Source Code – Fake News detection using ML

Project Structure

Executable_Files

```
├── data/
│   ├── train.csv          # Training dataset (news + labels)
│   ├── test.csv           # Testing dataset
│
├── notebooks/
│   ├── data_exploration.ipynb    # EDA (Exploratory Data Analysis) notebook
│   ├── model_training.ipynb      # Model training & evaluation notebook
│
├── src/
│   ├── data_preprocessing.py  # Data cleaning, text preprocessing functions
│   ├── model.py               # ML model creation, training functions
│   ├── predict.py             # Prediction logic for new input news
│
├── app/
│   ├── app.py                 # Flask/Django web app main file (if web-based)
│   ├── templates/             # HTML templates (if using Flask)
│   ├── static/                # CSS, JS files for frontend
│
├── requirements.txt           # Required Python packages
├── README.md                  # Project description & instructions
└── .gitignore                 # Git ignore file
```

data/

│

├── train.csv # Training dataset (news + labels)

test.csv # Testing dataset

Kaggle - Fake News Dataset

- Link: https://www.kaggle.com/datasets/clmentbisaillon/fake-and-real-news-dataset

data_exploration.ipynb

```
# Import libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# 1. Load training data
train = pd.read_csv('../data/train.csv')

# 2. Data overview
print("First 5 rows of training data:")
display(train.head())

print("\nShape of training data:", train.shape)

# 3. Check missing values
print("\nMissing values in each column:")
print(train.isnull().sum())

# 4. Label distribution
```

```
plt.figure(figsize=(6,4))
sns.countplot(x='label', data=train)
plt.title("Label Distribution (0=Real, 1=Fake)")
plt.show()

print("\nLabel counts:")
print(train['label'].value_counts())

# 5. Text length analysis
train['text_length'] = train['text'].astype(str).apply(len)

plt.figure(figsize=(8,5))
sns.histplot(train['text_length'], bins=50, kde=True)
plt.title("Distribution of News Text Length")
plt.xlabel("Number of characters")
plt.show()

# 6. Average length per class
avg_length = train.groupby('label')['text_length'].mean()
print("\nAverage text length per label:")
print(avg_length)

# 7. Sample texts from each class
print("\nSample Fake News:")
print(train[train['label']==1]['text'].iloc[0])
```

```
print("\nSample Real News:")
print(train[train['label']==0]['text'].iloc[0])
```

 # model_training.ipynb

```
# Step 1: Import libraries
import pandas as pd
import re
import string
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
import joblib  # model save/load ke liye

# Step 2: Load training data
train = pd.read_csv('../data/train.csv')
test = pd.read_csv('../data/test.csv')

# Step 3: Text preprocessing function
def preprocess_text(text):
    text = str(text).lower()  # lowercase
    text = re.sub(f"[{re.escape(string.punctuation)}]", "", text)  # punctuation hatao
    text = re.sub('\s+', ' ', text)  # multiple spaces replace karo
    return text.strip()
```

```
# Step 4: Preprocess train and test text columns
train['text'] = train['text'].apply(preprocess_text)
test['text'] = test['text'].apply(preprocess_text)

# Step 5: Vectorization (TF-IDF)
vectorizer = TfidfVectorizer(max_features=5000)
X_train = vectorizer.fit_transform(train['text'])
X_test = vectorizer.transform(test['text'])

y_train = train['label']
y_test = test['label']

# Step 6: Train Logistic Regression model
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)

# Step 7: Predictions and Evaluation
y_pred = model.predict(X_test)
print("Test Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n",
classification_report(y_test, y_pred))

# Step 8: Save model and vectorizer for future use
joblib.dump(model, '../src/logistic_model.pkl')
joblib.dump(vectorizer, '../src/vectorizer.pkl')
```

```
print("\nModel and vectorizer saved to src/ folder.")
```

 # src/data_preprocessing.py

```
import re
import string

def preprocess_text(text):
    """
    Function to clean and preprocess news text.
    Steps:
    - Convert to lowercase
    - Remove punctuation
    - Remove extra spaces
    """
    text = str(text).lower()  # lowercase
    text  =  re.sub(f"[{re.escape(string.punctuation)}]",  "",
text)  # punctuation remove
    text = re.sub('\s+', ' ', text)  # multiple spaces replace
with single space
    text = text.strip()
    return text
```

 # src/model.py

```
import pandas as pd
```

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
import joblib
from data_preprocessing import preprocess_text  # apne preprocessing function ko import kar rahe hain

class FakeNewsModel:
    def __init__(self):
        self.vectorizer = TfidfVectorizer(max_features=5000)
        self.model = LogisticRegression(max_iter=1000)

    def load_data(self, filepath):
        data = pd.read_csv(filepath)
        data['text'] = data['text'].apply(preprocess_text)
        return data

    def train(self, train_path, test_path):
        train_data = self.load_data(train_path)
        test_data = self.load_data(test_path)

        X_train = self.vectorizer.fit_transform(train_data['text'])
        y_train = train_data['label']

