**BODY FITNESS PREDICTON**

**Index**

1.Abstract

Introduction

Objective of Research

2.Problem Statement

3.Data collection

3.1.Data pre-processing

4.Model Building

5.Application Building

6. Findings and Suggestions

7. Conclusion

8. Reference

**Abstract**

Predicting the performance level of body activeness is an important and interesting problem. The main goal of the current study is to predict the performance of the body during exercising.

**Introduction**

The understanding of the term "physical fitness" was determined for a randomly selected sample (n = 94) of a population using a self-administered mailed questionnaire. Subjects were asked to state and give a reason for their perceived level of physical fitness, to state their perceived performance level in a number of physical fitness tests (muscular strength, daily physical work capacity, fatness, level of regular physical exercise, exercise speed, and body flexibility), and to rate how well these tests measure physical fitness. The reason most frequently stated for perceived level of physical fitness was the level of habitual physical activity (43%); significantly less frequently (P less than 0.01-0.0001) cited were reasons related to health (23%), physical performance (12%), and obesity (3%). The variation in perceived level of physical fitness was best explained by the variation in imagined regular exercise and fatness (r2 = 0.66, P less than 0.0001) with no significant additional contribution from imagined performance in remaining fitness tests. The measurement of regular exercise was most favored as a test of physical fitness. These results, taken together with evidence of the physical and psychological health benefits of regular exercise, imply that the most appropriate measure of physical fitness for the average person is an assessment of the habitual physical activity level.

**Objective of Research**

The given data set is about Prediction of Bool of active. The given data is to analyzed. Various objectives of the given data set are as follows:

a) To study the given data

b) To apply data cleaning methods to remove unknown data from the

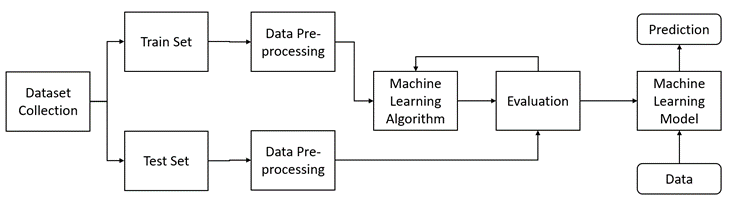
given data set.

c) Test the designed model’s working

d) Draw conclusions from the developed model

e) Predict whether the bool of active is satisfied or not

**Machine Learning Workflow:**



**Problem Statement**

It is with no doubt that regular exercise and physical activity and physical activity contribute to body fitness. some of the well-known benefits of exercise are controls weight, combats health issues, improves emotional stability, promotes better sleep.

**Data Collection**

The given data set is related to Body Fitness Prediction. It was taken from the website **kaggle.com**. The website provides various datasets from various domains.

**Data Preprocessing**

Importing required Libraries:

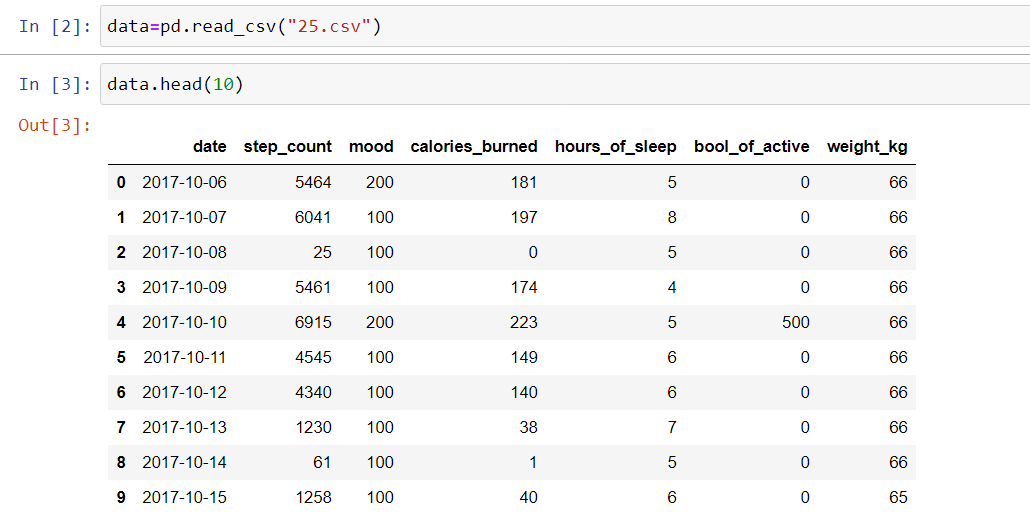


**Pandas:** It is a python library mainly used for data manipulation.

**NumPy:** This python library is used for numerical analysis.

**Matplotlib and Seaborn:** Both are the data visualization library used for plotting graph which will help us for understanding the data.

