

## **Information Retrieval for Data Science (P)**

Practical Work submitted to **ST. FRANCIS COLLEGE FOR WOMEN**  
in partial fulfillment of the requirements for the award of the degree of

### **MASTER OF DATA SCIENCE**

By

A.Banu Harshini

**121323030026**

Under the guidance and supervision of

**Ms. Afeefa Noorain**



**DEPARTMENT OF COMPUTER SCIENCE  
ST. FRANCIS COLLEGE FOR WOMEN  
BEGUMPET, HYDERABAD.**

**OCTOBER,2024**

## **CERTIFICATE**

This is to certify that the bonafide record of the lab entitled **Information Retrieval for Data Science (P)** in the **Semester III** submitted to the **St. Francis College for Women** in partial fulfillment of the requirements for the award of the **Degree of Masters in DATA SCIENCE**, is an original record work done by **A.Banu Harshini** bearing the **121323030026** during the academic Year **2023-2024** under the supervision and guidance of **Ms.Afeefa Noorain**, Department of Computer Science, St. Francis College for Women, Begumpet, Hyderabad-16.

Internal Examiner

External Examiner

## TABLE OF CONTENTS

S. NO	TOPIC	PAGE NO.
1	Text Processing	1
2	Term Document Matrix	3
3	Inverted Index	6
4	TF - IDF	8
5	Naive Bayes	11
6	Vector Space Model	13
7	Project	15

## 1.Text Processing

```
text1 = "Ethics are built right into the ideals and objectives of the United Nations "  
len(text1)
```

---

76

```
text2 = text1.split(' ')  
len(text2)
```

---

14

```
text2
```

```
['Ethics',  
 'are',  
 'built',  
 'right',  
 'into',  
 'the',  
 'ideals',  
 'and',  
 'objectives',  
 'of',  
 'the',  
 'United',  
 'Nations',  
 '']
```

```
[w for w in text2 if len(w) > 3]
```

```
['Ethics',  
 'built',  
 'right',  
 'into',  
 'ideals',  
 'objectives',  
 'United',  
 'Nations']
```

```
text3 = 'To be or not to be'  
text4 = text3.split(' ')  
len(text4)
```

---

6

```
text5 = 'ouagadougou'
text6 = text5.split('ou')
text6
```

---

```
['', 'agad', 'g', '']
```

```
[c for c in text5]
['o', 'u', 'a', 'g', 'a', 'd', 'o', 'u', 'g', 'o', 'u']
```

```
text8 = ' A quick brown fox jumped over the lazy dog'
text8.split(' ')
```

```
['',
 '',
 '',
 '',
 'A',
 'quick',
 'brown',
 'fox',
 'jumped',
 'over',
 'the',
 'lazy',
 'dog']
```

```
text9.replace('o','O')
'A quick brOwn fOx jumped Over the lazy dOg'
```

```
dateStr = '23-10-2002\n23/10/2002\n23/10/02\n10/23/2002\n23 Oct 2002\n23 October 2002\nOct 23,
2002\nOctober 23, 2002\n'
re.findall(r'\d{2}[/-]\d{2}[/-]\d{4}', dateStr)
```

