

Physical Activity Fitness Prediction Using IBM Watson

**A project on Physical Activity Fitness Prediction
Using IBM Watson with a small web application**

By **HEMANTH DURGA RAJESH PAYYAVULA**

Physical Activity Fitness Prediction Using IBM Watson

Project Idea:

Sedentary lifestyle is defined by the absence of physical activity practices throughout the day and causes a decrease in caloric expenditure. This behavior is explained by the inappropriate lifestyle, for example, too much time sitting or lying down and still eating unhealthy foods during this time of immobilization. Currently, a third of the adult world population is physically inactive and this generates 5 million deaths per year (The Lancet, 2012). In addition to contributing to several chronic diseases, physical inactivity also influences mood, sleep quality and body weight

By the end of this project:

- You'll be able to understand the problem to classify if it is a regression or a classification kind of problem.
- You will be able to know how to pre-process / clean the data using different data pre-processing techniques.
- You will be able to analysis or get insights of data through visualization.
- Applying different algorithms according to dataset and based on visualization.
- You will be able to know how to find the accuracy of the model.
- You will be able to know how to build a web application using the Flask framework.

Prerequisites :

To complete this project, you must require following software's, concepts and packages

1. In order to develop this project, we need to install Anaconda and PyCharm

1. **Anaconda Navigator:**

Anaconda Navigator is a free and open-source distribution of the Python and R

programming languages for data science and machine learning related applications. It can be installed on Windows, Linux, and macOS. Conda is an open-source, cross-platform, package management system. Anaconda comes with so very nice tools like Jupyter Lab, Jupyter Notebook,

For this project, we will be using Jupyter notebook and Spyder

2. To install Anaconda navigator and to know how to use Jupyter Notebook & Spyder using Anaconda watch the video with this

<https://youtu.be/5mDYijMfSzs>

Follow below steps to download required packages :

- Open the anaconda prompt.
- Type “pip install pandas” and click enter.
- Type “pip install matplotlib” and click enter.
- Type “pip install numpy” and click enter.
- Type “pip install scikit-image” and click enter.
- Type “pip install scikit-learn” and click enter.
- Type “pip install Flask” and click enter.

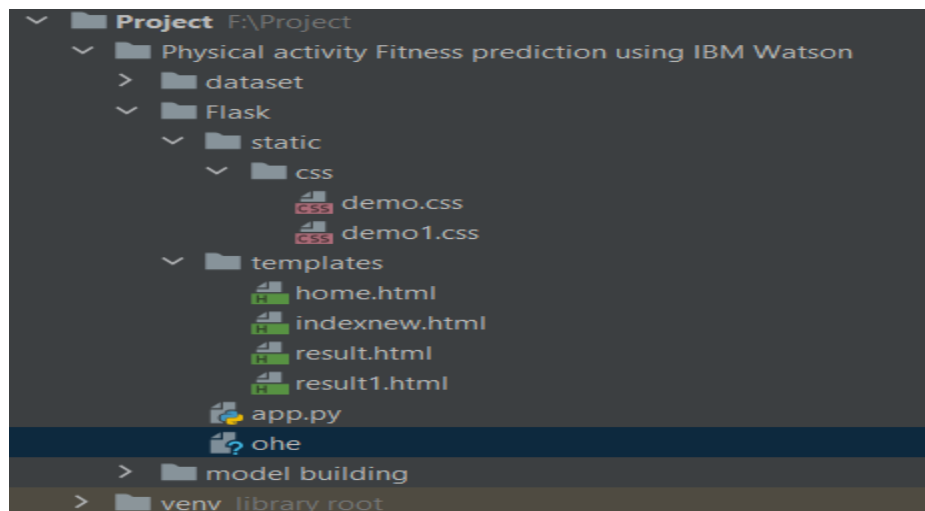
For Prior Knowledge

Watch video By

https://youtu.be/kE5QZ8G_78c

https://youtu.be/6za9_mh3uTE

https://youtu.be/lj4l_CvBnt0



folder which contains

The above image indicates the **Project Structure**

First

Dataset Collection

1. Collect The Dataset Or Create The Dataset

2. collect the dataset using this

<https://www.kaggle.com/aroojanwarkhan/fitness-data-trends#25.csv>

3. Here we are using a data set which you can find in the above link and you can download it from the reference.

Reading The Dataset

Read the Dataset using command

```
data=pd.read_csv(r" F:\Project\Physical activity Fitness prediction using IBM Watson\dataset\25.csv")
```

