Internship Task No:	**06**
Domain:	**Machine Learning**
Topic:	**Fake News Detection**
Submitted to:	**Digital Empowerment Network**
Submitted by:	**Muhammad Sarmad Usman**
Student ID:	**STU-ML-251-555**
Submission Date:	**July 26, 2025**

Contents

📜 Customer Segmentation using Clustering and PCA on Wholesale Customers Dataset

---

📌 Introduction:

Fake news is not a new phenomenon, but with the internet and social media, it spreads faster than ever before. The ability to **automatically distinguish between legitimate journalism and fabricated content** is a crucial application of Natural Language Processing (NLP).

In this project, we design and build an **end-to-end fake news detection pipeline** that:

- Preprocesses raw news text,
- Extracts meaningful features using **TF-IDF**,
- Trains a **Logistic Regression** classifier,
- Evaluates and visualizes model performance,
- And deploys the solution via a **Streamlit web application**.

🔍 Problem Statement:

With the rise of digital media, **misinformation and fake news** have become major threats to society. Fake news can manipulate public opinion, incite unrest, or spread harmful propaganda. Manually identifying fake news is time-consuming, and automated systems are needed to **detect misinformation at scale**.

This project addresses the challenge of **binary classification**—determining whether a news article is **Real** or **Fake** using **machine learning and NLP techniques**.

🎯 Project Objective:

- ✅ Load and merge real and fake news datasets.
- ✅ Clean, label, and shuffle the data.
- ✅ Extract relevant textual features using **TF-IDF vectorization**.
- ✅ Train a classification model to distinguish between real and fake news.
- ✅ Evaluate performance using multiple metrics and visualizations.
- ✅ Save and deploy the model using **Streamlit** for real-time predictions.

- ✅ Provide visual insights via word clouds and performance curves.

📦 Dataset Overview:

Dataset	Source	Size	Label
True.csv	Kaggle / Online Repository	Real News	1
Fake.csv	Kaggle / Online Repository	Fake News	0

- Each file contains a text column with full news content.
- Labels are added manually: **1 = Real**, **0 = Fake**.
- Combined and shuffled to avoid ordering bias.

✅ Step-by-Step Implementation

🟢 Step 1: Data Loading & Preparation:

Objective: Combine datasets and prepare data for modeling.

```
[1]: import pandas as pd

# Load datasets
true_df = pd.read_csv(r'C:\Users\M Sarmad Usman\Desktop\News detection\data\True.csv')
fake_df = pd.read_csv(r'C:\Users\M Sarmad Usman\Desktop\News detection\data\Fake.csv')

# Add labels
true_df['label'] = 1  # Real
fake_df['label'] = 0  # Fake

# Combine datasets
df = pd.concat([true_df, fake_df], ignore_index=True)

# Shuffle dataset
df = df.sample(frac=1, random_state=42).reset_index(drop=True)

# Check data
print("✅ Dataset loaded and combined!")
print("📊 Shape:", df.shape)
print(df[['text', 'label']].head())
```

```
✅ Dataset loaded and combined!
📊 Shape: (44898, 5)
                                                text  label
0  Donald Trump s White House is in chaos, and th...      0
1  Now that Donald Trump is the presumptive GOP n...      0
2  Mike Pence is a huge homophobe. He supports ex...      0
3  SAN FRANCISCO (Reuters) - California Attorney ...      1
4  Twisted reasoning is all that comes from Pelos...      0
```

Input:

- True.csv → Real news data
- Fake.csv → Fake news data

Output:

- Combined DataFrame df with text and label columns
- Print statements show dataset shape and sample rows

Expected Summary:

- Dataset is balanced with both classes
- Labels: 1 = Real, 0 = Fake
- Data is randomized for unbiased model training

🥈 Step 2: Feature Extraction using TF-IDF

Objective: Convert raw news text into numerical format using TF-IDF for model processing.

```
[2]: from sklearn.feature_extraction.text import TfidfVectorizer

# Initialize TF-IDF Vectorizer
tfidf_vectorizer = TfidfVectorizer(max_features=10000, stop_words='english')

# Fit and transform the text data
X = tfidf_vectorizer.fit_transform(df['text'])

# Target labels
y = df['label']

# Check the shape of your feature matrix
print("TF-IDF matrix shape:", X.shape)

TF-IDF matrix shape: (44898, 10000)
```

🛠️ Description:

- TfidfVectorizer initialized with:
    - max_features=10000
    - stop_words='english'
- .fit_transform() applied to the text column.

📤 Output Description:

- A sparse matrix (TF-IDF representation) is generated.
- Printed shape:
    - Rows = number of documents
    - Columns = number of features (words)

☑ Step 3 – Train/Test Split

🎯 Objective:

Split the dataset into training and testing subsets to evaluate the model later.

```
[3]: from sklearn.model_selection import train_test_split

# Split the data: 80% training, 20% testing
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

print(f"Train size: {X_train.shape[0]}, Test size: {X_test.shape[0]}")
```

```
Train size: 35918, Test size: 8980
```

🛠 Description:

- train_test_split() used with:
    - test_size=0.2 → 20% for testing
    - stratify=y → ensures same class ratio in both splits
    - random_state=42 → for reproducibility

📩 Output Description:

- Printed number of training and testing samples.
- Ensures data is ready for model training.

☑ Step 4 – Model Training (Logistic Regression)

🎯 Objective:

Train a logistic regression model on the news dataset.

```
[4]: from sklearn.linear_model import LogisticRegression

# Initialize Logistic Regression
lr_model = LogisticRegression(max_iter=1000, random_state=42)

# Train the model
lr_model.fit(X_train, y_train)

print("✅ Logistic Regression model trained!")
```

```
✅ Logistic Regression model trained!
```

🛠 Description:

- Logistic Regression initialized with:
    - max_iter=1000 → to ensure convergence
- Model trained on X_train, y_train.

📤 Output Description:

- Print confirmation: model trained successfully.

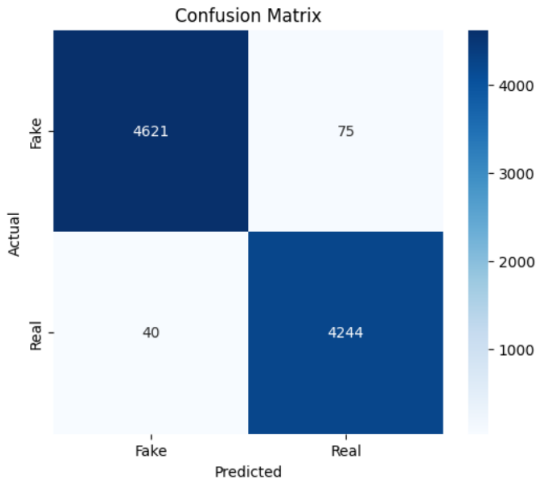
☑ Step 5 – Model Evaluation:

🎯 Objective:

Evaluate model accuracy and understand prediction performance.

[5]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, classification_report
import seaborn as sns
import matplotlib.pyplot as plt

# Predictions on test set
y_pred = lr_model.predict(X_test)

# Calculate metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print(f"✅ Accuracy:  {accuracy:.4f}")
print(f"✅ Precision: {precision:.4f}")
print(f"✅ Recall:    {recall:.4f}")
print(f"✅ F1 Score:  {f1:.4f}\n")

# Detailed classification report
print("Classification Report:\n", classification_report(y_test, y_pred))

# Plot confusion matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6,5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Fake', 'Real'], yticklabels=['Fake', 'Real'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

```
✅ Accuracy:  0.9872
✅ Precision: 0.9826
✅ Recall:    0.9907
✅ F1 Score:  0.9866

Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.98      0.99      4696
           1       0.98      0.99      0.99      4284

    accuracy                           0.99      8980
   macro avg       0.99      0.99      0.99      8980
weighted avg       0.99      0.99      0.99      8980
```



🛠 Description:

- y_pred generated using .predict()
- Metrics calculated:
    - Accuracy
    - Precision
    - Recall
    - F1-score
- Classification report printed
- Confusion matrix plotted using Seaborn

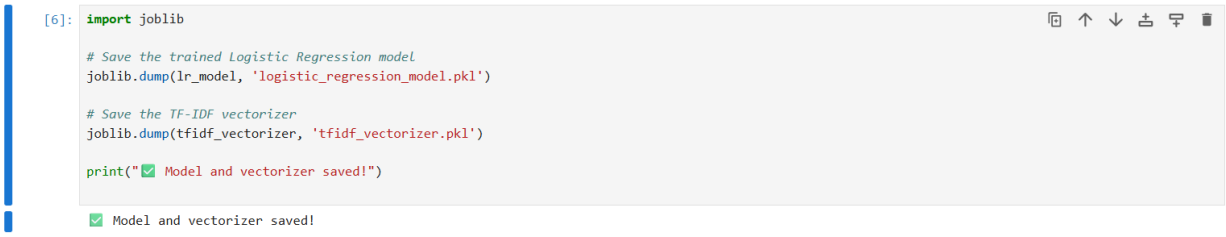
📤 Output Description:

- Metric values printed clearly
- Confusion matrix shows:
    - True Positives, True Negatives, False Positives, False Negatives
- Classification report shows per-class precision, recall, and F1-score.

