# News Article Recommendation System

*A project report submitted to ICT Academy of Kerala*

*in partial fulfillment of the requirements*

*for the certification of*

**CERTIFIED SPECIALIST**

**IN**

**ML & AI**

Submitted by

**Nimisha A**

**(Team 6)**

**Members: -Shibla T B, Mhd. Samarin, Kavya M, Vishnu P V**

**ICT ACADEMY OF KERALA**

**THIRUVANANTHAPURAM, KERALA, INDIA**

**Jan 2023**

## List of Figures

## List of Abbreviations

1.TF-IDF : Term Frequency-Inverse Document Frequency

2.NRS : News Recommendation System

## Table of Contents

# Abstract

Recommendation systems are a subclass of machine learning which generally deal with ranking or rating products / users. In other words, a recommender system is a system which predicts ratings a user might give to a specific item. These predictions will then be ranked and returned back to the user. From **Amazon** to **LinkedIn**, **Uber eats** to **Spotify**, **Netflix** to **Facebook**, **Recommender systems** are most extensively used to suggest "Similar items", "Relevant jobs", "preferred foods", "Movies of interest" etc to their users. There are basically two kinds of **recommendation** methods.

1. **Content based recommendation**
2. **Collaborative filtering**

In this project, we are going to discuss the **Content based recommendation** using the News **category** dataset. The dataset contains around 210k news headlines from 2012 to 2022 from HuffPost. This is one of the biggest news datasets and can serve as a benchmark for a variety of computational linguistic tasks. Each record in the dataset consists of attributes like category, headline, authors, link, short_description, date. The goal is to recommend **news articles** which are similar to the already read article by using attributes like article *headline*, *category*, *author* and *publishing date*.

# 1. Problem Definition

## 1.1 Overview

Our project works on a news article dataset collected from Kaggle. The core idea is to recommend items by finding similar items/users to the concerned **item/user** based on their **attributes**. We are going to discuss Content **based recommendation** using **News category** dataset.

## 1.2 Problem Statement

News publishers have decreased disseminating news through conventional newspapers and have migrated to the use of digital means like websites and purpose-built mobile applications. It is observed that news recommendation systems can automatically process lengthy articles and identify similar articles for readers considering predefined criteria. The objectives of the current work is to recommend **news articles** which are similar to the already read article by using attributes like article *headline*, *category*, *author* and *publishing date*.

# 2. Introduction

Recommender systems aim to predict users' interests and recommend product items that quite likely are interesting for them. They are among the most powerful machine learning systems that online retailers implement in order to drive sales.

Data required for recommender systems stems from explicit user ratings after watching a movie or listening to a song, from implicit search engine queries and purchase histories, or from other knowledge about the users/items themselves.

Sites like Spotify, YouTube or Netflix use that data in order to suggest playlists, so-called Daily mixes, or to make video recommendations, respectively.

**Why do we need recommender systems?**

Companies using recommender systems focus on increasing sales as a result of very personalized offers and an enhanced customer experience.

Recommendations typically speed up searches and make it easier for users to access content they're interested in, and surprise them with offers they would have never searched for.

What is more, companies are able to gain and retain customers by sending out emails with links to new offers that meet the recipients' interests, or suggestions of films and TV shows that suit their profiles.

The user starts to feel known and understood and is more likely to buy additional products or consume more content. By knowing what a user wants, the company gains competitive advantage and the threat of losing a customer to a competitor decreases.

Providing that added value to users by including recommendations in systems and products is appealing. Furthermore, it allows companies to position ahead of their competitors and eventually increase their earnings.

# 3. Data Collection

We have extracted the dataset from https://www.kaggle.com.The dataset contains  200583 observations with 6 features. Each record in the dataset consists of attributes like category, headline, authors, link, short_description, date.

**Data Description :**

All the variables are categorical except the date feature which is in datetime format.