---

```
['23-10-2002', '23/10/2002', '10/23/2002']
```

## 2.Term Document Matrix

```
review_1 = "The Glider II is a great soccer ball"
```

```
review_2 = "What a bad soccer ball"
```

```
review_3 = "I am happy with The glider"
```

```
docs = [review_1, review_2, review_3]
```

```
docs
```

```
['The Glider II is a great soccer ball',  
 'What a bad soccer ball',  
 'I am happy with The glider']
```

```
unique_terms = {term for doc in docs for term in doc.split()}
```

```
unique_terms
```

```
{'Glider',  
 'I',  
 'II',  
 'The',  
 'What',  
 'a',  
 'am',  
 'bad',  
 'ball',  
 'glider',  
 'great',  
 'happy',  
 'is',  
 'soccer',  
 'with'}
```

```
doc_term_matrix = { }
```

```
for term in unique_terms:
```

```
    doc_term_matrix[term] = []
```

```
    for doc in docs:
```

```
        if term in doc:
```

```
            doc_term_matrix[term].append(1)
```

```
        else: doc_term_matrix[term].append(0)
```

```
doc_term_matrix
```

```
{
  'II': [1, 0, 0],
  'great': [1, 0, 0],
  'I': [1, 0, 1],
  'is': [1, 0, 0],
  'The': [1, 0, 1],
  'soccer': [1, 1, 0],
  'ball': [1, 1, 0],
  'glider': [0, 0, 1],
  'happy': [0, 0, 1],
  'with': [0, 0, 1],
  'Glider': [1, 0, 0],
  'a': [1, 1, 1],
  'bad': [0, 1, 0],
  'am': [0, 0, 1],
  'What': [0, 1, 0]}

```

```
import numpy as np
```

```
docs_array = np.array(docs, dtype='object')
```

```
v1 = np.array(doc_term_matrix['Glider'])
```

```
v2 = np.array(doc_term_matrix['soccer'])
```

```
print(v1)
```

```
print(v2)
```

```
print('-----')
```

```
v3 = v1 & v2
```

```
print(v3)
```

```
[1 0 0]
```

```
[1 1 0]
```

```
-----
```

```
[1 0 0]
```

```
[doc for doc in v3 * docs_array if doc]
```

```
['The Glider II is a great soccer ball']
```

```
v1 = np.array(doc_term_matrix['a'])
```

```
v2 = np.array(doc_term_matrix['ball'])
```

```
print(v1)
```

```
print(v2)
```

```
print('-----')
```

```
v3 = v1 & v2
```

```
print(v3)
```

```
[1 1 1]
[1 1 0]
-----
[1 1 0]
```

```
[doc for doc in v3 * docs_array if doc]
```

```
['The Glider II is a great soccer ball', 'What a bad soccer ball']
```

```
v1 = np.array(doc_term_matrix['great'])
v2 = np.array(doc_term_matrix['bad'])
```

```
print(v1)
print(v2)
print('-----')
v3 = v1 | v2
print(v3)
```

```
[1 0 0]
[0 1 0]
-----
[1 1 0]
```

```
[doc for doc in v3 * docs_array if doc]
```

```
['The Glider II is a great soccer ball', 'What a bad soccer ball']
```



### 3. Inverted Index

```
import nltk
nltk.download()
```

```
-
True
```

```
from nltk.book import *
```

```
*** Introductory Examples for the NLTK Book ***
Loading text1, ..., text9 and sent1, ..., sent9
Type the name of the text or sentence to view it.
Type: 'texts()' or 'sents()' to list the materials.
text1: Moby Dick by Herman Melville 1851
text2: Sense and Sensibility by Jane Austen 1811
text3: The Book of Genesis
text4: Inaugural Address Corpus
text5: Chat Corpus
text6: Monty Python and the Holy Grail
text7: Wall Street Journal
text8: Personals Corpus
text9: The Man Who Was Thursday by G . K . Chesterton 1908
```

```
text1
```

```
<Text: Moby Dick by Herman Melville 1851>
```

```
sents()
```

```
sent1: Call me Ishmael .
sent2: The family of Dashwood had long been settled in Sussex .
sent3: In the beginning God created the heaven and the earth .
sent4: Fellow - Citizens of the Senate and of the House of Representatives :
sent5: I have a problem with people PMing me to lol JOIN
sent6: SCENE 1 : [ wind ] [ clop clop clop ] KING ARTHUR : Whoa there !
sent7: Pierre Vinken , 61 years old , will join the board as a nonexecutive director Nov. 29
.
sent8: 25 SEXY MALE , seeks attrac older single lady , for discreet encounters .
sent9: THE suburb of Saffron Park lay on the sunset side of London , as red and ragged as a
cloud of sunset .
```

```
sent1
```

```
['Call', 'me', 'Ishmael', '.']
```

```
text7
```

```
<Text: Wall Street Journal>
```

```
len(text7)
```

