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

The objective of the project is answering a simple question, “does exercise/working-out improve a person’s activeness?”. For the scope of this project a person’s activeness was the measure of their daily step-count (the number of steps they take in a day). We are going to build a Machine Learning model which predicts the activeness or inactiveness of a person based on the Mood and number of steps taken in a day. Mood was measured in either "Happy", "Neutral" or "Sad" which were given numeric values of 300, 200 and 100 respectively. Feeling of activeness was measured in either "Active" or "Inactive" which were given numeric values of 500 and 0 respectively.

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 PyChram

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/lj4I_CvBnt0

Tasks:

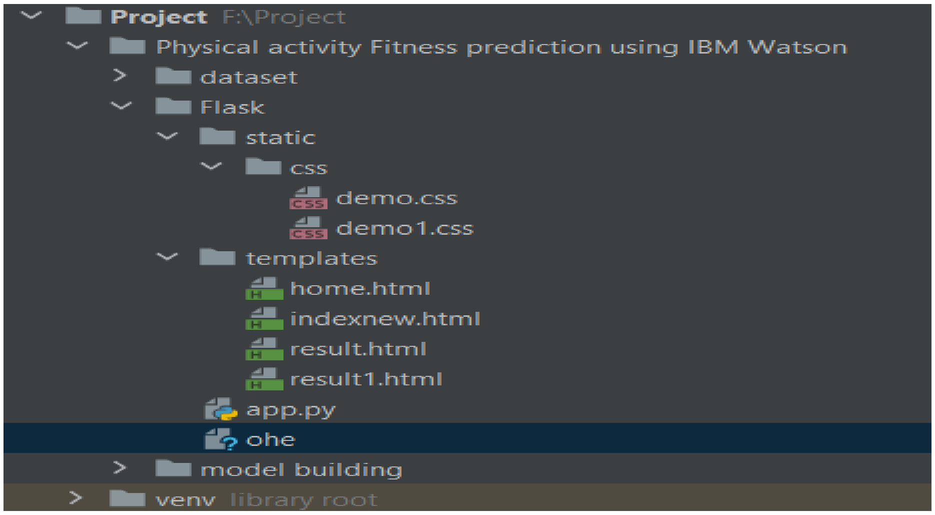
1. Data Collection.

   a.Collect the dataset or Create the dataset

1. Data Pre- processing.
   a. Import the Libraries.
   b. Importing the dataset.
   c. Exploratory Data Analysis
   d. Checking for Null Values.
   e. Data Visualization.
   f. Splitting Data into Train and Test.
2. Model Building
   a. Training and testing the model
   b. Evaluation of Model
3. Application Building
   a. Create an HTML file
   b. Build a Python Code.

Project Folder

We have three folders dataset, Flask and Model Building
A python file called app.py for server side scripting.
We need the model which is saved and the saved model in this content is (**fitness.pkl**).
Templates folder which contains home.html, result.html and resultnew.html files.
Static folder which contains css styles.css



```
Project F:\Project
  Physical activity Fitness prediction using IBM Watson
    dataset
    Flask
      static
        css
          demo.css
          demo1.css
      templates
        home.html
        indexnew.html
        result.html
        result1.html
      app.py
      ohe
    model building
  venv library root
```

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.

Second

Data Pre-Processing

Import libraries

1.Pandas

2.Numpy

3.Sklearn

4.Joblib

5.Pickle

6.Flask

7 Matplotlib and Seaborn

8. Train test split

9. Counter

Reading The Dataset

Read the Dataset using command

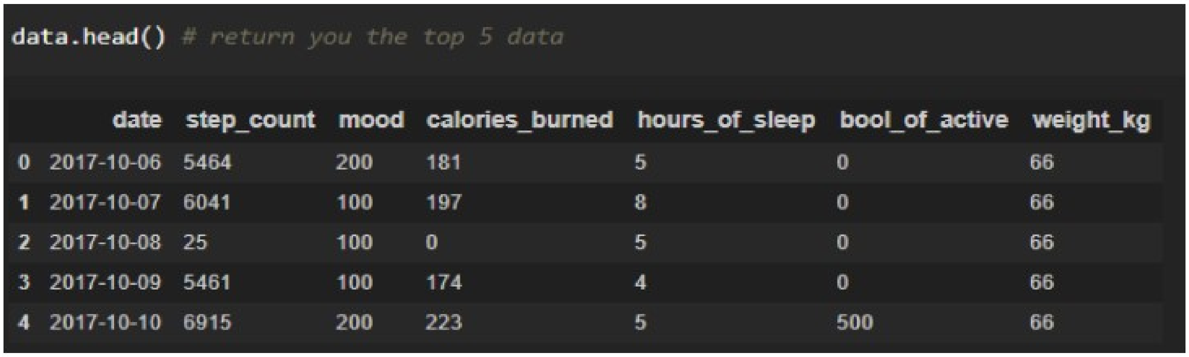
data=pd.read_csv(r” F:\Project\Physical activity Fitness prediction using IBM Watson\dataset\25.csv”)

Or

Data=pd.read_csv(“25.csv”)

Exploratory Data Analysis

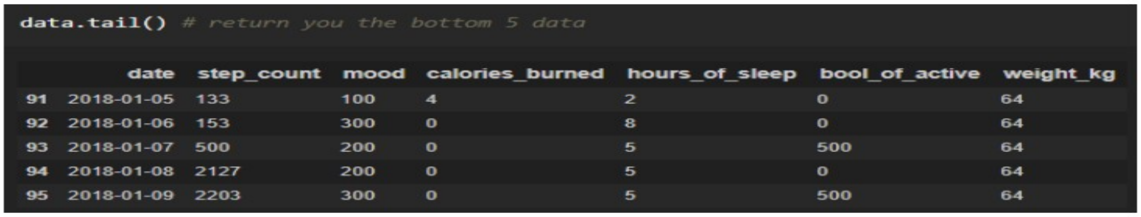
- To check the first five rows of the dataset, we have a function called head( ).



```
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

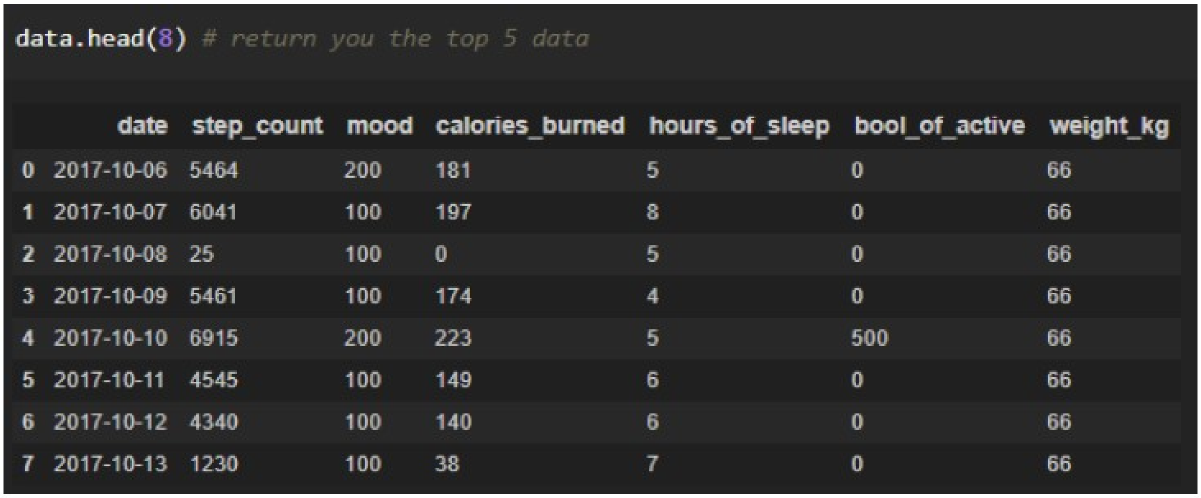
- To check the last five rows of the dataset, we have a function called tail().