Or

```
Data=pd.read_csv("25.csv")
```

Exploratory Data Analysis

- To check the first five rows of the dataset, we have a function called `head()`.

```
data.head() # return you the top 5 data
```

	date	step_count	mood	calories_burned	hours_of_sleep	bool_of_active	weight_kg
0	2017-10-06	5464	200	181	5	0	66
1	2017-10-07	6041	100	197	8	0	66
2	2017-10-08	25	100	0	5	0	66
3	2017-10-09	5461	100	174	4	0	66
4	2017-10-10	6915	200	223	5	500	66

- To check the last five rows of the dataset, we have a function called `tail()`.

```
data.tail() # return you the bottom 5 data
```

	date	step_count	mood	calories_burned	hours_of_sleep	bool_of_active	weight_kg
91	2018-01-05	133	100	4	2	0	64
92	2018-01-06	153	300	0	8	0	64
93	2018-01-07	500	200	0	5	500	64
94	2018-01-08	2127	200	0	5	0	64
95	2018-01-09	2203	300	0	5	500	64

- However, you can always specify how many rows to show such as `dataset.head(8)` to show 8 rows.

```
data.head(8) # return you the top 5 data
```

	date	step_count	mood	calories_burned	hours_of_sleep	bool_of_active	weight_kg
0	2017-10-06	5464	200	181	5	0	66
1	2017-10-07	6041	100	197	8	0	66
2	2017-10-08	25	100	0	5	0	66
3	2017-10-09	5461	100	174	4	0	66
4	2017-10-10	6915	200	223	5	500	66
5	2017-10-11	4545	100	149	6	0	66
6	2017-10-12	4340	100	140	6	0	66
7	2017-10-13	1230	100	38	7	0	66

- For finding the names of the columns present in the dataset we make use of columns

```
data.columns # column names present in the dataset
```

```
Index(['date', 'step_count', 'mood', 'calories_burned', 'hours_of_sleep',  
      'bool_of_active', 'weight_kg'],  
      dtype='object')
```

Understanding Data Type and Summary of features

see how our dataset is, by using the info()

```
data.info() #information about the data
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 96 entries, 0 to 95  
Data columns (total 7 columns):  
#   Column                Non-Null Count  Dtype  
---  -  
0   date                   96 non-null    object  
1   step_count             96 non-null    int64  
2   mood                   96 non-null    int64  
3   calories_burned        96 non-null    int64  
4   hours_of_sleep         96 non-null    int64  
5   bool_of_active         96 non-null    int64  
6   weight_kg              96 non-null    int64  
dtypes: int64(6), object(1)  
memory usage: 5.4+ KB
```

Selecting the Categorical features

Creating a function to fetch the number of unique elements present in each column. The columns which are having a minimum number of unique elements or category present will be

considered as the categorical columns and the remaining columns will be a numerical column.

```
def category(data): # function
    for i in data.columns: # looping with each column data
        print(i)
        print(data[i].unique()) # finding unique data
        print("x"*90)
category(data) # calling our function
```

Label Encoding

Label Encoding is a popular encoding technique for handling categorical variables. In this technique, each label is assigned a unique integer based on alphabetical ordering.

Label Encoding the target column

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
data['bool_of_active']=le.fit_transform(data['bool_of_active'])
data
```

•

Train And Test The Model Using Decision Tree

There are several Machine learning algorithms to be used depending on the data you are going to process such as images, sound, text, and numerical values. The algorithms that you can choose according to the objective that you might have may be Classification algorithms are Regression algorithms.

Example: 1. Linear Regression.

2. Logistic Regression.

3. Random Forest Regression / Classification.

4. Decision Tree Regression / Classification.

Now we apply the Decision Tree algorithm on our dataset.

A decision tree is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource

costs, and utility. It is one way to display an algorithm that only contains conditional control statements.

Build the model with the DecisionTreeClassifier.

We're going to use `x_train` and `y_train` obtained above in `train_test_split` section to train our decision tree classifier model. We're using the `fit` method and passing the parameters as shown below.

Predict the values

Once the model is trained, it's ready to make predictions. We can use the **predict** method on the model and pass **x_test** as a parameter to get the output as **dt_y_train**.

Notice that the prediction output is an array of real numbers corresponding to the input array.