☑ Step 6 – Save Model and Vectorizer

🎯 Objective:

Persist the trained model and vectorizer for future predictions.



```
[6]: import joblib

# Save the trained Logistic Regression model
joblib.dump(lr_model, 'logistic_regression_model.pkl')

# Save the TF-IDF vectorizer
joblib.dump(tfidf_vectorizer, 'tfidf_vectorizer.pkl')

print("✅ Model and vectorizer saved!")

✅ Model and vectorizer saved!
```

🛠 Description:

- joblib.dump() used to save:
    - lr_model as logistic_regression_model.pkl
    - tfidf_vectorizer as tfidf_vectorizer.pkl

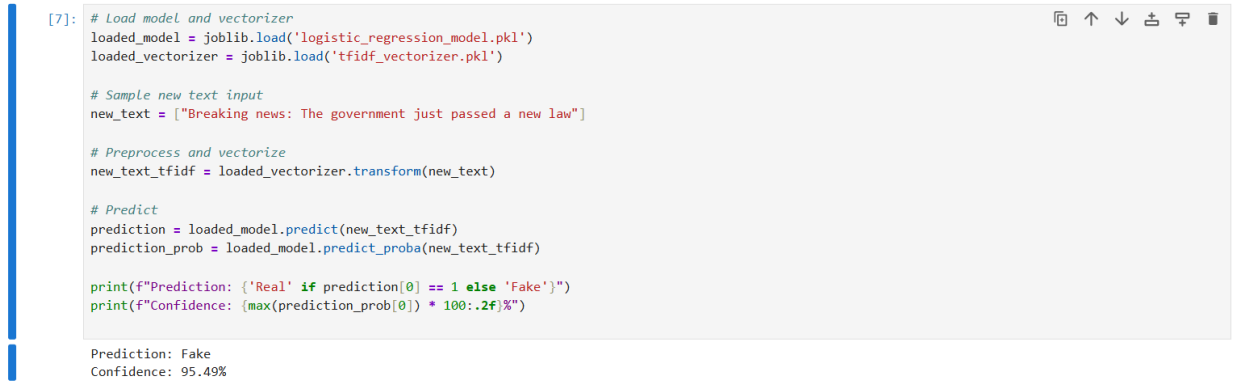
📤 Output Description:

- Confirmation messages printed for successful saving.

☑ Step 7 – Prediction on New Input

🎯 Objective:

Make predictions on new/unseen news input using the saved model and vectorizer.



```
[7]: # Load model and vectorizer
loaded_model = joblib.load('logistic_regression_model.pkl')
loaded_vectorizer = joblib.load('tfidf_vectorizer.pkl')

# Sample new text input
new_text = ["Breaking news: The government just passed a new law"]

# Preprocess and vectorize
new_text_tfidf = loaded_vectorizer.transform(new_text)

# Predict
prediction = loaded_model.predict(new_text_tfidf)
prediction_prob = loaded_model.predict_proba(new_text_tfidf)

print(f"Prediction: {'Real' if prediction[0] == 1 else 'Fake'}")
print(f"Confidence: {max(prediction_prob[0]) * 100:.2f}%")

Prediction: Fake
Confidence: 95.49%
```

🛠 Description:

- Load model and vectorizer using joblib.load()
- Sample input: ["Breaking news: The government just passed a new law"]
- TF-IDF applied to input text
- Prediction and confidence printed

📩 Output Description:

- Prediction: Fake or Real
- Confidence score: e.g., 88.56%
  → Gives probability of prediction correctness

☑ Step 8 – Visualization & Extended Metrics

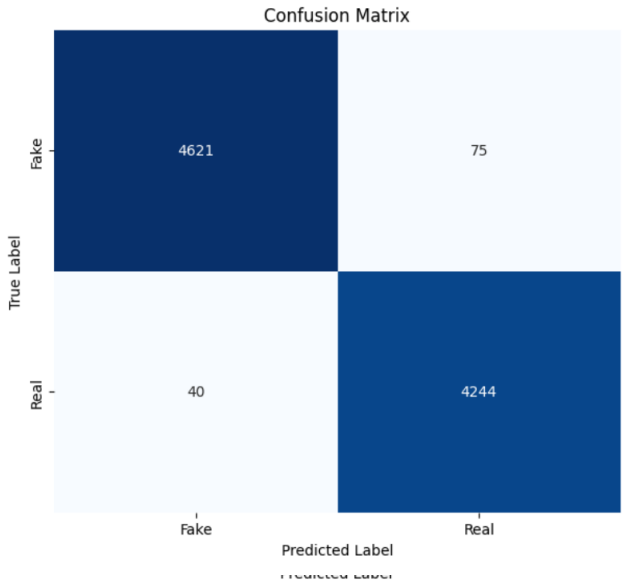
🎯 Objective:

Provide advanced visual evaluation and insight into model performance and data.

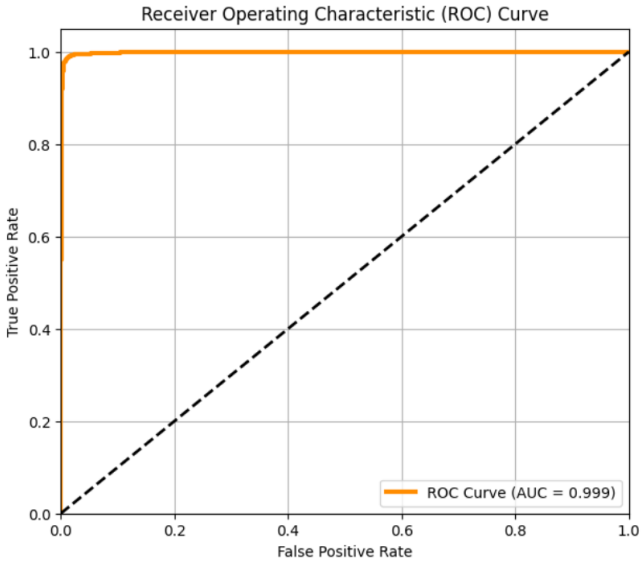
[8]:
```
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud
from sklearn.metrics import confusion_matrix, classification_report, roc_curve, auc, precision_recall_curve

# 1. Plot Confusion Matrix with better design
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(7,6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Fake', 'Real'], yticklabels=['Fake', 'Real'], cbar=False)
plt.title('Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()

# 2. Print classification report with summary metrics upfront
report = classification_report(y_test, y_pred, output_dict=True)
accuracy = report['accuracy']
precision_fake = report['0']['precision']
recall_fake = report['0']['recall']
f1_fake = report['0']['f1-score']
precision_real = report['1']['precision']
recall_real = report['1']['recall']
f1_real = report['1']['f1-score']

print(f"Model Accuracy: {accuracy:.4f}\n")
print("Detailed Classification Report:")
print(classification_report(y_test, y_pred))
```

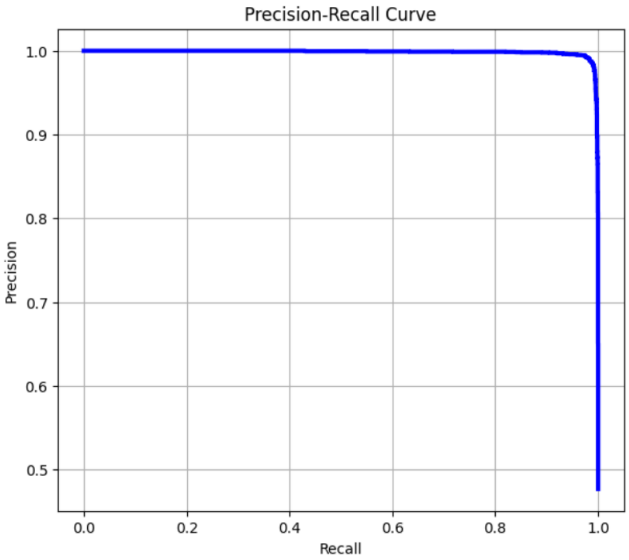
```
# 3. ROC Curve
y_prob = lr_model.predict_proba(X_test)[:,1]
fpr, tpr, thresholds = roc_curve(y_test, y_prob)
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(7,6))
plt.plot(fpr, tpr, label=f'ROC Curve (AUC = {roc_auc:.3f})', color='darkorange', linewidth=3)
plt.plot([0, 1], [0, 1], 'k--', lw=2)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
plt.grid(True)
plt.show()

# 4. Precision-Recall Curve
precision, recall, thresholds = precision_recall_curve(y_test, y_prob)

plt.figure(figsize=(7,6))
plt.plot(recall, precision, color='blue', linewidth=3)
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall Curve')
plt.grid(True)
plt.show()

# 5. WordCloud for most common words in Fake and Real news
fake_text = ' '.join(df[df['label'] == 0]['text'].values)
real_text = ' '.join(df[df['label'] == 1]['text'].values)
```