```
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

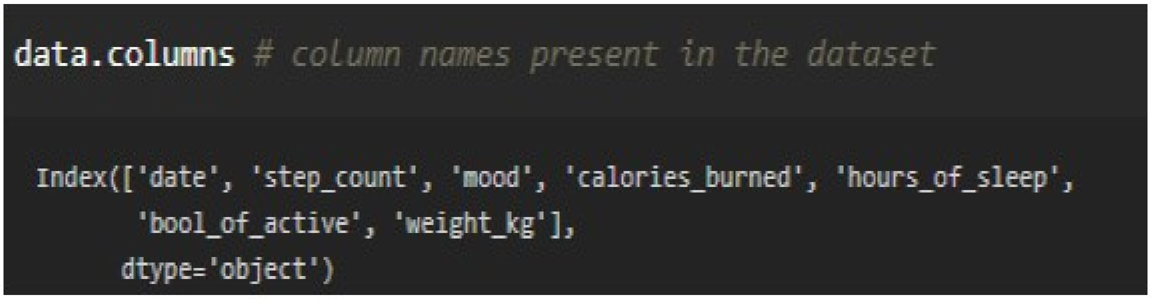
- However, you can always specify how many rows to show such as dataset.head(8) to show 8 rows.



```
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

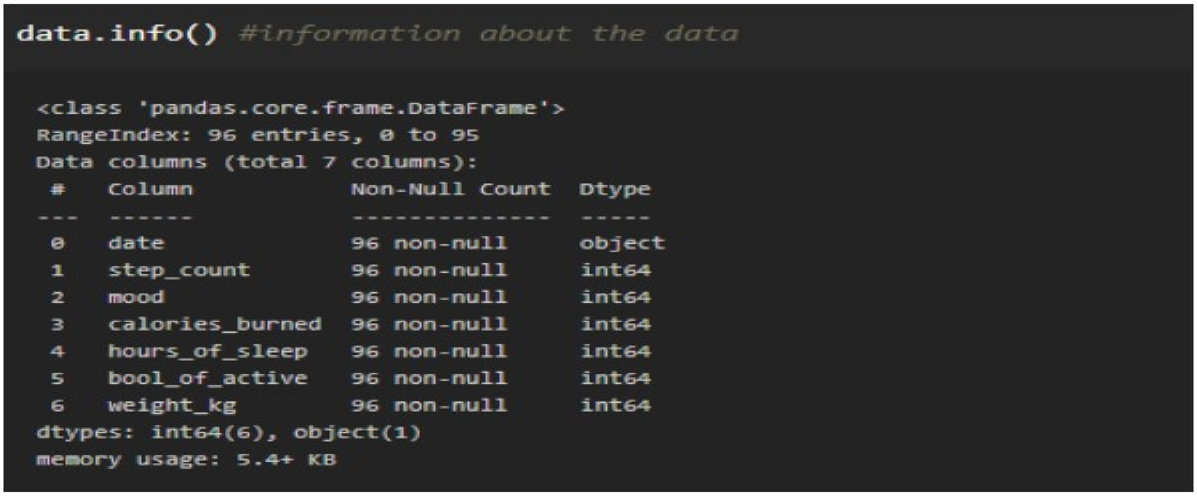


```
data.columns # column names present in the dataset

 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()

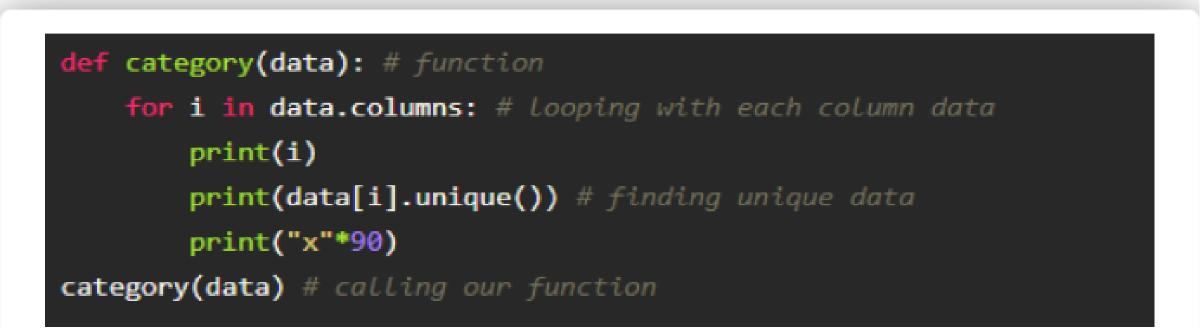


```
data.info() #information about the data

 <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

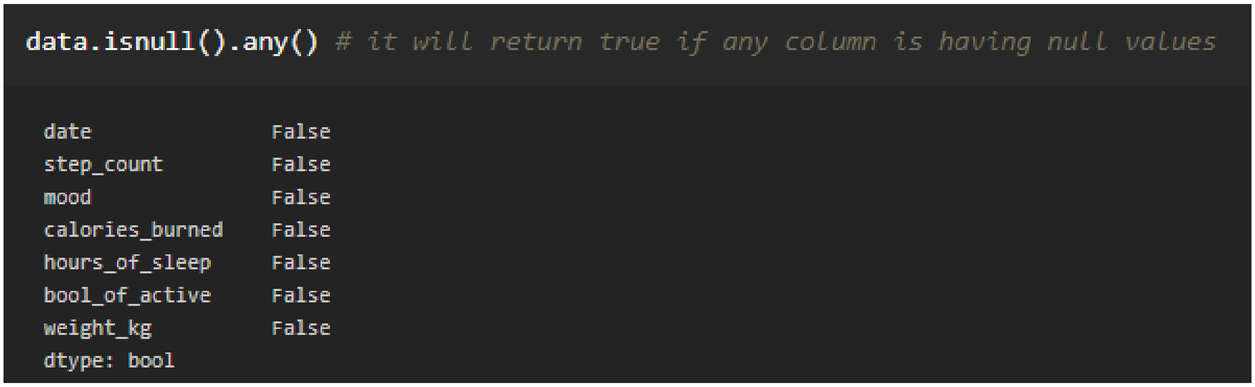
Creating a function to fetch the number of unique elements present in each column. The columns which are having a minimum number of unique elements or category present will be considered as the categorical columns and the remaining columns will be a numerical column.



```
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
```

Checking For Null Values

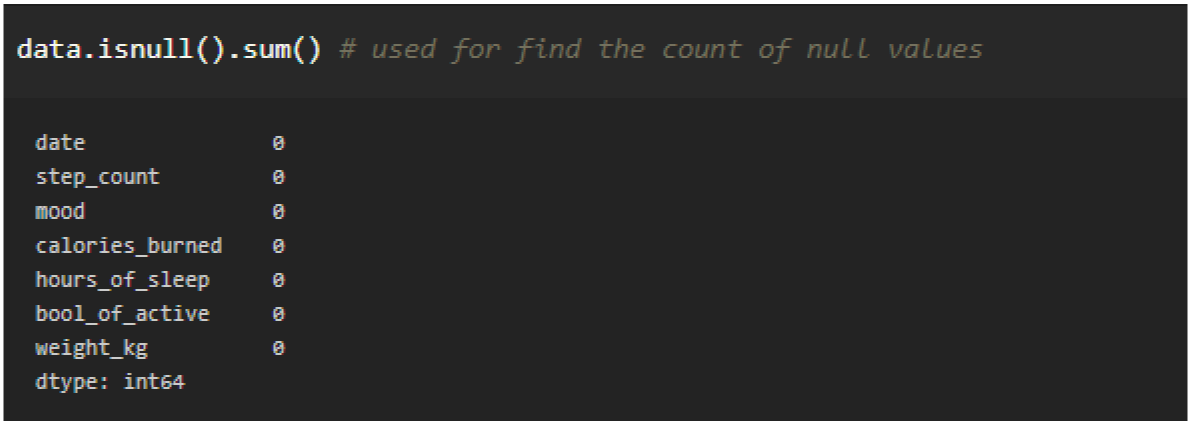
- We will be using **isnull().any()** method to see which column has missing values.
- We get this as result



