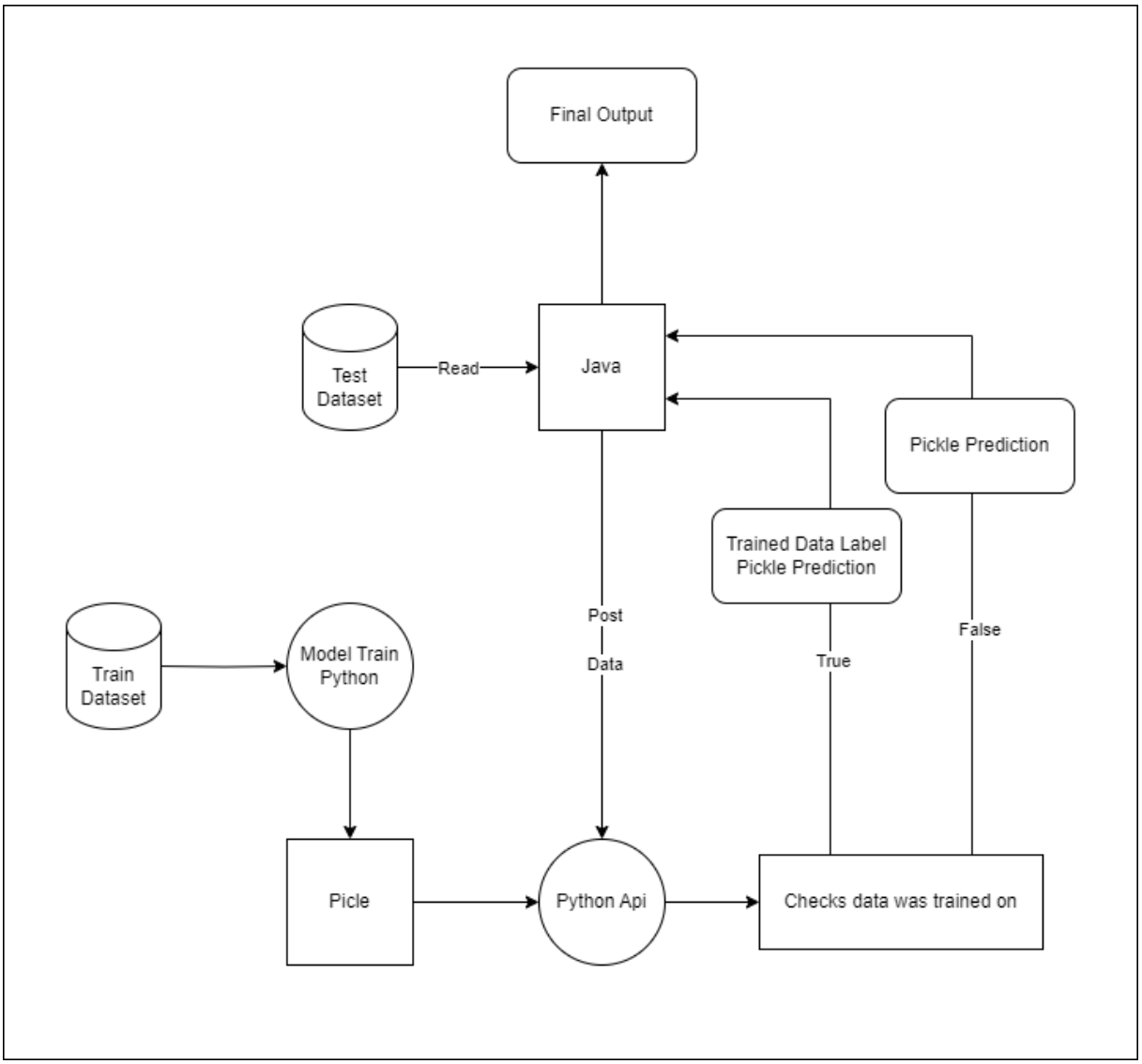

Python api

```
from flask import Flask, request, jsonify
import pickle
import numpy as np
import pandas as pd

app = Flask(__name__)

# Load the model (pipeline including preprocessing and classifier)
with open('C:\\Users\\xxtra\\OneDrive\\Desktop\\ave staj\\javaliödev\\decision_tree_model.pkl', 'rb') as file:
    model = pickle.load(file)

def is_data_exists(data, dataset):
    dataset_wo_last_col = dataset.iloc[:, :-1]  # Exclude the last column from the dataset
    mask = dataset_wo_last_col.isin(data.values).all(axis=1)
    if mask.any():
```

```
        matching_indices = dataset_wo_last_col.index[mask]
        print(matching_indices)
        return True, matching_indices
    else:
        return False, None



thatcsv=pd.read_csv("C:\\Users\\xxtra\\OneDrive\\Desktop\\ave staj\\javalıödev\\training_data_with_clusters.csv")



@app.route('/predict', methods=['POST'])
def predict():
    try:
        # Get the data from the POST request
        data = request.get_json(force=True)
        df = pd.DataFrame([data])
        exists_in_data,match = is_data_exists(df.iloc[0].astype(str), thatcsv.astype(str))
        print(exists_in_data)


        if exists_in_data:
            loaded_y_pred = model.predict(df)

            response = {
                "message": "Data exists in the training set",
                "predicted_label": str(loaded_y_pred[0]),
                "training_cluster": str(thatcsv.iloc[match[0]]["Cluster"])
            }
            return jsonify(response), 200
        else:
            loaded_y_pred = model.predict(df)
            response = {
                "message": "Data is not in the training set",
                "predicted_label": str(loaded_y_pred[0])
            }
            return jsonify(response), 201
    except Exception as e:
        app.logger.error(f"Error in /predict endpoint: {str(e)}")
        return jsonify({"error": str(e)}), 500



if __name__ == '__main__':
    app.run(port=5000,debug=True)
```

