

# Load your dataset

df = pd.read\_csv('C:\\Users\\AISHWARYA D\\Downloads\\BBC News.csv') # Replace with your actual file path

# Apply the cleaning function to the 'Text' column df['Cleaned\_Text'] = df['Text'].apply(clean\_text)

# Show the first few rows of the cleaned dataset print(df[['Text', 'Cleaned\_Text']].head())

def create\_wordcloud(words):

wordcloud = WordCloud(width=800, height=500, random\_state=21, max\_font\_size=110).generate(words)

plt.figure(figsize=(10, 7))

plt.imshow(wordcloud, interpolation="bilinear") plt.axis('off')

plt.show()def create\_wordcloud(words):

wordcloud = WordCloud(width=800, height=500, random\_state=21, max\_font\_size=110).generate(words)

plt.figure(figsize=(10, 7)) plt.imshow(wordcloud, interpolation="bilinear") plt.axis('off')

plt.show()

subset=data[data.Category=="entertainment"] text=subset.Text.values

words =" ".join(text) create\_wordcloud(words)

subset=data[data.Category=="politics"] text=subset.Text.values

words =" ".join(text) create\_wordcloud(words)

subset=data[data.Category=="sport"] text=subset.Text.values

words =" ".join(text) create\_wordcloud(words)

subset=data[data.Category=="tech"] text=subset.Text.values

words =" ".join(text) create\_wordcloud(words)

def process\_text(text):

text = text.lower().replace('\n',' ').replace('\r','').strip() text = re.sub(' +', ' ', text)

text = re.sub(r'[^\w\s]','',text)

stop\_words = set(stopwords.words('english')) word\_tokens = word\_tokenize(text)

filtered\_sentence = [w for w in word\_tokens if not w in stop\_words] filtered\_sentence = []

for w in word\_tokens:

if w not in stop\_words: filtered\_sentence.append(w) text = " ".join(filtered\_sentence) return text

data['Text\_parsed'] = data['Text'].apply(process\_text) data.head()

from sklearn import preprocessing label\_encoder = preprocessing.LabelEncoder()

data['Category\_target']= label\_encoder.fit\_transform(data['Category']) data.head()

data.to\_csv('BBC\_News\_processed.csv')

X\_train, X\_test, y\_train, y\_test = train\_test\_split(data['Text\_parsed'], data['Category\_target'],

test\_size=0.2, random\_state=8)

X\_train.shapedata.to\_csv('BBC\_News\_processed.csv')

X\_train, X\_test, y\_train, y\_test = train\_test\_split(data['Text\_parsed'],

data['Category\_target'], test\_size=0.2, random\_state=8)

X\_train.shapedata.to\_csv('BBC\_News\_processed.csv')

X\_train, X\_test, y\_train, y\_test = train\_test\_split(data['Text\_parsed'],

data['Category\_target'], test\_size=0.2, random\_state=8

X\_train.shape X\_test.shape

ngram\_range = (1,2)

min\_df = 10

max\_df = 1.

max\_features = 300

tfidf = TfidfVectorizer(encoding='utf-8',

ngram\_range=ngram\_range, stop\_words=None, lowercase=False, max\_df=max\_df,

min\_df=min\_df,

max\_features=max\_features, norm='l2', sublinear\_tf=True)

features\_train = tfidf.fit\_transform(X\_train).toarray() labels\_train = y\_train

print(features\_train)

features\_test = tfidf.transform(X\_test).toarray()

labels\_test = y\_test print(features\_test.shape)ngram\_range = (1,2)

min\_df = 10

max\_df = 1.

max\_features = 300

tfidf = TfidfVectorizer(encoding='utf-8',

ngram\_range=ngram\_range, stop\_words=None, lowercase=False, max\_df=max\_df, min\_df=min\_df, max\_features=max\_features, norm='l2', sublinear\_tf=True)

features\_train = tfidf.fit\_transform(X\_train).toarray() labels\_train = y\_train

print(features\_train)

features\_test = tfidf.transform(X\_test).toarray() labels\_test = y\_test

print(features\_test.shape)

