**Cyber Bullying Detection System in Social Media**

* **.gitignore**

# Logs

logs

\*.log

npm-debug.log\*

yarn-debug.log\*

yarn-error.log\*

pnpm-debug.log\*

lerna-debug.log\*

node\_modules

dist

dist-ssr

\*.local

# Editor directories and files

.vscode/\*

!.vscode/extensions.json

.idea

.DS\_Store

\*.suo

\*.ntvs\*

\*.njsproj

\*.sln

\*.sw?

* **Main.jsx**

import React from 'react'

import NavBar from './components/NavBar'

import { Outlet } from 'react-router-dom'

const Main = () => {

return (

<>

<div>

<div className="fixed top-0 h-20 w-full bg-white z-50">

<NavBar/>

</div>

<div className='absolute top-0 z-10'>

<Outlet/>

</div>

</div>

</>

)

}

export default Main

* **app.py**

from flask import Flask, request, jsonify

import pandas as pd

import os

import pickle

from flask\_cors import CORS

from flask\_jwt\_extended import create\_access\_token, jwt\_required, JWTManager, get\_jwt\_identity

from sklearn.feature\_extraction.text import TfidfVectorizer

# Initialize Flask app

app = Flask(\_\_name\_\_)

# CORS

CORS(app)

app.config["JWT\_SECRET\_KEY"] = "supersecretkey"

jwt = JWTManager(app)

# Load the trained model and vectorizer from disk

with open('model.pkl', 'rb') as model\_file:

model = pickle.load(model\_file)

with open('vectorizer.pkl', 'rb') as vec\_file:

vectorizer = pickle.load(vec\_file)

#for pie chart data

#Load datasets

dataset\_folder = os.path.join(os.path.dirname(\_\_file\_\_), "datasets")

bullying\_file = os.path.join(dataset\_folder,"Aggressive\_All.csv")

non\_bullying\_file = os.path.join(dataset\_folder,"Non\_Aggressive\_All.csv")

bullying\_data = pd.read\_csv(bullying\_file,encoding="utf-8")

non\_bullying\_data = pd.read\_csv(non\_bullying\_file,encoding="utf-8")

bullying\_data["label"] = 1

non\_bullying\_data["label"] = 0

df = pd.concat([bullying\_data,non\_bullying\_data],ignore\_index=True)

df = df.dropna(subset=['Message'])

class\_counts = df['label'].value\_counts()

total = class\_counts.sum()

bullying\_percentage = (class\_counts[1] / total) \* 100 if 1 in class\_counts else 0

non\_bullying\_percentage = (class\_counts[0] / total) \* 100 if 0 in class\_counts else 0

VALID\_USERNAME = "Admin"

VALID\_PASSWORD = "Admin"

# login

@app.route('/login',methods=['POST'])

def login():

data = request.json

username = data.get("username")

password = data.get("password")

if username == VALID\_USERNAME and password == VALID\_PASSWORD:

access\_token = create\_access\_token(identity=username)

return jsonify({"token" : access\_token, "username" : username}),200

else:

return jsonify({"message" : "Invalid credentials"}),401

@app.route('/')

def home():

return "Welcome to the Cyberbullying Detection API!"

@app.route('/predict', methods=['POST'])

def predict():

# Get the input text from the user (assuming it's sent as JSON)

input\_data = request.get\_json() # JSON input

input\_text = input\_data['text'] # Get the text field from the JSON input

# Vectorize the input text using the loaded vectorizer

input\_text\_vec = vectorizer.transform([input\_text])

print("input text:",input\_text\_vec.shape)

# Predict using the loaded model

prediction = model.predict(input\_text\_vec)

print("Prediction:", prediction)

# Return the prediction result as a response

result = "Bullying" if prediction[0] == 1 else "Non-Bullying"

return jsonify({"prediction": result})

@app.route('/data-distribution',methods=['GET'])

def get\_data\_distribution():

return jsonify({

"bullying" : bullying\_percentage,

"nonbullying" : non\_bullying\_percentage

})

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

* **eslint.config.js**

import js from '@eslint/js'

import globals from 'globals'

import reactHooks from 'eslint-plugin-react-hooks'

import reactRefresh from 'eslint-plugin-react-refresh'

export default [

{ ignores: ['dist'] },

{

files: ['\*\*/\*.{js,jsx}'],

languageOptions: {

ecmaVersion: 2020,

globals: globals.browser,

parserOptions: {

ecmaVersion: 'latest',

ecmaFeatures: { jsx: true },

sourceType: 'module',

},

},

plugins: {

'react-hooks': reactHooks,

'react-refresh': reactRefresh,

},

rules: {

...js.configs.recommended.rules,

...reactHooks.configs.recommended.rules,

'no-unused-vars': ['error', { varsIgnorePattern: '^[A-Z\_]' }],

'react-refresh/only-export-components': [

'warn',

{ allowConstantExport: true },

],

},

},

]

* **index.html**

<!doctype html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<link rel="icon" type="image/svg+xml" href="/vite.svg" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>Vite + React</title>

</head>

<body>

<div id="root"></div>

<script type="module" src="/src/main.jsx"></script>

</body>

</html>

* **package-lock.json**

{

"name": "cyberbullying-frontend",

"private": true,

"version": "0.0.0",

"type": "module",

"scripts": {

"dev": "vite",

"build": "vite build",

"lint": "eslint .",

"preview": "vite preview"