```
data.isnull().any() # it will return true if any column is having null values
```

```
date               False
step_count         False
mood               False
calories_burned    False
hours_of_sleep     False
bool_of_active     False
weight_kg          False
dtype: bool
```

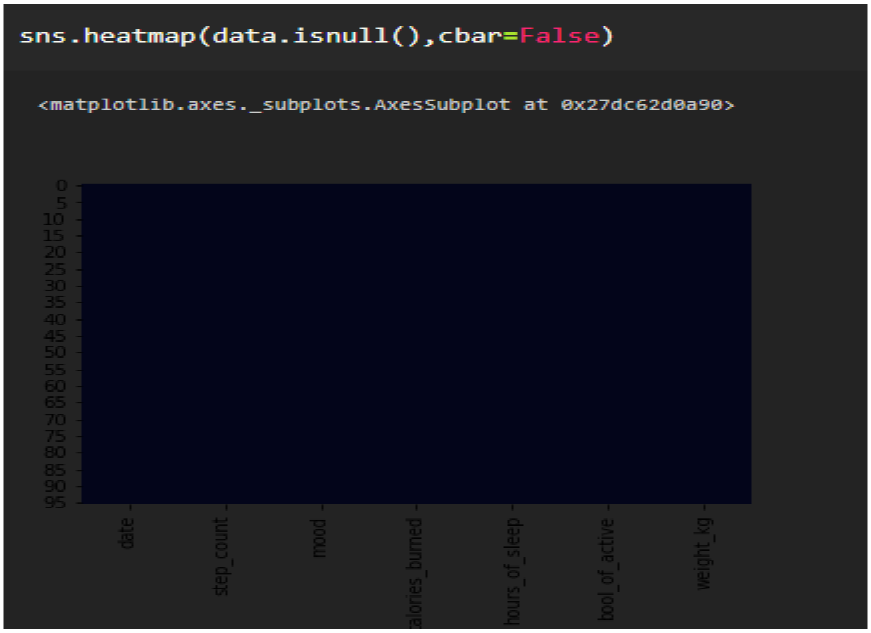
- Word “True” that the particular column has missing values, we can also see the count of null values in each column by using **isnull().sum()**.



```
data.isnull().sum() # used for find the count of null values
```

```
date               0
step_count         0
mood               0
calories_burned    0
hours_of_sleep     0
bool_of_active     0
weight_kg          0
dtype: int64
```

- As our data is not having any null values so we don’t have to handle the null values but by using the seaborn heatmap libraries we can observe the null values graphically.

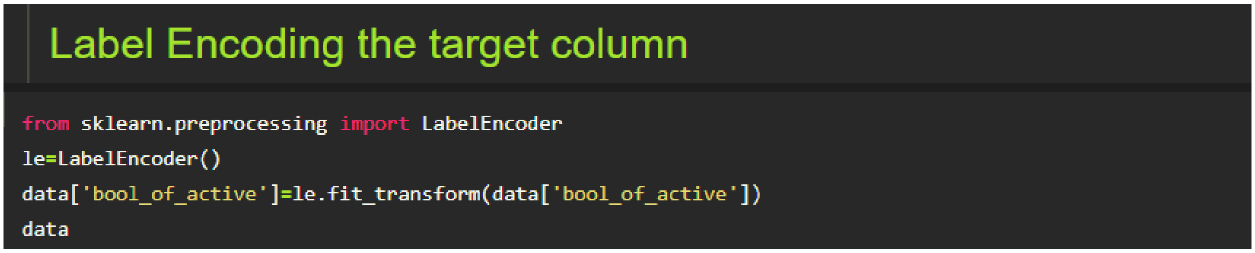


```
sns.heatmap(data.isnull(),cbar=False)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x27dc62d0a90>
```

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
```

Data Visualization

- For

  step_count
  calories_burned
  weights_kg
  hours_of_sleep

Finding correlation between the independent Columns using :

- finding the correlation between variables we have corr() available.

  corr=data.corr()
  sns.heatmap(corr,xticklabels=corr.columns,yticklabels=corr.columns)

- Plotting a pair plot to showcase the relationship among all the different columns

  sns.set_style("whitegrid")
  sns.pairplot(data)

Bivariate Analysis

- step count vs mood
- Step_count vs Activeness

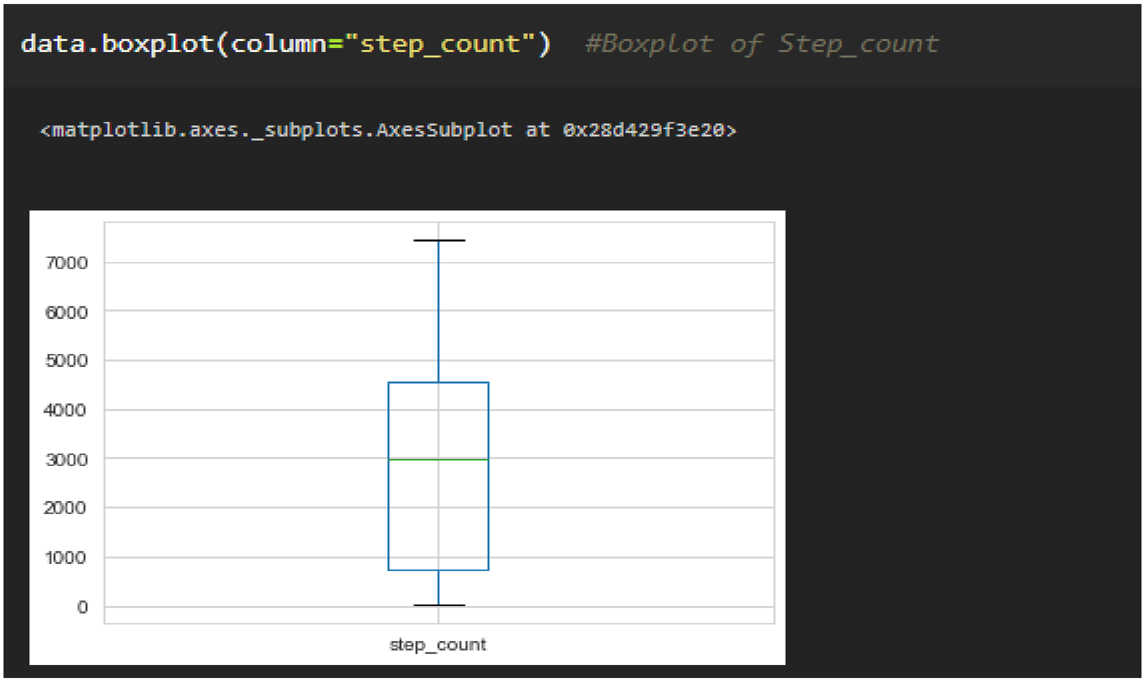
- Mood vs Calories_burned

- Calories_burned vs Activeness
- Sleeping_hours vs mood
- Sleeping_hours vs Activeness
- Mood vs Activeness