# Step 1: Define the CNN Model class CNNClassifier(nn.Module):

def \_init\_(self, input\_dim, num\_classes): super(CNNClassifier, self).\_init\_()

self.conv1 = nn.Conv1d(in\_channels=1, out\_channels=128, kernel\_size=3) self.pool = nn.MaxPool1d(kernel\_size=2)

self.fc1 = nn.Linear(128 \* 149, num\_classes) # Adjust based on input size after convolution and pooling

def forward(self, x):

x = self.pool(torch.relu(self.conv1(x))) # Apply conv and pooling x = x.view(x.size(0), -1) # Flatten the output from conv layer

x = self.fc1(x) # Fully connected layer return x

# Step 2: Prepare Data

# Convert features to torch tensors

X\_train\_tensor = torch.tensor(features\_train, dtype=torch.float32) X\_test\_tensor = torch.tensor(features\_test, dtype=torch.float32) y\_train\_tensor = torch.tensor(labels\_train.values, dtype=torch.long) y\_test\_tensor = torch.tensor(labels\_test.values, dtype=torch.long)

# Reshaping input for Conv1d layer (should be [batch\_size, 1, seq\_len])

X\_train\_tensor = X\_train\_tensor.unsqueeze(1) # Add an extra dimension for channels

X\_test\_tensor = X\_test\_tensor.unsqueeze(1) # Add an extra dimension for channels

# Step 3: Create DataLoader for Training and Testing train\_dataset = TensorDataset(X\_train\_tensor, y\_train\_tensor) test\_dataset = TensorDataset(X\_test\_tensor, y\_test\_tensor)

train\_loader = DataLoader(train\_dataset, batch\_size=64, shuffle=True) test\_loader = DataLoader(test\_dataset, batch\_size=64, shuffle=False) # Step 4: Initialize Model, Loss Function, and Optimizer

input\_dim = X\_train\_tensor.shape[2] # 300 from TF-IDF features

num\_classes = len(np.unique(labels\_train)) # Number of output classes (5 categories)

cnn\_model = CNNClassifier(input\_dim=input\_dim, num\_classes=num\_classes) criterion = nn.CrossEntropyLoss() # Suitable loss for multi-class classification

optimizer = optim.Adam(cnn\_model.parameters(), lr=0.001) # Step 5: Train the Model

num\_epochs = 10# Number of epochs to train for epoch in range(num\_epochs):

cnn\_model.train() running\_loss = 0.0

for inputs, labels in train\_loader: optimizer.zero\_grad()

outputs = cnn\_model(inputs) loss = criterion(outputs, labels) loss.backward() optimizer.step()

running\_loss += loss.item()

print(f"Epoch [{epoch+1}/{num\_epochs}], Loss: {running\_loss/len(train\_loader):.4f}")

# Step 6: Evaluate the Model cnn\_model.eval() predictions = []

true\_labels = []

with torch.no\_grad():

for inputs, labels in test\_loader:

outputs = cnn\_model(inputs)

\_, predicted = torch.max(outputs, 1) predictions.extend(predicted.numpy()) true\_labels.extend(labels.numpy())

# Calculate Accuracy

accuracy = accuracy\_score(true\_labels, predictions) print(f"Accuracy: {accuracy:.4f}")

# Print the Classification Report print("Classification Report:") print(classification\_report(true\_labels, predictions)) # Load dataset and prepare data

df = pd.read\_csv("C:\\Users\\AISHWARYA D\\Downloads\\BBC News.csv") # Replace with your dataset path

X = TfidfVectorizer(max\_features=300).fit\_transform(df['Text']).toarray() y = LabelEncoder().fit\_transform(df['Category'])

In [36]: # Split data into train and test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Convert to PyTorch tensors

X\_train\_tensor = torch.tensor(X\_train, dtype=torch.float32) X\_test\_tensor = torch.tensor(X\_test, dtype=torch.float32) y\_train\_tensor = torch.tensor(y\_train, dtype=torch.long) y\_test\_tensor = torch.tensor(y\_test, dtype=torch.long)

# Reshape for RNN (add an extra dimension for sequence length)

X\_train\_tensor = X\_train\_tensor.unsqueeze(1) # Shape: [batch\_size, 1, input\_dim]