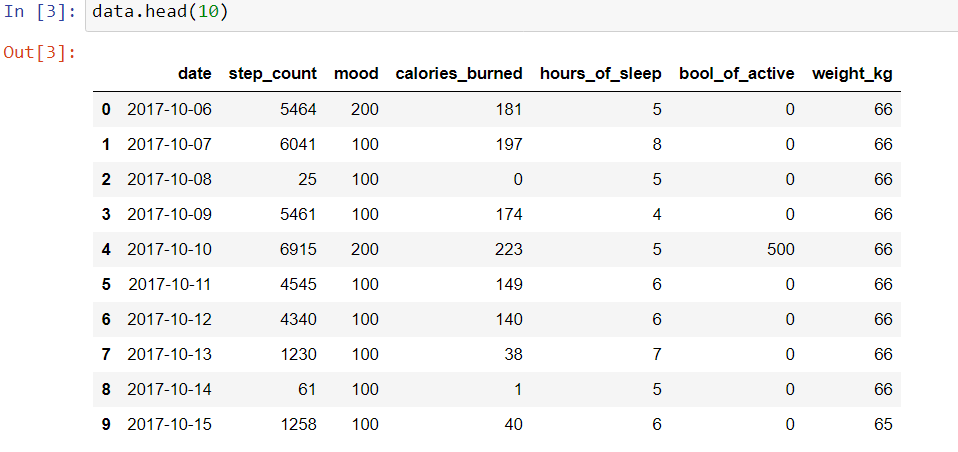
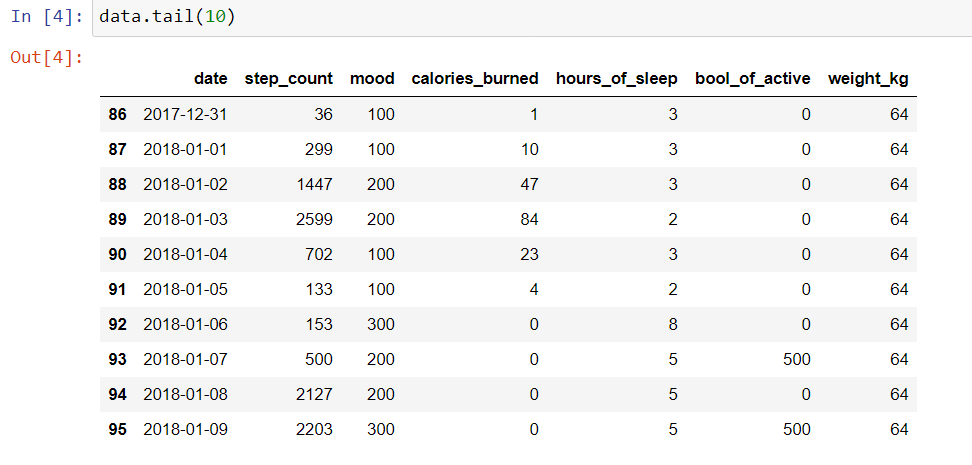
**Importing the dataset**

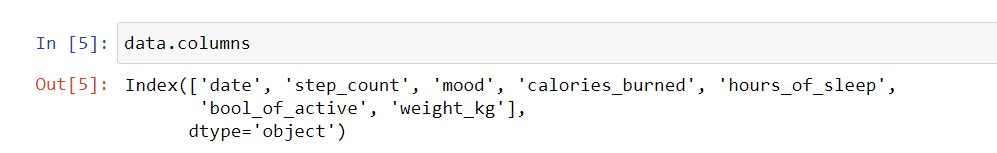


* You might have your data in .csv files. excel files or .tsv files or something else. But the goal is the same in all cases. If you want to analyse that data using pandas, the first step will be to read it into a data structure that’s compatible with pandas.
* Let’s load a .csv data file into pandas. There is a function for it, called **read\_csv().** We will need to locate the directory of the CSV file at first (it’s more efficient to keep the dataset in the same directory as your program).
* Path names on Windows tend to have backslashes in them. But we want them to mean actual backslashes, not special characters.

