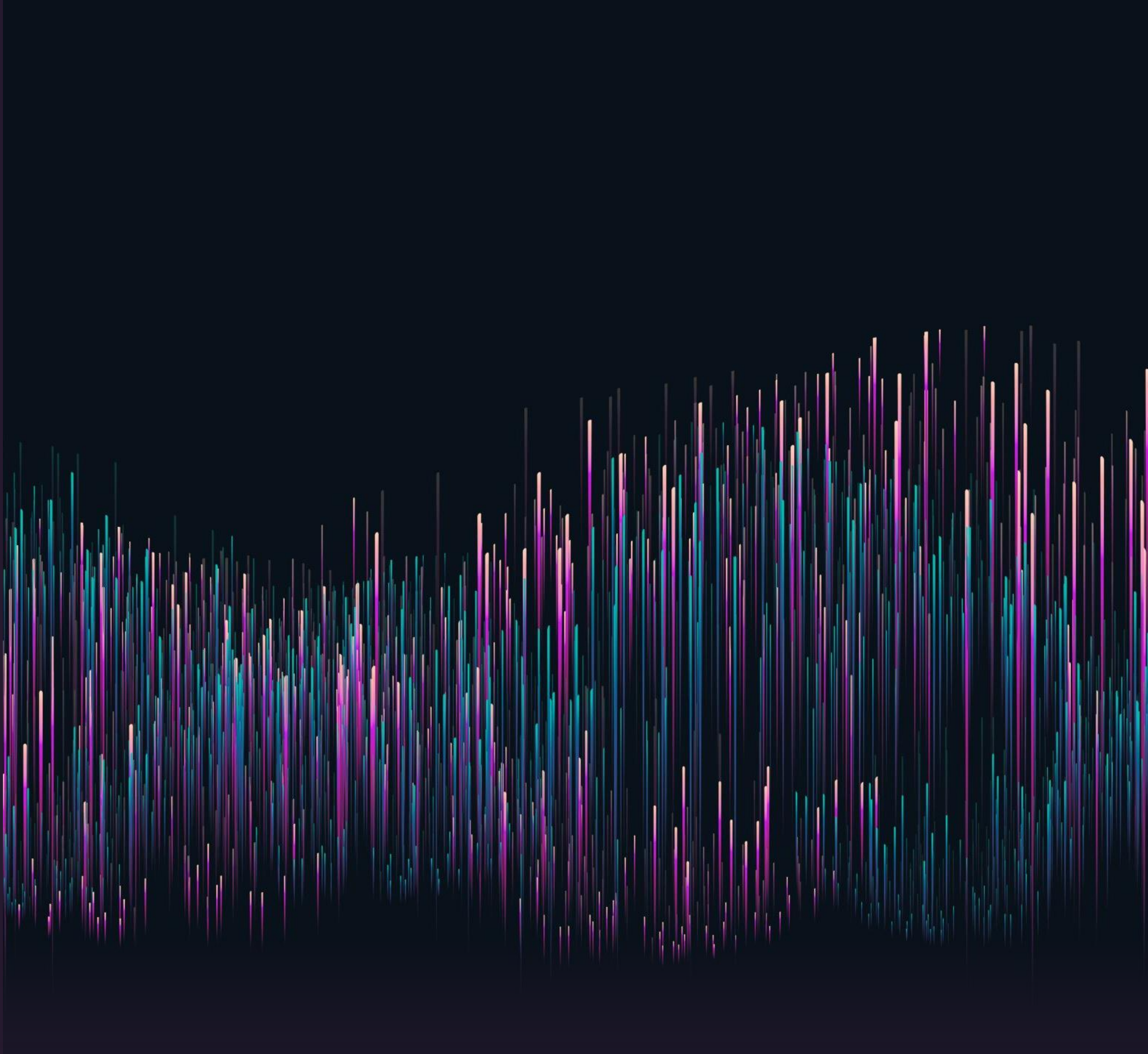
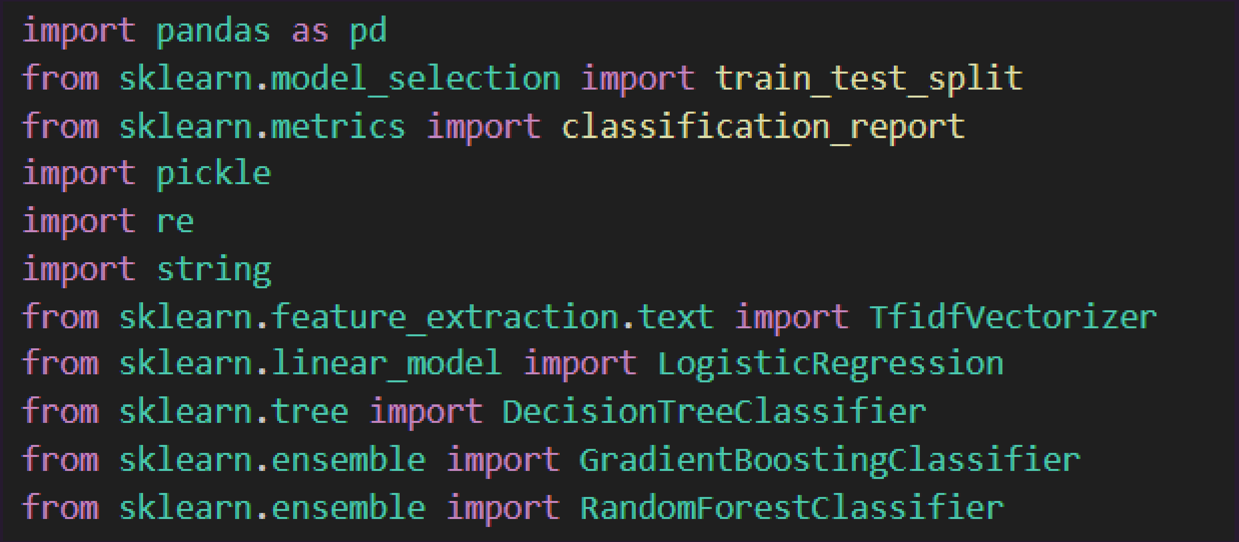
Fake news detection



Followed steps

1. Find true and false datasets online.
2. Preprocess all the data within both files.
3. Train four models (Logistic Regression, Decision Tree Classifier, Gradient Boosting Classifier, Random Forest Classifier) using vectorized data.
4. Save all the models and the vectorization technique for reuse in the backend application.
5. Develop a Python backend application capable of responding to "detect" requests to predict whether the news is fake or not.