```
dt_y_train=dtc.predict([[ '0.0', '0.0', '1.0', '4435.0', '141.0', '5.0', '64.0']])
dt_y_train

array([1], dtype=int64)
```

Model Evaluation

Finally, we need to check to see how well our model is performing on the test data.

```
dtc.score(x_train,y_train)

0.9883720930232558
```

Save the Model

```
import pickle
pickle.dump(dtc, open('fitness.pkl','wb'))
```

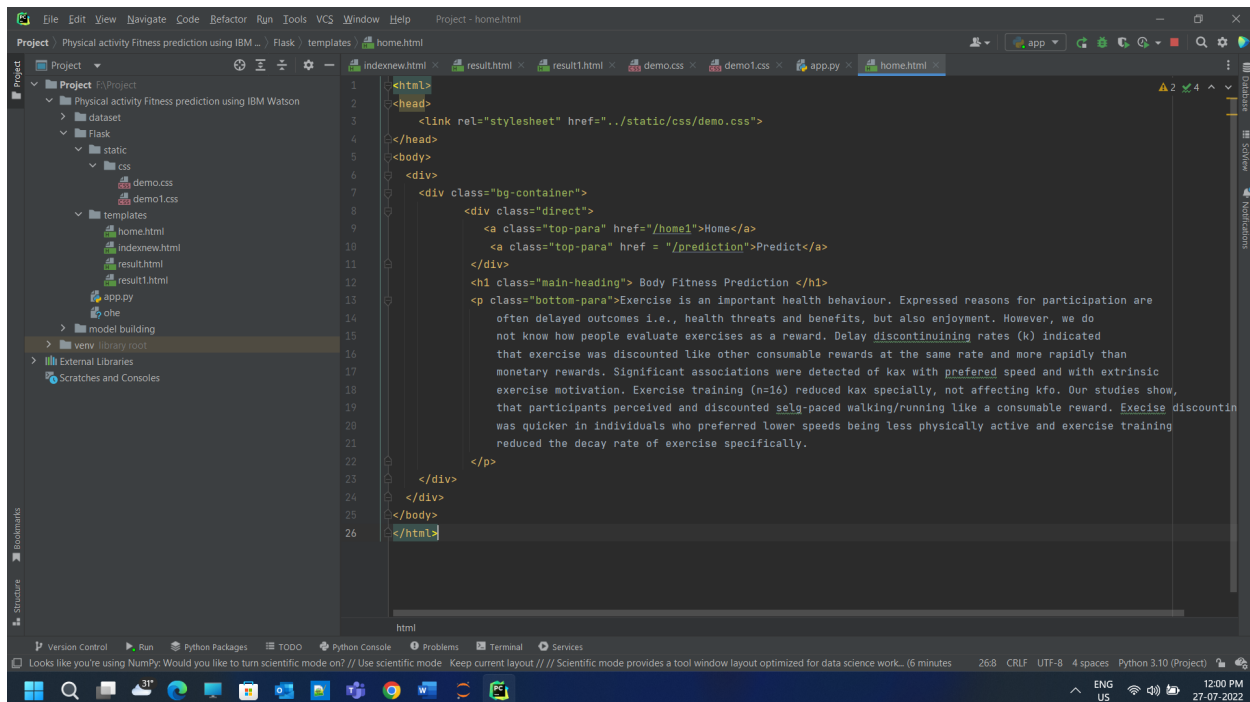
Here, **dtc** is our decision tree classifier class with saving as **fitness.pkl** file. **Wb** is the write binary in bytes.

Application Building

Create An HTML File

- We use HTML to create the front end part of the web page.
- Here, we created 2 html pages- index.html, web.html.
- index.html displays the home page.
- web.html accepts the values from the input and displays the prediction.
- For more information regarding [HTML](#)

We also use [JavaScript-main.js](#) and [CSS-main.css](#) to enhance our functionality and view of HTML pages