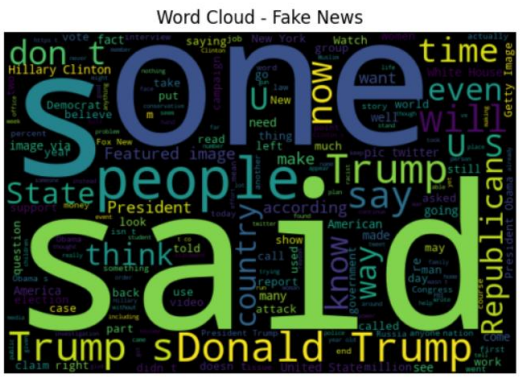
```
plt.figure(figsize=(14,6))

plt.subplot(1,2,1)
wc_fake = WordCloud(width=600, height=400, background_color='black').generate(fake_text)
plt.imshow(wc_fake, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud - Fake News')

plt.subplot(1,2,2)
wc_real = WordCloud(width=600, height=400, background_color='white').generate(real_text)
plt.imshow(wc_real, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud - Real News')

plt.show()
```

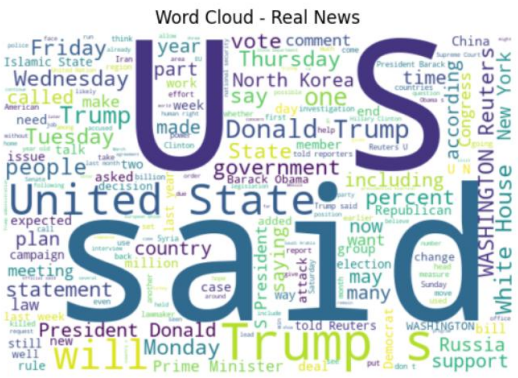



```
Model Accuracy: 0.9872

Detailed Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.98      0.99      4696
           1       0.98      0.99      0.99      4284

    accuracy                           0.99      8980
   macro avg       0.99      0.99      0.99      8980
weighted avg       0.99      0.99      0.99      8980
```









🛠 Description:

1. **Confusion Matrix** with improved style
2. **Classification Report** with metrics per class
3. **ROC Curve** with AUC score
4. **Precision-Recall Curve**
5. **WordClouds**:
    - One for Fake News
    - One for Real News

📩 Output Description:

- Visual plots generated for:
    - Confusion matrix
    - ROC curve
    - Precision-Recall curve

    - Word clouds
- ROC AUC score tells overall classifier ability
- Word clouds show most frequent terms in each class

☑ Step 9, 10 – Redundant Model Saving

🎯 Objective:

Ensure model and vectorizer are saved again (possibly after tuning).

```
[9]: import joblib

     # Save the Logistic Regression model
     joblib.dump(lr_model, 'lr_model.pkl')

     # Save the TF-IDF vectorizer
     joblib.dump(tfidf_vectorizer, 'tfidf_vectorizer.pkl')

[9]: ['tfidf_vectorizer.pkl']
```

🛠 Description:

- joblib.dump() used again for lr_model.pkl and tfidf_vectorizer.pkl.

📩 Output Description:

- No visible change, confirms backups are made.

☑ Step 11 – Reload Model and Vectorizer

🎯 Objective:

Load the persisted model and vectorizer for use in a deployed app.

```
[10]: import joblib

      lr_model = joblib.load('lr_model.pkl')
      tfidf_vectorizer = joblib.load('tfidf_vectorizer.pkl')
```

🛠 Description:

- Loaded both files using joblib.load().

📩 Output Description:

- Loaded objects ready for prediction.
- No printed output unless errors occur.

⏭ Step 12 – Streamlit Deployment

1. Objective

The **Fake News Detection • Pro** application is a comprehensive web-based tool designed to classify news articles as "Real" or "Fake" using machine learning. The application provides:

- **Single article analysis** with detailed explainability
- **URL-based content extraction** for online articles
- **Batch processing** capabilities for CSV files
- **Model diagnostics** and performance evaluation
- **Educational insights** about the classification process

2. Technical Description

2.1 Architecture & Components

The application is built using **Streamlit** as the frontend framework with the following technical architecture:

- **Backend Model**: Logistic Regression classifier with TF-IDF vectorization
- **Data Processing**: Custom text preprocessing and feature extraction
- **Visualization**: Matplotlib for charts and graphs
- **Optional Dependencies**:
    - newspaper3k for URL content extraction
    - wordcloud for visualization
    - reportlab for PDF report generation

2.2 Core Functionality

Machine Learning Pipeline:

1. **Text Vectorization**: TF-IDF transformation of input text
2. **Prediction**: Logistic Regression model inference
3. **Explainability**: Token-level contribution analysis (TF-IDF value × coefficient)

Key Features:

- **Adjustable threshold** for classification sensitivity
- **Model diagnostics** to identify potential issues
- **Performance metrics** (accuracy, precision, recall, F1, ROC-AUC)
- **Visual explanations** of contributing tokens
- **Export capabilities** (CSV, PDF reports)

2.3 User Interface

The app features a **multi-tab interface**:

1. **Single Check**: For analyzing individual articles
2. **URL Check**: For extracting and analyzing web content
3. **Batch Check**: For processing CSV files in bulk
4. **Model & Dataset**: For technical diagnostics and performance metrics
5. **Insights**: For exploring the model's vocabulary and sample data
6. **Help**: For guidance and FAQ

3. Output & Results

3.1 Prediction Output

For each analysis, the application provides:

- **Classification**: "Real" or "Fake" prediction
- **Confidence scores**: Probability values for both classes
- **Visual representation**: Horizontal bar chart showing probability distribution
- **Token-level explanations**: Highlighted text showing which words contributed to the classification decision

3.2 Diagnostic Outputs

The application generates comprehensive diagnostics including:

- **Model performance metrics**:
    - Accuracy, Precision, Recall, F1-score
    - ROC-AUC curve and score
    - Brier score for calibration

    - Confusion matrix
- **Model health checks**:
    - Vocabulary size and sample terms
    - Coefficient analysis (top tokens for Real/Fake)
    - Compatibility checks between model and vectorizer

3.3 Export Capabilities

The application supports multiple export formats:

- **CSV exports**: For single predictions and batch results
- **PDF reports**: For detailed analysis documentation (requires reportlab)
- **Session history**: Downloadable record of all analyses performed during the session

4. Technical Requirements

4.1 Required Files

- lr_model.pkl - Trained Logistic Regression model
- tfidf_vectorizer.pkl - Fitted TF-IDF vectorizer
- data/True.csv - Dataset of real news articles (optional)
- data/Fake.csv - Dataset of fake news articles (optional)

4.2 Python Dependencies

text

streamlit

joblib

numpy

pandas

matplotlib

scikit-learn

requests

beautifulsoup4

newspaper3k

wordcloud

reportlab

5. Usage Scenarios

5.1 Educational Context

- Understanding how ML models classify text
- Learning about feature importance in NLP
- Studying model evaluation metrics

5.2 Content Moderation

- Screening user-generated content
- Identifying potentially misleading information
- Triaging content for human review