100676

```
len(set(text7))
```

12408

```
text11 = "Children shouldn't drink a sugary drink before"  
text11.split(' ')
```

---

```
['Children', '', "shouldn't", 'drink', 'a', 'sugary', 'drink', 'before']
```

```
nltk.word_tokenize(text11)
```

```
['Children', 'should', "n't", 'drink', 'a', 'sugary', 'drink', 'before']
```

```
text12 = " This is the first sentence. A gallon of milk in the U.S. costs $2.99.', 'Is this the third sentence?', 'Yes, it is!'"
```

```
sentence = nltk.sent_tokenize(text12)
```

```
len(sentence)
```

4

```
sentence
```

```
[' This is the first sentence.',  
 'A gallon of milk in the U.S. costs $2.99.',  
 '', 'Is this the third sentence?',  
 '', 'Yes, it is!']
```

## 4.TF- IDF

```
# Import required module
from sklearn.feature_extraction.text import TfidfVectorizer

# Assign documents
d0 = 'Geeks for geeks'
d1 = 'Geeks'
d2 = 'r2j'

# Merge documents into a single corpus
documents = [d0, d1, d2]

# Create the TfidfVectorizer object
tfidf = TfidfVectorizer()

# Fit the model and get the TF-IDF values
result = tfidf.fit_transform(documents)

# Get IDF values and feature names (words)
print('\nIDF values:')
for word, idf_value in zip(tfidf.get_feature_names_out(), tfidf.idf_):
    print(f"{word} : {idf_value}")

IDF values:
for : 1.6931471805599454
geeks : 1.2876820724517808
r2j : 1.6931471805599454

# get indexing
print('\nWord indexes:')
print(tfidf.vocabulary_)

# display tf-idf values
print('\ntf-idf value:')
print(result)

# in matrix form
print('\ntf-idf values in matrix form:')
print(result.toarray())
```

Word indexes:

```
{'geeks': 1, 'for': 0, 'r2j': 2}
```

tf-idf value:

(0, 0)	0.5493512310263033
(0, 1)	0.8355915419449176
(1, 1)	1.0
(2, 2)	1.0

tf-idf values in matrix form:

```
[[0.54935123 0.83559154 0.      ]
 [0.         1.         0.      ]
 [0.         0.         1.      ]]
```

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
# assign documents
```

```
d0 = 'geek1'
```

```
d1 = 'geek2'
```

```
d2 = 'geek3'
```

```
d3 = 'geek4'
```

```
# merge documents into a single corpus
```

```
string = [d0, d1, d2, d3]
```

```
# create object
```

```
tfidf = TfidfVectorizer()
```

```
# get tf-df values
```

```
result = tfidf.fit_transform(string)
```

```
# get indexing
```

```
print("\nWord indexes:")
```

```
print(tfidf.vocabulary_)
```

```
# display tf-idf values
```

```
print("\ntf-idf values:")
```

```
print(result)
```

Word indexes:

```
{'geek1': 0, 'geek2': 1, 'geek3': 2, 'geek4': 3}
```

tf-idf values:

(0, 0)	1.0
(1, 1)	1.0
(2, 2)	1.0
(3, 3)	1.0

```
# import required module
from sklearn.feature_extraction.text import TfidfVectorizer
# assign documents
d0 = 'Geeks for geeks!'
d1 = 'Geeks for geeks!'
# merge documents into a single corpus
string = [d0, d1]
# create object
tfidf = TfidfVectorizer()
# get tf-df values
result = tfidf.fit_transform(string)
# get indexing
print('\nWord indexes:')
print(tfidf.vocabulary_)
# display tf-idf values
print('\ntf-idf values:')
print(result)
```

```
Word indexes:
{'geeks': 1, 'for': 0}

tf-idf values:
(0, 0)      0.4472135954999579
(0, 1)      0.8944271909999159
(1, 0)      0.4472135954999579
(1, 1)      0.8944271909999159
```