        X_test = self.vectorizer.transform(test_data['text'])
```

```
        y_test = test_data['label']

        self.model.fit(X_train, y_train)

        y_pred = self.model.predict(X_test)

        print(f"Test Accuracy: {accuracy_score(y_test,
y_pred):.4f}")
        print("\nClassification Report:\n",
classification_report(y_test, y_pred))

        # Save model and vectorizer
        joblib.dump(self.model, 'logistic_model.pkl')
        joblib.dump(self.vectorizer, 'vectorizer.pkl')
        print("Model and vectorizer saved!")

    def predict(self, texts):
        """
        Predict if given news texts are fake or real.
        texts: list of raw news strings
        returns: list of predictions (0 = real, 1 = fake)
        """
        clean_texts = [preprocess_text(text) for text in texts]
        vect_texts = self.vectorizer.transform(clean_texts)
        preds = self.model.predict(vect_texts)
        return preds.tolist()
```

```
# Usage example (if running as script):
if __name__ == "__main__":
    fn_model = FakeNewsModel()
    fn_model.train('../data/train.csv', '../data/test.csv')
```

 # src/predict.py

```
import joblib
from data_preprocessing import preprocess_text

class FakeNewsPredictor:
    def __init__(self, model_path='logistic_model.pkl', vectorizer_path='vectorizer.pkl'):
        self.model = joblib.load(model_path)
        self.vectorizer = joblib.load(vectorizer_path)

    def predict(self, news_list):
        """
        news_list: list of raw news strings
        returns: list of predictions (0 = real, 1 = fake)
        """
        processed_news = [preprocess_text(news) for news in news_list]
        vect_news = self.vectorizer.transform(processed_news)
        preds = self.model.predict(vect_news)
        return preds.tolist()
```

```
if __name__ == "__main__":
    # Example usage:
    predictor = FakeNewsPredictor()
    sample_news = [
        "Breaking: New vaccine shows 99% effectiveness!",
        "Aliens have landed on Earth, government confirms."
    ]
    predictions = predictor.predict(sample_news)
    for news, pred in zip(sample_news, predictions):
        label = "Fake" if pred == 1 else "Real"
        print(f"News: {news}\nPrediction: {label}\n")
```

 # app/app.py

```
from flask import Flask, request, render_template
import sys
import os
sys.path.append(os.path.abspath("../src"))    #  src  folder  ko
import path me add kar diya

from predict import FakeNewsPredictor

app = Flask(__name__)
predictor = FakeNewsPredictor(
    model_path="../logistic_model.pkl",
    vectorizer_path="../vectorizer.pkl"
```

```
)

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/predict', methods=['POST'])
def predict():
    news_text = request.form['news']
    prediction = predictor.predict([news_text])[0]
    label = "Fake" if prediction == 1 else "Real"
    return    render_template('index.html',    prediction=label,
news=news_text)

if __name__ == '__main__':
    app.run(debug=True)
```

<!-- app/templates/index.html -->

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8" />
    <title>Fake News Detection</title>
</head>
<body>
```

```
    <h1>Fake News Detection System</h1>
    <form method="POST" action="/predict">
        <textarea name="news" rows="6" cols="60"
placeholder="Paste news here..." required></textarea><br>
        <button type="submit">Predict</button>
    </form>

    {% if prediction %}
    <h2>Prediction: {{ prediction }}</h2>
    <p><b>News:</b> {{ news }}</p>
    {% endif %}
</body>
</html>
```

/* app/static/style.css */

```
body {
    font-family: Arial, sans-serif;
    background-color: #f0f2f5;
    display: flex;
    justify-content: center;
    padding: 50px 20px;
}

.container {
    background-color: #fff;
```

```
    padding: 25px 30px;
    border-radius: 8px;
    box-shadow: 0 0 20px rgba(0,0,0,0.1);
    max-width: 600px;
    width: 100%;
}

h1 {
    text-align: center;
    color: #333;
    margin-bottom: 25px;
}

textarea {
    width: 100%;
    padding: 15px;
    font-size: 16px;
    border-radius: 6px;
    border: 1px solid #ccc;
    resize: vertical;
    box-sizing: border-box;
}

button {
    background-color: #007bff;
    color: white;
```

```
    border: none;
    padding: 14px;
    border-radius: 6px;
    width: 100%;
    font-size: 16px;
    margin-top: 15px;
    cursor: pointer;
    transition: background-color 0.3s ease;
}

button:hover {
    background-color: #0056b3;
}

.result {
    margin-top: 30px;
    padding: 15px;
    font-weight: bold;
    font-size: 18px;
    text-align: center;
    border-radius: 6px;
}

.real {
    color: #155724;
    background-color: #d4edda;
```

```
    border: 1px solid #c3e6cb;
}

.fake {
    color: #721c24;
    background-color: #f8d7da;
    border: 1px solid #f5c6cb;
}

.news-text {
    margin-top: 10px;
    font-style: italic;
    color: #555;
    white-space: pre-wrap;
}
```

requirements.txt

```
flask
pandas
numpy
scikit-learn
nltk
joblib
```

README.md

```
# Fake News Detection
```

This project implements a Fake News Detection system using Machine Learning. The system can classify news articles as **Real** or **Fake** based on their content.