},

"dependencies": {

"@tailwindcss/vite": "^4.0.14",

"axios": "^1.8.3",

"chart.js": "^4.4.8",

"react": "^19.0.0",

"react-chartjs-2": "^5.3.0",

"react-dom": "^19.0.0",

"react-router-dom": "^7.3.0",

"tailwindcss": "^4.0.14"

},

"devDependencies": {

"@eslint/js": "^9.21.0",

"@types/react": "^19.0.10",

"@types/react-dom": "^19.0.4",

"@vitejs/plugin-react": "^4.3.4",

"eslint": "^9.21.0",

"eslint-plugin-react-hooks": "^5.1.0",

"eslint-plugin-react-refresh": "^0.4.19",

"globals": "^15.15.0",

"vite": "^6.2.0"

}

}

* **package.json**

{

"name": "cyberbullying-frontend",

"private": true,

"version": "0.0.0",

"type": "module",

"scripts": {

"dev": "vite",

"build": "vite build",

"lint": "eslint .",

"preview": "vite preview"

},

"dependencies": {

"@tailwindcss/vite": "^4.0.14",

"axios": "^1.8.3",

"chart.js": "^4.4.8",

"react": "^19.0.0",

"react-chartjs-2": "^5.3.0",

"react-dom": "^19.0.0",

"react-router-dom": "^7.3.0",

"tailwindcss": "^4.0.14"

},

"devDependencies": {

"@eslint/js": "^9.21.0",

"@types/react": "^19.0.10",

"@types/react-dom": "^19.0.4",

"@vitejs/plugin-react": "^4.3.4",

"eslint": "^9.21.0",

"eslint-plugin-react-hooks": "^5.1.0",

"eslint-plugin-react-refresh": "^0.4.19",

"globals": "^15.15.0",

"vite": "^6.2.0"

}

}

* **train\_model.py**

import pandas as pd

import os

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.ensemble import RandomForestClassifier

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

from sklearn.metrics import classification\_report

from imblearn.over\_sampling import SMOTE

from imblearn.under\_sampling import RandomUnderSampler

from sklearn.linear\_model import LogisticRegression

import pickle

dataset\_folder = os.path.join(os.path.dirname(\_\_file\_\_), "datasets")

bullying\_file = os.path.join(dataset\_folder,"Aggressive\_All.csv")

non\_bullying\_file = os.path.join(dataset\_folder,"Non\_Aggressive\_All.csv")

bullying\_data = pd.read\_csv(bullying\_file,encoding='utf-8') # Update with actual file path

non\_bullying\_data = pd.read\_csv(non\_bullying\_file,encoding='utf-8')

# Ensure DataFrame type consistency after assigning labels

bullying\_data.reset\_index(drop=True, inplace=True)

non\_bullying\_data.reset\_index(drop=True, inplace=True)

# Rename the "message" column to "text" for consistency

bullying\_data.rename(columns={"Message" : "text"}, inplace=True)

non\_bullying\_data.rename(columns={"Message" : "text"}, inplace=True)

# Check sample data before labeling

print("Bullying Data Sample (should contain aggressive text):")

print(bullying\_data[['text']].sample(10))

print("Non-Bullying Data Sample (should contain non-aggressive text):")

print(non\_bullying\_data[['text']].sample(10))

# Assign labels: 1 for bullying, 0 for non-bullying

bullying\_data["label"] = 1

non\_bullying\_data["label"] = 0 # Assign label

# Debugging: Check if non\_bullying\_data still has text values

print("Non-Bullying Data After Label Assignment:")

print(non\_bullying\_data.sample(5))

# Combine both datasets

df = pd.concat([bullying\_data, non\_bullying\_data],ignore\_index=True)

df = df.dropna(subset=['text'])

# Convert the data to a pandas DataFrame

df = df[['text','label']]

df.reset\_index(drop=True, inplace=True)

print(df.info())

print(df.sample(10))

print(df.head())

# Split the data into features (X) and labels (y)

X = df['text']

y = df['label']

# Split the dataset into training and testing sets (80% train, 20% test)

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

# Vectorize the text data (Convert text to numerical features)

vectorizer = TfidfVectorizer(stop\_words='english', ngram\_range=(1, 2), max\_df=0.7, min\_df=10, max\_features=20000)

# Fit and transform the training data and transform the test data

X\_train\_vec = vectorizer.fit\_transform(X\_train)

X\_test\_vec = vectorizer.transform(X\_test)

# Check original class distribution

print("Original class distribution:")

print(y\_train.value\_counts())

rus = RandomUnderSampler(random\_state=42)

X\_train\_vec, y\_train = rus.fit\_resample(X\_train\_vec, y\_train)

# Check class distribution after undersampling

print("After undersampling class distribution:")

print(pd.Series(y\_train).value\_counts())

# Create a LogisticRegression model

lr\_model = LogisticRegression(class\_weight="balanced", max\_iter=1000)

# Train the model on the training data

lr\_model.fit(X\_train\_vec, y\_train)

# Test the model

y\_pred = lr\_model.predict(X\_test\_vec)

# Print classification report to evaluate performance

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

# Save the trained model and vectorizer to disk

with open('model.pkl', 'wb') as model\_file:

pickle.dump(lr\_model, model\_file)

with open('vectorizer.pkl', 'wb') as vec\_file:

pickle.dump(vectorizer, vec\_file)

* **vite.config.js**

import { defineConfig } from 'vite'

import tailwindcss from '@tailwindcss/vite'

import react from '@vitejs/plugin-react'

// https://vite.dev/config/

export default defineConfig({

plugins: [react(),tailwindcss()],

})