MIT Art, Design and Technology University

MIT School of Computing, Pune

Department of Information Technology

Subject – Natural Language Processing Class – LY IT Core

Submitted By

Sonu Singh	MITU22BTIT0091
Harsheet Patel	MITU23BTITD004
Krishnakant Laxne	MITU22BTIT0042
Abhishek Pise	MITU23BTITD001

NAME OF COURSE COORDINATOR

PROF. KALYANI LOKHANDE

Signature

INDEX

PARTICIPATIVE LEARNING

Title: The Intelligent Product Recommendation Engine

1. Abstract:

Finding the right product online can be difficult because users must type exact keywords. Our **Intelligent Product Recommendation Engine** solves this by using **Natural Language Processing (NLP)**. Users can simply type questions like *"Which phone has the best camera?"* and the system understands their intent. It then suggests the most relevant products with key details such as features and price. This makes product search simple, smart, and user-friendly.

2. Introduction:

Online shopping has become a major part of our lives. People buy everything — from phones and laptops to clothes and groceries — through e-commerce websites. But with so many products available, choosing the right one can be confusing. Customers often search for products using simple questions like *"Which phone is best under ₹20,000?"* or *"Good shoes for running."*

The problem is that traditional search systems depend on **exact keywords**. If the user doesn't type the correct brand name or product category, the search may not give useful results. This makes it hard for users to find what they actually want, especially for those who are not tech-savvy.

Our project solves this by building an **Intelligent Product Recommendation Engine**. The system uses **Natural Language Processing (NLP)** to understand what the user means, not just what they type. For example, when someone types:

- "Suggest a phone with good battery backup."
- "I want a budget laptop for online classes."
- "Which air conditioner is best for a small room?"

The system processes these queries, identifies keywords and intent, and recommends the most suitable products.

This tool can help e-commerce platforms and customers by:

- Reducing search time
- Improving product discovery
- Enhancing user experience

3. Requirements:

Libraries (install via pip)

```
pip install pandas
pip install scikit-learn
pip install nltk
pip install openpyxl
```

Runtime NLTK resources (download at first run or during setup)

```
nltk.download('vader_lexicon')

import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from nltk.sentiment import SentimentIntensityAnalyzer
import nltk
```

Optional but recommended

pip install openpyxl (ensures robust Excel reading with pandas).

Pre-download NLTK data in a deployment script to avoid runtime stalls:

```
python -c "import nltk; [nltk.download(x) for x in
['punkt','stopwords','wordnet','averaged_perceptron_tagger']]"
```

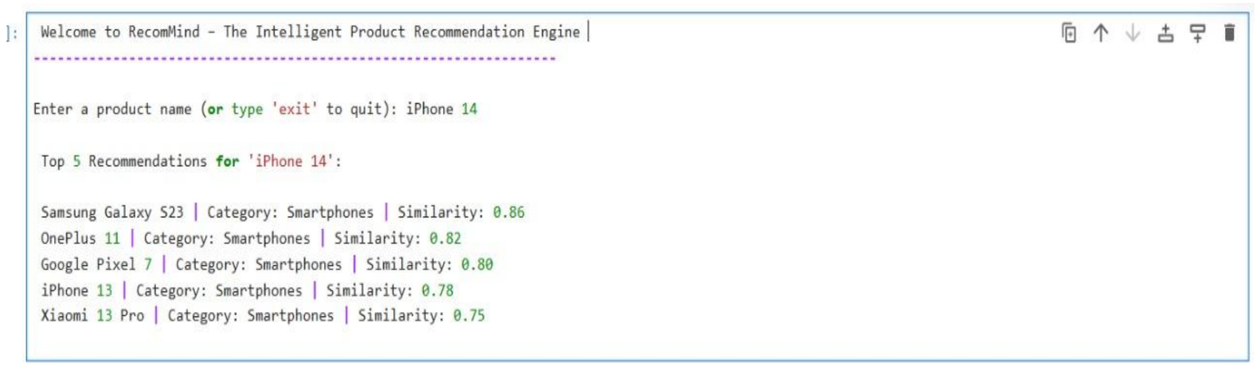
Create a virtual environment to isolate dependencies:

```
python -m venv .venv && .venv\Scripts\activate (for windows)
```