- **Outliers Detection using boxplot**

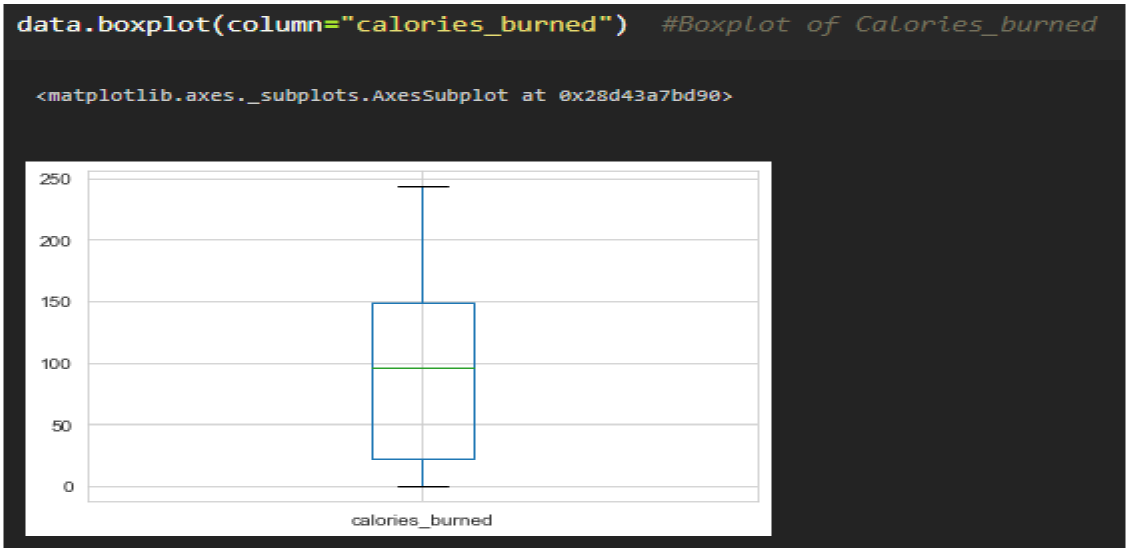
- Outliers are observations in a dataset that don't fit in some way or you can say those values are of no use and also may effect our results.

- So with the help of boxplot we can visualize and check whether the data contains any outliers or not



```
data.boxplot(column="step_count")  #Boxplot of Step_count
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x28d429f3e20>
```

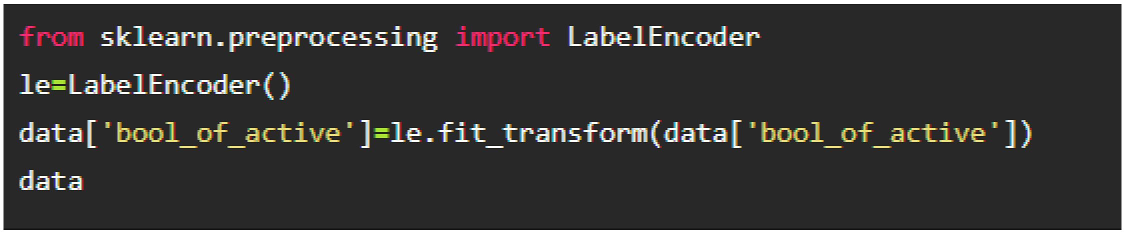


```
data.boxplot(column="calories_burned")  #Boxplot of Calories_burned
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x28d43a7bd90>
```

Label Encoding of target data

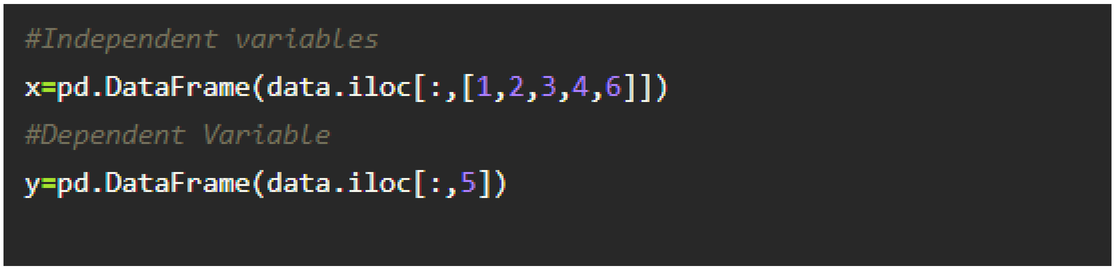
As in our target data we have two classes i,e. 0(Inactive) , 500 (active) we will be using a label encoding technique to encode it to 0 and 1 for better understanding of the classes



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

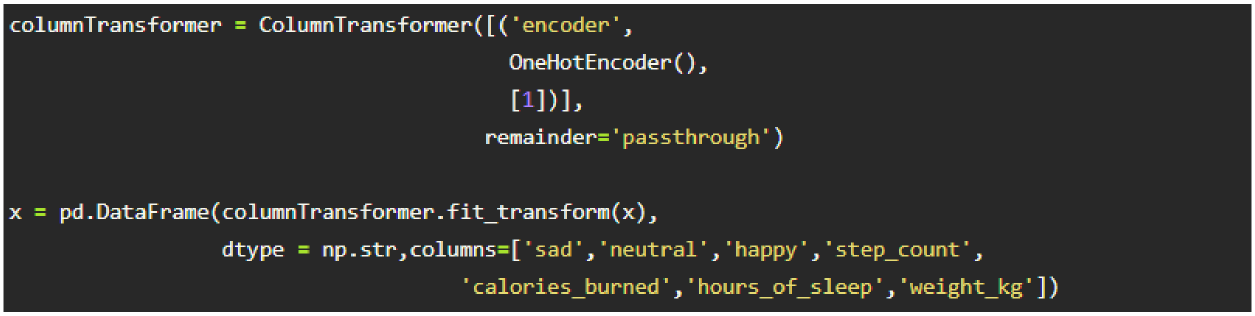
Splitting The Dataset Into Dependent And Independent Variables

Let’s create out independent and dependent variables:



```
#Independent variables
x=pd.DataFrame(data.iloc[:,[1,2,3,4,6]])
#Dependent Variable
y=pd.DataFrame(data.iloc[:,5])
```

Column Transformer is a new approach where we apply both labels and one hot encoding in a single step.



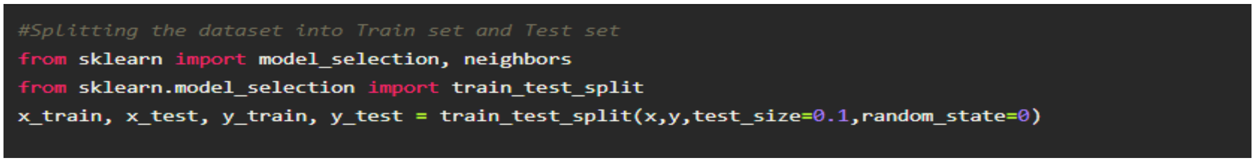
```
columnTransformer = ColumnTransformer([('encoder',
                                        OneHotEncoder(),
                                        [1])],
                                      remainder='passthrough')

x = pd.DataFrame(columnTransformer.fit_transform(x),
                 dtype = np.str,columns=['sad','neutral','happy','step_count',
                                    'calories_burned','hours_of_sleep','weight_kg'])
```

Split The Dataset Into Train Set And Test Set

The train-test split is a technique for evaluating the performance of a machine learning algorithm.

- **Train Dataset**: Used to fit the machine learning model.
- **Test Dataset**: Used to evaluate the fit machine learning model.



```
#Splitting the dataset into Train set and Test set
from sklearn import model_selection, neighbors
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.1,random_state=0)
```

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.

