Practical Assignment – 1

* Deploying ML model with Flask

NAME: Navneetkumar Ramanbhai Thakor

ID: 21CP031

SUBJECT: Machine Learning

**Application:** Find ideal Weight from height

**Model used:** Linear Regression

This model is predicting ideal weight of user from his/her height provided.

I had used dataset from Kaggle for this practical assignment

Link is following: <https://www.kaggle.com/datasets/burnoutminer/heights-and-weights-dataset>

Additionally, I had used sklearn and numpy library in this assignment.

Code section –

1)HTML: (templates/index.html)

-> I had used Javascript in this page it self.

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <link rel="stylesheet" href="../static/index.css"/>

    <title>project 1</title>

</head>

<body>

    <h3>Navneetkumar R. Thakor : 21CP031</h3>

    <div id="formContainer">

        <h1>Linear Regration</h1>

        <div id="formquestions">

            <div>

                <input type="text" required name="inputdata" id="inputdata" />

                <label for="inputdata">Enter you height (cem)</label>

            </div>

            <div>

                <input type="text" required name="ans" id="ans" />

                <label for="inputdata">calculated weight (Kg)</label>

            </div>

        </div>

        <button id="btn">Calculate</button>

    </div>

    <script>

        const button  = document.getElementById("btn");

        button.onclick = async () =>{

            const val = document.getElementById("inputdata");

            console.log(val.value)

            const url = `http://localhost:5000/calculate?height=${val.value}`;

            const response = await fetch(url,{

                method: "POST",

                headers:{

                    "Content-Type": "application/json"

                },

            })

            const ans = await response.json();

            document.getElementById("ans").value = ans;

        }

    </script>

</body>

</html>

2) CSS (static/index.css):

/\* css reset  \*/

\*{

    padding: 0;

    margin: 0;

    box-sizing: border-box;

}

body{

    display: flex;

    flex-direction: column;

    align-items: center;

    justify-content: center;

    min-height: 100vh;

    width: 100vw;

    overflow-x: hidden;

    background-color: rgb(41, 41, 46);

    color: white;

}

/\* form styling  \*/

#formContainer{

    width: 80vw;

    height: 70vh;

    border: 2px solid gray;

    border-radius: 8px;

    display: flex;

    flex-direction: column;

    align-items: center;

    justify-content: center;

}

#formquestions{

    position: relative;

    width: 50%;

    display: flex;

    margin-top: 10vh;

}

#formquestions > div{

    position: relative;

    width: 40%;

    margin-left: 5%;

    display: flex;

    border-bottom: 2px solid lightgray;

    /\* margin-top: 10vh; \*/

}

#formquestions label{

    position: absolute;

    bottom: 0;

    left: 0;

    transition: 0.5s;

}

input{

    height: 5vh;

    background-color: transparent;

    color: white;

    border: none;

    outline: none;

}

#formquestions  input:focus ~ label,

#formquestions  input:valid ~ label{

    transform: translateY(-5vh);

}

/\* button stling  \*/

button{

    margin-top: 5vh;

    font-size: large;

    color: rgb(33, 33, 128);

    background-color: rgb(65, 177, 185);

    height: 5vh;

    width: 20%;

    border: none;

    outline: none;

    transition: 0.5s;

}

button:active{

    background-color: aquamarine;

    color: brown;

}

3) app.py :

from flask import Flask, render\_template, request

from flask\_cors import CORS

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

CORS(app)

"""

training model with our data

"""

import numpy as np

from sklearn.linear\_model import LinearRegression

import csv

X\_train = np.array([])

Y\_train = np.array([])

with open('data.csv', 'r') as csvFile:

    csv\_reader = csv.reader(csvFile)

    for row in csv\_reader:

        X\_train = np.concatenate((X\_train, [float(row[0])]))

        Y\_train = np.concatenate((Y\_train, [float(row[1])]))

# Creating a linear regression model

model = LinearRegression()

# Training the model

model.fit(X\_train.reshape(-1, 1), Y\_train)

"""

our routes

"""

@app.route('/')

def hello\_world():

    return render\_template('index.html')

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

def calculate():

    if request.method == 'POST':

        height = request.args.get('height')

        height = int(height)

        ans = model.predict(np.array([height]).reshape(1,-1))

        # ans = ((height\*0.39) + 3)

        return str(ans)

    return "5"

if \_\_name\_\_ == "\_\_main\_\_":

    app.run(debug=True)

4)requirements.txt

-> it will be useful for 3rd person to understand and download all the dependencies

blinker==1.7.0

click==8.1.7

colorama==0.4.6

Flask==3.0.2

Flask-Cors==4.0.0

gunicorn==21.2.0

itsdangerous==2.1.2

Jinja2==3.1.3

MarkupSafe==2.1.5

packaging==24.0

Werkzeug==3.0.1

Photos of final output:

|  |
| --- |
|  |
|  |
|  |