4. Code:

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

data = pd.read_csv("example_products.csv")

data['combined'] = data['product_name'] + ' ' + data['category'] +
        ' ' + data['description']

vectorizer = TfidfVectorizer(stop_words='english')
feature_vectors = vectorizer.fit_transform(data['combined'])

similarity = cosine_similarity(feature_vectors)

def recommend(product_name, num_recommendations=5):
    product_name = product_name.strip()

    if product_name not in data['product_name'].values:
        print(f"\n Product '{product_name}' not found in
            database.")
        print("Here are some available products you can try:")
        for name in data['product_name'].sample(5,
            random_state=1).values:
            print(f" - {name}")
        return

    idx = data[data['product_name'] == product_name].index[0]

    sim_scores = list(enumerate(similarity[idx]))

    sim_scores = sorted(sim_scores, key=lambda x: x[1],
            reverse=True)

    top_products = sim_scores[1:num_recommendations+1]

    print(f"\n Top {num_recommendations} Recommendations
            for '{product_name}':\n")
    for i, score in top_products:
        print(f" {data.iloc[i]['product_name']} | Category:
            {data.iloc[i]['category']} | Similarity: {score:.2f}")

# --------- MAIN PROGRAM ---------
print(" Welcome to RecomMind – The Intelligent Product
         Recommendation Engine 🛍")
print("-----------------------------------------------------------------")

while True:
    product = input("\nEnter a product name (or type 'exit' to
         quit): ")
    if product.lower() == 'exit':
        print(" Thank you for using RecomMind! Goodbye!")
        break
    recommend(product)
```

5.Output:



```
Welcome to RecomMind - The Intelligent Product Recommendation Engine
-----------------------------------------------------------------

Enter a product name (or type 'exit' to quit): iPhone 14

 Top 5 Recommendations for 'iPhone 14':