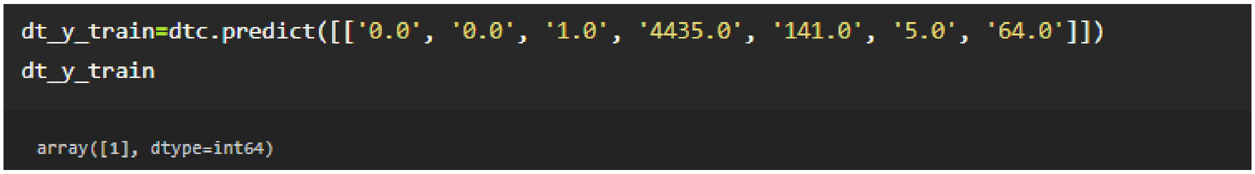
Build the model with the DecisionTreeClassifier.

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.

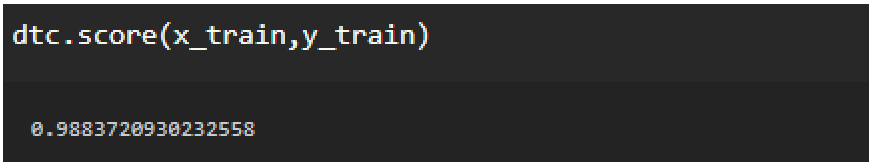


```
dt_y_train=dtc.predict([['0.0', '0.0', '1.0', '4435.0', '141.0', '5.0', '64.0']])
dt_y_train

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

Model Evaluation

Finally, we need to check to see how well our model is performing on the test data. There are many evaluation techniques. For this, we evaluate scores produced by the model.

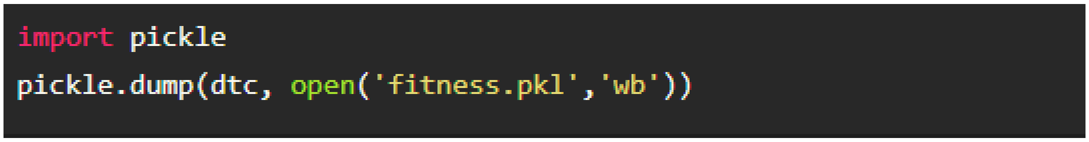


```
dtc.score(x_train,y_train)
```

```
0.9883720930232558
```

Save the Model

Pickle is used for serializing and de-serializing Python object structures, also called marshalling or flattening. Serialization refers to the process of converting an object in memory to a byte stream that can be stored on disk or sent over a network. Later on, this character stream can then be retrieved and de-serializedback to a Python object.Save our model by importing pickle file



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

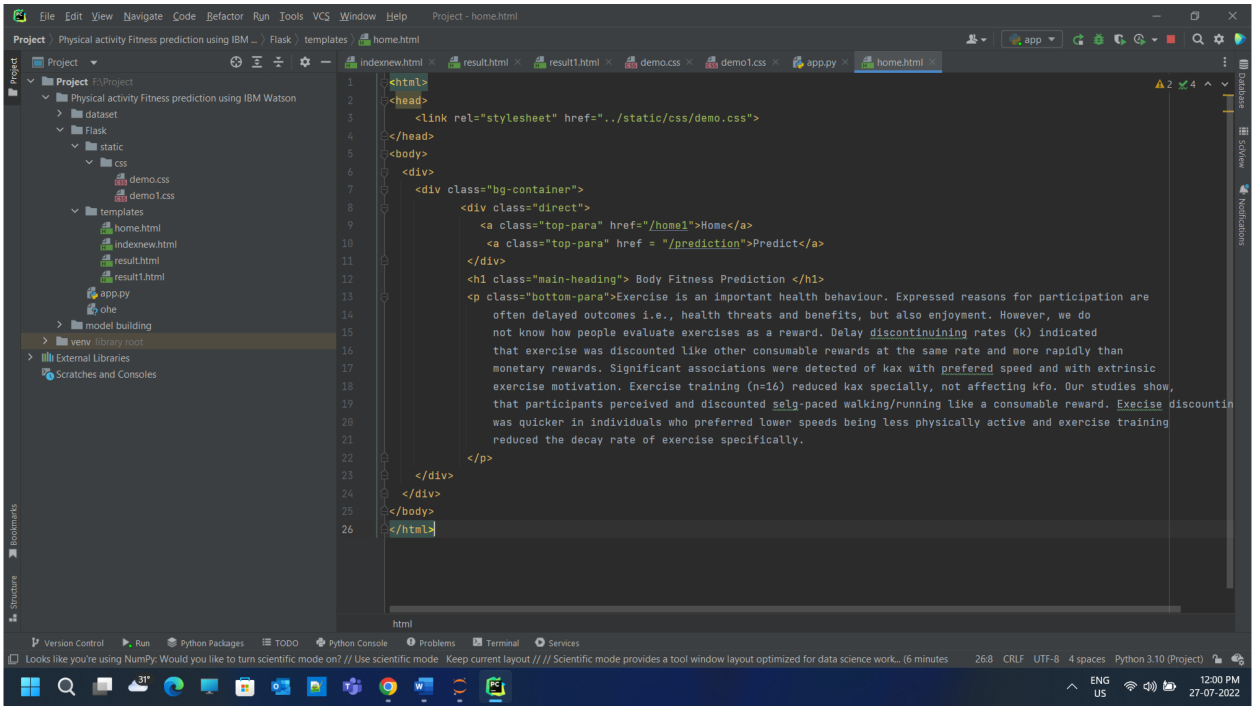
Here, **dtc** is our decision tree classifier class with saving as **fitness.pkl** file. **Wb** is the write binary in bytes.

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 discontinuining 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 prefered speed and with extrinsic
                exercise motivation. Exercise training (n=16) reduced kax specially, not affecting kfo. Our studies show,
                that participants perceived and discounted selg-paced walking/running like a consumable reward. Execise discountin
                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>
```

Looks like you're using NumPy: Would you like to turn scientific mode on? // Use scientific mode Keep current layout // // Scientific mode provides a tool window layout optimized for data science work... (6 minutes

Above is the home.html page

Like this create indexnew.html(for Prediction)

result.html(you are Active....)

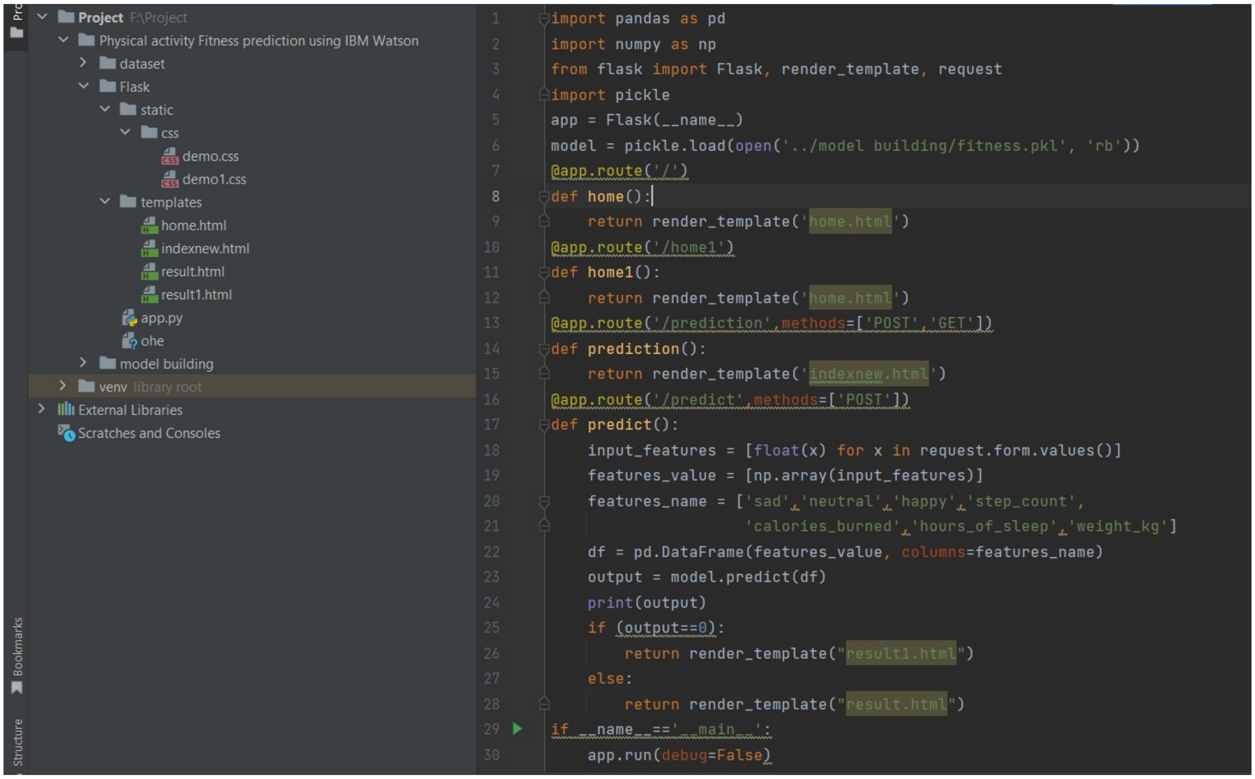
result.html(you are Lazzy.....)