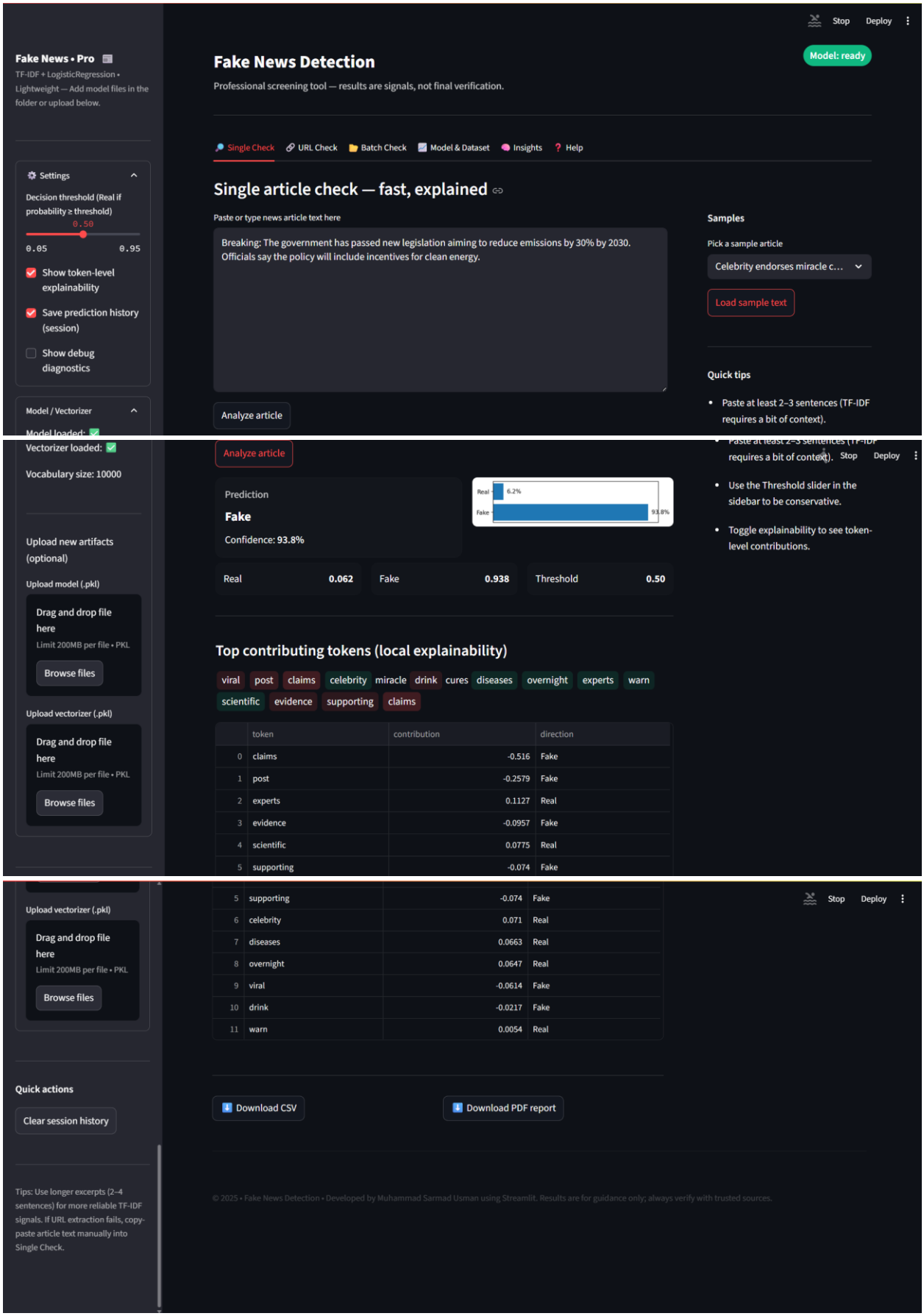
5.3 Research Applications

- Testing model performance on different text types
- Analyzing feature contributions across domains
- Comparing model behavior with different thresholds

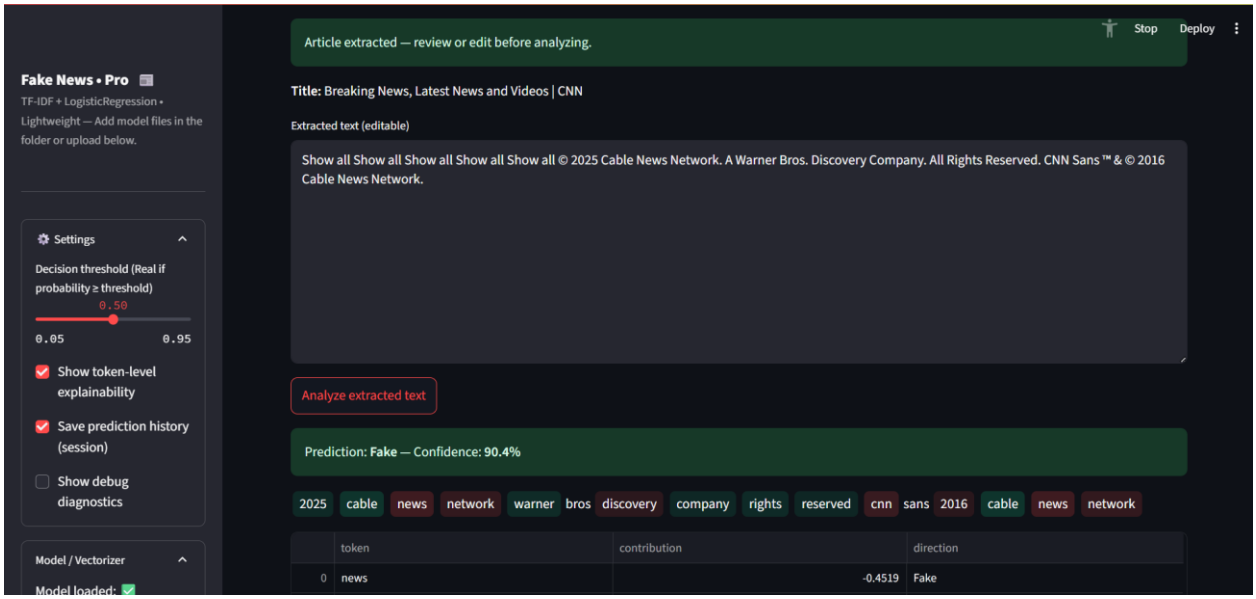
6. Limitations & Considerations

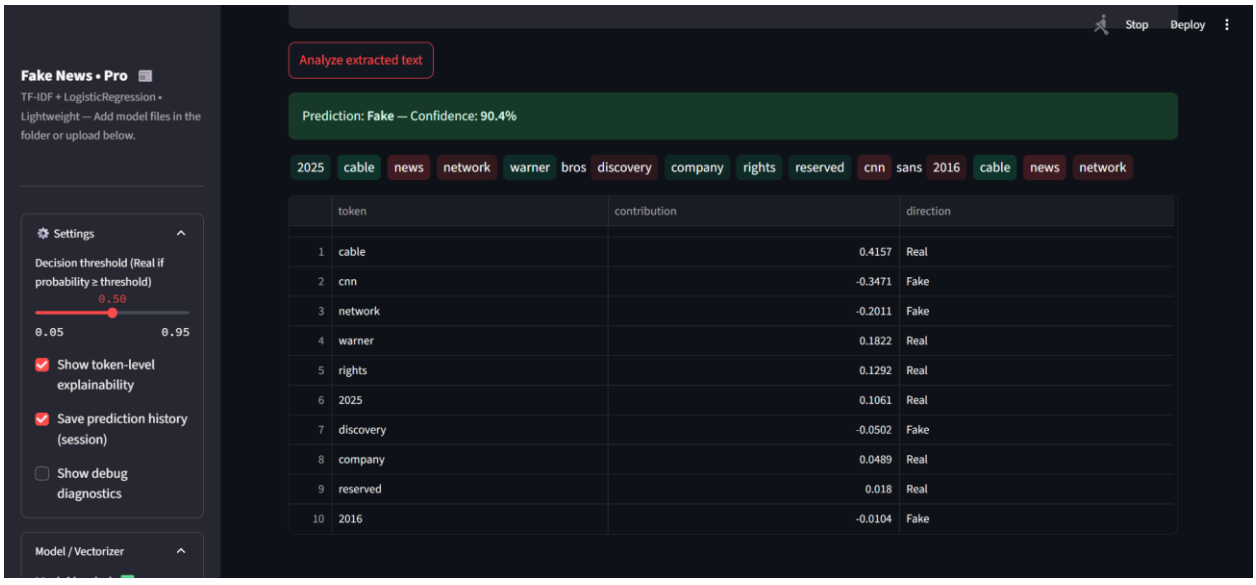
1. **Model Dependency**: Requires pre-trained model and vectorizer files
2. **Text Length Sensitivity**: Works best with 2-4 sentences of context
3. **Domain Specificity**: Performance depends on training data relevance
4. **URL Extraction**: Limited by website anti-scraping measures
5. **Binary Classification**: Limited to Real/Fake without nuance degrees