Java

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.net.URL;

public class Main {

    public static void main(String[] args) {
        String csvFile = "C:\\Users\\xxtra\\OneDrive\\Desktop\\ave staj\\javalıödev\\test_data.csv"; // CSV dosyasının yolu
        String line;
        String cvsSplitBy = ","; // CSV dosyasındaki ayırıcı karakter
        int numberOfRowsToSend = 13; // Gönderilecek satır sayısı
```

```
    // Kolon isimlerini belirtiyoruz (id'yi hariç tutuyoruz)
    String[] columnNames = {"Gender", "Age", "Height", "Weight", "family_history_with_overweight",
        "FAVC", "FCVC", "NCP", "CAEC", "SMOKE", "CH2O", "SCC", "FAF", "TUE",
        "CALC", "MTRANS"};

    try (BufferedReader br = new BufferedReader(new FileReader(csvFile))) {
      int currentRow = 0;

      // İlk satırları oku ve POST isteği gönder
      while ((line = br.readLine()) != null && currentRow < numberOfRowsToSend) {
        // İlk satırı atlıyoruz
        if (currentRow == 0) {
          currentRow++;
          continue;
        }

        // Satırı ayırarak parçalara böl (id'yi hariç tutuyoruz)
        String[] columns = line.split(cvsSplitBy);

        // JSON formatında veri hazırla
        StringBuilder jsonInputString = new StringBuilder("{");
        for (int i = 1; i < columns.length; i++) { // i = 1 ile başlayarak id'yi atlıyoruz
          jsonInputString.append("\"").append(columnNames[i - 1]).append("\": \"").append(columns[i]).append("\"");
          if (i < columns.length - 1) {
            jsonInputString.append(", ");
          }
        }
        jsonInputString.append("}");

        // POST isteği gönder
        URL url = new URL("http://127.0.0.1:5000/predict");
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();
        conn.setRequestMethod("POST");
        conn.setRequestProperty("Content-Type", "application/json; utf-8");
        conn.setRequestProperty("Accept", "application/json");
        conn.setDoOutput(true);

        try (OutputStream os = conn.getOutputStream()) {
          byte[] input = jsonInputString.toString().getBytes("utf-8");
          os.write(input, 0, input.length);
        }

        int responseCode = conn.getResponseCode();
        System.out.println("POST Response Code for row " + currentRow + " :: " + responseCode);

        // Yanıtı oku ve konsola yazdır
        try (BufferedReader in = new BufferedReader(new InputStreamReader(conn.getInputStream()))) {
          String inputLine;
          StringBuilder response = new StringBuilder();

          while ((inputLine = in.readLine()) != null) {
            response.append(inputLine);
          }
          // Yanıtı konsola yazdır
          System.out.println("Response for row " + currentRow + " :: " + response.toString());
        }

        // Bağlantıyı kapat
        conn.disconnect();

        currentRow++;
      }
    } catch (IOException e) {
      e.printStackTrace();
    }
  }
}
```

Results

```
C:\Users\xxtra\.jdks\openjdk-22.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.1.2\lib\idea_rt.jar=51177:C:\Program Files\JetBrains\Intell
POST Response Code for row 1 :: 201
Response for row 1 :: {  "message": "Data is not in the training set",  "predicted_label": "Obesity_Type_I"}
POST Response Code for row 2 :: 201
Response for row 2 :: {  "message": "Data is not in the training set",  "predicted_label": "Overweight_Level_II"}
POST Response Code for row 3 :: 201
Response for row 3 :: {  "message": "Data is not in the training set",  "predicted_label": "Obesity_Type_I"}
POST Response Code for row 4 :: 201
Response for row 4 :: {  "message": "Data is not in the training set",  "predicted_label": "Overweight_Level_I"}
POST Response Code for row 5 :: 201
Response for row 5 :: {  "message": "Data is not in the training set",  "predicted_label": "Obesity_Type_III"}
POST Response Code for row 6 :: 201
Response for row 6 :: {  "message": "Data is not in the training set",  "predicted_label": "Obesity_Type_I"}
POST Response Code for row 7 :: 200
Response for row 7 :: {  "message": "Data exists in the training set",  "predicted_label": "Obesity_Type_II",  "training_cluster": "Obesity_Type_II"}
POST Response Code for row 8 :: 201
Response for row 8 :: {  "message": "Data is not in the training set",  "predicted_label": "Insufficient_Weight"}
POST Response Code for row 9 :: 200
Response for row 9 :: {  "message": "Data exists in the training set",  "predicted_label": "Normal_Weight",  "training_cluster": "Normal_Weight"}
POST Response Code for row 10 :: 200
Response for row 10 :: {  "message": "Data exists in the training set",  "predicted_label": "Obesity_Type_III",  "training_cluster": "Obesity_Type_III"}
POST Response Code for row 11 :: 200
Response for row 11 :: {  "message": "Data exists in the training set",  "predicted_label": "Obesity_Type_I",  "training_cluster": "Obesity_Type_I"}
POST Response Code for row 12 :: 200
Response for row 12 :: {  "message": "Data exists in the training set",  "predicted_label": "Insufficient_Weight",  "training_cluster": "Insufficient_Weight"}

Process finished with exit code 0
```