X\_test\_tensor = X\_test\_tensor.unsqueeze(1) # Shape: [batch\_size, 1, input\_dim] # Create DataLoader

train\_dataset = TensorDataset(X\_train\_tensor, y\_train\_tensor) test\_dataset = TensorDataset(X\_test\_tensor, y\_test\_tensor)

In [40]: train\_loader = DataLoader(train\_dataset, batch\_size=64, shuffle=True) test\_loader = DataLoader(test\_dataset, batch\_size=64, shuffle=False)

# Define RNN model

class RNNClassifier(nn.Module):

def \_init\_(self, input\_dim, hidden\_dim, num\_classes): super(RNNClassifier, self).\_init\_() self.hidden\_dim = hidden\_dim

self.rnn = nn.RNN(input\_dim, hidden\_dim, batch\_first=True) # Simple RNN

self.fc = nn.Linear(hidden\_dim, num\_classes) def forward(self, x):

h0 = torch.zeros(1, x.size(0), self.hidden\_dim).to(x.device) # Initialize hidden state

out, \_ = self.rnn(x, h0) # RNN forward pass

out = out[:, -1, :] # Only take the output of the last time step out = self.fc(out) # Fully connected layer

return out

# Model parameters

input\_dim = X\_train\_tensor.shape[2] # 300 features from TF-IDF hidden\_dim = 128

num\_classes = len(np.unique(y)) # Initialize the model

model = RNNClassifier(input\_dim, hidden\_dim, num\_classes) # Loss function and optimizer

criterion = nn.CrossEntropyLoss()

optimizer = optim.Adam(model.parameters(), lr=0.001) # Training loop

num\_epochs = 10

device = torch.device('cuda' if torch.cuda.is\_available() else 'cpu') # Use GPU if available

model.to(device)

for epoch in range(num\_epochs): model.train()

running\_loss = 0.0

for inputs, labels in train\_loader:

inputs, labels = inputs.to(device), labels.to(device) optimizer.zero\_grad()

# Forward pass

outputs = model(inputs)

# Compute loss

loss = criterion(outputs, labels)

# Backward pass and optimization loss.backward()

optimizer.step()

running\_loss += loss.item()

print(f"Epoch [{epoch+1}/{num\_epochs}], Loss: {running\_loss/len(train\_loader):.4f}") model.eval()

predictions = [] true\_labels = []

with torch.no\_grad():

for inputs, labels in test\_loader:

inputs, labels = inputs.to(device), labels.to(device) outputs = model(inputs)

\_, predicted = torch.max(outputs, 1) predictions.extend(predicted.cpu().numpy()) true\_labels.extend(labels.cpu().numpy())

# Calculate accuracy

accuracy = accuracy\_score(true\_labels, predictions) print(f"Accuracy: {accuracy:.4f}") print("Classification Report:") print(classification\_report(true\_labels, predictions))

# After training your model, save the trained model torch.save(model.state\_dict(), 'rnn\_model.pth') # Saves in the current directory

# Save the trained model weights to 'rnn\_model.pth' in the current directory torch.save(model.state\_dict(), 'rnn\_model.pth')

print('Model saved to the current directory as rnn\_model.pth')

class RNNClassifier(nn.Module):

def \_init\_(self, input\_dim, hidden\_dim, num\_classes): super(RNNClassifier, self).\_init\_() self.hidden\_dim = hidden\_dim

self.rnn = nn.RNN(input\_dim, hidden\_dim, batch\_first=True) self.fc = nn.Linear(hidden\_dim, num\_classes)

def forward(self, x):

h0 = torch.zeros(1, x.size(0), self.hidden\_dim).to(x.device) out, \_ = self.rnn(x, h0)

out = out[:, -1, :] out = self.fc(out) return out

# Load your model

input\_dim = 300 # TF-IDF vectorizer's output dimensions

hidden\_dim = 128 # Hidden dimension used in training num\_classes = 5 # Adjust based on your dataset categories

# Load model

model = RNNClassifier(input\_dim, hidden\_dim, num\_classes) model.load\_state\_dict(torch.load('rnn\_model.pth')) model.eval()

# Load the previously saved TF-IDF vectorizer with open('tfidf\_vectorizer.pkl', 'rb') as file:

tfidf\_vectorizer = pickle.load(file) def extract\_text\_from\_url(url):

