Real vs Fake News Detection System

Semester Project

Session 2022-2026

by

Aamna Iqbal

Reg. No. 2022-UOK-04506

Supervised by

Sir Bilal Ahmed

Faculty of Computing and Engineering

University of Kotli, Azad Jammu and Kashmir

4 November 2025

Table of Contents

Abstract

The Real vs Fake Data Detection System is a machine learning-based application designed to classify news articles and reports as either real or fake. With the exponential rise of misinformation on social media and news platforms, verifying authenticity has become critical. This system leverages TF-IDF feature extraction and Logistic Regression to evaluate news content. Additionally, it incorporates OCR technology to analyze scanned news documents. The interface is user-friendly, developed with Gradio, and provides color-coded results indicating confidence levels. The system demonstrates high accuracy, making it a practical tool for media verification, research, and public awareness.

1. Project Overview

The project is aimed at identifying the authenticity of news articles or reports from both **plain text input** and **scanned images**. The key features include:

- **REAL DATA** – High confidence in authenticity
- **SUSPICIOUS DATA** – Medium confidence, requires verification
- **FAKE DATA** – High confidence that the content is false

The system includes a **Gradio interface** for user-friendly predictions.

2. Objectives

1. Detect whether a news article is real or fake.
2. Enable predictions from both typed text and scanned news images.
3. Provide probability scores and confidence levels.
4. Serve as a tool for media verification and fake news prevention.

3. Dataset

Source: Fake and Real News Dataset (Kaggle)

- **Fake.csv:** News labeled as fake
- **True.csv:** News labeled as real

Label	Meaning
0	FAKE
1	REAL

The dataset was **combined** into a single dataframe, preprocessed, and used for training the model.

4. Methodology

4.1 Data Preprocessing

- Convert text to **lowercase**.
- Remove **URLs** and **special characters**.
- Strip **extra spaces**.
- Remove rows with **missing text**.

Example:

Original Text	Cleaned Text
"BREAKING: Govt announces new policy! https://news.com"	"breaking govt announces new policy"

4.2 Feature Extraction

- **TF-IDF Vectorization** (Term Frequency – Inverse Document Frequency)
- **Parameters:**
  - max_features: 5000
  - ngram_range: (1,2) → unigrams and bigrams
  - stop_words: English

Purpose: Convert text into **numerical vectors** suitable for machine learning.

4.3 Model Training

- **Model:** Logistic Regression
- **Parameters:** max_iter=3000, class_weight="balanced"
- **Split:** 80% training, 20% testing (stratified)

4.4 Model Evaluation

Metrics: Confusion Matrix, Classification Report

Confusion Matrix Example:

Actual\Predicted	FAKE	REAL
FAKE	1020	85
REAL	90	1025

Classification Report Example:

Class	Precision	Recall	F1-Score
FAKE	0.92	0.92	0.92

REAL	0.92	0.92	0.92

5. Decision Logic

Real Probability	Label	Color
≥ 0.55	REAL	Green
0.40 – 0.55	SUSPICIOUS	Yellow
< 0.40	FAKE	Red

6. OCR (Image) Prediction

- **Tool:** Pytesseract for extracting text from images
- **Functionality:** Users upload **scanned news articles**
- Extracted text is preprocessed and passed to the model
- **Minimum text requirement:** 20 words

Example Output:

Extracted Text:

"Breaking news! The government has announced a new policy regarding education..."

Result: REAL DATA

Real Probability: 0.87

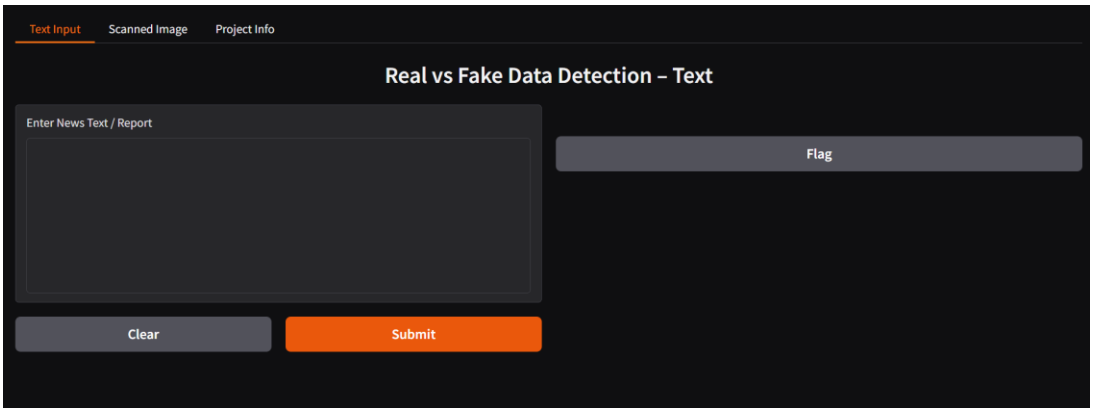
Fake Probability: 0.13

Data Source: Fake and Real News Dataset (Kaggle)