```
<html>
<head>
  <link rel="stylesheet" href="../static/css/demo.css">
</head>
<body>
  <div>
    <div class="bg-container">
      <div class="direct">
        <a class="top-para" href="/home1">Home</a>
        <a class="top-para" href="/prediction">Predict</a>
      </div>
      <h1 class="main-heading">Body Fitness Prediction </h1>
      <p class="bottom-para">Exercise is an important health behaviour. Expressed reasons for participation are often delayed outcomes i.e., health threats and benefits, but also enjoyment. However, we do not know how people evaluate exercises as a reward. Delay discontinuing rates (k) indicated that exercise was discounted like other consumable rewards at the same rate and more rapidly than monetary rewards. Significant associations were detected of kax with preferred speed and with extrinsic exercise motivation. Exercise training (n=16) reduced kax specially, not affecting kfo. Our studies show, that participants perceived and discounted self-paced walking/running like a consumable reward. Exercise discounting was quicker in individuals who preferred lower speeds being less physically active and exercise training reduced the decay rate of exercise specifically.
      </p>
    </div>
  </div>
</body>
</html>
```

Above is the home.html page

Like this create indexnew.html(for Prediction)

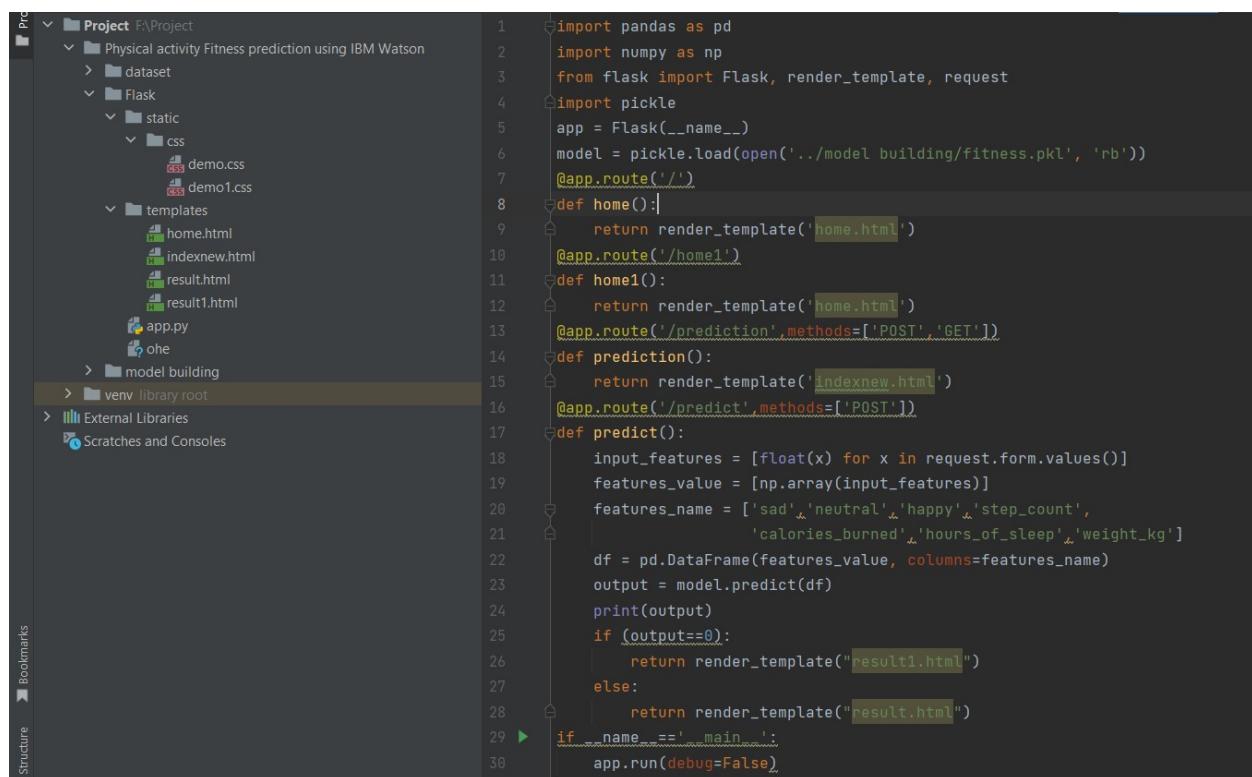
result.html(you are Active...)

result.html(you are Lazy.....)

Next create python Flask file(app.py) for web framework

Build Python Code

- Let us build a flask file 'app.py' which is a web framework written in python for server-side scripting. Let's see the step by step procedure for building the backend application.
- App starts running when the "__name__" constructor is called in main.
- render_template is used to return an html file.
- "GET" method is used to take input from the user.
- "POST" method is used to display the output to the user.



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with folders like 'Physical activity Fitness prediction using IBM Watson', 'dataset', 'static', 'css', 'templates', 'model building', and 'venv'. The code editor shows the following Python code:

```
1 import pandas as pd
2 import numpy as np
3 from flask import Flask, render_template, request
4 import pickle
5 app = Flask(__name__)
6 model = pickle.load(open('../model building/fitness.pkl', 'rb'))
7 @app.route('/')
8 def home():
9     return render_template('home.html')
10 @app.route('/home1')
11 def home1():
12     return render_template('home.html')
13 @app.route('/prediction', methods=['POST', 'GET'])
14 def prediction():
15     return render_template('indexnew.html')
16 @app.route('/predict', methods=['POST'])
17 def predict():
18     input_features = [float(x) for x in request.form.values()]
19     features_value = np.array(input_features)
20     features_name = ['sad', 'neutral', 'happy', 'step_count',
21                     'calories_burned', 'hours_of_sleep', 'weight_kg']
22     df = pd.DataFrame(features_value, columns=features_name)
23     output = model.predict(df)
24     print(output)
25     if (output==0):
26         return render_template("result1.html")
27     else:
28         return render_template("result.html")
29 if __name__ == '__main__':
30     app.run(debug=False)
```

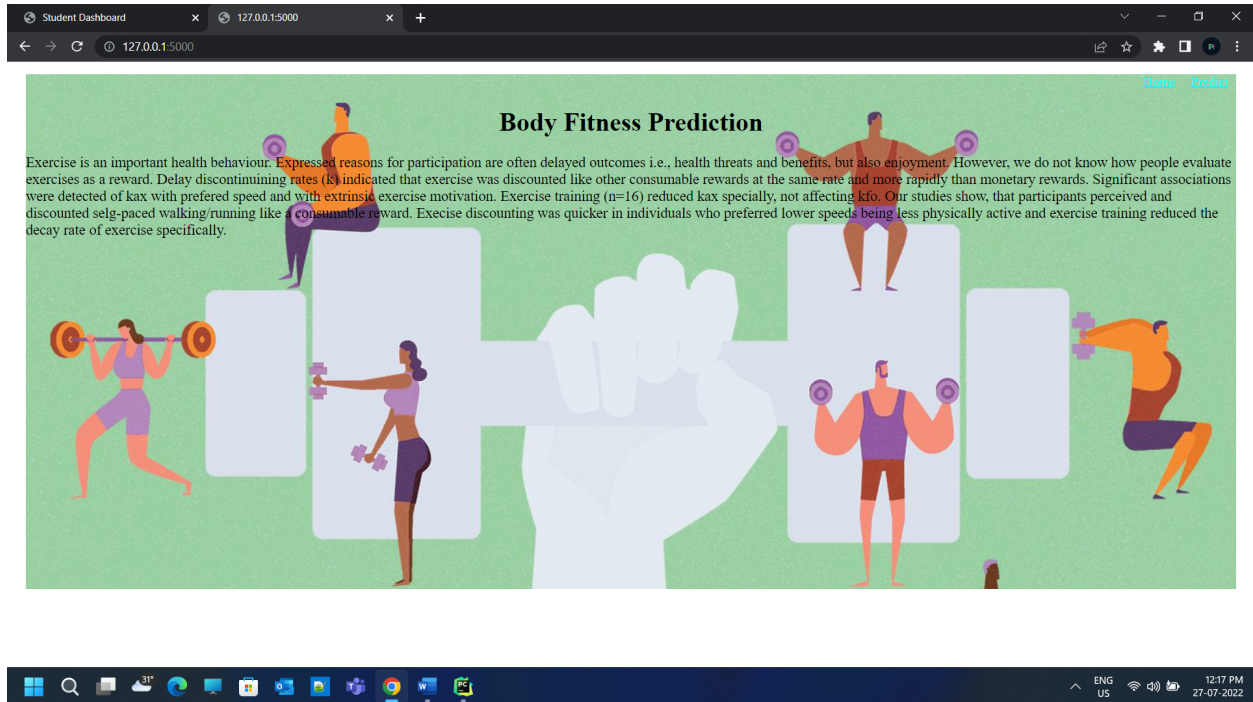
Here we are running it on localhost:5000

<http://127.0.0.1:5000>

Run The App

- Open anaconda prompt from the start menu
- Navigate to the folder where your python script is.
- Now type “python app.py” command
- Navigate to the localhost where you can view your web page

- Showcasing The UI