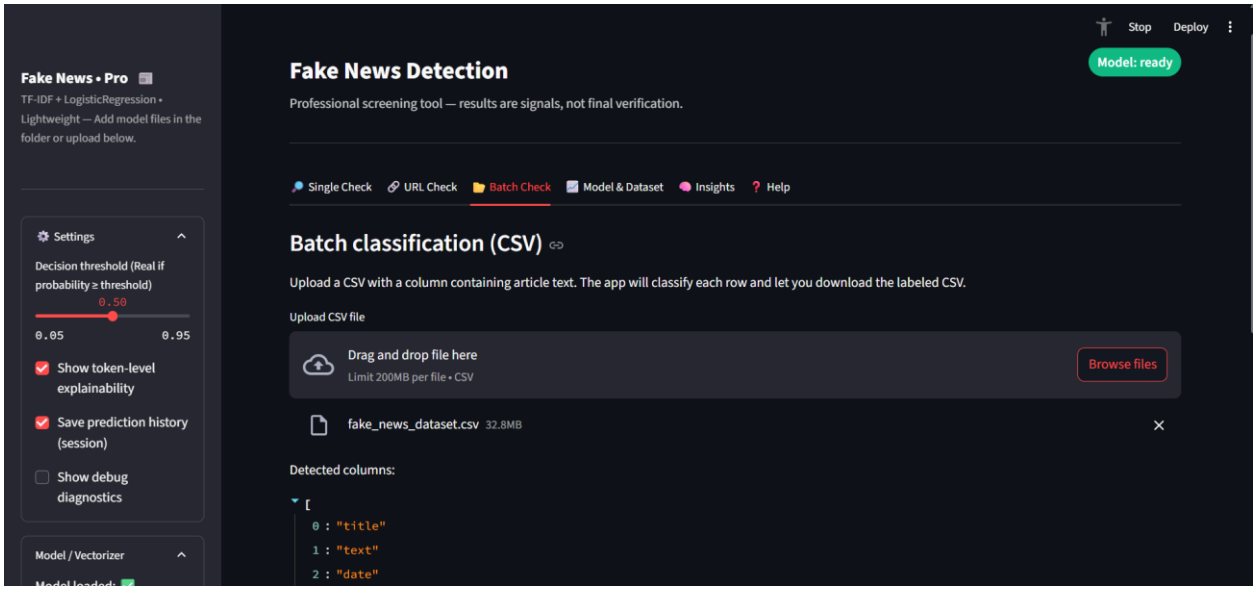
7.Streamlit User Interface

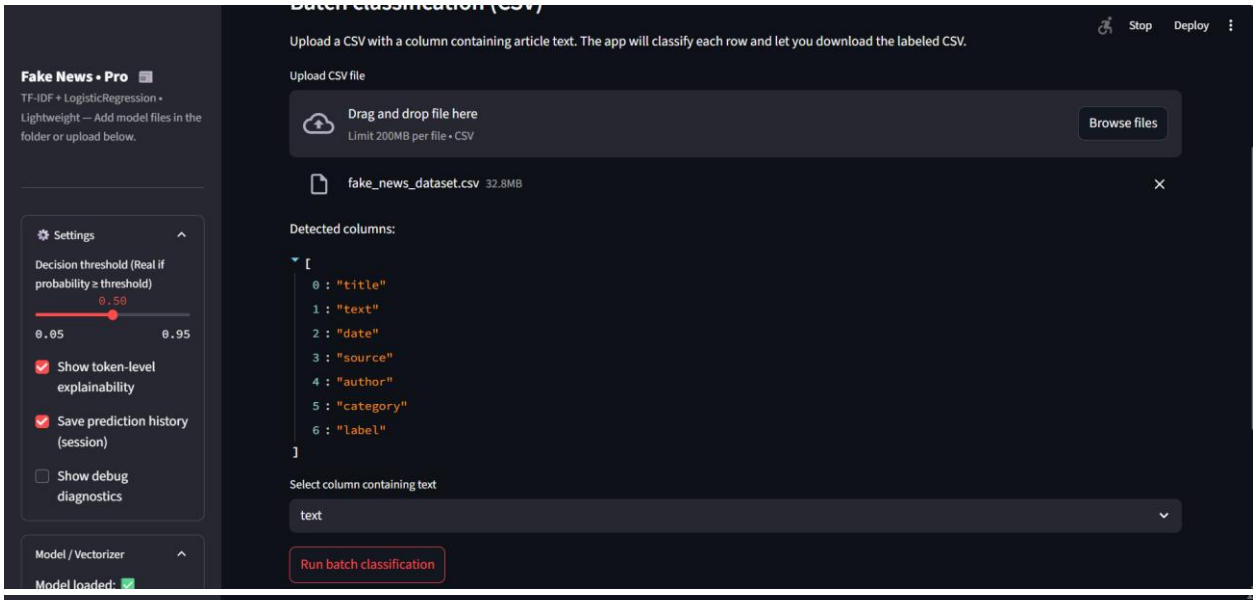


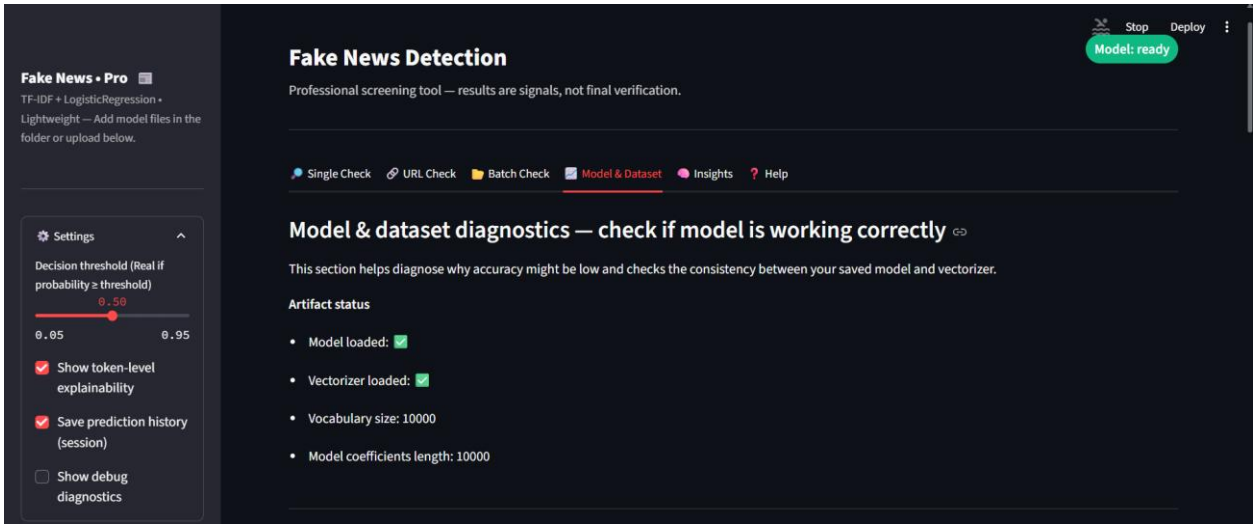

Fake News • Pro
TF-IDF + LogisticRegression • Lightweight — Add model files in the folder or upload below.
Settings
Decision threshold (Real if probability ≥ threshold)
0.50
0.05
0.95
Show token-level explainability
Save prediction history (session)
Show debug diagnostics
Model / Vectorizer
Model loaded:
Stop
Deploy
Model: ready
Fake News Detection
Professional screening tool — results are signals, not final verification.
Single Check
URL Check
Batch Check
Model & Dataset
Insights
Help
URL-based article extraction & analysis
Paste an article URL and the app will attempt to extract the text. If extraction fails, we'll show diagnostics and what you can try.
Article URL
https://edition.cnn.com/
Fetch & analyze
Extraction diagnostics
{
"status": "ok",
"reason": "requests_bs4",
"status_code": "200",
"content_type": "text/html; charset=utf-8",
"len": "166"
}


Fake News • Pro
TF-IDF + LogisticRegression • Lightweight — Add model files in the folder or upload below.
Settings
Decision threshold (Real if probability ≥ threshold)
0.50
0.05
0.95
Show token-level explainability
Save prediction history (session)
Show debug diagnostics
Model / Vectorizer
Model loaded:
Stop
Deploy
Article extracted — review or edit before analyzing.
Title: Breaking News, Latest News and Videos | CNN
Extracted text (editable)
Show all Show all Show all Show all Show all © 2025 Cable News Network. A Warner Bros. Discovery Company. All Rights Reserved. CNN Sans ™ & © 2016 Cable News Network.
Analyze extracted text
Prediction: Fake — Confidence: 90.4%
2025 cable news network warner bros discovery company rights reserved cnn sans 2016 cable news network
token
contribution
direction
0 news -0.4519 Fake