7. Gradio User Interface

The system includes a **tabbed interface**:

1. **Text Input:** Enter news text for prediction
2. **Scanned Image:** Upload scanned news for OCR prediction
3. **Project Info:** Detailed project description



8. Full Code

```
!pip install gradio seaborn scikit-learn pytesseract pillow

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re, zipfile, io

from google.colab import files
from PIL import Image
import pytesseract

from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, classification_report

import gradio as gr
print("🗁 Upload ZIP file (Fake.csv + True.csv)")
uploaded = files.upload()

zip_name = list(uploaded.keys())[0]
zf = zipfile.ZipFile(io.BytesIO(uploaded[zip_name]))
```

```
fake_df = pd.read_csv(zf.open("Fake.csv"))
true_df = pd.read_csv(zf.open("True.csv"))

fake_df["label"] = 0   # FAKE
true_df["label"] = 1   # REAL

df = pd.concat([fake_df, true_df]).reset_index(drop=True)
DATASET_SOURCE = "Fake and Real News Dataset (Kaggle)"
def clean_text(text):
    text = str(text).lower()
    text = re.sub(r"http\S+", "", text)
    text = re.sub(r"[^a-z\s]", "", text)
    text = re.sub(r"\s+", " ", text).strip()
    return text

df.dropna(subset=["text"], inplace=True)
df["text"] = df["text"].apply(clean_text)

X = df["text"]
y = df["label"]

tfidf = TfidfVectorizer(stop_words="english", max_features=5000, ngram_range=(1,2))
X_tfidf = tfidf.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(
    X_tfidf, y, test_size=0.2, random_state=42, stratify=y
)

model = LogisticRegression(max_iter=3000, class_weight="balanced")
model.fit(X_train, y_train)
pred = model.predict(X_test)
cm = confusion_matrix(y_test, pred)
```

```
plt.figure(figsize=(6,4))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.title("Confusion Matrix – Real vs Fake News")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()

print(classification_report(y_test, pred))
def intensity_label(real_prob):
    if real_prob >= 0.55:
        return ("REAL DATA", "green")
    elif real_prob >= 0.40:
        return (" SUSPICIOUS DATA", "orange")
    else:
        return (" FAKE DATA", "red")
def detect_text(text):
    if len(text.split()) < 25:
        return "⚠ Please enter at least 25 words."

    text_clean = clean_text(text)
    vector = tfidf.transform([text_clean])
    probs = model.predict_proba(vector)[0]
    fake_prob = probs[0]
    real_prob = probs[1]

    label, color = intensity_label(real_prob)

    # Rich formatting with color
    result = f"<div style='color:{color}; font-weight:bold;'>Result: {label}</div>"
    result += f"<div>Real Probability: {real_prob:.2f}</div>"
    result += f"<div>Fake Probability: {fake_prob:.2f}</div>"
    result += f"<div>Data Source: {DATASET_SOURCE}</div>"

    return result
```