6. Build a Node.js application to facilitate communication between the frontend page and the Python app.
7. Create a basic HTML file where users can input news and send it to the backend to determine if it's fake or not.

Importing necessary libraries

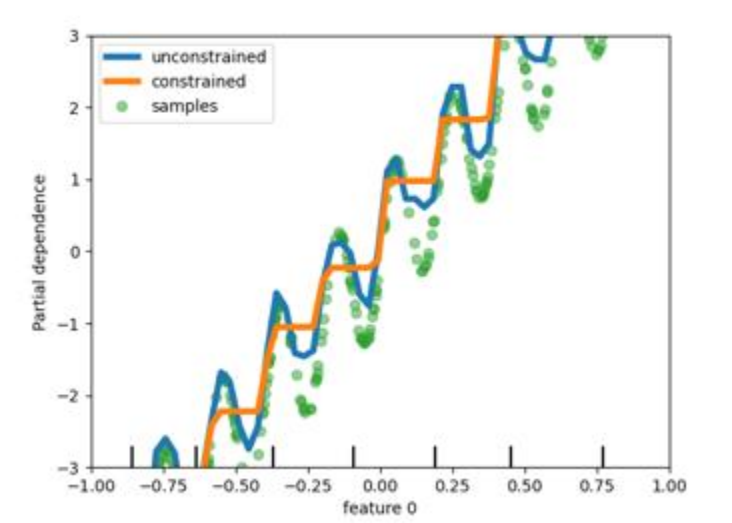


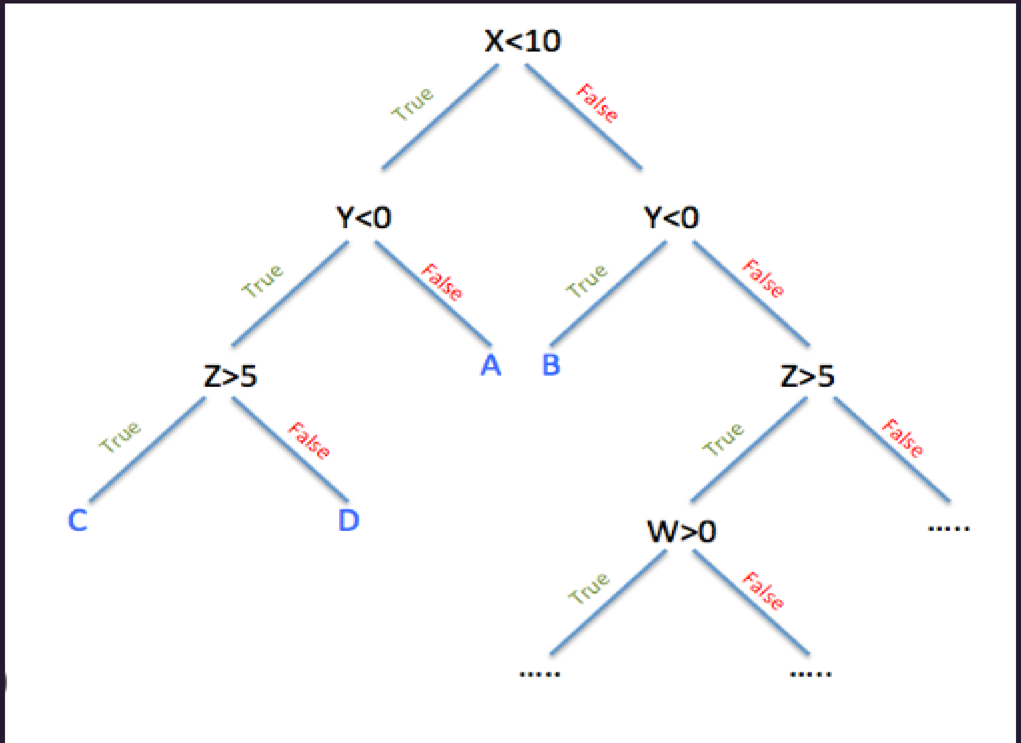
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
import pickle
import re
import string
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import RandomForestClassifier
```

Used models

Logistic Regression:

- Precision: 0.99 for both classes
