Fake News & Hate Speech Detector — Full Project

This project implements two lightweight NLP classifiers (TF-IDF + Logistic Regression) to detect **fake news** and **hate speech**, and exposes them via a **Telegram moderation bot** so you can test text in chat. The code is designed to run locally in VS Code.

Project Structure

```
fake-news-hate-speech-detector/
|-- src/
|   |-- bot.py            # Telegram bot (reply & moderation)
|   |-- train.py          # Train both classifiers and save models
|   |-- preprocess.py     # Text cleaning + tokenizer utilities
|   |-- model_utils.py    # Model save/load + predict helpers
|
|-- data/
|   |-- fake_news_sample.csv
|   |-- hate_speech_sample.csv
|
|-- models/
|   |-- fake_news_model.joblib
|   |-- hate_speech_model.joblib
|
|-- requirements.txt
|-- README.md
```

---

src/preprocess.py

```python
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer

# Download NLTK resources on first run
nltk.download('stopwords', quiet=True)
nltk.download('wordnet', quiet=True)
nltk.download('omw-1.4', quiet=True)

STOPWORDS = set(stopwords.words('english'))
LEMMATIZER = WordNetLemmatizer()

URL_PATTERN = re.compile(r'https?://\S+|www\.\S+')
```

```
NON_ALPHANUM = re.compile(r'[^a-zA-Z0-9\s]')
MULTI_SPACE = re.compile(r'\s+')


def clean_text(text: str) -> str:
    """Basic text cleaning: lowercase, remove URLs, punctuation, stopwords
and lemmatize."""
    if not isinstance(text, str):
        return ''
    text = text.lower()
    text = URL_PATTERN.sub(' ', text)
    text = NON_ALPHANUM.sub(' ', text)
    text = MULTI_SPACE.sub(' ', text).strip()
    tokens = [t for t in text.split() if t not in STOPWORDS]
    lemmas = [LEMMATIZER.lemmatize(t) for t in tokens]
    return ' '.join(lemmas)
```

---

src/model_utils.py

```
import joblib
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
import pandas as pd


def build_pipeline() -> Pipeline:
    return Pipeline([
        ('tfidf', TfidfVectorizer(max_features=10000, ngram_range=(1,2))),
        ('clf', LogisticRegression(max_iter=1000))
    ])


def train_and_save(csv_path: str, text_col: str, label_col: str, model_path:
str) -> dict:
    df = pd.read_csv(csv_path)
    df = df.dropna(subset=[text_col, label_col])
    X = df[text_col].astype(str)
    y = df[label_col]

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42, stratify=y)
    pipe = build_pipeline()
    pipe.fit(X_train, y_train)

    preds = pipe.predict(X_test)
```

```
    report = classification_report(y_test, preds, output_dict=True)

    joblib.dump(pipe, model_path)
    return report


def load_model(path: str):
    return joblib.load(path)


def predict_text(model, texts):
    return model.predict(texts), model.predict_proba(texts)
```

---

src/train.py

```
"""Train both fake-news and hate-speech models using sample CSVs.

Run:
    python src/train.py

This will write model files to `models/` and print evaluation reports.
"""
import os
from src.model_utils import train_and_save

os.makedirs('models', exist_ok=True)

print('Training Fake News model...')
report_fn = train_and_save('data/fake_news_sample.csv', text_col='text',
label_col='label', model_path='models/fake_news_model.joblib')
print('Fake News classification report:')
print(report_fn)

print('\nTraining Hate Speech model...')
report_hs = train_and_save('data/hate_speech_sample.csv', text_col='text',
label_col='label', model_path='models/hate_speech_model.joblib')
print('Hate Speech classification report:')
print(report_hs)
```