Fake News • Pro
TF-IDF + LogisticRegression • Lightweight — Add model files in the folder or upload below.
Settings
Decision threshold (Real if probability ≥ threshold)
0.50
0.05
0.95
Show token-level explainability
Save prediction history (session)
Show debug diagnostics
Model / Vectorizer
Stop
Deploy
Analyze extracted text
Prediction: Fake — Confidence: 90.4%
2025 cable news network warner bros discovery company rights reserved cnn sans 2016 cable news network
token
contribution
direction
1 cable 0.4157 Real
2 cnn -0.3471 Fake
3 network -0.2011 Fake
4 warner 0.1822 Real
5 rights 0.1292 Real
6 2025 0.1061 Real
7 discovery -0.0502 Fake
8 company 0.0489 Real
9 reserved 0.018 Real
10 2016 -0.0104 Fake


Stop
Deploy
Model: ready
Fake News • Pro
TF-IDF + LogisticRegression • Lightweight — Add model files in the folder or upload below.
Fake News Detection
Professional screening tool — results are signals, not final verification.
Single Check
URL Check
Batch Check
Model & Dataset
Insights
Help
Batch classification (CSV)
Upload a CSV with a column containing article text. The app will classify each row and let you download the labeled CSV.
Settings
Decision threshold (Real if probability ≥ threshold)
0.50
0.05
0.95
Show token-level explainability
Save prediction history (session)
Show debug diagnostics
Upload CSV file
Drag and drop file here
Limit 200MB per file • CSV
Browse files
fake_news_dataset.csv 32.8MB
Detected columns:
[
0 : "title"
1 : "text"
2 : "date"
Model / Vectorizer


Stop
Deploy
Upload a CSV with a column containing article text. The app will classify each row and let you download the labeled CSV.
Fake News • Pro
TF-IDF + LogisticRegression • Lightweight — Add model files in the folder or upload below.
Upload CSV file
Drag and drop file here
Limit 200MB per file • CSV
Browse files
fake_news_dataset.csv 32.8MB
Settings
Decision threshold (Real if probability ≥ threshold)
0.50
0.05
0.95
Show token-level explainability
Save prediction history (session)
Show debug diagnostics
Detected columns:
[
0 : "title"
1 : "text"
2 : "date"
3 : "source"
4 : "author"
5 : "category"
6 : "label"
]
Select column containing text
text
Run batch classification
Model / Vectorizer


Stop
Deploy
Model: ready
Fake News • Pro
TF-IDF + LogisticRegression • Lightweight — Add model files in the folder or upload below.
Fake News Detection
Professional screening tool — results are signals, not final verification.
Single Check
URL Check
Batch Check
Model & Dataset
Insights
Help
Model & dataset diagnostics — check if model is working correctly
This section helps diagnose why accuracy might be low and checks the consistency between your saved model and vectorizer.
Artifact status
Model loaded:
Vectorizer loaded:
Vocabulary size: 10000
Model coefficients length: 10000
Settings
Decision threshold (Real if probability ≥ threshold)
0.50
0.05
0.95
Show token-level explainability
Save prediction history (session)
Show debug diagnostics

Fake News • Pro

TF-IDF + LogisticRegression • Lightweight — Add model files in the folder or upload below.

Settings

Decision threshold (Real if probability ≥ threshold)

0.50

0.05 0.95

Show token-level explainability

Save prediction history (session)

Show debug diagnostics

Stop Deploy

Quick sanity checks

Top tokens toward Real

	token	coef
0	reuters	27.7032
1	said	17.539
2	washington	6.71
3	wednesday	5.7733
4	tuesday	5.3566
5	thursday	5.0932
6	republican	5.0052
7	friday	4.5771
8	nov	4.3817
9	monday	4.2433

Top tokens toward Fake

	token	coef
0	just	-6.2396
1	read	-5.7909
2	image	-5.7236
3	featured	-5.5693
4	gop	-5.3666
5	mr	-5.0529
6	com	-4.9183
7	hillary	-4.7085
8	america	-4.1844
9	watch	-4.1262

Fake News • Pro

TF-IDF + LogisticRegression • Lightweight — Add model files in the folder or upload below.

Settings

Decision threshold (Real if probability ≥ threshold)

0.50

0.05 0.95

Show token-level explainability

Save prediction history (session)

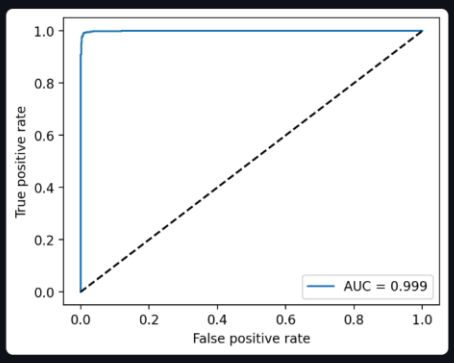
Show debug diagnostics

Stop Deploy

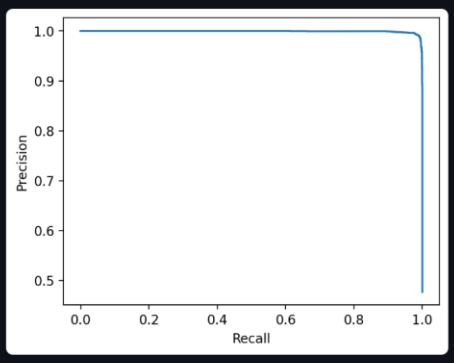
Evaluate on dataset (if available)

Accuracy	Precision	Recall	F1	Roc_auc	Brier
0.9909	**0.9897**	**0.9911**	**0.9904**	**0.9994**	**0.0135**

ROC Curve



Precision–Recall Curve



Fake News • Pro

TF-IDF + LogisticRegression • Lightweight — Add model files in the folder or upload below.

Settings

Decision threshold (Real if probability ≥ threshold)

0.50

0.05 0.95

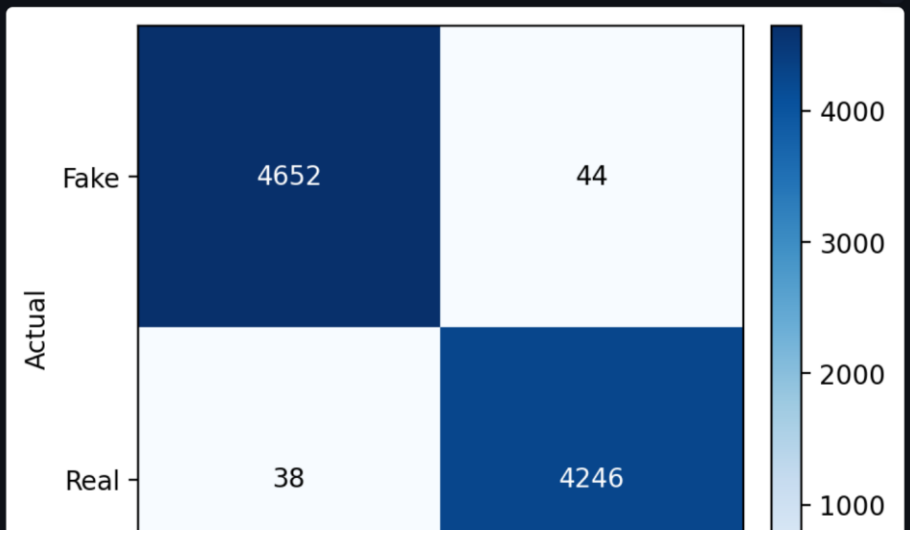
Show token-level explainability

Save prediction history (session)