1. category : Describes to which category of the news article belongs to. eg : Crime, Politics, Entertainment etc. There are around 41 different categories present.
2. headline : Headline of the news.
3. authors : The authors of the article
4. link : Link corresponding the news article
5. short_description : Short description on the content of the article
6. date : Date on which the article was published

# 4.Data Preprocessing

## *4.a) Fetching only the articles from 2018*

Dataset from 2018 is only considered for the modeling since as we go further past the relevance of the data may be affected.

```
news_articles = news_articles[news_articles['date'] >= pd.Timestamp(2018,1,1)]
```

After filtering the dataset, the new data consists of 8583 observations with 6 features.

## 4.b) Removing all the short headline articles

Very short headlines are removed from the datasets (i.e, headlines having length less than 5 words) as most of them seem to be meaningless sentences.

```
news_articles = news_articles[news_articles['headline'].apply(lambda x: len(x.split())>5)]
```

After removing the smaller headlines the observations in the dataset got reduced to 8530.

## 4.c) Checking for missing values

```
news_articles.isna().sum()
```

```
category             0
headline             0
authors              0
link                 0
short_description    0
date                 0
dtype: int64
```

No missing values are present in the dataset.

# 5. Basic Data Exploration

## 5.a) Basic statistics - Number of articles,authors,categories

```
print("Total number of articles : ", news_articles.shape[0])
print("Total number of authors : ", news_articles["authors"].nunique())
print("Total number of unqiue categories : ", news_articles["category"].nunique())
```

```
Total number of articles :  8485
Total number of authors :  892
Total number of unqiue categories :  26
```

## 5.b) Distribution of articles category-wise

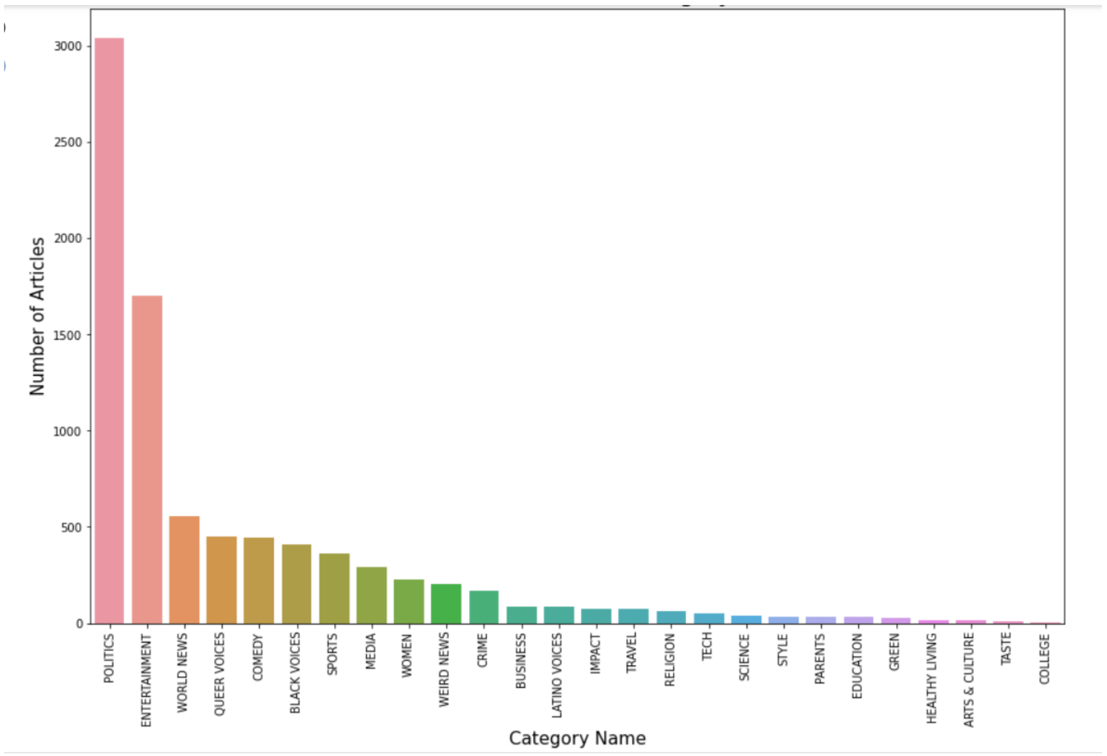


*fig. 5.1 Distribution of Articles category wise*

**Finding :**

The Politics category has the highest number of articles then entertainment and so on.