---

src/bot.py

```
"""Telegram bot for text moderation demo.

Set environment variable TELEGRAM_BOT_TOKEN with your bot token.
```

```
Commands:
 - /start : show usage
 - /check <text> : classify a short text for fake-news and hate-speech

To run locally:
    export TELEGRAM_BOT_TOKEN='123:ABC'  # Windows: set
TELEGRAM_BOT_TOKEN=...
    python src/bot.py
"""
import os
import logging
from telegram import Update
from telegram.ext import ApplicationBuilder, CommandHandler, ContextTypes,
MessageHandler, filters
from src.preprocess import clean_text
from src.model_utils import load_model

logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)

BOT_TOKEN = os.getenv('TELEGRAM_BOT_TOKEN')
if not BOT_TOKEN:
    logger.error('Please set TELEGRAM_BOT_TOKEN environment variable and
restart the bot.')

# Load models (expect models/ directory)
fake_model = None
hate_model = None
try:
    fake_model = load_model('models/fake_news_model.joblib')
    hate_model = load_model('models/hate_speech_model.joblib')
    logger.info('Models loaded successfully')
except Exception as e:
    logger.warning('Could not load models. Make sure you ran src/train.py
first. %s', e)


async def start(update: Update, context: ContextTypes.DEFAULT_TYPE):
    await update.message.reply_text(
        'Hi — I am a moderation demo bot.\\n\\nUse /check followed by your
text to classify for fake news and hate speech.\\nExample:\\n/check Some
headline about vaccination'
    )


async def check(update: Update, context: ContextTypes.DEFAULT_TYPE):
    text = ' '.join(context.args)
    if not text:
        await update.message.reply_text('Please provide text after /check
command.')
        return
```

```
    cleaned = clean_text(text)
    response_lines = [f'Original: {text}\\n', f'Cleaned: {cleaned}\\n']

    if fake_model is not None:
        pred, probs = fake_model.predict([cleaned]),
fake_model.predict_proba([cleaned])
        response_lines.append(f'Fake News Prediction: {pred[0]} (confidence
{probs[0].max():.2f})')
    else:

response_lines.append('Fake News Prediction: model not available. Run `python
src/train.py`.')

    if hate_model is not None:
        pred2, probs2 = hate_model.predict([cleaned]),
hate_model.predict_proba([cleaned])
        response_lines.append(f'Hate Speech Prediction: {pred2[0]}
(confidence {probs2[0].max():.2f})')
    else:
        response_lines.append('Hate Speech Prediction: model not available.
Run `python src/train.py`.')

    await update.message.reply_text('\\n'.join(response_lines))


async def echo(update: Update, context: ContextTypes.DEFAULT_TYPE):
    # Allow users to just send messages without /check
    text = update.message.text
    cleaned = clean_text(text)
    if fake_model is None or hate_model is None:
        await update.message.reply_text('Models missing — run `python src/
train.py` to train and save models in models/.')
        return
    pred_fn = fake_model.predict([cleaned])[0]
    pred_hs = hate_model.predict([cleaned])[0]
    await update.message.reply_text(f'Fake: {pred_fn} | Hate: {pred_hs}')


def main():
    if not BOT_TOKEN:

print('Set TELEGRAM_BOT_TOKEN environment variable before running the bot.')
        return

    app = ApplicationBuilder().token(BOT_TOKEN).build()

    app.add_handler(CommandHandler('start', start))
    app.add_handler(CommandHandler('check', check))
    app.add_handler(MessageHandler(filters.TEXT & ~filters.COMMAND, echo))
```

```
    print('Bot starting... Press Ctrl+C to stop.')
    app.run_polling()


if __name__ == '__main__':
    main()
```

---

data/fake_news_sample.csv (example rows)

```
text,label
"New study shows chocolate cures cancer",fake
"Government announces new tax relief for low income families",real
"Celebrity died after drinking soda — sources say",fake
"Local elections scheduled on next Monday",real
```

data/hate_speech_sample.csv (example rows)

```
text,label
"I hate those people and they should be removed",hate
"I dislike the policy but not the people",neutral
"Such a disgusting group, they are all criminals",hate
"We should discuss alternatives peacefully",neutral
```

---

requirements.txt

```
pandas
scikit-learn
joblib
nltk
python-telegram-bot==20.4
```

README.md (to paste into repo)

```
# Fake News & Hate Speech Detector (Telegram Bot)

Lightweight dual-classifier for detecting misinformation (fake news) and hate
speech in text. Built with TF-IDF + Logistic Regression and exposed through a
Telegram bot for live testing.
```

```
## Quickstart
1. Clone the repo
```

git clone https://github.com/<your-username>/fake-news-hate-speech-detector cd fake-news-hate-speech-detector

```
2. Create virtual environment and install
```

python -m venv venv source venv/bin/activate # Windows: venv\Scripts\activate pip install -r requirements.txt

```
3. Train models (uses small sample CSVs provided)
```

python src/train.py

```
4. Run the Telegram bot
```

export TELEGRAM_BOT_TOKEN='YOUR_TOKEN' python src/bot.py

```
Then message your bot or use `/check` command.
```

How to run in VS Code

1. Open the folder in VS Code
2. Create a Python virtual environment (`python -m venv venv`) and activate it
3. Install requirements with `pip install -r requirements.txt`
4. Run `python src/train.py` in the Terminal to train models
5. Set your Telegram token and run `python src/bot.py`

If you want, I can now: - Generate the actual Python files in the canvas separately so you can copy/paste each file faster, OR - Create a polished GitHub `README.md` (already included), OR - Generate the one-line + three-bullet resume entry tailored to a specific job role (reply with role number 1-5 like before).

Which of those next?