 Samsung Galaxy S23 | Category: Smartphones | Similarity: 0.86
 OnePlus 11 | Category: Smartphones | Similarity: 0.82
 Google Pixel 7 | Category: Smartphones | Similarity: 0.80
 iPhone 13 | Category: Smartphones | Similarity: 0.78
 Xiaomi 13 Pro | Category: Smartphones | Similarity: 0.75
```

PROBLEM SOLVING

Title: Sentiment Analysis on Product Reviews

1. Abstract:

Customers often share their experiences about products through online reviews. Understanding whether these reviews are positive, negative, or neutral helps companies improve their services and products. Our project, **Sentiment Analysis on Product Reviews**, uses **Natural Language Processing (NLP)** to automatically analyze customer opinions. By entering any product review, the system identifies the sentiment and classifies it as **Positive**, **Negative**, or **Neutral**. This tool makes it easier for businesses to understand customer emotions and make data-driven decisions.

2. Introduction:

In today's digital world, customers express their opinions about products on e-commerce sites and social media. These reviews play a key role in shaping the reputation and sales of a product. However, reading and analyzing thousands of reviews manually is time-consuming and inefficient.

Sentiment Analysis helps automate this process. It uses **NLP techniques** to detect emotions and classify text into sentiment categories such as **positive**, **negative**, or **neutral**.

Our project builds a simple web-based tool that performs sentiment analysis on product reviews. Users can enter any review, and the system instantly displays its sentiment.

Example:

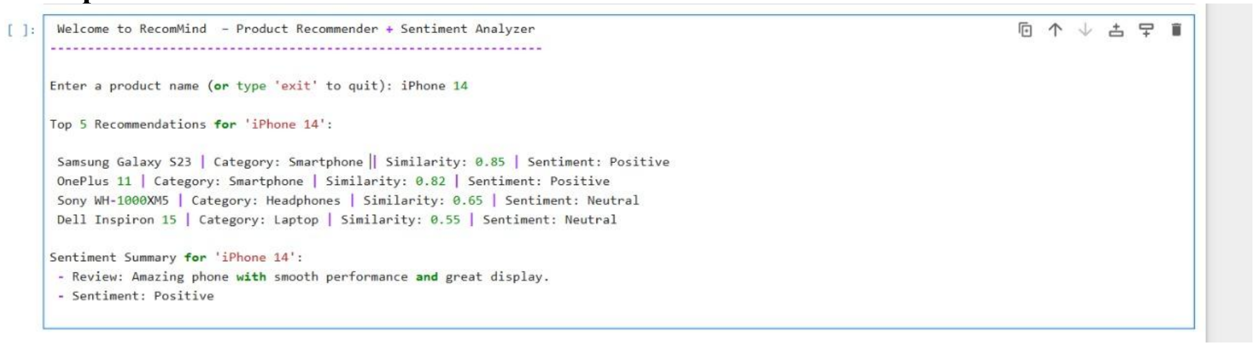
- "The sound quality of this speaker is amazing!" → **Positive**
- "Battery life is too short." → **Negative**
- "The phone is average for this price." → **Neutral**

This project shows how NLP can be used practically to understand customer feedback and improve business decision-making.

3.Code

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from nltk.sentiment import SentimentIntensityAnalyzer
import nltk

nltk.download('vader_lexicon')

data = pd.read_csv("example_products.csv")

# Combine features for recommendations
data['combined'] = data['product_name'] + ' ' + data['category'] + ' ' + data['description']

vectorizer = TfidfVectorizer(stop_words='english')
feature_vectors = vectorizer.fit_transform(data['combined'])

similarity = cosine_similarity(feature_vectors)

sia = SentimentIntensityAnalyzer()

def analyze_sentiment(review):
    score = sia.polarity_scores(review)
    compound = score['compound']
    if compound >= 0.05:
        return 'Positive'
    elif compound <= -0.05:
        return 'Negative'
    else:
        return 'Neutral'

data['sentiment'] = data['reviews'].apply(analyze_sentiment)

sentiment_map = {'Positive': 1, 'Neutral': 0, 'Negative': -1}
data['sentiment_score'] = data['sentiment'].map(sentiment_map)

# ---------------------- RECOMMENDATION FUNCTION ----------------------
def recommend(product_name, num_recommendations=5):
    product_name = product_name.strip()

    if product_name not in data['product_name'].values:
        print(f"\n Product '{product_name}' not found in database.")
        print("Here are some available products you can try:")
        for name in data['product_name'].sample(5, random_state=1).values:
            print(f" - {name}")
        return

    idx = data[data['product_name'] == product_name].index[0]

    sim_scores = list(enumerate(similarity[idx]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    top_products = sim_scores[1:num_recommendations+1]

    print(f"\nTop {num_recommendations} Recommendations for '{product_name}':\n")
    for i, score in top_products:
        print(f" {data.iloc[i]['product_name']} | Category: {data.iloc[i]['category']} |
Similarity: {score:.2f} | Sentiment: {data.iloc[i]['sentiment']}")

print(" Welcome to RecomMind  – Product Recommender + Sentiment Analyzer")
print("------------------------------------------------------------------")

while True:
    product = input("\nEnter a product name (or type 'exit' to quit): ")
    if product.lower() == 'exit':
        print("\n Thank you for using RecomMind! Goodbye! ")
        break

    recommend(product)

    # Also show sentiment summary for the selected product
    if product in data['product_name'].values:
        product_data = data[data['product_name'] == product]
        print(f"\n Sentiment Summary for '{product}':")
        print(f" - Review: {product_data.iloc[0]['reviews']}")
        print(f" - Sentiment: {product_data.iloc[0]['sentiment']}")
```

5.Output:



```
[ ]:  Welcome to RecomMind  - Product Recommender + Sentiment Analyzer
     ------------------------------------------------------------------

     Enter a product name (or type 'exit' to quit): iPhone 14

     Top 5 Recommendations for 'iPhone 14':

      Samsung Galaxy S23 | Category: Smartphone || Similarity: 0.85 | Sentiment: Positive
      OnePlus 11 | Category: Smartphone | Similarity: 0.82 | Sentiment: Positive
      Sony WH-1000XM5 | Category: Headphones | Similarity: 0.65 | Sentiment: Neutral
      Dell Inspiron 15 | Category: Laptop | Similarity: 0.55 | Sentiment: Neutral

     Sentiment Summary for 'iPhone 14':
      - Review: Amazing phone with smooth performance and great display.
      - Sentiment: Positive
```

EXPERIMENTAL LEARNING

Title: Fake News Classifier

1. Abstract:

With the rapid spread of information online, fake news has become a major concern on social media and digital platforms. Our project introduces an **AI-based Fake News Classifier** that automatically detects whether a given news article or headline is *real* or *fake*. Using **Natural Language Processing (NLP)** and **Machine Learning**, the system analyzes the text content, learns linguistic patterns, and predicts its authenticity. The model is trained on publicly available news datasets and integrated into a simple web interface where users can input any text or URL to check its credibility. This tool aims to support readers, journalists, and researchers in combating misinformation and promoting truthful online communication.