- Recall: 0.99 for both classes
- F1-score: 0.99 for both classes
- The Logistic Regression model shows high precision, recall, and F1-score for both classes, indicating excellent performance in classifying fake and not fake news.



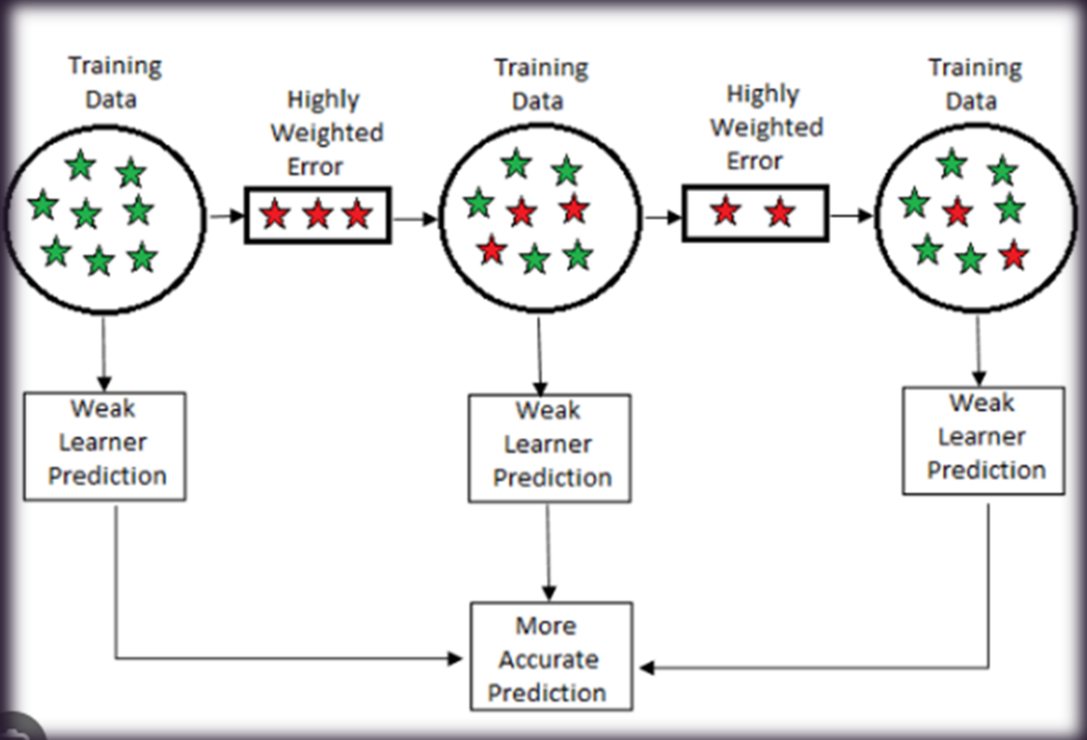


Decision Tree Classifier:

- Precision: 1.00 for both classes
- Recall: 1.00 for both classes
- F1-score: 1.00 for both classes
- The Decision Tree Classifier achieves perfect precision, recall, and F1-score for both classes, indicating optimal performance in classifying fake and not fake news.