Show debug diagnostics

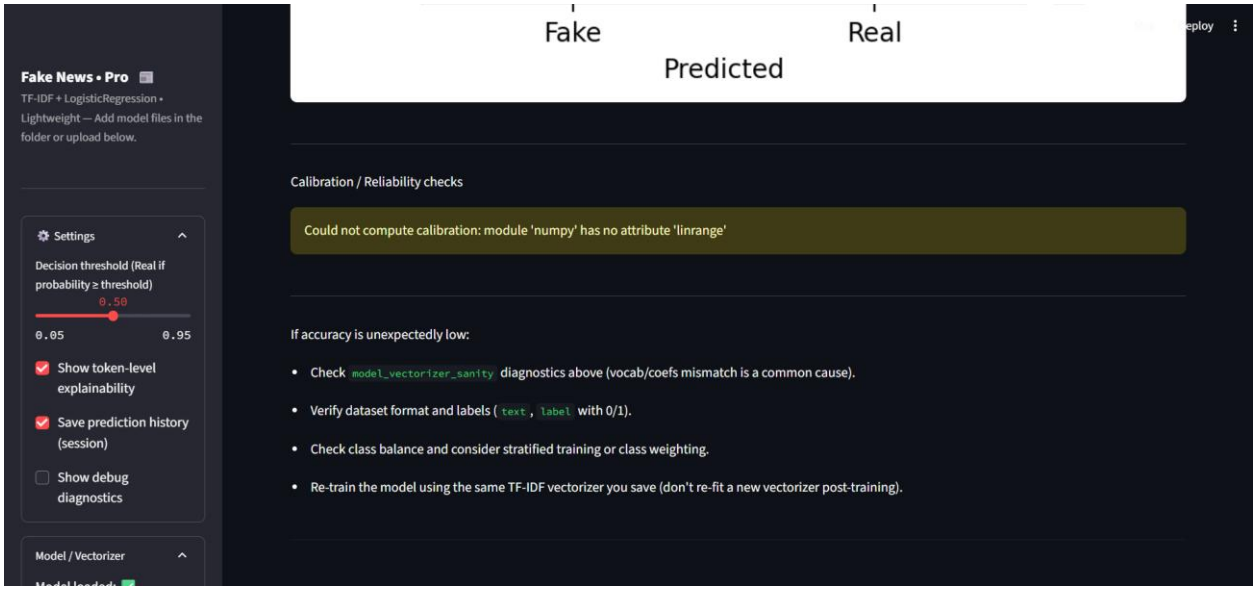
Stop Deploy

Confusion matrix (on test split)

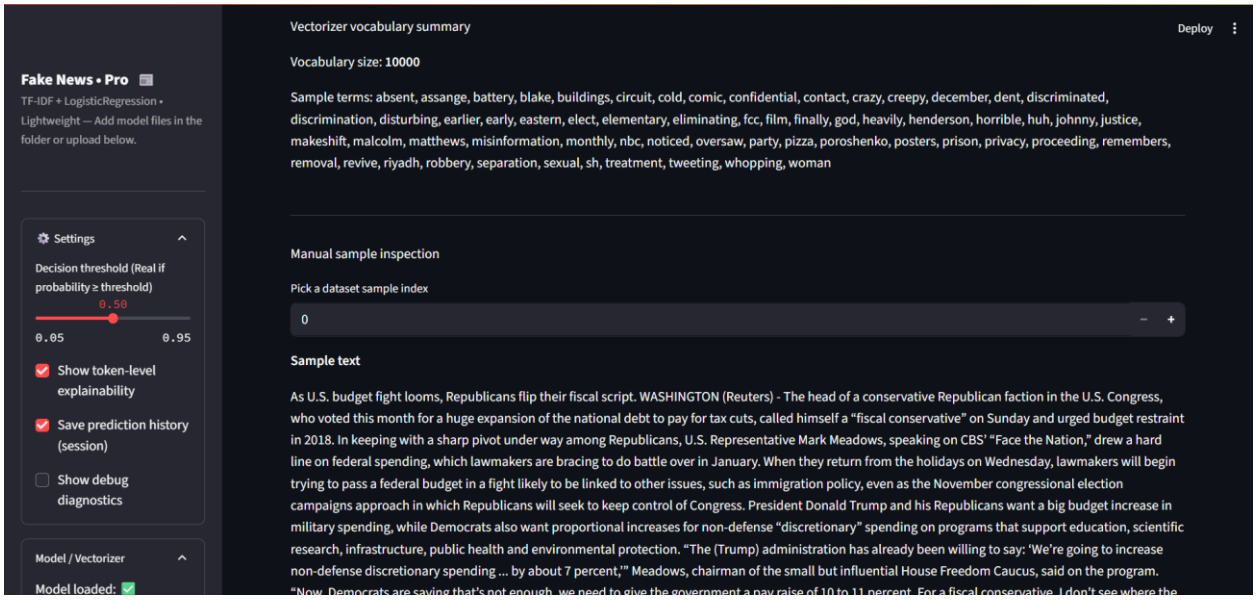


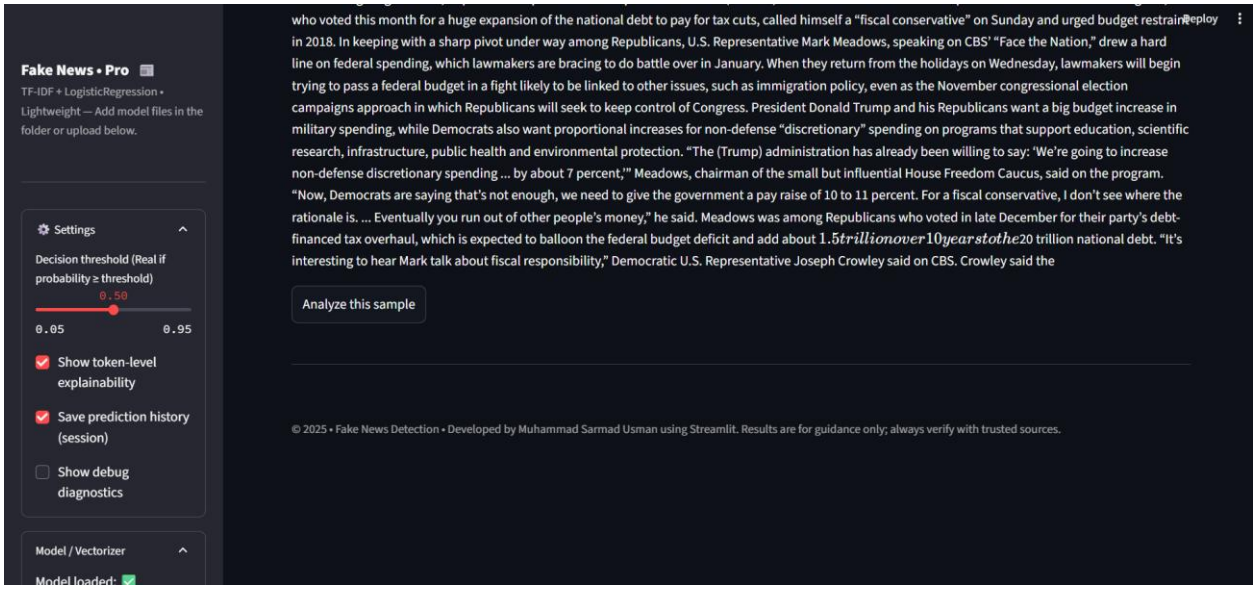
Model / Vectorizer

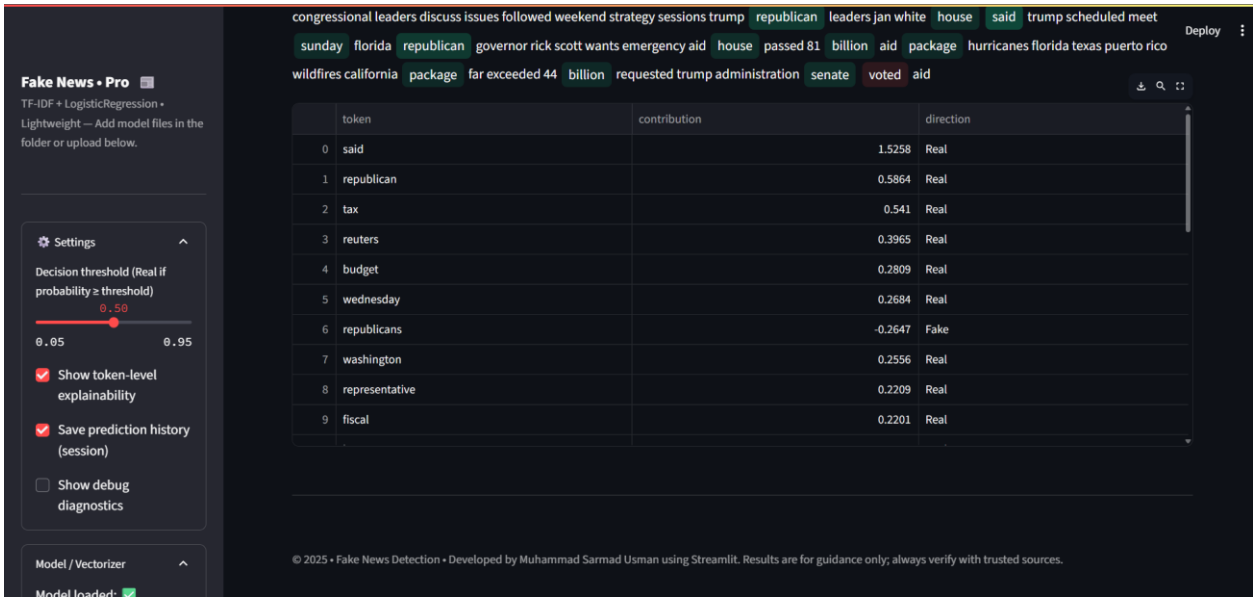
Model loaded:


Fake
Real
Predicted
Deploy
Fake News • Pro
TF-IDF + LogisticRegression • Lightweight — Add model files in the folder or upload below.
Settings
Decision threshold (Real if probability ≥ threshold)
0.50
0.05
0.95
Show token-level explainability
Save prediction history (session)
Show debug diagnostics
Model / Vectorizer
Calibration / Reliability checks
Could not compute calibration: module 'numpy' has no attribute 'linrange'
If accuracy is unexpectedly low:
Check model_vectorizer_sanity diagnostics above (vocab/coefs mismatch is a common cause).
Verify dataset format and labels (text , label with 0/1).
Check class balance and consider stratified training or class weighting.
Re-train the model using the same TF-IDF vectorizer you save (don't re-fit a new vectorizer post-training).

