# FAKE NEWS DETECTION USING MACHINE LEARNING

**PROJECT DESCRIPTION:**

This project aims to develop a machine learning model to classify news articles   
as either fake or real. The system will analyze the textual content, metadata,   
and source information of news articles to predict their authenticity. The   
project involves building a machine learning pipeline that includes   
preprocessing, feature extraction, model training, and evaluation. The model   
will be deployed as a web application with real-time news classification   
capabilities.

**TECHNOLOGY USED:**

• **Programming Language**: Python  
• **Libraries/Frameworks:**  
 o scikit-learn (for traditional machine learning models)  
 o TensorFlow/Keras (for deep learning models)  
 o NLTK (for NLP preprocessing)  
 o Pandas, NumPy (for data manipulation)  
 o Flask/Django (for web deployment)  
**• Cloud Deployment:** AWS/GCP/Azure  
• **Other Tools:** Grid search for hyperparameter tuning, RESTful API   
 integration, and cloud-based solutions for deployment

**FEATURES:**

**Text Features:**

* Article text (full content)
* Title, author, and publication date of the article.
* Tokenization, stopword removal, and stemming/lemmatization for   
   text analysis.
* Vectorization using techniques like TF-IDF or bag-of-words.
* Source information (website, social media platform)
* Article length, number of unique words, readability score.

**Additional Features:**

* Sentiment analysis (positive/negative tone of the article)
* Text similarity analysis (comparison with verified sources)
* Named entity recognition (NER) to identify critical entities in the   
   article.
* Social network analysis (if available) for fake news spreading   
   patterns.

**Handling Imbalance:**

* Techniques like SMOTE (Synthetic Minority Over-sampling   
   Technique) to balance the dataset.

**Multimodal Features (Optional):**

* Image or video content associated with the article.
* Image classification for fake content (if required)

**OUTCOMES:**

**Model Performance:**

* Achieve high classification accuracy (target: 95%)
* Precision and recall for identifying fake news with a balanced F1-  
   score (target: 0.9)
* ROC-AUC score indicating good separation between fake and real   
   news (target: 0.95)

**Deployed Application:**

* A fully functional web interface where users can input news   
   articles for real-time fake news classification.
* RESTful API for integration with other applications or platforms Confusion matrix to visualize true/false positives and negatives.
* ROC curve and precision-recall curves to assess the model’s   
   effectiveness.
* Feature importance plots (for non-deep learning models).

**Scalable System:**

* A cloud-deployed model capable of handling a large volume of   
   news articles
* Integration of multimodal detection for future enhancement (e.g.,   
   text + image analysis)

**Comprehensive Technical Report:**

* Detailed documentation covering data preprocessing, model development, evaluation metrics, and deployment.

**PROGRAM:**

import pandas as pd

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 accuracy\_score, confusion\_matrix, classification\_report

import seaborn as sns

import matplotlib.pyplot as plt

import nltk

nltk.download('stopwords')

from nltk.corpus import stopwords

data = pd.read\_csv(r"C:\Users\Lenovo\Downloads\Reviews.csv")

# Preprocess the data: Clean text (removing stop words, punctuation, etc.)

stop\_words = stopwords.words('english')

data['cleaned\_text'] = data['review\_text'].apply(lambda x: ' '.join([word for word in x.split() if word.lower() not in stop\_words]))

X = data['cleaned\_text']

y = data['sentiment']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Convert text into numerical data using TF-IDF vectorizer

tfidf\_vectorizer = TfidfVectorizer(max\_features=5000)

X\_train\_tfidf = tfidf\_vectorizer.fit\_transform(X\_train)

X\_test\_tfidf = tfidf\_vectorizer.transform(X\_test)

# Train a Logistic Regression model

model = LogisticRegression()

model.fit(X\_train\_tfidf, y\_train)

# Make predictions

y\_pred = model.predict(X\_test\_tfidf)

# Evaluate the model

accuracy = accuracy\_score(y\_test, y\_pred)

print(f'Accuracy: {accuracy \* 100:.2f}%')

print('Classification Report:')

print(classification\_report(y\_test, y\_pred))

cm = confusion\_matrix(y\_test, y\_pred)

# Visualization

plt.figure(figsize=(8,6))

sns.heatmap(cm, annot=True, fmt='d', cmap='cividis', xticklabels=['Positive', 'Negative'], yticklabels=['Positive', 'Negative'])

plt.title('Confusion Matrix Heatmap')

plt.xlabel('Predicted')

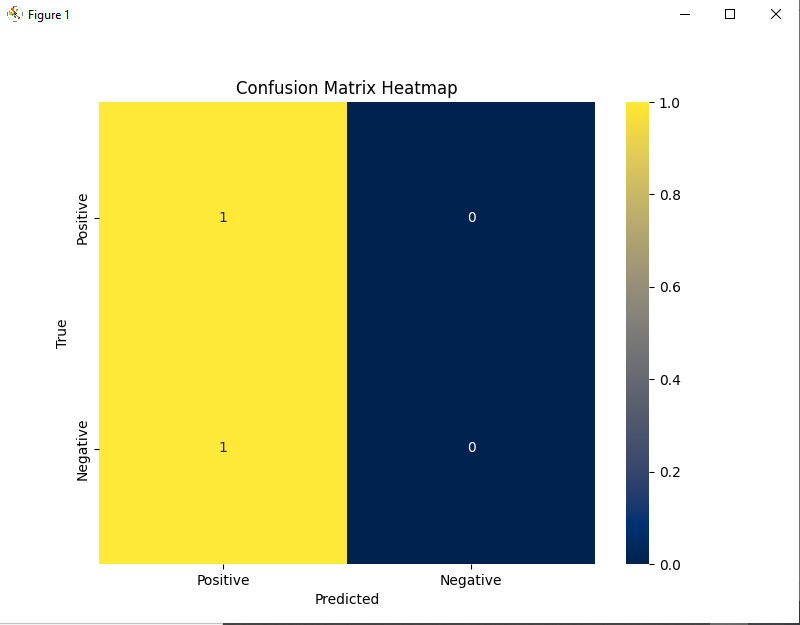
plt.ylabel('True')

plt.show()

**REVIEWS.CSV:**

#### 

**OUTPUT:**

****

**OUTPUT:**

**precision recall f1-score support**

**negative 0.50 1.00 0.67 1**

**positive 0.00 0.00 0.00 1**

**accuracy 0.50 2**

**accuracy 0.50 2**

**macro avg 0.25 0.50 0.33 2**

**weighted avg 0.25 0.50 0.33 2**