This is the prediction page where we get to choose the input from our local system and predict the output.

Student Dashboard x 127.0.0.1:5000/prediction x +

127.0.0.1:5000/prediction

Body Fitness Prediction

A machine learning Web App, Built with Flask

Your mood sad | yes ▾

Your mood neutral | yes ▾

Your mood happy | yes ▾

Enter the step counts :

Enter the calories_burned :

Enter the hours of sleep :

Enter your weight in kg :

Windows taskbar: 12:17 PM 27-07-2022

Finally, the prediction for the given input features is shown.


Student Dashboard x 127.0.0.1:5000/predict x +

127.0.0.1:5000/predict

Body Fitness Prediction

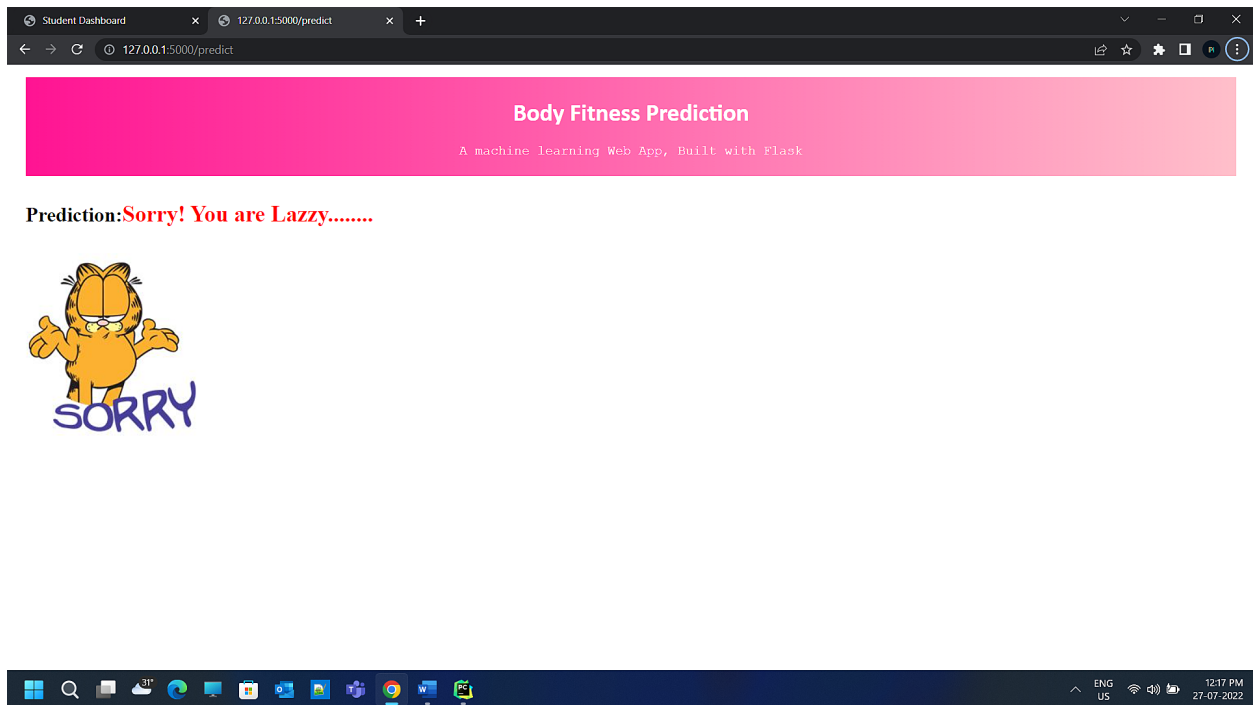
A machine learning Web App, Built with Flask

Prediction:Great! You are ACTIVE.....



Finally, the prediction for the given input features is shown.

Windows taskbar: 12:18 PM 27-07-2022



Finally

Train The Model On IBM

In this milestone, you will learn how to build a Machine Learning Model and deploy it on the IBM Cloud.

Train The ML Model On IBM

Train ML Model with IBM Notebook

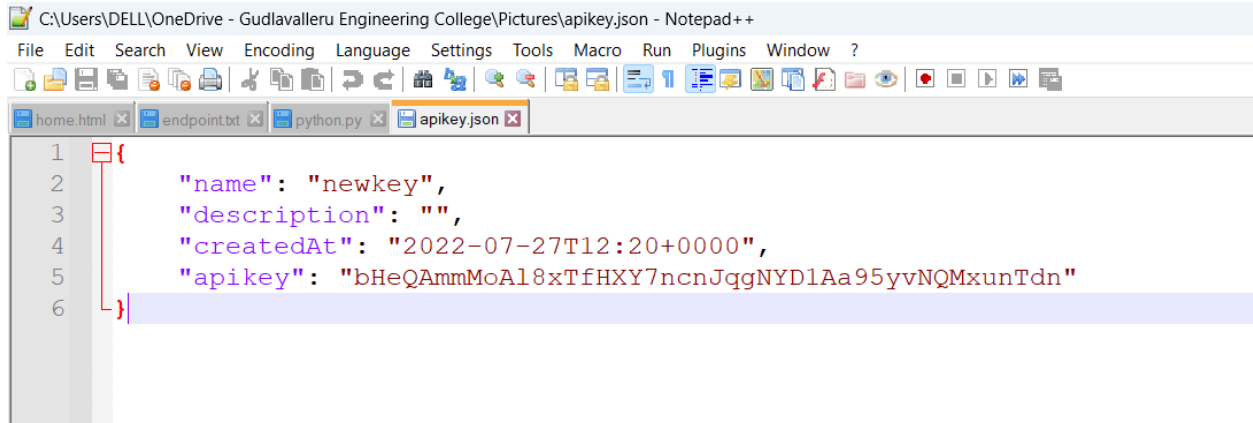
while Train ML Model with IBM Notebook we should use the (.ipynb) file in the in IBM Jupyter Notebook and upload the our own dataset

And we should install IBM watson machine Learning packing our notebook and it with our model to create IBM model.

APIKEY

APIkey is key use to link your IBM cloud account to your ibmapp.py to use the cloud dataset

APIkey is in the JSON format and to look like this