## 5.c) Number of articles per month

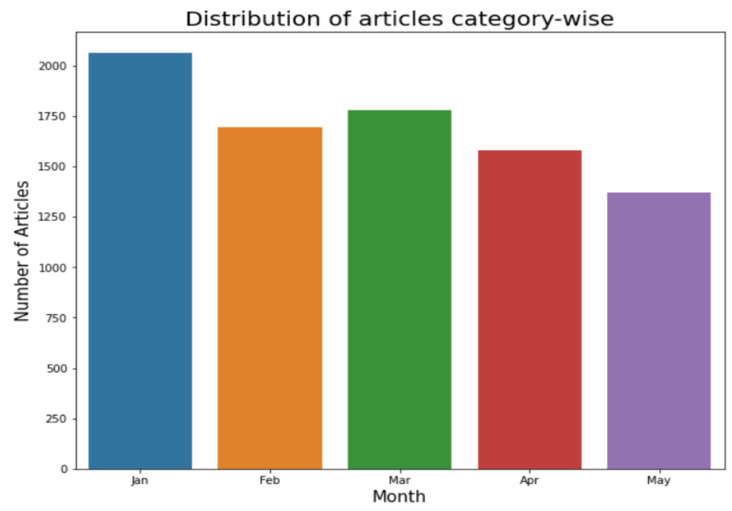


*fig 5.2.Number of Articles per month*

**Finding :**

January has the highest number of articles then March and so on.

## 5.d) PDF for the length of headlines

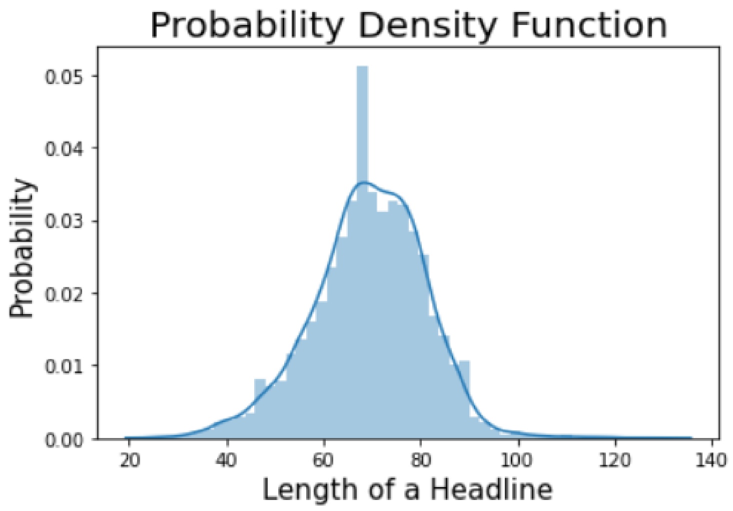


*fig 5.3. PDF for the length of headlines*

**Finding :**

The probability distribution function of headline length is almost similar to a Guassian distribution, where most of the headlines are 58 to 80 words long in length

**INFERENCES :**

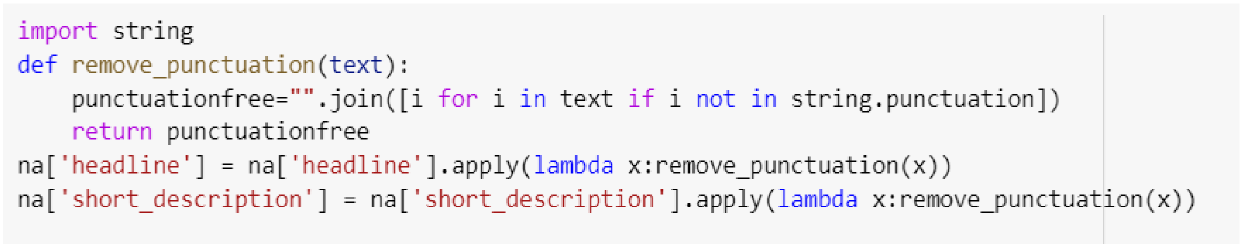
- Major part of the news articles comes under the Politics and Entertainment category.
- Most of the headlines have words between 58 to 80.