"""Extracts text from a news article URL.""" response = requests.get(url)

soup = BeautifulSoup(response.content, 'html.parser')

# Extract text from <p> tags paragraphs = soup.find\_all('p')

text = ' '.join([para.get\_text() for para in paragraphs]) return text

nltk.download('stopwords') nltk.download('punkt')

# Define stopwords

stop\_words = set(stopwords.words('english')) def preprocess\_text(text):

"""Preprocess text: lowercase, remove punctuation, stopwords, etc.""" text = text.lower() # Convert to lowercase

text = re.sub(r'[^\w\s]', '', text) # Remove punctuation tokens = word\_tokenize(text) # Tokenize

tokens = [word for word in tokens if word not in stop\_words] # Remove stopwords

return ' '.join(tokens)

# Define the categories (update with your dataset labels) category\_names = ['Business', 'Entertainment', 'Politics', 'Sport', 'Tech'] def predict\_category\_from\_url(url):

"""Predicts the category of a news article given its URL.""" # Step 1: Extract text

raw\_text = extract\_text\_from\_url(url)

print("\nExtracted Text:\n", raw\_text[:500]) # Show snippet of the article

# Step 2: Preprocess text

cleaned\_text = preprocess\_text(raw\_text)

# Step 3: Transform text to TF-IDF features

text\_tfidf = tfidf\_vectorizer.transform([cleaned\_text]).toarray()

# Step 4: Convert to PyTorch tensor

text\_tensor = torch.tensor(text\_tfidf, dtype=torch.float32).unsqueeze(0)

# Step 5: Predict using RNN model with torch.no\_grad():

output = model(text\_tensor)

\_, predicted = torch.max(output, 1)

# Step 6: Return the predicted category return category\_names[predicted.item()]

# Example usage

url = "https://dailybharat.org/sports/ms-dhoni-reveals-why-he-batted-at-no-8-in- ipl-2024/" # Replace with a valid article URL

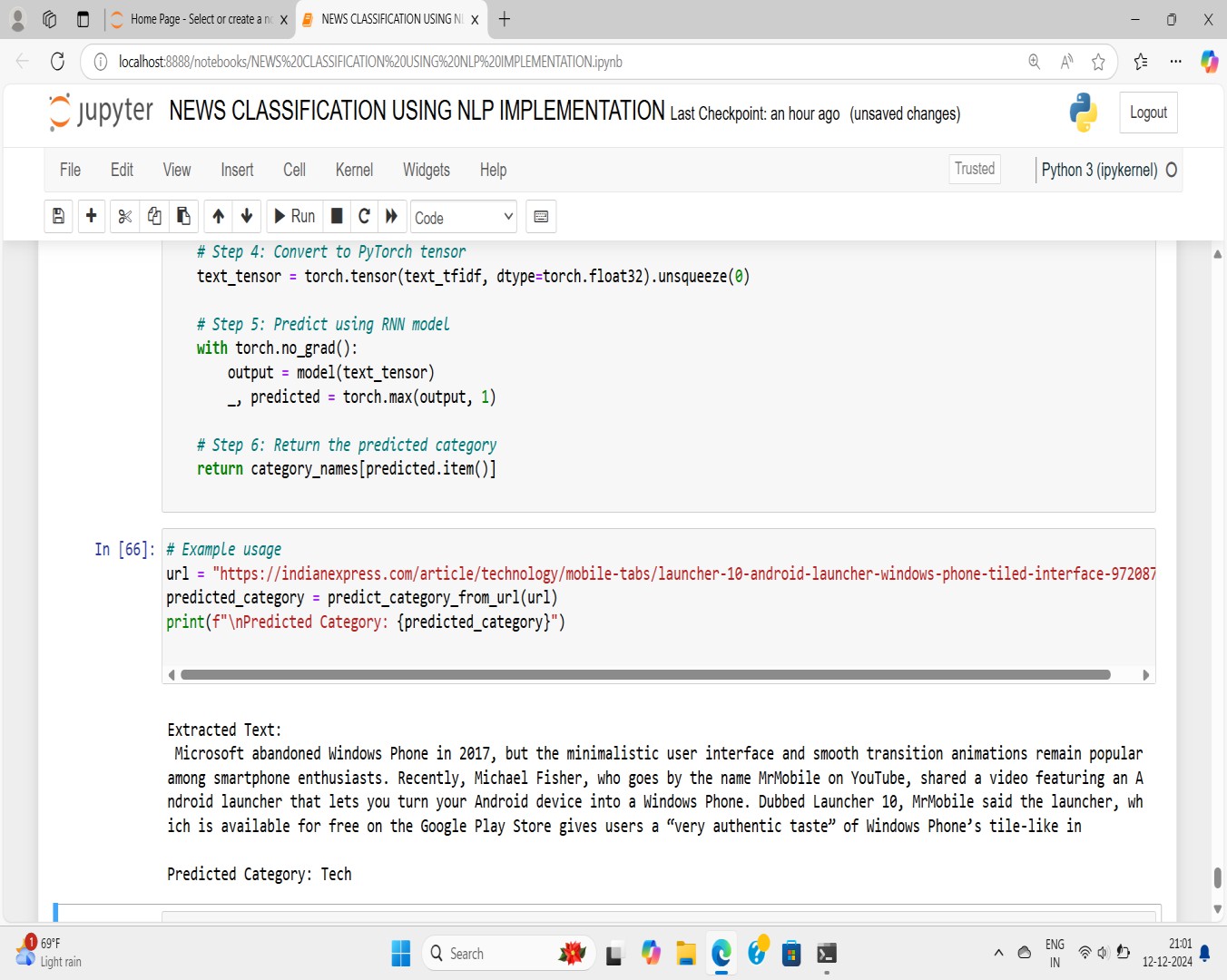
predicted\_category = predict\_category\_from\_url(url) print(f"\nPredicted Category: {predicted\_category}")

