**1. Imports and Initialization**

python

Copy code

import pandas as pd import re import string from flask import Flask, request, jsonify, render\_template from sklearn.model\_selection import train\_test\_split from sklearn.feature\_extraction.text import TfidfVectorizer from sklearn.linear\_model import LogisticRegression from sklearn.tree import DecisionTreeClassifier from sklearn.ensemble import RandomForestClassifier import psycopg2 app = Flask(\_\_name\_\_)

* **Imports:** The script imports necessary libraries for data manipulation (**pandas**), regular expressions (**re**), string operations, machine learning (**sklearn**), web framework (**Flask**), and database connection (**psycopg2**).
* **Flask App Initialization:** A Flask application is initialized.

**2. Database Connection**

python

Copy code

def get\_db\_connection(): conn = psycopg2.connect( dbname='newsdetectiondb', user='tanaya', password='cQsflHxOnfKr7Un7hCW7ZJVViGlqU1Sq', host='dpg-cp4sfj21hbls73f5ven0-a.singapore-postgres.render.com', port='5432' ) return conn

* **Database Connection Function:** Establishes a connection to a PostgreSQL database using credentials and returns the connection object.

**3. Loading Data from Database**

python

Copy code

def load\_data\_from\_db(): conn = get\_db\_connection() query\_fake = "SELECT text, 0 AS class FROM Fake\_News" query\_true = "SELECT text, 1 AS class FROM True\_News" data\_fake = pd.read\_sql(query\_fake, conn) data\_true = pd.read\_sql(query\_true, conn) conn.close() return pd.concat([data\_fake, data\_true]).sample(frac=1).reset\_index(drop=True)

* **Load Data Function:** Executes SQL queries to fetch data from **Fake\_News** and **True\_News** tables. The data is combined, shuffled, and returned as a DataFrame.

**4. Text Preprocessing**

python

Copy code

def wordopt(text): text = text.lower() text = re.sub(r'\[.\*?\]', '', text) text = re.sub(r"\\W", " ", text) text = re.sub(r'https?://\S+|www\.\S+', '', text) text = re.sub(r'[%s]' % re.escape(string.punctuation), '', text) text = re.sub(r'\n', '', text) text = re.sub(r'\w\*\d\w\*', '', text) return text data['text'] = data['text'].apply(wordopt)

* **Text Preprocessing Function:** Cleans the text data by converting it to lowercase, removing special characters, URLs, punctuation, newlines, and words with digits.

**5. Splitting Data**

python

Copy code

x = data['text'] y = data['class'] x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.25, random\_state=0)

* **Train-Test Split:** The dataset is split into training and testing sets with 25% of the data reserved for testing.

**6. Text Vectorization**

python

Copy code

vectorizer = TfidfVectorizer() xv\_train = vectorizer.fit\_transform(x\_train) xv\_test = vectorizer.transform(x\_test)

* **TF-IDF Vectorization:** Converts the text data into numerical vectors using TF-IDF (Term Frequency-Inverse Document Frequency).

**7. Model Training**

python

Copy code

models = { "Logistic Regression": LogisticRegression(), "Decision Tree": DecisionTreeClassifier(), "Random Forest": RandomForestClassifier(random\_state=0) } for model\_name, model in models.items(): model.fit(xv\_train, y\_train)

* **Training Models:** Initializes and trains three different machine learning models: Logistic Regression, Decision Tree, and Random Forest.

**8. Flask Routes**

python

Copy code

@app.route('/') def home(): return render\_template('index.html') @app.route('/about') def about(): return render\_template('about.html') @app.route('/contact') def contact(): return render\_template('contact.html')

* **Routes for Static Pages:** Defines routes for the home, about, and contact pages, rendering corresponding HTML templates.

**9. Prediction Endpoint**

python

Copy code

@app.route('/predict', methods=['POST']) def predict(): if request.method == 'POST': title = request.form.get('title', '') title\_processed = wordopt(title) # Check the input title against the database conn = get\_db\_connection() cur = conn.cursor() cur.execute("SELECT title FROM True\_News WHERE title = %s", (title,)) db\_result = cur.fetchone() conn.close() if db\_result: db\_verdict = "True News" else: db\_verdict = "Fake News" return render\_template('index.html', result=db\_verdict)

* **Prediction Route:** Handles POST requests for predicting whether a given news title is true or fake. It preprocesses the title, checks if it exists in the **True\_News** table, and renders the result on the home page.

**10. Running the Application**

python

Copy code

if \_\_name\_\_ == '\_\_main\_\_': app.run(debug=True)

* **Running the Flask App:** Starts the Flask development server with debug mode enabled.

**Summary**

* The script sets up a Flask web application that connects to a PostgreSQL database.
* It loads and preprocesses news data, then trains three machine learning models.
* It defines routes for static pages and a prediction endpoint to classify news titles based on database entries.

Top of Form