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.



```python
import pandas as pd
import numpy as np
from flask import Flask, render_template, request
import pickle
app = Flask(__name__)
model = pickle.load(open('../model building/fitness.pkl', 'rb'))
@app.route('/')
def home():
    return render_template('home.html')
@app.route('/home1')
def home1():
    return render_template('home.html')
@app.route('/prediction',methods=['POST','GET'])
def prediction():
    return render_template('indexnew.html')
@app.route('/predict',methods=['POST'])
def predict():
    input_features = [float(x) for x in request.form.values()]
    features_value = [np.array(input_features)]
    features_name = ['sad','neutral','happy','step_count',
                     'calories_burned','hours_of_sleep','weight_kg']
    df = pd.DataFrame(features_value, columns=features_name)
    output = model.predict(df)
    print(output)
    if (output==0):
        return render_template("result1.html")
    else:
        return render_template("result.html")
if __name__=='__main__':
    app.run(debug=False)
```

Here we are running it on localhost:5000

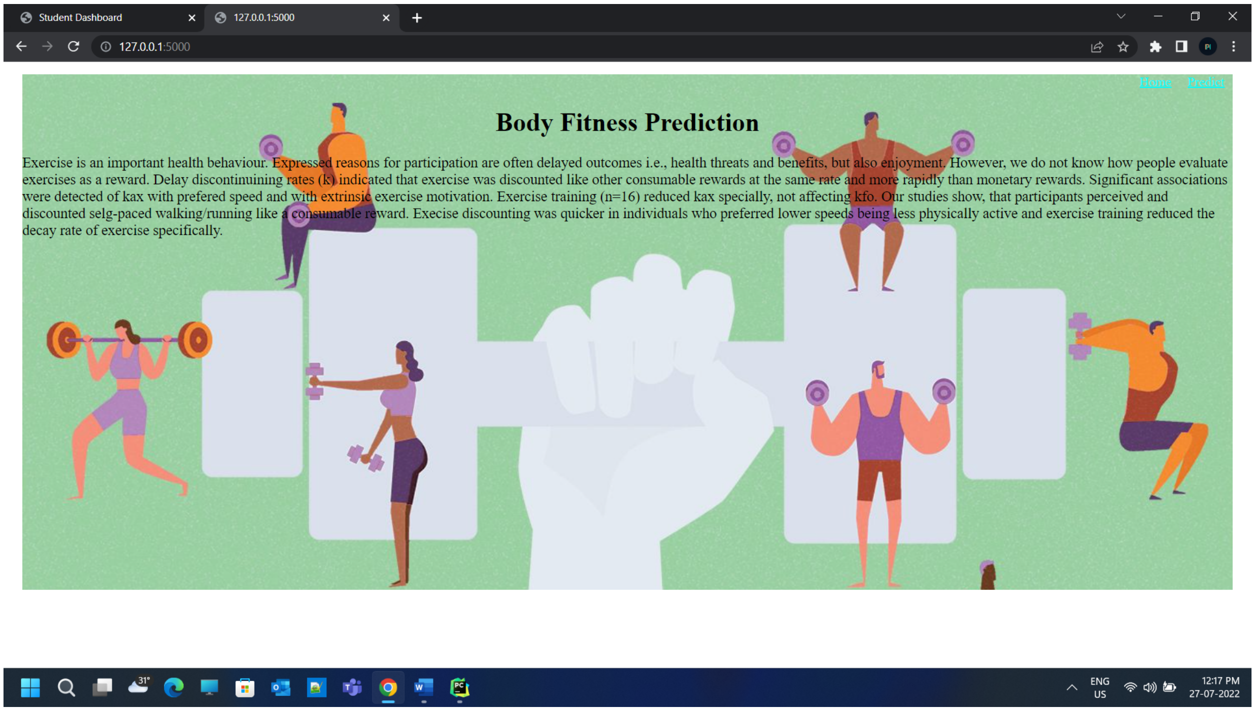
http://127.0.0.1:5000

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

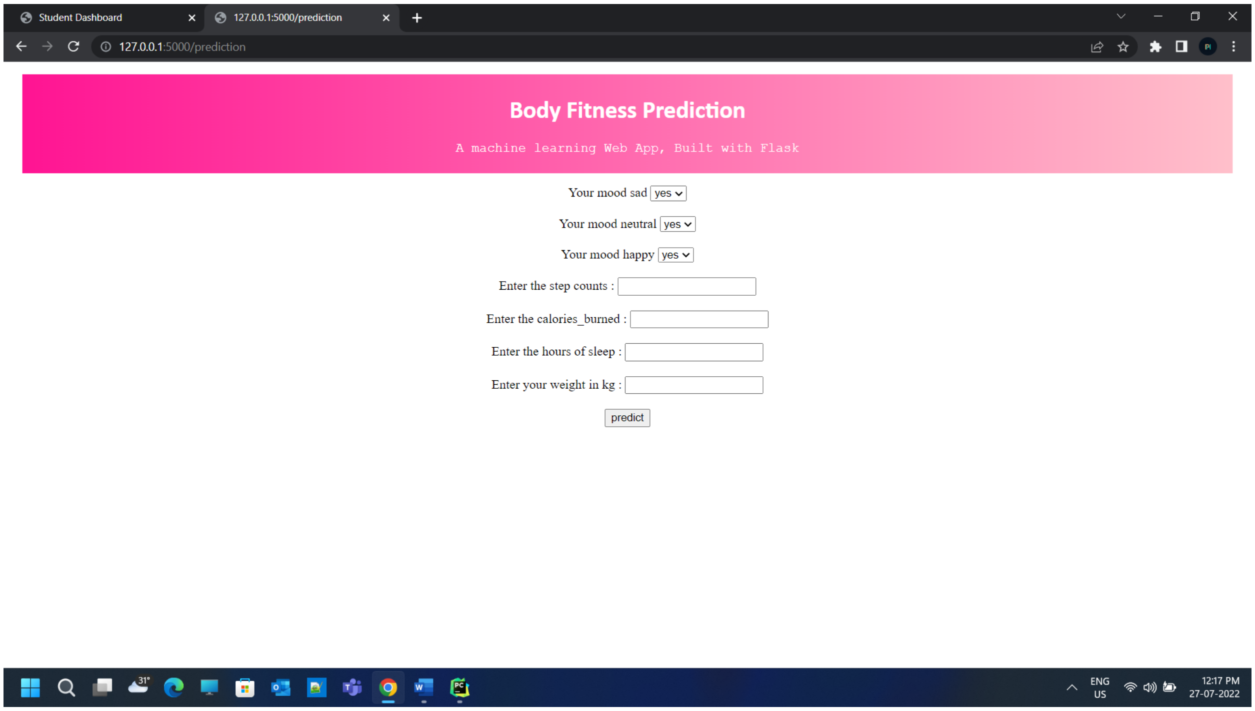
- Showcasing The UI



Body Fitness Prediction

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 discontinuining 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 prefered speed and with extrinsic exercise motivation. Exercise training (n=16) reduced kax specially, not affecting kfo. Our studies show, that participants perceived and discounted selg-paced walking/running like a consumable reward. Execise discounting was quicker in individuals who preferred lower speeds being less physically active and exercise training reduced the decay rate of exercise specifically.