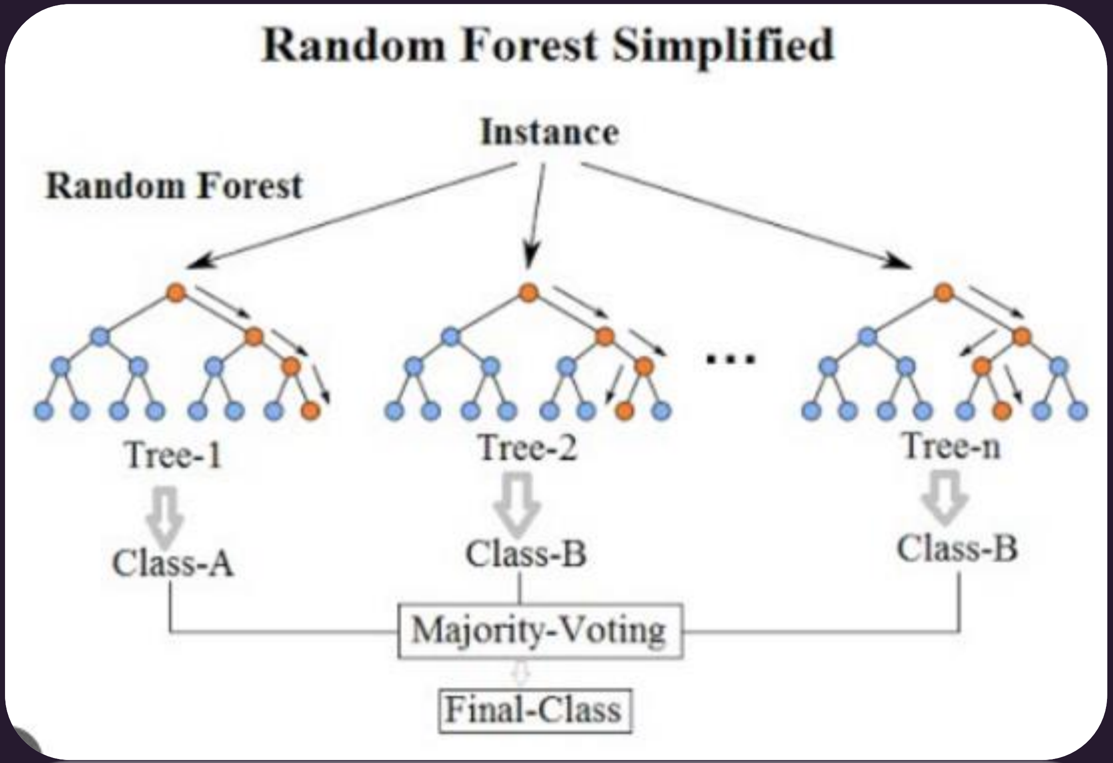
Gradient Boosting Classifier:

- Precision: 0.99 for class 0, 1.00 for class 1
- Recall: 0.99 for class 0, 1.00 for class 1
- F1-score: 0.99 for class 0, 1.00 for class 1
- The Gradient Boosting Classifier demonstrates high precision, recall, and F1-score for both classes, with a slightly higher performance in classifying not fake news.