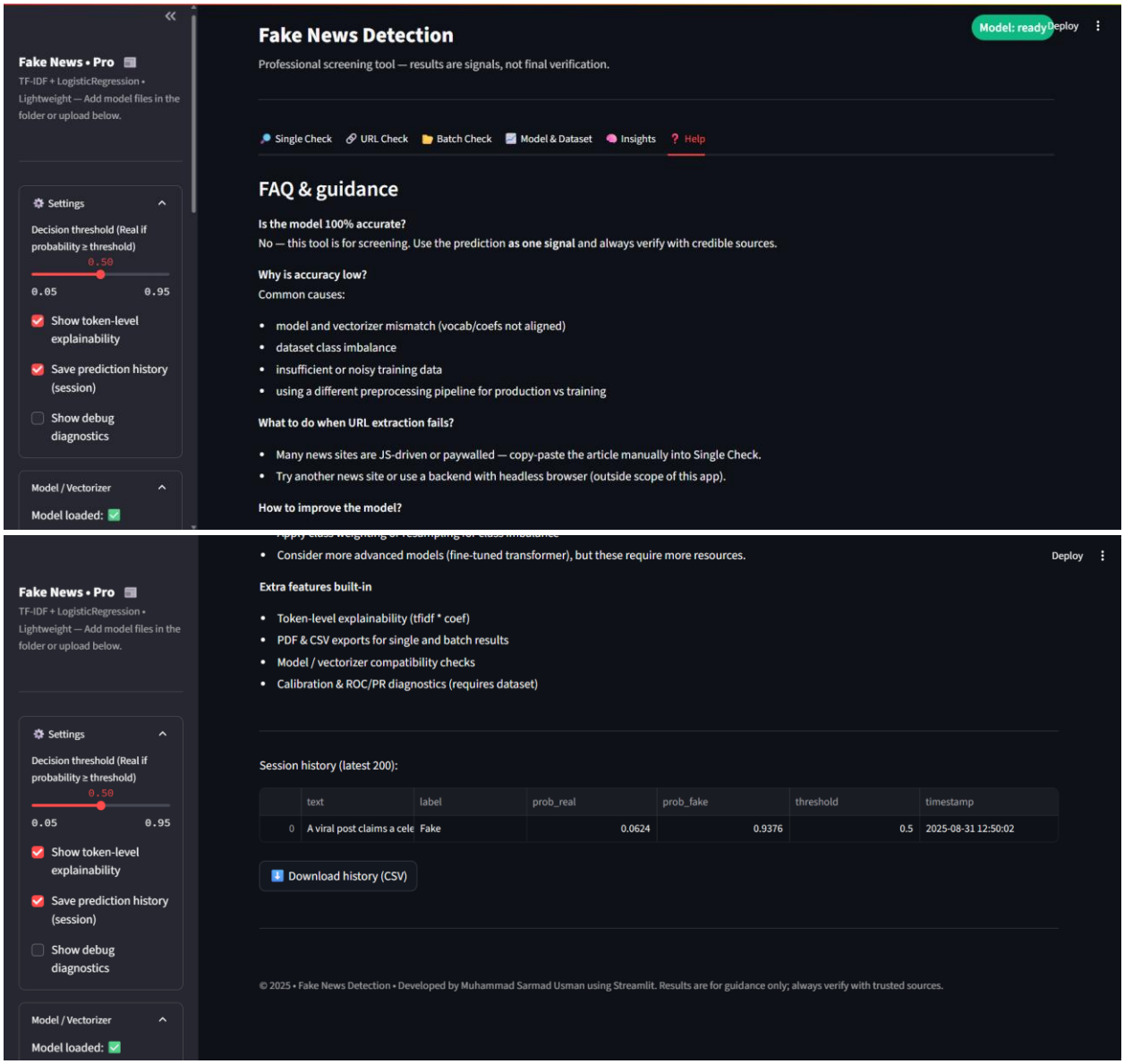

Fake News Detection
Professional screening tool — results are signals, not final verification.
Stop
Deploy
Model: ready
Single Check
URL Check
Batch Check
Model & Dataset
Insights
Help
Insights — vocabulary, wordclouds, and sample inspection
Dataset loaded with 44898 rows
WordCloud — Fake
WordCloud — Real
Fake News • Pro
TF-IDF + LogisticRegression • Lightweight — Add model files in the folder or upload below.
Settings
Decision threshold (Real if probability ≥ threshold)
0.50
0.05
0.95
Show token-level explainability
Save prediction history (session)
Show debug diagnostics
Model / Vectorizer
Model loaded:


Vectorizer vocabulary summary
Deploy
Vocabulary size: 10000
Sample terms: absent, assange, battery, blake, buildings, circuit, cold, comic, confidential, contact, crazy, creepy, december, dent, discriminated, discrimination, disturbing, earlier, early, eastern, elect, elementary, eliminating, fcc, film, finally, god, heavily, henderson, horrible, huh, johnny, justice, makeshift, malcolm, matthews, misinformation, monthly, nbc, noticed, oversaw, party, pizza, poroshenko, posters, prison, privacy, proceeding, remembers, removal, revive, riyadh, robbery, separation, sexual, sh, treatment, tweeting, whopping, woman
Manual sample inspection
Pick a dataset sample index
0
Sample text
As U.S. budget fight looms, Republicans flip their fiscal script. WASHINGTON (Reuters) - The head of a conservative Republican faction in the U.S. Congress, who voted this month for a huge expansion of the national debt to pay for tax cuts, called himself a “fiscal conservative” on Sunday and urged budget restraint in 2018. In keeping with a sharp pivot under way among Republicans, U.S. Representative Mark Meadows, speaking on CBS' “Face the Nation,” drew a hard line on federal spending, which lawmakers are bracing to do battle over in January. When they return from the holidays on Wednesday, lawmakers will begin trying to pass a federal budget in a fight likely to be linked to other issues, such as immigration policy, even as the November congressional election campaigns approach in which Republicans will seek to keep control of Congress. President Donald Trump and his Republicans want a big budget increase in military spending, while Democrats also want proportional increases for non-defense “discretionary” spending on programs that support education, scientific research, infrastructure, public health and environmental protection. “The (Trump) administration has already been willing to say: ‘We’re going to increase non-defense discretionary spending ... by about 7 percent,’” Meadows, chairman of the small but influential House Freedom Caucus, said on the program. “Now, Democrats are saying that’s not enough, we need to give the government a pay raise of 10 to 11 percent. For a fiscal conservative, I don’t see where the
Fake News • Pro
TF-IDF + LogisticRegression • Lightweight — Add model files in the folder or upload below.
Settings
Decision threshold (Real if probability ≥ threshold)
0.50
0.05
0.95
Show token-level explainability
Save prediction history (session)
Show debug diagnostics
Model / Vectorizer
Model loaded:





	token	contribution	direction
0	said	1.5258	Real
1	republican	0.5864	Real
2	tax	0.541	Real
3	reuters	0.3965	Real
4	budget	0.2809	Real
5	wednesday	0.2684	Real
6	republicans	-0.2647	Fake
7	washington	0.2556	Real
8	representative	0.2209	Real
9	fiscal	0.2201	Real



8. Conclusion

The Fake News Detection • Pro application provides a **professional-grade interface** for analyzing news content using machine learning. It combines **technical robustness** with **user-friendly design**, making advanced NLP capabilities accessible to non-technical users while providing detailed diagnostics for experts.

The application successfully demonstrates how **explainable AI** principles can be implemented in practice, offering transparency into the model's decision-making process through token-level contributions and comprehensive performance metrics.

Note: The application is designed as a screening tool rather than a definitive verification system, emphasizing that results should be treated as signals rather than final determinations.
