Fake News Detection using Machine Learning

# Abstract

The spread of fake news on online platforms has become a significant issue in recent years. Misinformation can mislead people, create confusion, and influence opinions. This project, **Fake News Detection using Machine Learning**, aims to build a system capable of classifying news articles as fake or real. Using text preprocessing, TF–IDF feature extraction, and Logistic Regression as the baseline classifier, the system achieves efficient and interpretable results. The project is structured step-by-step for reproducibility and is organized for upload to GitHub as a major project.

# Introduction

The goal of this minor project is to build a baseline automatic fake-news classifier that can separate fake and real news articles using only textual information (title and body). The work focuses on a classical machine-learning baseline (TF–IDF + Logistic Regression) that is easy to reproduce, explain, and submit for evaluation.

# Objectives

* To design and develop a machine learning pipeline that detects fake news.
* To preprocess and clean textual datasets (titles and content).
* To extract features using TF–IDF vectorization.
* To build a classification model (Logistic Regression baseline).
* To evaluate performance using accuracy, precision, recall, and F1-score.
* To prepare a well-documented repository suitable for GitHub upload.

# Dataset

* The dataset used consists of news articles with attributes:

Title, text, label (0 = real, 1 = fake)

* Example public sources: Kaggle Fake News Dataset, LIAR Dataset, FakeNewsNet.
* Only a sample dataset (sample\_news.csv) will be stored in the GitHub repo. Full datasets must be kept outside the repo and added to .gitignore.

# Methodology

### ****Step 1: Data Preprocessing****

* Lowercasing the text.
* Removing URLs, HTML tags, punctuation, and special characters.
* Tokenization and optional lemmatization.
* Combining **title + text** into one field.

### *****Step 2: Feature Extraction*****

* TF–IDF Vectorizer with unigrams and bigrams.
* Limit to max\_features = 10,000 for efficiency.

### ****Step 3: Model Training****

* Logistic Regression used as baseline classifier.
* Other algorithms like SVM or Random Forest can also be tried.

### ****Step 4: Model Evaluation****

* Metrics: Accuracy, Precision, Recall, F1-Score.
* Confusion Matrix and ROC AUC for visualization.

# Implementation Steps

1. **Organize dataset** into a CSV file with columns: **title, text, label** (0 = real, 1 = fake).
2. **Preprocess text** (lowercasing, removing URLs/HTML, punctuation, optional stop words & lemmatization).
3. **Extract features** using TF–IDF vectorization (unigrams + bigrams).
4. **Define baseline model** (Logistic Regression classifier inside a Scikit-learn pipeline).
5. **Compile/Train model** with train-test split (e.g., 80/20) and stratified sampling.
6. **Evaluate model** using Accuracy, Precision, Recall, F1-score, Confusion Matrix, and ROC AUC.
7. **Save trained pipeline** with **joblib** for later prediction.
8. **Test predictions** on new/unseen news text using the saved pipeline.

# Python Code Implementation

## Traning Script

#!/usr/bin/env python3

"""

Fake News Detector - Training Script

Baseline: TF-IDF + Logistic Regression

"""

import argparse

import joblib

import pandas as pd

import re

from pathlib import Path

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.linear\_model import LogisticRegression

from sklearn.model\_selection import train\_test\_split

from sklearn.pipeline import Pipeline

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix, roc\_auc\_score

# Text cleaning function

def clean\_text(text):

if not isinstance(text, str):

text = str(text)

text = text.lower()

text = re.sub(r"http\\S+", " ", text) # remove URLs

text = re.sub(r"<.\*?>", " ", text) # remove HTML tags

text = re.sub(r"[^a-z\\s]", " ", text) # keep only alphabets

return " ".join(text.split())

def main(args):

# Load dataset

df = pd.read\_csv(args.data)

# Combine title + text if both available

if "title" in df.columns and "text" in df.columns:

df["content"] = df["title"].fillna("") + " " + df["text"].fillna("")

elif "text" in df.columns:

df["content"] = df["text"]

else:

raise ValueError("Dataset must have at least 'text' column")

# Ensure label column exists

if args.label\_col not in df.columns:

raise ValueError(f"Dataset must contain a '{args.label\_col}' column")

# Clean text

df["content"] = df["content"].apply(clean\_text)

X = df["content"]

y = df[args.label\_col]

# Train/test split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y, test\_size=args.test\_size, stratify=y, random\_state=42

)

# Build pipeline

pipeline = Pipeline([

("tfidf", TfidfVectorizer(max\_features=args.max\_features, ngram\_range=(1, 2))),

("clf", LogisticRegression(max\_iter=1000))

])

# Train model

print("Training model...")

pipeline.fit(X\_train, y\_train)

# Evaluate

print("Evaluating...")