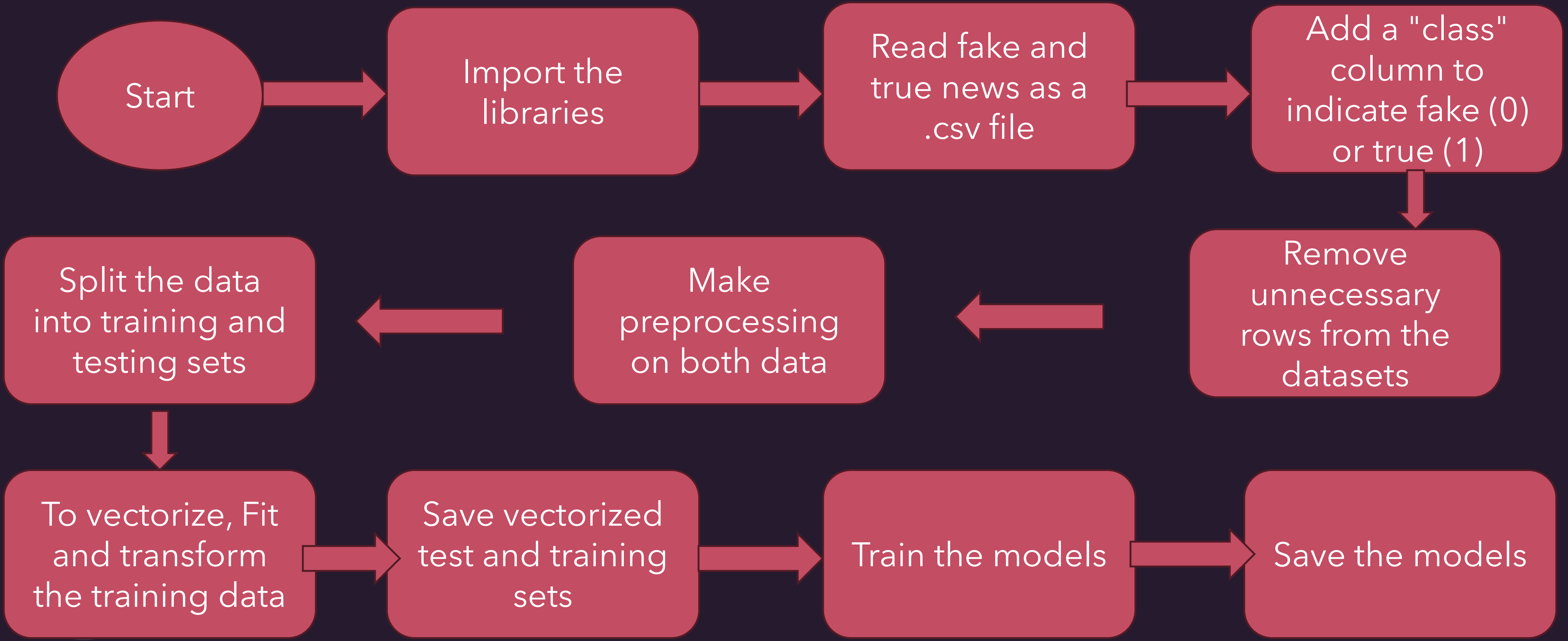


Random Forest Classifier:

- Precision: 0.99 for both classes
- Recall: 0.99 for both classes
- F1-score: 0.99 for both classes
- The Random Forest Classifier achieves high precision, recall, and F1-score for both classes, indicating strong performance in classifying fake and not fake news.