# 6.Text Preprocessing

Text preprocessing involves transforming text into a clean and consistent format that can then be fed into a model for further analysis and learning. We have performed the text preprocessing on the feature 'headline' and 'short-description'.

## 6.a) Removing Punctuation

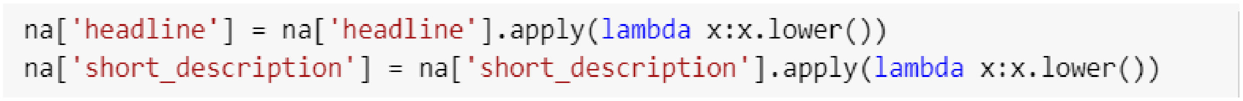
The punctuation removal process will help to treat each text equally. We have removed punctuations using the python's string module.



```python
import string
def remove_punctuation(text):
    punctuationfree="".join([i for i in text if i not in string.punctuation])
    return punctuationfree
na['headline'] = na['headline'].apply(lambda x:remove_punctuation(x))
na['short_description'] = na['short_description'].apply(lambda x:remove_punctuation(x))
```

## 6.b) Lowering

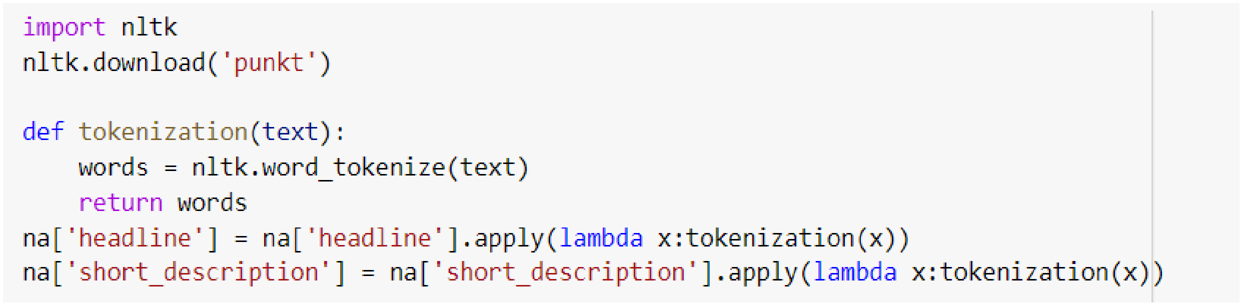
Converting all your data to lowercase helps in the process of preprocessing and in later stages in the NLP application like stopwords removal because stopwords are in lowercase.



```python
na['headline'] = na['headline'].apply(lambda x:x.lower())
na['short_description'] = na['short_description'].apply(lambda x:x.lower())
```

## 6.c) Tokenization

Tokenization is the process of tokenizing or splitting a string, text into a list of tokens. Here we used the *word_tokenize()* module *of* nltk library for tokenization.



```python
import nltk
nltk.download('punkt')

def tokenization(text):
    words = nltk.word_tokenize(text)
    return words
na['headline'] = na['headline'].apply(lambda x:tokenization(x))
na['short_description'] = na['short_description'].apply(lambda x:tokenization(x))
```

## 6.d) Stopwords Removal

The idea is simply removing the words that occur commonly across all the documents in the corpus. Typically, articles and pronouns are generally classified as stop words.Removal of stop words definitely reduces the dataset size and thus reduces the training time due to the fewer number of tokens involved in the training.We removed stopwords using nltk library.

```
nltk.download('stopwords')
stopwords = nltk.corpus.stopwords.words('english')
def remove_stopwords(text):
    output= [i for i in text if i not in stopwords]
    return output
na['headline'] = na['headline'].apply(lambda x:remove_stopwords(x))
na['short_description'] = na['short_description'].apply(lambda x:remove_stopwords(x))
```