```
C:\Users\DELL\OneDrive - Gudlavalleru Engineering College\Pictures\apikey.json - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
home.html endpoint.txt python.py apikey.json
1 {
2   "name": "newkey",
3   "description": "",
4   "createdAt": "2022-07-27T12:20+0000",
5   "apikey": "bHeQAmmMoAl8xTfHXY7ncnJqgNYD1Aa95yvNQMxunTdn"
6 }
```

IBM MODEL FILE

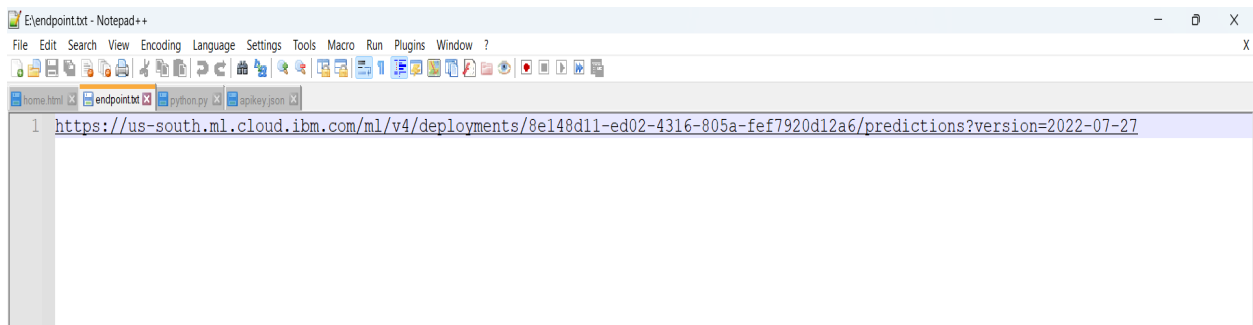
After

Using the this APIkey and IBM Machine Learning should create the IBM Model which is used to store in the IBM cloud and depolyed

DEPOLYMENT

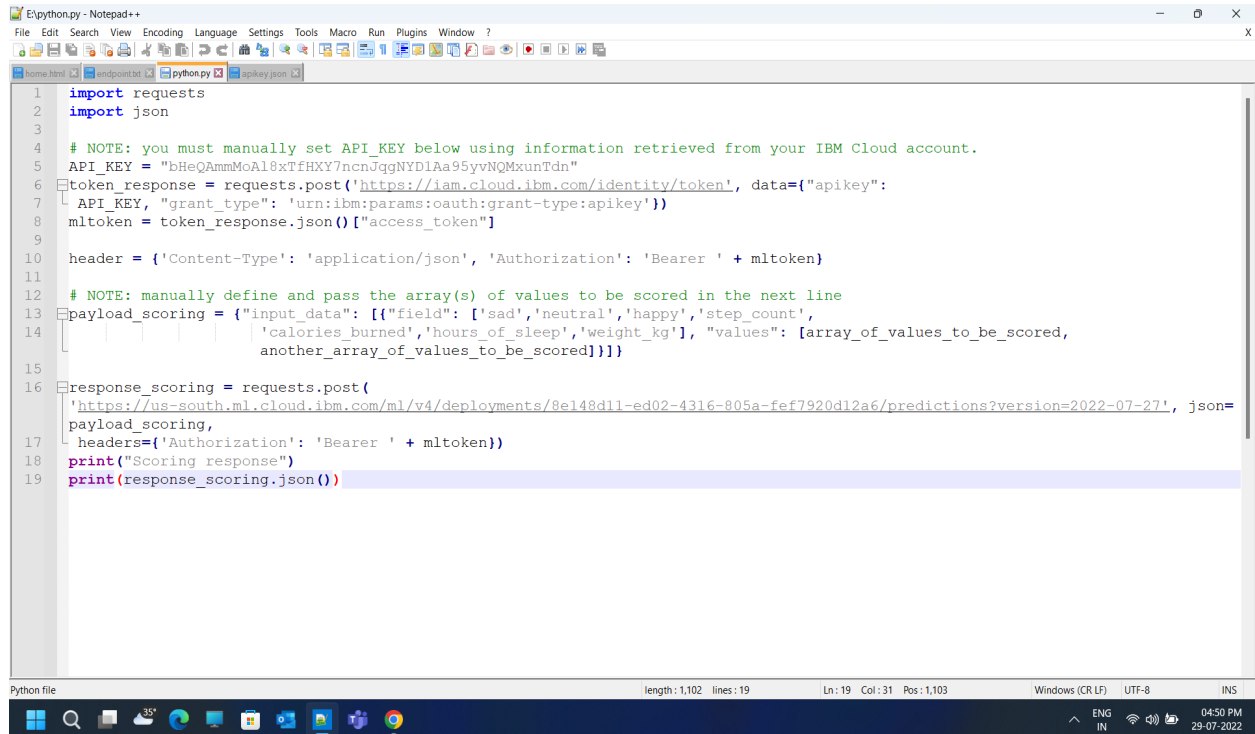
Depolyment is use in model for cloud storge of the IBM .Depolyment is done in the online method to get endpoint and python socring code

ENDPOINT