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



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 :

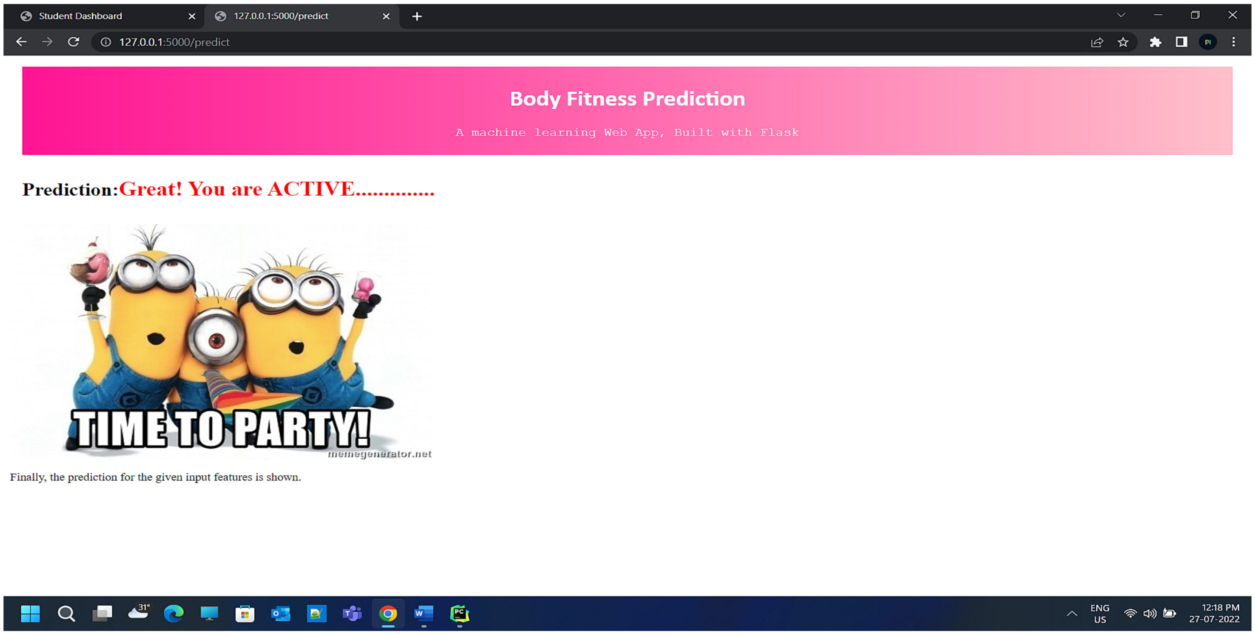
Enter the calories_burned :

Enter the hours of sleep :

Enter your weight in kg :

predict

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

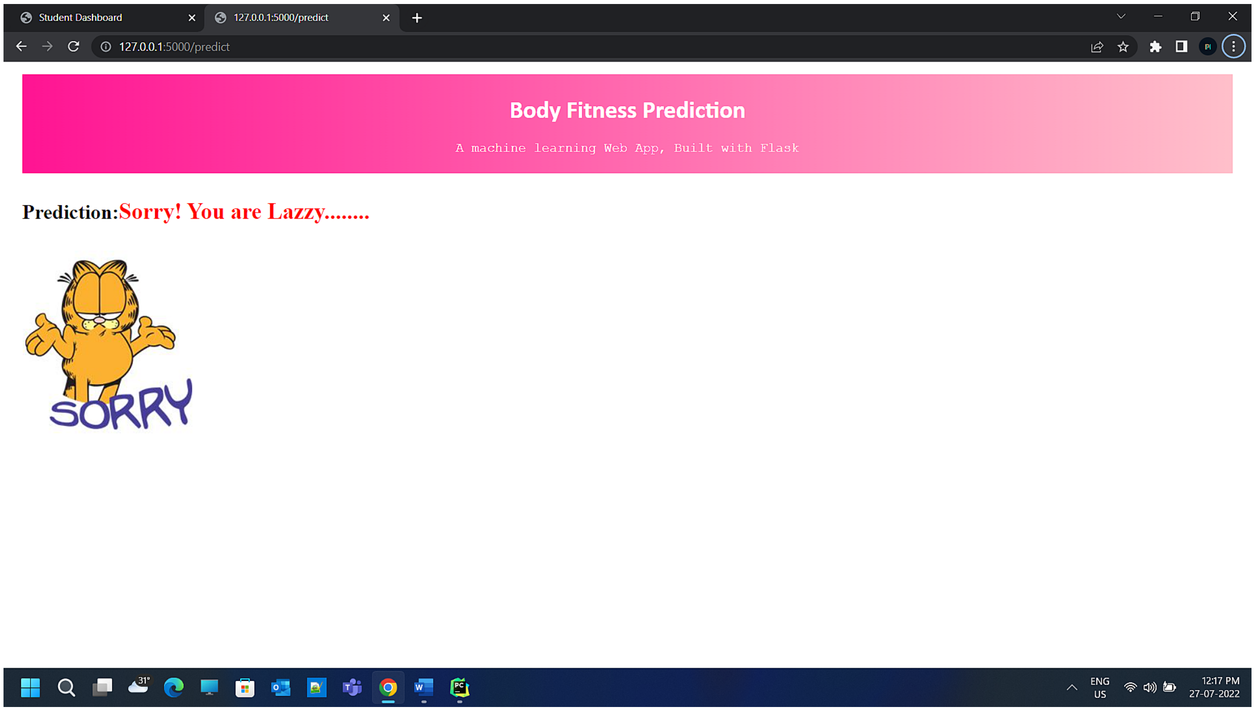


Body Fitness Prediction

A machine learning Web App, Built with Flask

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

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



Body Fitness Prediction

A machine learning Web App, Built with Flask

Prediction:Sorry! You are Lazzy........

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

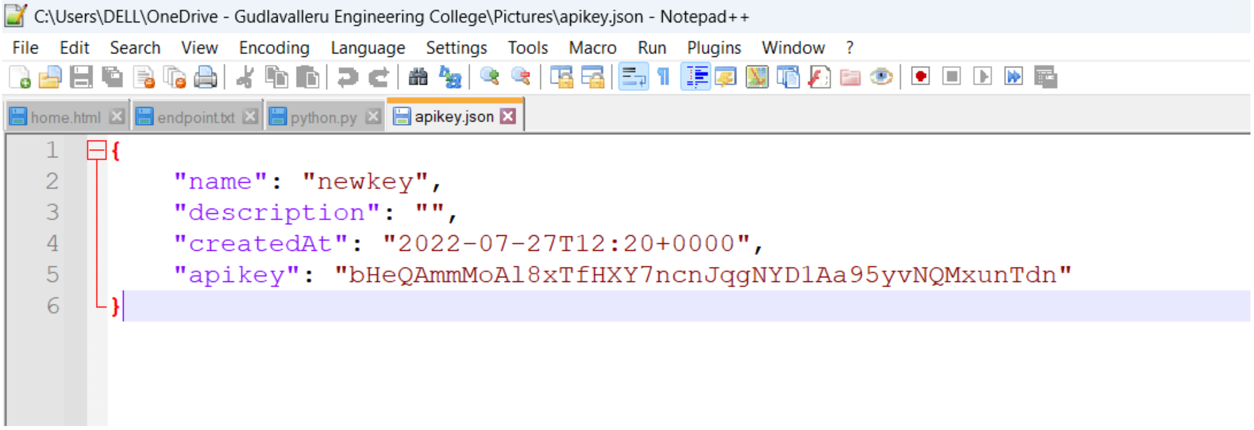
while Train ML Model with IBM Notebook we should use the (.ipynb) file in the in IBM Jpyter Notebook and upload the our own dataset