## 6.e) Lemmatization

Lemmatization is the process of replacing a word with its *root* or *head word* called lemma. Aim is to reduce inflectional forms to a common base form.We have used WordNetLemmatizer, a library that is imported from *nltk.stem* which looks for lemmas of words from the WordNet Database.

```
from nltk.stem import WordNetLemmatizer
nltk.download('wordnet')
nltk.download('omw-1.4')
lemm=WordNetLemmatizer()
wordnet_lemmatizer = WordNetLemmatizer()
def lemmatizer(text):
  lemm_text = [wordnet_lemmatizer.lemmatize(word) for word in text]
  return lemm_text
na['headline'] = na['headline'].apply(lambda x:lemmatizer(x))
na['short_description'] = na['short_description'].apply(lambda x:lemmatizer(x))
```

# 7.Modelling

For the modeling we tried with Bag of words and TD-IF method. TD-IF method was tried in feature “Headline” with euclidean space.Bag of words was tried with feature “Headline” and with euclidean distance. Also with feature “Short_description” and cosine_similarity. Here the latter one gave better results which was taken for web hosting.

## Bag-of-Words

```
from sklearn.feature_extraction.text import CountVectorizer
cv=CountVectorizer(max_features=5000,stop_words='english')
vectors=cv.fit_transform(na['short_description']).toarray()
from sklearn.metrics.pairwise import cosine_similarity
similarity=cosine_similarity(vectors)
```

```
def recommendations(news):
    news_index=na[na['headline']==news].index[0]
    dist_per_news=similarity[4]
    news_list=sorted(list(enumerate(dist_per_news)),reverse=True,key=lambda x:x[1])[1:11]
    for i in news_list:
        print(na['headline'][i[0]])
        print(dist_per_news)
```

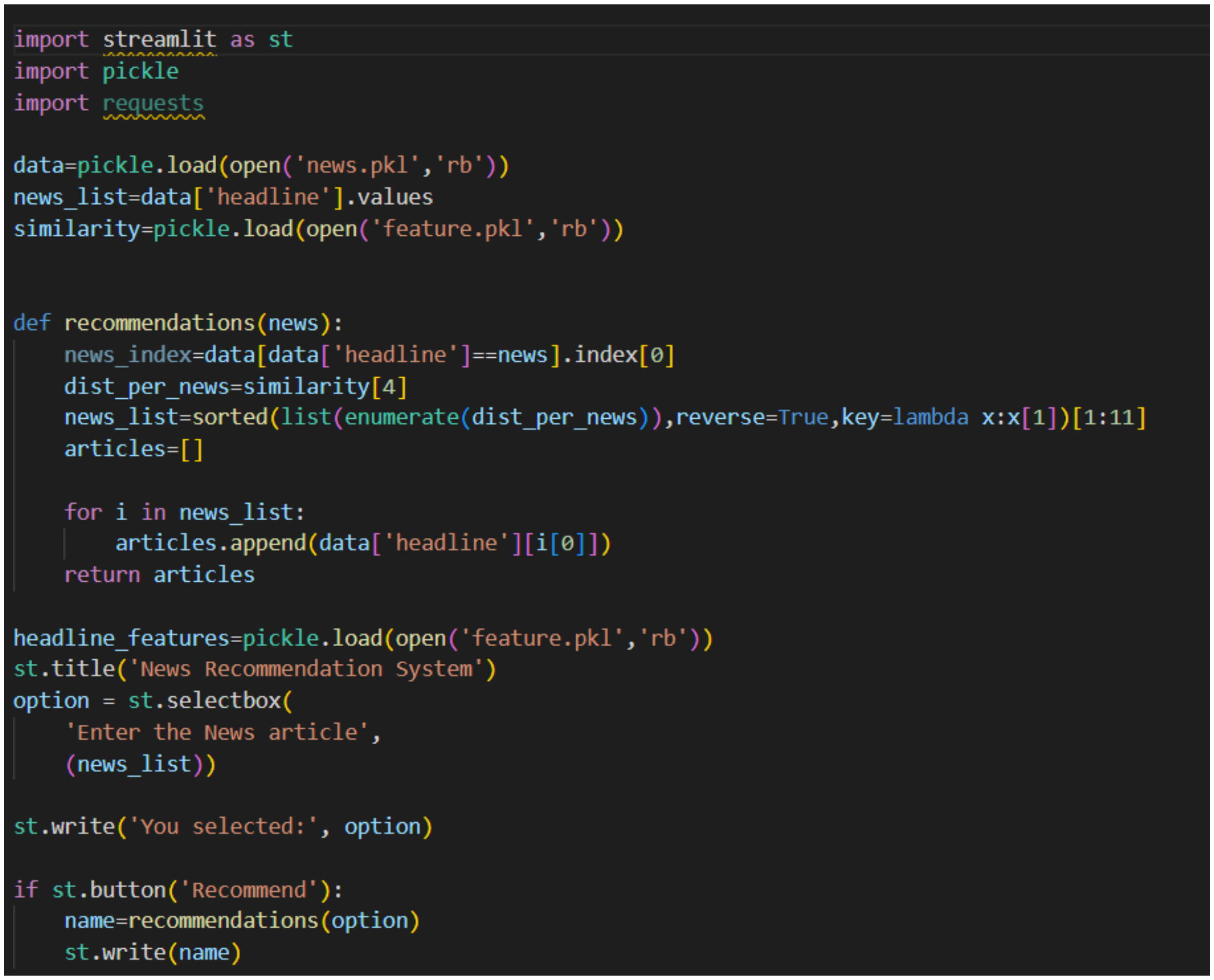
After that the model was converted into a pickle file for web hosting purpose.

```
import pickle
pickle.dump(na,open('/content/drive/MyDrive/Project_team_6/news.pkl','wb'))
pickle.dump(similarity,open('/content/drive/MyDrive/Project_team_6/feature.pkl','wb'))
```

```
!python3 "/content/drive/MyDrive/Project_team_6/App.ipynb"
```