```
# import required module
from sklearn.feature_extraction.text import TfidfVectorizer
# assign corpus
string = ['Geeks geeks']*5
# create object
tfidf = TfidfVectorizer()
# get tf-df values
result = tfidf.fit_transform(string)
# get indexing
print('\nWord indexes:')
print(tfidf.vocabulary_)
# display tf-idf values
print('\ntf-idf values:')
print(result)
```

```
Word indexes:
{'geeks': 0}

tf-idf values:
(0, 0)      1.0
(1, 0)      1.0
(2, 0)      1.0
(3, 0)      1.0
(4, 0)      1.0
```

## 5.Naive Bayes

```
import numpy as np
from collections import defaultdict

class NaiveBayesTextClassifier:
    def __init__(self):
        self.classes = []
        self.class_word_counts = defaultdict(lambda: defaultdict(int))
        self.class_doc_counts = defaultdict(int)
        self.total_doc_count = 0
        self.vocab = set()

    def train(self, documents, labels):
        """Train the classifier with labeled training data."""
        self.classes = set(labels)
        self.total_doc_count = len(labels)

        for doc, label in zip(documents, labels):
            self.class_doc_counts[label] += 1
            words = doc.split()
            self.vocab.update(words)
            for word in words:
                self.class_word_counts[label][word] += 1

    def predict(self, document):
        """Predict the class of a given document."""
        words = document.split()
        log_probabilities = {}

        for c in self.classes:
            log_prob = np.log(self.class_doc_counts[c] / self.total_doc_count)
            total_words_in_class = sum(self.class_word_counts[c].values())

            for word in words:
                word_count = self.class_word_counts[c][word] + 1 # Laplace smoothing
                log_prob += np.log(word_count / (total_words_in_class + len(self.vocab)))

            log_probabilities[c] = log_prob

        return max(log_probabilities, key=log_probabilities.get)

documents = [
    "the economy is improving",
    "sports are popular",
    "the government is stable",
```

```
"football and basketball are sports",  
"the economy needs help",  
"governments support the economy"  
]  
labels = ["economy", "sports", "government", "sports", "economy", "government"]  
  
nb_classifier = NaiveBayesTextClassifier()  
nb_classifier.train(documents, labels)  
  
# Test with a new document  
test_document = "the economy needs improvement"  
predicted_class = nb_classifier.predict(test_document)  
  
print(f"The predicted class for '{test_document}' is: {predicted_class}")
```

---

```
The predicted class for 'the economy needs improvement' is: economy
```

## 6.Vector Space Model

```
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

class VectorSpaceModel:
    def __init__(self):
        self.vectorizer = TfidfVectorizer()

    def fit_transform(self, documents):
        """Transform the documents into tf-idf vectors."""
        tfidf_matrix = self.vectorizer.fit_transform(documents)
        return tfidf_matrix

    def query(self, query, document_vectors):
        """Transform the query and compute similarity to documents."""
        query_vec = self.vectorizer.transform([query])
        similarity_scores = cosine_similarity(query_vec, document_vectors).flatten()
        return similarity_scores

# Example usage
documents = [
    "the economy is improving",
    "sports are popular",
    "the government is stable",
    "football and basketball are sports",
    "the economy needs help",
    "governments support the economy"
]

# Initialize the Vector Space Model
vsm = VectorSpaceModel()

# Fit the model with the documents and convert them to vectors
document_vectors = vsm.fit_transform(documents)

# Define a query
query = "economy and government"

# Compute similarity scores
similarity_scores = vsm.query(query, document_vectors)

# Display the similarity scores for each document
for i, score in enumerate(similarity_scores):
    print(f"Document {i}: {documents[i]}")
```



```
print(f"Similarity Score: {score}\n")
Document 0: the economy is improving
Similarity Score: 0.1923760136208081