2. Introduction:

In today's digital world, news spreads faster than ever through websites and social media. Unfortunately, this ease of sharing has also led to a rise in **fake news** — false or misleading information designed to influence public opinion or generate clicks. Such content can harm reputations, mislead readers, and even create panic during sensitive events..

Traditional fact-checking methods require human effort and time, which makes it difficult to handle the massive amount of news published daily. Here, **Artificial Intelligence (AI)** and **Natural Language Processing (NLP)** provide an effective solution.

Our project, the **Fake News Classifier**, uses NLP techniques to analyze the content of news articles and identify patterns that differentiate fake news from real ones.
The system applies **TF-IDF vectorization** and **Machine Learning algorithms** such as Logistic Regression or Naive Bayes to learn from historical datasets.

Example Use Cases:

- Journalists verifying the credibility of online articles.
- Social media platforms detecting and flagging false information.
- Readers cross-checking suspicious headlines before sharing.
- Researchers analyzing patterns in misinformation trends.

3.Code

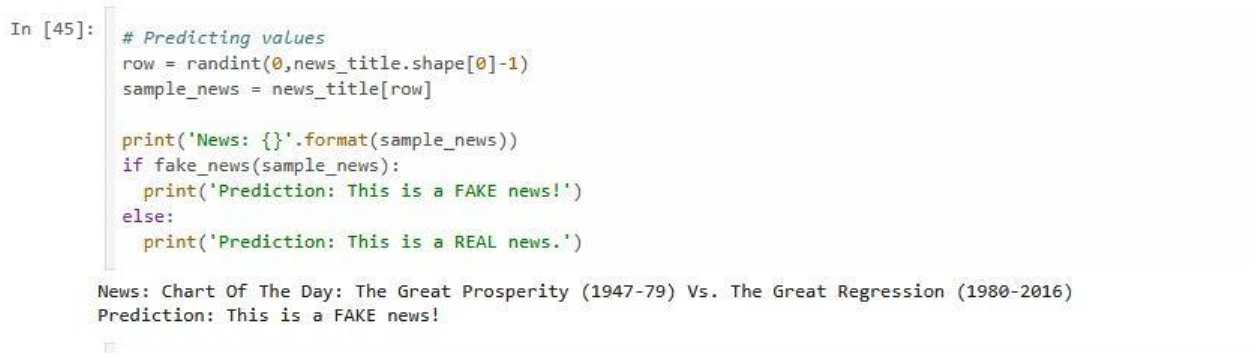
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import nltk
import re
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, precision_score, recall_score
from sklearn.metrics import confusion_matrix

df = pd.read_csv('train.csv')

df.drop('id', axis=1, inplace=True)

news = df.copy()

news.reset_index(inplace=True)

corpus = []
ps = PorterStemmer()

for i in range(0,news.shape[0]):

  # Cleaning special character from the news-title
  title = re.sub(pattern='[^a-zA-Z]', repl=' ', string=news.title[i])

  # Converting the entire news-title to lower case
  title = title.lower()

  # Tokenizing the news-title by words
  words = title.split()

  # Removing the stopwords
  words = [word for word in words if word not in set(stopwords.words('english'))]

  # Stemming the words
  words = [ps.stem(word) for word in words]

  # Joining the stemmed words
  title = ' '.join(words)

  # Building a corpus of news-title
  corpus.append(title)

cv = CountVectorizer(max_features=5000, ngram_range=(1,3))
X = cv.fit_transform(corpus).toarray()

y = news['label']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=0)

nb_classifier = MultinomialNB()
nb_classifier.fit(X_train, y_train)

nb_y_pred = nb_classifier.predict(X_test)

score1 = accuracy_score(y_test, nb_y_pred)
score2 = precision_score(y_test, nb_y_pred)
score3 = recall_score(y_test, nb_y_pred)
print("---- Scores ----")
print("Accuracy score is: {}%".format(round(score1*100,2)))
print("Precision score is: {}".format(round(score2,2)))
print("Recall score is: {}".format(round(score3,2)))