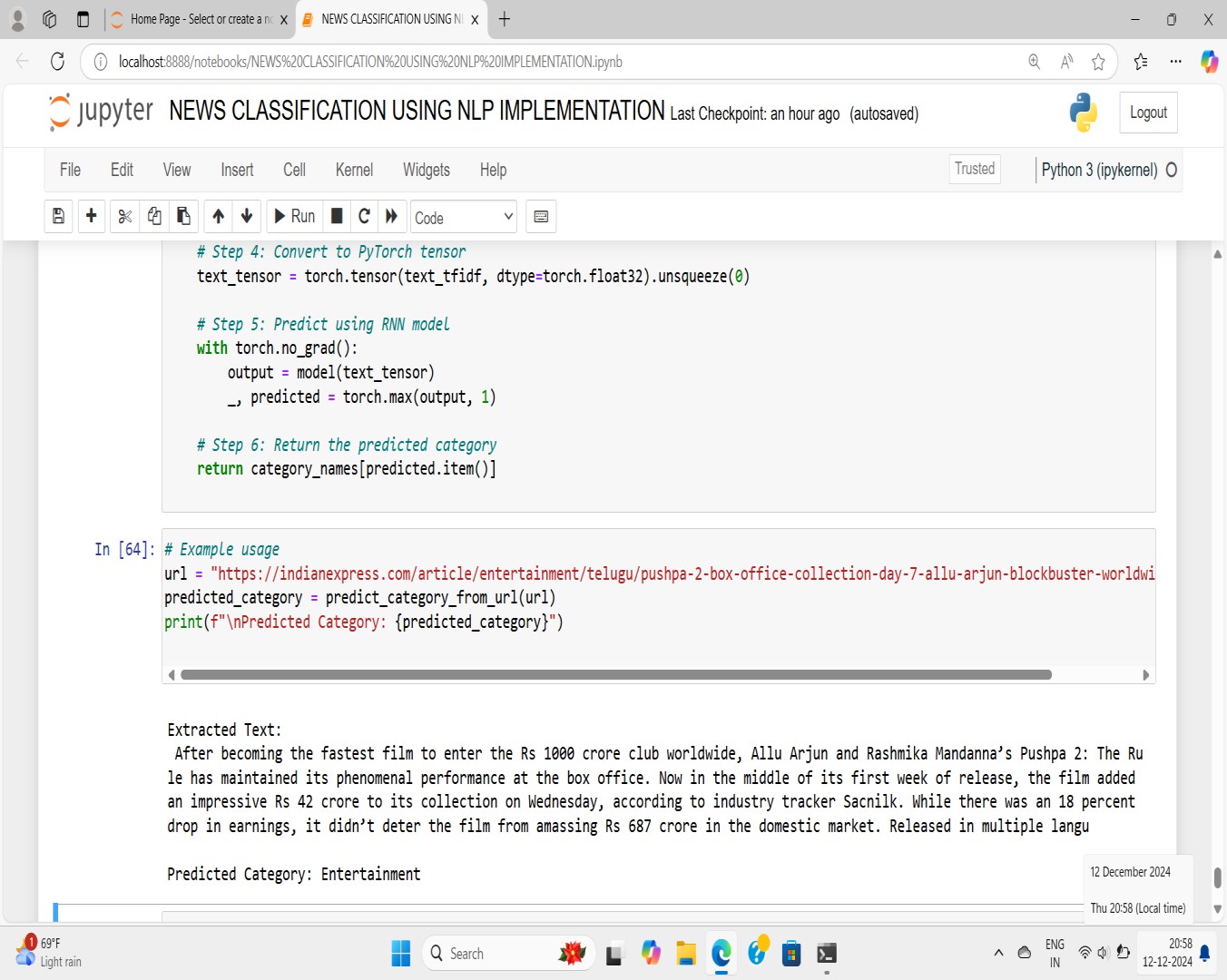
output: Extracted Text: MS Dhoni recently shed light on the selfless reasoning behind his decision to bat lower down the order for the Chennai Super Kings (CSK) in the IPL 2024 season. Positioned at No. 8, Dhoni’s choice surprised fans who expected him to take on a more prominent batting role. Explaining his decision, the former India captain shared that he wanted to provide more chances to CSK players vying for sports in India’s T20 World Cup 2024 squad. Dhoni saw little benefit in promoting himself and preferred to a.