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

```
{
        "name": "newkey",
        "description": "",
        "createdAt": "2022-07-27T12:20+0000",
        "apikey": "bHeQAmmMoAl8xTfHXY7ncnJqgNYD1Aa95yvNQMxunTdn"
}
```

IBM MODEL FILE

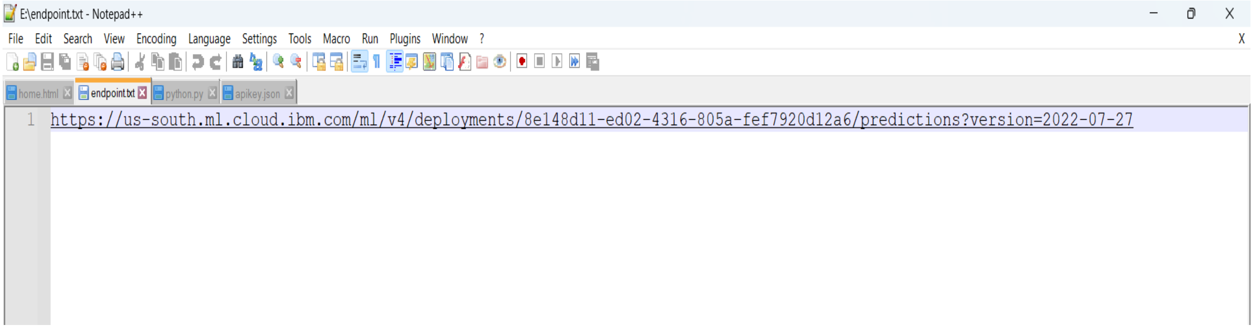
After

Using the this APIkey and IBM Machine Learning should creaate 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

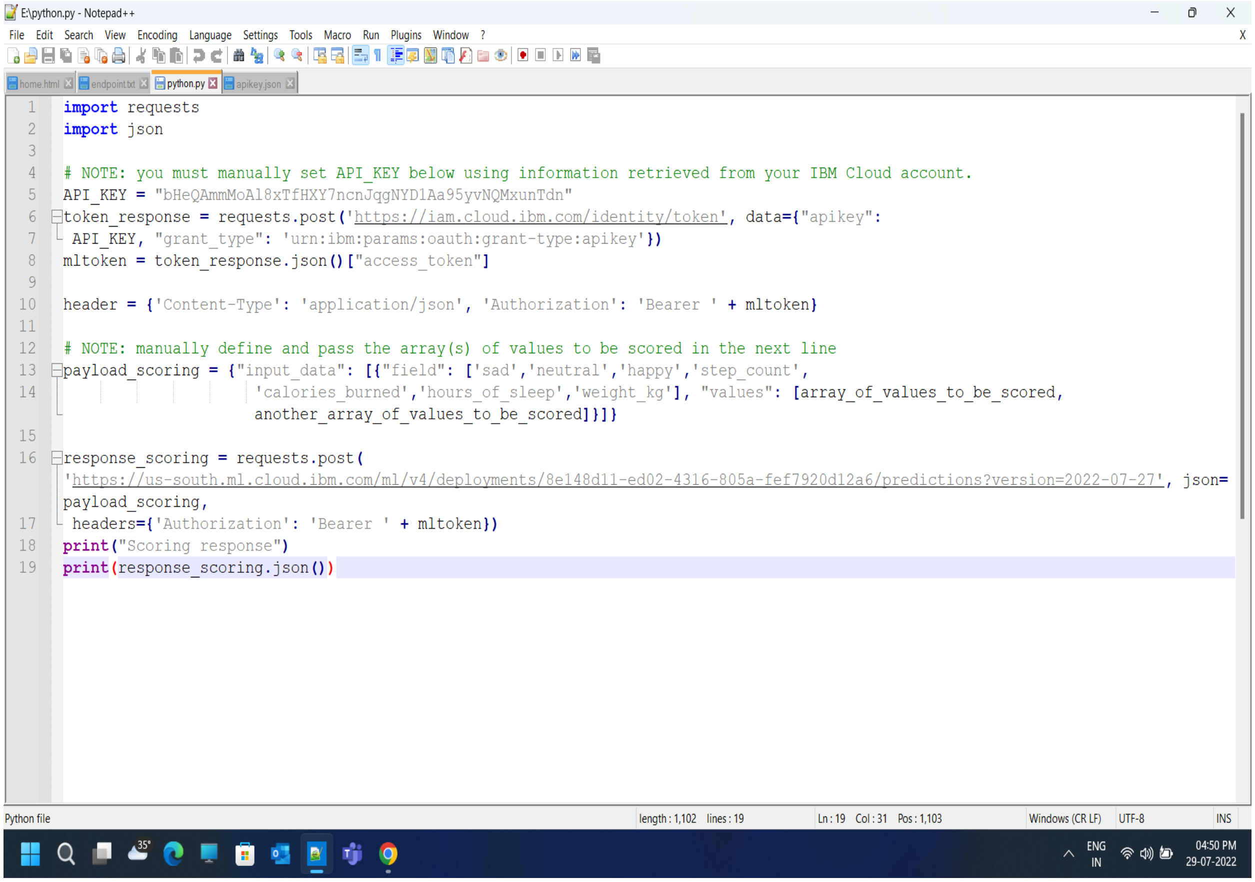
ENDPOINT



E:\endpoint.txt - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
home.html
endpoint.txt
python.py
apikey.json

```
https://us-south.ml.cloud.ibm.com/ml/v4/deployments/8e148d11-ed02-4316-805a-fef7920d12a6/predictions?version=2022-07-27
```

PYTHON CODE OF IBM



```python
import requests
import json

# NOTE: you must manually set API_KEY below using information retrieved from your IBM Cloud account.
API_KEY = "bHeQAmmMoAl8xTfHXY7ncnJqgNYDlAa95yvNQMxunTdn"
token_response = requests.post('https://iam.cloud.ibm.com/identity/token', data={"apikey":
 API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type:apikey'})
mltoken = token_response.json()["access_token"]

header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}

# NOTE: manually define and pass the array(s) of values to be scored in the next line
payload_scoring = {"input_data": [{"field": ['sad','neutral','happy','step_count',
                      'calories_burned','hours_of_sleep','weight_kg'], "values": [array_of_values_to_be_scored,
                      another_array_of_values_to_be_scored]}]}

response_scoring = requests.post(
'https://us-south.ml.cloud.ibm.com/ml/v4/deployments/8e148d11-ed02-4316-805a-fef7920d12a6/predictions?version=2022-07-27', json=
payload_scoring,
 headers={'Authorization': 'Bearer ' + mltoken})
print("Scoring response")
print(response_scoring.json())
```

E:\python.py - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
home.html
endpoint.txt
python.py
apikey.json
Python file
length : 1,102 lines : 19
Ln : 19 Col : 31 Pos : 1,103
Windows (CR LF)
UTF-8
INS
35°
ENG IN
04:50 PM
29-07-2022

AFTER DEPOLYMENT

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

For the Reference
I used this videos

FOR DEPOLYMENT
https://youtu.be/TysuP3KgSzc

FOR INTEGRATE
https://youtu.be/ST1ZYLmYw2U

By this we successfully complete our project

Thank you