Document 1: sports are popular
Similarity Score: 0.0

Document 2: the government is stable
Similarity Score: 0.36518892421204585

Document 3: football and basketball are sports
Similarity Score: 0.3046832133490678

Document 4: the economy needs help
Similarity Score: 0.18090504710401742

Document 5: governments support the economy
Similarity Score: 0.18090504710401742
```

## 7.Project

**Title : “DETECTING FAKE AND REAL NEWS WITH NLP AND ML”**

### **ABSTRACT :**

The rapid proliferation of fake news across digital platforms poses significant challenges to public trust and societal well-being. This project aims to develop a machine learning-based system for detecting fake news using textual and topical features derived from news articles. The dataset employed contains both real and fake news articles, labeled accordingly, with features including article titles, content, subject categories, and publication dates. Using natural language processing (NLP) techniques, the project transforms these features into numerical representations through methods such as TF-IDF vectorization. Various classification algorithms, such as Logistic Regression, Support Vector Machines, and Random Forests, are employed to identify fake news. The system provides both batch and real-time prediction capabilities. Results are evaluated using accuracy, precision, recall, and F1-score metrics, demonstrating the effectiveness of the approach in distinguishing real from fake news. This project contributes to combating misinformation by offering an automated and scalable solution to fake news detection.

## **Tools Used :**

### **1.Python:**

The core programming language used to develop the emotion detection model, implement data preprocessing, and integrate machine learning algorithms.

### **2.Pycharm:**

PyCharm is an integrated development environment (IDE) designed specifically for Python programming, offering powerful code editing, debugging, and testing tools.

### **3.Scikit-learn:**

A machine learning library used for building the Logistic Regression model, including text vectorization through TF-IDF and model training.

### **4.NLTK (Natural Language Toolkit):**

A Python library for processing human language data (text). It is used for text preprocessing, including tokenization, stemming, and stopword removal.

### **5.Tkinter:**

Tkinter is Python's standard library for creating graphical user interfaces (GUIs). It allows developers to build windows and interactive elements like buttons, labels, and text boxes easily.

### **6.CSS :**

Basic CSS is used within Pycharm to style the interface and apply custom background images to the app for better visual appeal.

These tools collectively enable the emotion detection system to preprocess text, predict emotions, and deliver a rich, interactive user experience.

**Code :****Fake and Real news prediction:**

```
import numpy as np
import pandas as pd
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score
```

```
# Download stopwords if needed
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data] C:\Users\harsh\AppData\Roaming\nltk_data...
[nltk_data] Package stopwords is already up-to-date!
```

```
True
```

```
# Load the dataset
news_df = pd.read_csv('fake_real.csv') # Make sure the file exists in the directory
```

```
news_df.head()
```

	title	text	subject	date	label
0	John Lewis Has the PERFECT Birthday Message F...	Rep. John Lewis (D-GA) is a civil rights icon....	News	26-Oct-17	0
1	Nazis Whine Over Impending Release Of An Anti...	With the rise of Donald Trump has also come th...	News	08-Oct-17	0
2	Final Republican tax bill slashes U.S. corpora...	WASHINGTON (Reuters) - Congressional Republica...	politicsNews	15-Dec-17	1
3	Obama?s Head Of Disaster Relief TRASHES Trump...	It s no secret that Donald Trump s response to...	News	30-Sep-17	0
4	Nuclear plan backer denies Inauguration Day te...	WASHINGTON (Reuters) - A company promoting a p...	politicsNews	11-Dec-17	1

```
news_df.tail()
```

	title	text	subject	date	label
1995	Rupert Murdoch Told Roger Ailes To Help Elect...	A new report has just revealed that Fox News p...	News	19-Jul-17	0
1996	Republican U.S. Senators Rubio, Lee want boost...	WASHINGTON (Reuters) - Republican U.S. Senator...	politicsNews	09-Nov-17	1
1997	Trump Throws Hissy Fit After New York Times R...	Donald Trump woke up on Wednesday morning and ...	News	28-Jun-17	0
1998	Democrats Give Trump A Big F**ck You After He...	Donald Trump s Twitter feed is always a goldmi...	News	28-Nov-17	0
1999	Factbox: Big-ticket items at center of Congres...	(Reuters) - The U.S. Congress on Thursday appr...	politicsNews	21-Dec-17	1

## PREPROCESSING:

```
news_df.shape
```

```
(2000, 5)
```

```
# Fill missing values with a blank space
news_df = news_df.fillna(' ')
```

```
news_df.isna().sum()
```

```
title      0
text       0
subject    0
date       0
label      0
dtype: int64
```

```
# Combine the 'title' and 'text' columns into a single content column
news_df['content'] = news_df['title'] + " " + news_df['text']
```

news\_df

	title	text	subject	date	label	content
0	John Lewis Has the PERFECT Birthday Message F...	Rep. John Lewis (D-GA) is a civil rights icon....	News	26-Oct-17	0	John Lewis Has the PERFECT Birthday Message F...
1	Nazis Whine Over Impending Release Of An Anti...	With the rise of Donald Trump has also come th...	News	08-Oct-17	0	Nazis Whine Over Impending Release Of An Anti...
2	Final Republican tax bill slashes U.S. corpora...	WASHINGTON (Reuters) - Congressional Republi...	politicsNews	15-Dec-17	1	Final Republican tax bill slashes U.S. corpora...
3	Obama?s Head Of Disaster Relief TRASHES Trump...	It s no secret that Donald Trump s response to...	News	30-Sep-17	0	Obama?s Head Of Disaster Relief TRASHES Trump...
4	Nuclear plan backer denies Inauguration Day te...	WASHINGTON (Reuters) - A company promoting a p...	politicsNews	11-Dec-17	1	Nuclear plan backer denies Inauguration Day te...
...	...	...	...	...	...	...
1995	Rupert Murdoch Told Roger Ailes To Help Elect...	A new report has just revealed that Fox News p...	News	19-Jul-17	0	Rupert Murdoch Told Roger Ailes To Help Elect...
1996	Republican U.S. Senators Rubio, Lee want boost...	WASHINGTON (Reuters) - Republican U.S. Senator...	politicsNews	09-Nov-17	1	Republican U.S. Senators Rubio, Lee want boost...
1997	Trump Throws Hissy Fit After New York Times R...	Donald Trump woke up on Wednesday morning and ...	News	28-Jun-17	0	Trump Throws Hissy Fit After New York Times R...
1998	Democrats Give Trump A Big F**ck You After He...	Donald Trump s Twitter feed is always a goldmi...	News	28-Nov-17	0	Democrats Give Trump A Big F**ck You After He...
1999	Factbox: Big-ticket items at center of Congres...	(Reuters) - The U.S. Congress on Thursday appr...	politicsNews	21-Dec-17	1	Factbox: Big-ticket items at center of Congres...

2000 rows x 6 columns

news\_df['content']

```

0      John Lewis Has the PERFECT Birthday Message F...
1      Nazis Whine Over Impending Release Of An Anti...
2      Final Republican tax bill slashes U.S. corpora...
3      Obama?s Head Of Disaster Relief TRASHES Trump...
4      Nuclear plan backer denies Inauguration Day te...
...
1995    Rupert Murdoch Told Roger Ailes To Help Elect...
1996    Republican U.S. Senators Rubio, Lee want boost...
1997    Trump Throws Hissy Fit After New York Times R...
1998    Democrats Give Trump A Big F**ck You After He...
1999    Factbox: Big-ticket items at center of Congres...
Name: content, Length: 2000, dtype: object

```

## STEMMING:

```

# Stemming function
ps = PorterStemmer()
def stemming(content):
    stemmed_content = re.sub('[^a-zA-Z]', ' ', content) # Remove non-alphabetic characters
    stemmed_content = stemmed_content.lower()           # Lowercase the text
    stemmed_content = stemmed_content.split()           # Tokenize by splitting
    stemmed_content = [ps.stem(word) for word in stemmed_content if word not in stopwords.words('english')]
    stemmed_content = ' '.join(stemmed_content)         # Join back into a single string
    return stemmed_content

```