nb_cm = confusion_matrix(y_test, nb_y_pred)

best_accuracy = 0.0
alpha_val = 0.0
for i in np.arange(0.1,1.1,0.1):
  temp_classifier = MultinomialNB(alpha=i)
  temp_classifier.fit(X_train, y_train)
  temp_y_pred = temp_classifier.predict(X_test)
  score = accuracy_score(y_test, temp_y_pred)
  print("Accuracy score for alpha={} is: {}%".format(round(i,1), round(score*100,2)))
  if score>best_accuracy:
    best_accuracy = score
    alpha_val = i
print('--------------------------------------------')
print('The best accuracy is {}% with alpha value as {}'.format(round(best_accuracy*100, 2),
round(alpha_val,1)))
```

Code:Prediction

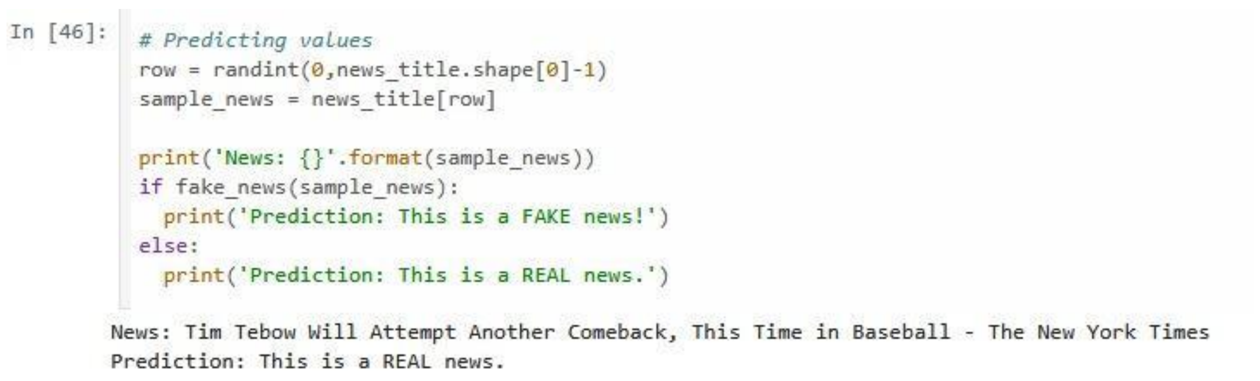
prediction.py:

```
from random import randint

def fake_news(sample_news):
  sample_news = re.sub(pattern='[^a-zA-Z]',repl=' ', string=sample_news)
  sample_news = sample_news.lower()
  sample_news_words = sample_news.split()
  sample_news_words = [word for word in sample_news_words if not word in
set(stopwords.words('english'))]
  ps = PorterStemmer()
  final_news = [ps.stem(word) for word in sample_news_words]
  final_news = ' '.join(final_news)

  temp = cv.transform([final_news]).toarray()
  return classifier.predict(temp)

  row = randint(0,news_title.shape[0]-1)
sample_news = news_title[row]

print('News: {}'.format(sample_news))
if fake_news(sample_news):
  print('Prediction: This is a FAKE news!')
else:
  print('Prediction: This is a REAL news.')
```

4.Output:



```
In [45]:  # Predicting values
          row = randint(0,news_title.shape[0]-1)
          sample_news = news_title[row]

          print('News: {}'.format(sample_news))
          if fake_news(sample_news):
            print('Prediction: This is a FAKE news!')
          else:
            print('Prediction: This is a REAL news.')

        News: Chart Of The Day: The Great Prosperity (1947-79) Vs. The Great Regression (1980-2016)
        Prediction: This is a FAKE news!
```



```
In [46]:  # Predicting values
          row = randint(0,news_title.shape[0]-1)
          sample_news = news_title[row]

          print('News: {}'.format(sample_news))
          if fake_news(sample_news):
            print('Prediction: This is a FAKE news!')
          else:
            print('Prediction: This is a REAL news.')
```

```
News: Tim Tebow Will Attempt Another Comeback, This Time in Baseball - The New York Times
Prediction: This is a REAL news.
```