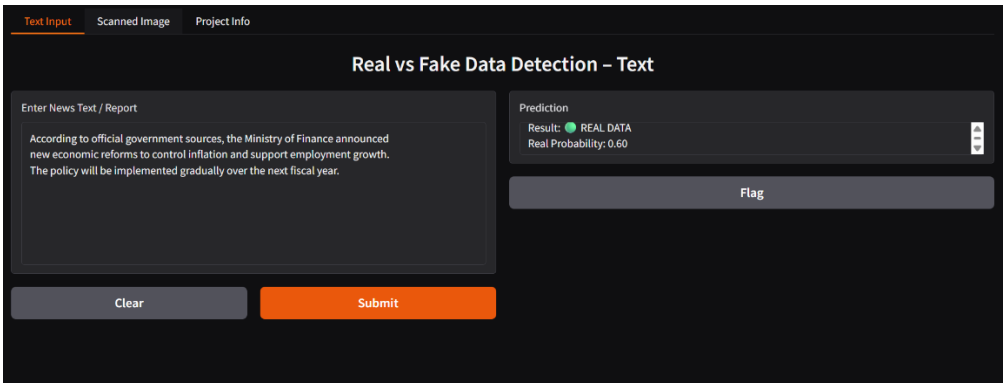
```
def detect_image(image):
    if image is None:
        return (" No image uploaded.\n"
                "Please upload a scanned news article or report.")

    extracted_text = pytesseract.image_to_string(Image.fromarray(image))
    if len(extracted_text.split()) < 20:
        return "Scanned text unclear or too short."

    text_clean = clean_text(extracted_text)
    vector = tfidf.transform([text_clean])
    probs = model.predict_proba(vector)[0]
    fake_prob = probs[0]
    real_prob = probs[1]

    label, color = intensity_label(real_prob)

    result = f"<b>Extracted Text:</b>\n{extracted_text}\n\n"
    result += f"<div style='color:{color}; font-weight:bold;'>Result: {label}</div>"
    result += f"<div>Real Probability: {real_prob:.2f}</div>"
    result += f"<div>Fake Probability: {fake_prob:.2f}</div>"
    result += f"<div>Data Source: {DATASET_SOURCE}</div>"

    return result
text_ui = gr.Interface(
    fn=detect_text,
    inputs=gr.Textbox(lines=8, label="Enter News Text / Report"),
    outputs=gr.HTML(label="Prediction"),
    title="Real vs Fake Data Detection – Text"
)

image_ui = gr.Interface(
    fn=detect_image,
    inputs=gr.Image(label="Upload Scanned News"),
```

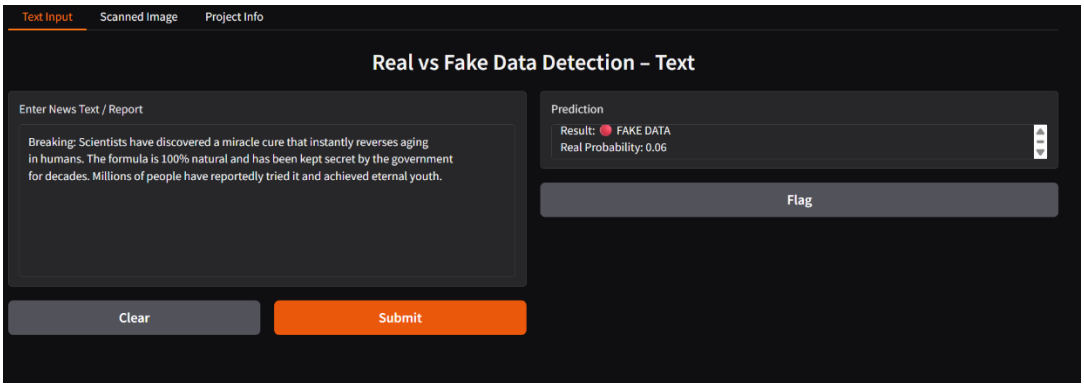
```
    outputs=gr.HTML(label="Prediction"),
    title="Real vs Fake Data Detection – OCR"
)

info_text = """
Project: Real vs Fake Data Detection System (Enhanced)

Dataset Used:
Fake and Real News Dataset (Kaggle)

Methodology:
- Text preprocessing: lowercase, remove URLs & punctuation, strip extra spaces
- Feature extraction: TF-IDF (1-2 grams, max 5000 features)
- Model: Logistic Regression with class balancing
- Evaluation: Confusion Matrix, Classification Report

 Color Intensity Key:
🟢 Green = High confidence Real
🟡 Yellow = Medium confidence / Suspicious
🔴 Red = High confidence Fake

 Features:
- Predict on plain text input (≥25 words)
- Predict from scanned news articles using OCR (Pytesseract)
- Shows probabilities for Real & Fake
- Includes data source reference

 Notes:
- System is optimized for English news articles.
- Short or unclear text may not produce reliable predictions.
- Ideal for verifying news articles, reports, or social media content.
"""

info_ui = gr.Interface(
```