y\_pred = pipeline.predict(X\_test)

y\_proba = pipeline.predict\_proba(X\_test)[:, 1]

print("Accuracy:", accuracy\_score(y\_test, y\_pred))

print("Classification Report:\n", classification\_report(y\_test, y\_pred))

print("Confusion Matrix:\n", confusion\_matrix(y\_test, y\_pred))

try:

print("ROC AUC:", roc\_auc\_score(y\_test, y\_proba))

except Exception:

pass

# Save model

out\_path = Path(args.out)

out\_path.parent.mkdir(parents=True, exist\_ok=True)

joblib.dump(pipeline, out\_path)

print(f"Model saved to {out\_path}")

if \_\_name\_\_ == "\_\_main\_\_":

parser = argparse.ArgumentParser()

parser.add\_argument("--data", required=True, help="Path to dataset CSV")

parser.add\_argument("--out", default="models/fake\_news\_pipeline.joblib", help="Output model path")

parser.add\_argument("--label-col", default="label", help="Name of label column")

parser.add\_argument("--test-size", type=float, default=0.2, help="Test size fraction")

parser.add\_argument("--max-features", type=int, default=10000, help="Maximum TF-IDF features")

args = parser.parse\_args()

main(args)

## Prediction Script

#!/usr/bin/env python3

"""

Fake News Detector - Prediction Script

Loads trained pipeline and predicts label for input text.

"""

import sys

import joblib

# Load trained pipeline

model = joblib.load("models/fake\_news\_pipeline.joblib")

# Take input text from command line

if len(sys.argv) < 2:

print("Usage: python predict.py \"Your news article text here...\"")

sys.exit(1)

text = " ".join(sys.argv[1:])

pred = model.predict([text])[0]

# If probabilities available

if hasattr(model, "predict\_proba"):

proba = model.predict\_proba([text])[0][1]

print(f"Prediction: {'FAKE' if pred == 1 else 'REAL'} (probability of fake: {proba:.4f})")

else:

print(f"Prediction: {'FAKE' if pred == 1 else 'REAL'}")

# Results

After training the Logistic Regression model (TF–IDF features, max 10,000), the following performance was achieved on the test dataset:

* **Accuracy:** 99.13%
* **Precision (Real):** 0.99
* **Precision (Fake):** 0.99
* **Recall (Real):** 0.99
* **Recall (Fake):** 0.99
* **F1-Score (Real):** 0.99
* **F1-Score (Fake):** 0.99
* **ROC AUC:** 0.9994

**Classification Report:**

| **Class** | **Precision** | **Recall** | **F1-Score** | **Support** |
| --- | --- | --- | --- | --- |
| Real (0) | 0.99 | 0.99 | 0.99 | 4284 |
| Fake (1) | 0.99 | 0.99 | 0.99 | 4696 |
| **Accuracy** |  |  | **0.9913** | 8980 |
| Macro Avg | 0.99 | 0.99 | 0.99 | 8980 |
| Weighted Avg | 0.99 | 0.99 | 0.99 | 8980 |
|  |  |  |  |  |

# LimitatioN

* Logistic Regression only captures shallow textual patterns.
* Does not use contextual embeddings (e.g., BERT).
* Performance may drop if dataset is small or imbalanced.

# Conclusion

The Fake News Detection (ML) system demonstrates a reproducible baseline for classifying fake and real news articles. The step-by-step methodology ensures ease of understanding, reproducibility, and suitability for academic evaluation. With further extensions, the system can evolve into a powerful real-time detection tool.

# Future Work

* Implement advanced models like BERT, RoBERTa, or XLNet.
* Deploy as a web app (Flask/FastAPI).
* Add visualization dashboards (confusion matrix, word importance).
* Enable real-time news verification with APIs.

# References

1. **Scikit-learn Documentation** — Machine Learning in Python https://scikit-learn.org/stable/
2. **Pandas Documentation** — Data analysis with Python  
   https://pandas.pydata.org/
3. **NLTK Documentation** — Natural Language Toolkit  
   https://www.nltk.org/
4. **Kaggle: Fake News Dataset** — Public dataset for fake/real news classification  
   [fake-and-real-news-dataset](https://www.kaggle.com/datasets/clmentbisaillon/fake-and-real-news-dataset?utm_source=chatgpt.com)
5. **LIAR Dataset** — Short statements labeled for truthfulness  
   https://www.cs.ucsb.edu/~william/data/liar\_dataset.zip
6. **FakeNewsNet** — A dataset for fake news research  
   <https://github.com/KaiDMML/FakeNewsNet>
7. **Joblib Documentation** — Serialization of Python objects (used for saving models)  
   https://joblib.readthedocs.io/
8. **Python Official Documentation**  
   https://docs.python.org/3/