```
# Apply stemming to the 'content' column
news_df['content'] = news_df['content'].apply(stemming)
```

## separating the data and label:

```
# Separate the features (X) and the target label (y)
X = news_df['content'].values
y = news_df['label'].values
```

## converting the textual data to numerical data:

```
# Convert text data into numerical data using TF-IDF vectorization
vectorizer = TfidfVectorizer()
X_tfidf = vectorizer.fit_transform(X)
```

## Splitting the dataset to training & test data:

```
# Split the data into training and testing sets (80% training, 20% testing)
X_train_tfidf, X_test_tfidf, y_train, y_test = train_test_split(X_tfidf, y, test_size=0.2, stratify=y, random_state=1)
```

## Training the Model:

```
# Train Naive Bayes classifier
model = MultinomialNB()
model.fit(X_train_tfidf, y_train)
```

```
▼ MultinomialNB
MultinomialNB()
```

```
# Predict on the test set
y_pred = model.predict(X_test_tfidf)
```

```
# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Model Accuracy: {accuracy * 100:.2f}%")
```

Model Accuracy: 95.25%

## INTERFACE:

```
import tkinter as tk
from tkinter import messagebox, Scrollbar, Text
from tkinter.font import Font

# Function to classify news
def classify_news():
    user_input = entry.get("1.0", "end-1c") # Get the user input from the text box
    if not user_input.strip():
        messagebox.showwarning("Input Error", "Please enter some news text.")
        return

    user_input_tfidf = vectorizer.transform([user_input]) # Transform input to match training data
    prediction = model.predict(user_input_tfidf)

    if prediction == 1:
        result = "This news is Real."
        status_label.config(text="Prediction: Real News", fg="green")
    else:
        result = "This news is Fake."
        status_label.config(text="Prediction: Fake News", fg="red")

    messagebox.showinfo("Prediction", result)

# Function to reset the input area and status label
def reset_input():
    entry.delete("1.0", "end") # Clear the text area
    status_label.config(text="Prediction: Awaiting input...", fg="black") # Reset status label

# Create the main window
window = tk.Tk()
window.title("Fake News Detector")
window.geometry("600x400") # Set window size
```



```
# Create the main window
window = tk.Tk()
window.title("Fake News Detector")
window.geometry("600x400") # Set window size

# Set background color and padding
window.configure(bg="#F0F0F0")
window.resizable(False, False)

# Set font styles
title_font = Font(family="Helvetica", size=16, weight="bold")
label_font = Font(family="Arial", size=12)
button_font = Font(family="Arial", size=10, weight="bold")

# Title Label
title_label = tk.Label(window, text="News Detector", font=title_font, bg="#F0F0F0", pady=10)
title_label.pack()

# Add a label for instructions
label = tk.Label(window, text="Enter the news text below for classification:", font=label_font, bg="#F0F0F0")
label.pack()