Predicted Category: Sport

**CHAPTER 5**

# SNAPSHOTS





**CONCLUSION**

The NLP-driven News Topic Classifier provides an innovative solution for automating the categorization of news articles using advanced natural language processing and machine learning techniques.

With a user-friendly Back end, this project enhances accessibility and user engagement. The implementation of various models aims to achieve high classification accuracy.

This project not only addresses current challenges in news classification but also opens avenues for future enhancements, making it a valuable tool for users and organizations alike.

**REFERENCES**

1. Ganesh Gopal Devarajan , SeniorMember, MuruganNagarajan,Member,IEEE,Sardar Irfanullah Amanullah,S.A.Sahaaya Arul Mary,and Ali Kashif Bashir,Senior member,IEEE”AI-Assisted Deep NLP\_based Approach for prediction of fake news from social media users” IEEE ,August 2024.
2. Sahana M\*1, Sandarsh Gowda MM\*2,” News Classification using Natural Language Processing”, Volume:04/Issue:06/June-2022 Impact Factor- 6.752 [www.irjmets.com](http://www.irjmets.com/) .
3. RitikPatil, Rohan Patil, Prathamesh Patil,” News Classification usingNatural Language Processing, Volume: 09 Issue: 04 | Apr 2022 [www.irjet.net](http://www.irjet.net/)
4. Zhanhe Bi,” Application of Natural Language Processing Technology in News Event Retrieval and Classification” 2023 International Conference on Evolutionary Algorithms and Soft Computing Techniques (EASCT).
5. K.Yasaswi,Vijaya kumar kambala,M.Sreya,V.Jasmika,”News Classification Using Natural Language Processing”,2023 3rd International Conference on Intelligent Engineering and Management(ICIEM).
6. Roshan Kumar ,Shah,Sahil ,Kumar,Shashank,”Multilabel News Category Classification using Machine Learning”,in access IEEE xplore ICCES 2023.
7. Parvathavarthini S ,Shreekanth M,Vignesh Kumar S ,Santhosh N S ,”News Category Classification using Natural Language Processing Transformer”,in access IEEE Xplore 2023,ICAISS.
8. B.Logesh Kumaran,N.V.Ravindran,”NLP based Text Summarization Techniques for News Articles”,in access 2024 IEEE,(IATMSI)