```
E:\endpoint.txt - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
home.html endpoint.txt python.py apikey.json
1 https://us-south.ml.cloud.ibm.com/ml/v4/deployments/8e148d11-ed02-4316-805a-fef7920d12a6/predictions?version=2022-07-27
```

PYTHON CODE OF IBM



```
1 import requests
2 import json
3
4 # NOTE: you must manually set API_KEY below using information retrieved from your IBM Cloud account.
5 API_KEY = "bHeQAmM0a18xTfHXY7ncnJggNYD1Aa95yvNQMxunTdn"
6 token_response = requests.post('https://iam.cloud.ibm.com/identity/token', data={"apikey":
7 API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type:apikey'})
8 mltoken = token_response.json()["access_token"]
9
10 header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}
11
12 # NOTE: manually define and pass the array(s) of values to be scored in the next line
13 payload_scoring = {"input_data": [{"field": ['sad','neutral','happy','step_count',
14 'calories_burned','hours_of_sleep','weight_kg'], "values": [array_of_values_to_be_scored,
15 another_array_of_values_to_be_scored]}]}
16 response_scoring = requests.post(
17 'https://us-south.ml.cloud.ibm.com/ml/v4/deployments/8e148d11-ed02-4316-805a-fef7920d12a6/predictions?version=2022-07-27', json=
18 payload_scoring,
19 headers={'Authorization': 'Bearer ' + mltoken})
20 print("Scoring response")
21 print(response_scoring.json())
```

AFTER DEPLOYMENT

we should integrate this code with our app.py code to access the IBM Model file

For the Reference
I used this videos

FOR DEPLOYMENT

<https://youtu.be/TysuP3KgSzc>

FOR INTEGRATE

<https://youtu.be/ST1ZYLmYw2U>

By this we successfully complete our project

Thank you