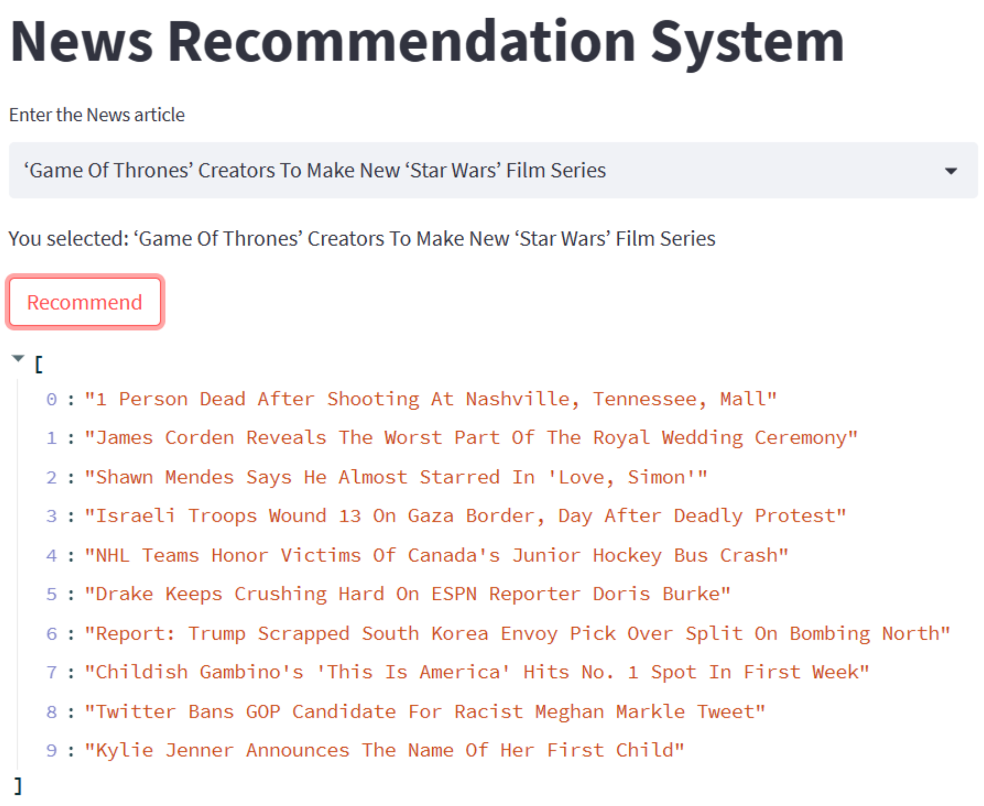
# 8.Web Hosting

Web hosting of the best model is done using Streamlit.



```
import streamlit as st
import pickle
import requests

data=pickle.load(open('news.pkl','rb'))
news_list=data['headline'].values
similarity=pickle.load(open('feature.pkl','rb'))


def recommendations(news):
    news_index=data[data['headline']==news].index[0]
    dist_per_news=similarity[4]
    news_list=sorted(list(enumerate(dist_per_news)),reverse=True,key=lambda x:x[1])[1:11]
    articles=[]

    for i in news_list:
        articles.append(data['headline'][i[0]])
    return articles

headline_features=pickle.load(open('feature.pkl','rb'))
st.title('News Recommendation System')
option = st.selectbox(
    'Enter the News article',
    (news_list))

st.write('You selected:', option)

if st.button('Recommend'):
    name=recommendations(option)
    st.write(name)
```

The following was the hosting page with an example where 10 articles are recommended with Headline " 'Game of Thrones' creators To make new 'Star wars' film series " input



News Recommendation System

Enter the News article

'Game Of Thrones' Creators To Make New 'Star Wars' Film Series

You selected: 'Game Of Thrones' Creators To Make New 'Star Wars' Film Series

```
[
  0 : "1 Person Dead After Shooting At Nashville, Tennessee, Mall"
  1 : "James Corden Reveals The Worst Part Of The Royal Wedding Ceremony"
  2 : "Shawn Mendes Says He Almost Starred In 'Love, Simon'"
  3 : "Israeli Troops Wound 13 On Gaza Border, Day After Deadly Protest"
  4 : "NHL Teams Honor Victims Of Canada's Junior Hockey Bus Crash"
  5 : "Drake Keeps Crushing Hard On ESPN Reporter Doris Burke"
  6 : "Report: Trump Scrapped South Korea Envoy Pick Over Split On Bombing North"
  7 : "Childish Gambino's 'This Is America' Hits No. 1 Spot In First Week"
  8 : "Twitter Bans GOP Candidate For Racist Meghan Markle Tweet"