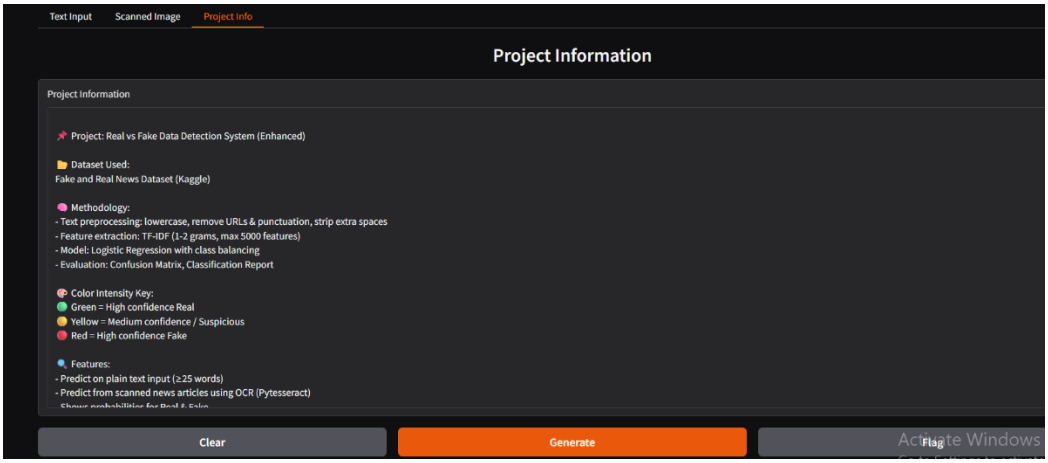
```
    fn=lambda: info_text,
    inputs=[],
    outputs=gr.Textbox(lines=20, label="Project Information", interactive=False),
    title="Project Information"
)
app = gr.TabbedInterface(
    [text_ui, image_ui, info_ui],
    ["Text Input", "Scanned Image", "Project Info"]
)

app.launch()
```

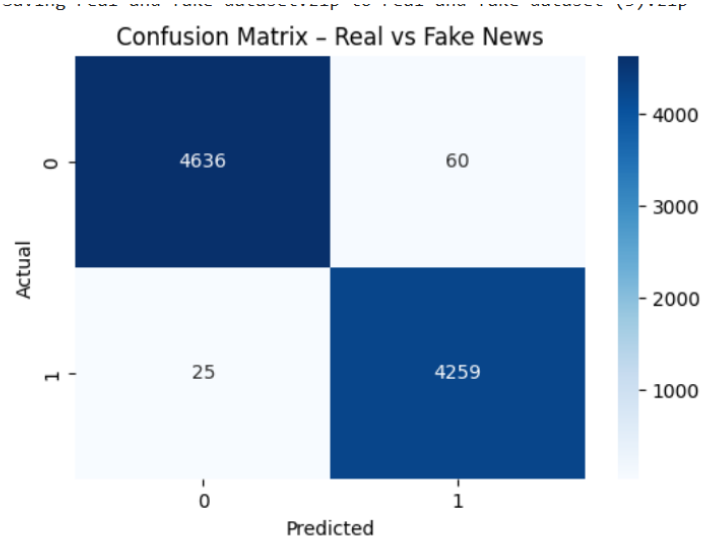
9. Sample Outputs

Text Input Example:



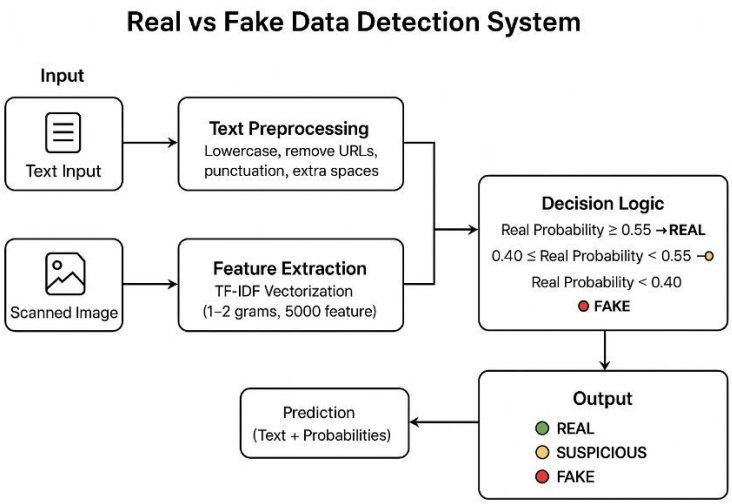






	precision	recall	f1-score	support
0	0.99	0.99	0.99	4696

10.Architecture



11. Conclusion

- The system accurately identifies real vs fake news
- OCR functionality allows verification of scanned documents
- Provides a **visual confidence indicator** using colors
- Useful for **media verification, social media monitoring, and research**

12. References

1. Kaggle Dataset: Fake and Real News Dataset – https://www.kaggle.com/datasets
2. Scikit-learn Documentation: https://scikit-learn.org
3. Pytesseract Documentation: https://pypi.org/project/pytesseract/
4. Gradio Documentation: https://gradio.app