# Frame to contain the Text widget and Scrollbar
text_frame = tk.Frame(window, bg="#F0F0F0")
text_frame.pack(pady=10)

# Scrollbar for the text box
scrollbar = Scrollbar(text_frame)
scrollbar.pack(side=tk.RIGHT, fill=tk.Y)

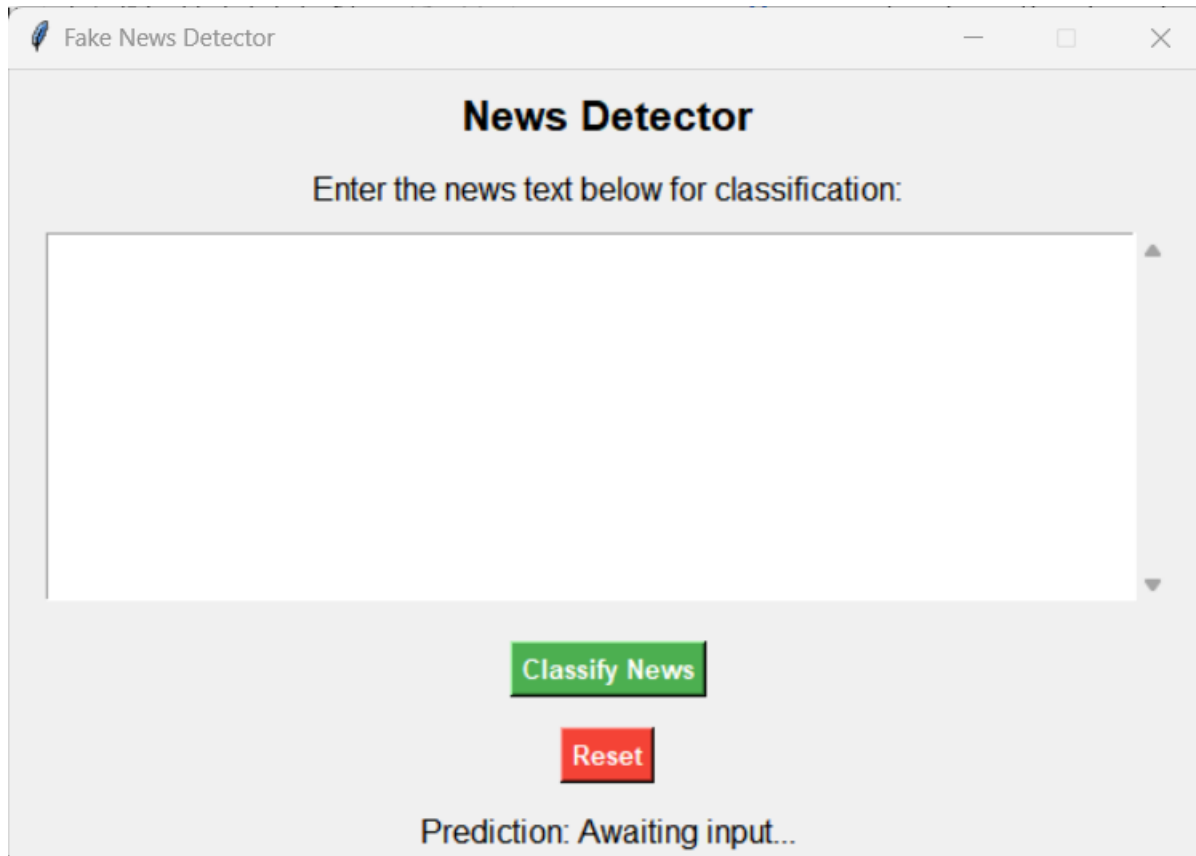
# Text entry box for input (Scrollable)
entry = Text(text_frame, height=10, width=60, font=("Arial", 12), yscrollcommand=scrollbar.set)
entry.pack(side=tk.LEFT)
scrollbar.config(command=entry.yview)

# Classify button with padding and custom font
classify_button = tk.Button(window, text="Classify News", font=button_font, bg="#4CAF50", fg="white", command=classify_news)
classify_button.pack(pady=10)

# Reset button to clear input and status
reset_button = tk.Button(window, text="Reset", font=button_font, bg="#f44336", fg="white", command=reset_input)
reset_button.pack(pady=5)

# Status label to display prediction result dynamically
status_label = tk.Label(window, text="Prediction: Awaiting input...", font=label_font, bg="#F0F0F0", pady=10)
status_label.pack()

# Run the application
window.mainloop()
```

**OUTPUT :**

The screenshot shows a web browser window with the title 'Fake News Detector'. The main heading is 'News Detector'. Below the heading is a text input area with the placeholder text 'Enter the news text below for classification:'. There are two buttons: a green 'Classify News' button and a red 'Reset' button. At the bottom, the text 'Prediction: Awaiting input...' is displayed.