Training part diagram of the system



Start

Import the libraries

Read fake and true news as a .csv file

Add a "class" column to indicate fake (0) or true (1)

Remove unnecessary rows from the datasets

Make preprocessing on both data

Split the data into training and testing sets

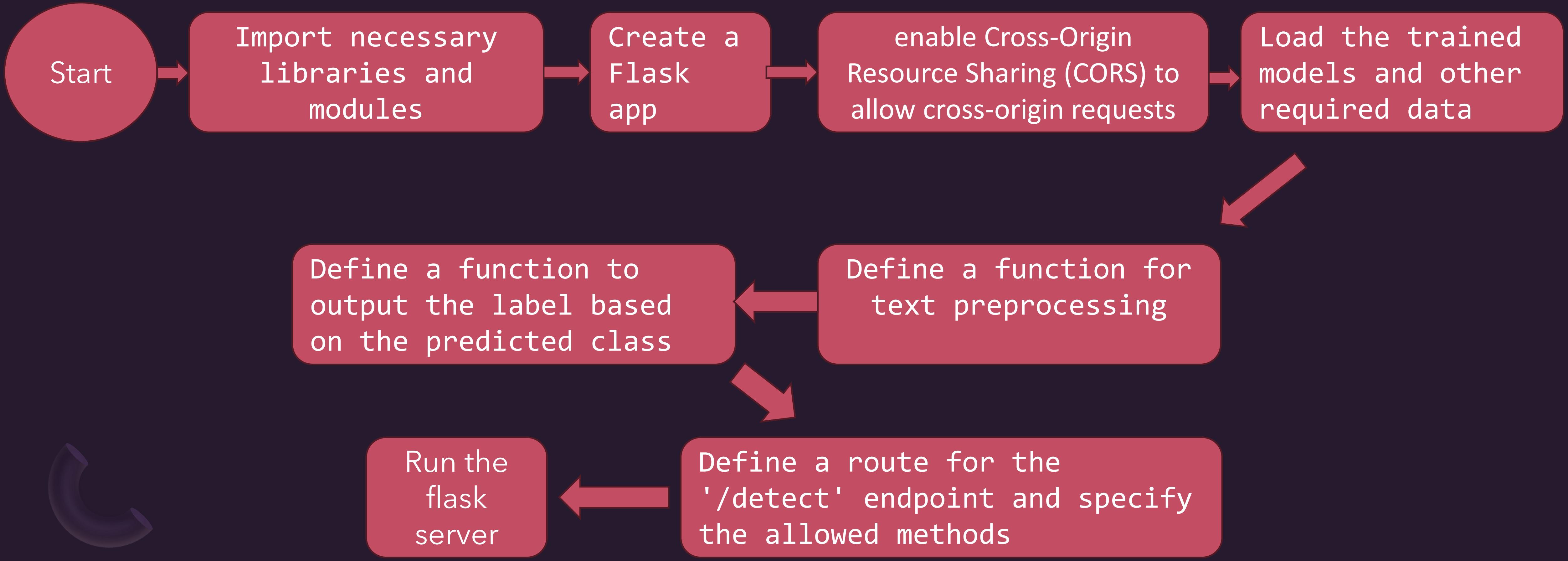
To vectorize, Fit and transform the training data

Save vectorized test and training sets

Train the models

Save the models

Python application diagram



Start

Import necessary libraries and modules

Create a Flask app

enable Cross-Origin Resource Sharing (CORS) to allow cross-origin requests

Load the trained models and other required data