  9 : "Kylie Jenner Announces The Name Of Her First Child"
]
```

*fig 8.1. News Recommendation system Web page*

# 9.Result

News articles recommendation System deals with a content based recommendation problem. The chosen dataset includes about 6 features. Aim of the project was to recommend articles similar to the given headline. The best model was Bag-of-Words based on Cosine similarity.

# 10.Conclusion

NRS has been increasingly used in recent years to provide better suggestions to end users so that they can consume online news from various sources. There are many unique challenges associated with the NRS, most of which are inherited from the news domain. Out of these challenges, the issues related to timeliness, evolving readers' preferences over dynamically generated news, quality of news content and the effects of news recommendations on users' behavior are prominent ones.The general recommendation algorithms are insufficient to provide news recommendations since they need to be modified, varied or extended to a large extent. Recently, the DL-based solutions have addressed much of those limitations of conventional recommenders. Accuracy is considered as a standard evaluation measure to assess the quality of a recommender system. However, beyond accuracy, other aspects such as diversity, coverage, novelty, serendipity are also important to provide better user experience in an NRS. Datasets, open recommendation platforms and evaluation protocols together play a role in developing recommendation solutions in the news domain. This project is a basic version of recommending articles based on content read by the reader, which can be further enhanced with the availability of ratings or viewer votings.

# 11.Reference

[1] News Recommendation System using Python
https://thecleverprogrammer.com/2022/09/12/news-recommendation-system-using-python/