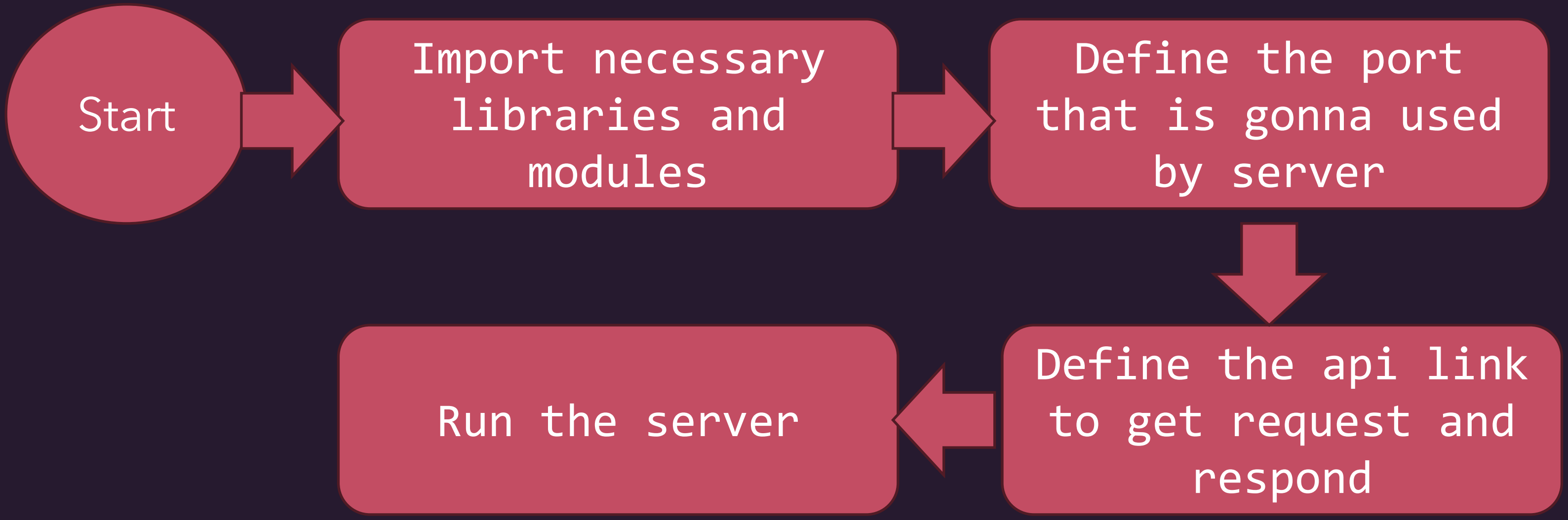
Define a function for text preprocessing

Define a function to output the label based on the predicted class

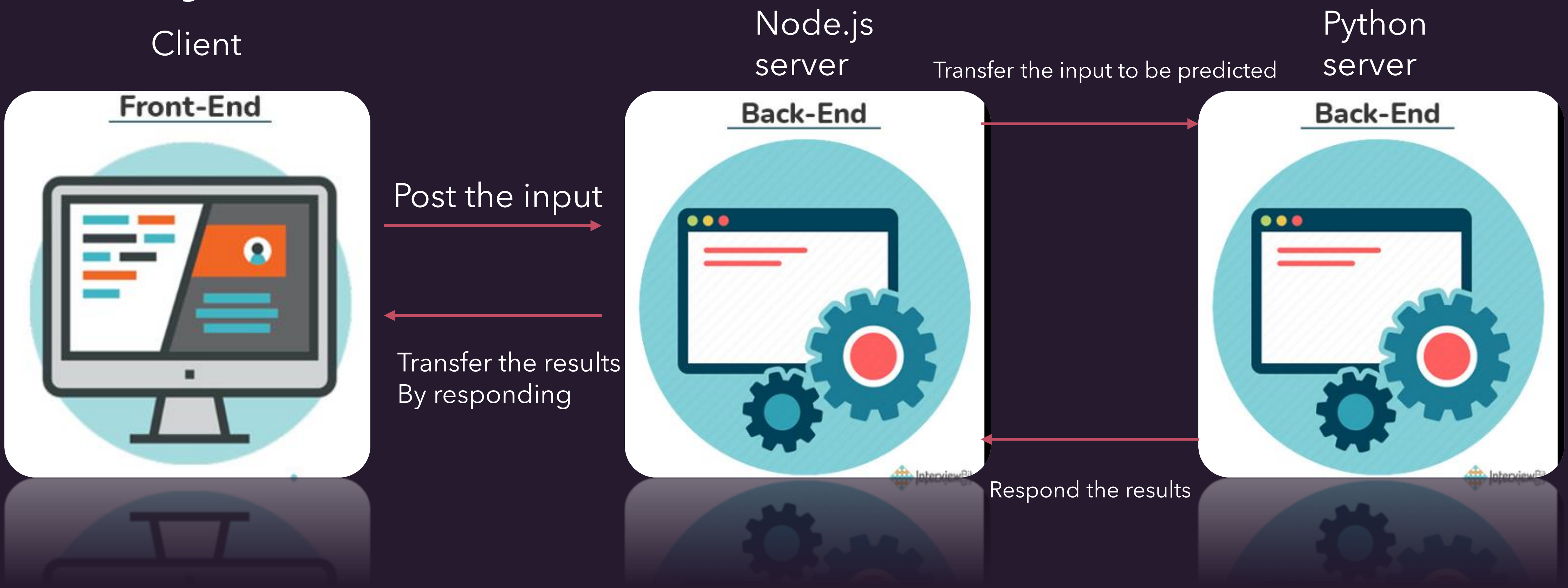
Define a route for the '/detect' endpoint and specify the allowed methods

Run the flask server

Node application diagram


Start
Import necessary libraries and modules
Define the port that is gonna used by server
Define the api link to get request and respond
Run the server

Life cycle of servers and client



Client

Node.js server

Python server

Post the input

Transfer the results
By responding

Transfer the input to be predicted

Respond the results