**Data Visualization**

Exploratory data analysis is an approach to analysing data sets to summarize their main characteristics, often with visual methods and used for determine how best to manipulate data sources to get the answers you need, making it easier for data scientists to discover patterns, spot anomalies, test a hypothesis, or check assumptions.

* To check first five rows of dataset, we have a function call **head().**
* This head () function returns the first 5 rows for the object based on position, it is useful for quickly testing if your object has the right type of data in it.
* To check last five rows of dataset, we have a function call **tail ().**
* For finding the names of the columns present in the dataset we make use of **columns**

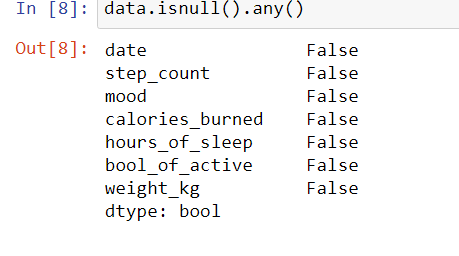


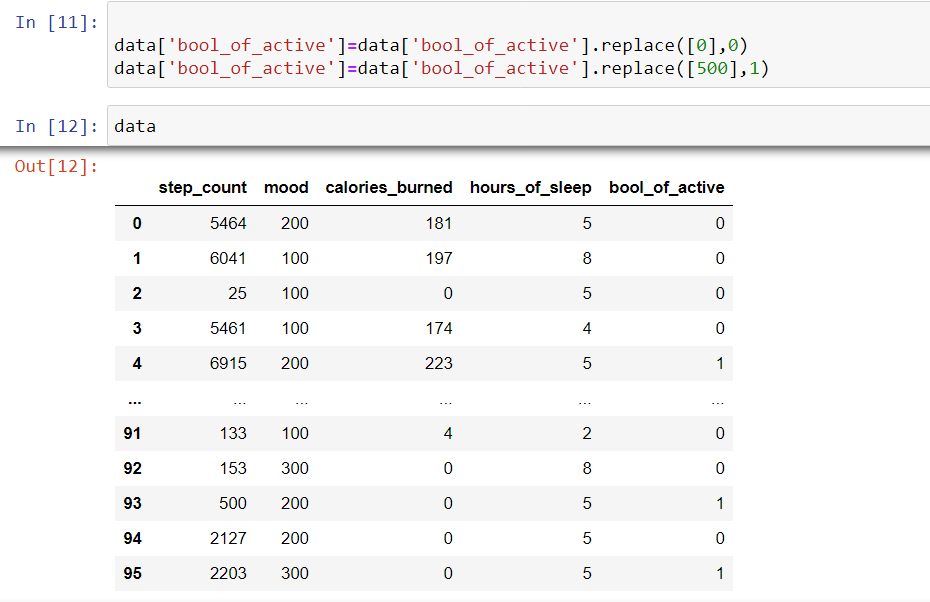
* **data.columns** will return you all the column names which are present in your data.

**Taking care of Missing Data**

Sometimes you may find some data are missing in the dataset. We need to be equipped to handle the problem when we come across them. Obviously, you could remove the entire line of data but what if you are unknowingly removing crucial information? Of course, we would not want to do that. One of the most common ideas to handle the problem is to take a mean of all the values for continuous and for categorical we make use of mode values and replace the missing data.

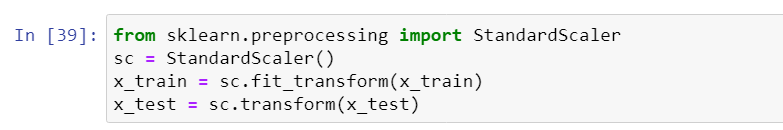
* We will be using **isnull(). any ()** method to see which column has missing values.







**Feature Scaling**



**Splitting Data into Train and Test**

When you are working on a model and you want to train it, you obviously have a dataset. But after training, we have to test the model on some test dataset. For this, you will a dataset which is different from the training set you used earlier. But it might not always be possible to have so much data during the development phase. In such cases, the solution is to split the dataset into two sets, one for training and the other for testing.

But the question is, how do you split the data? You can’t possibly manually split the dataset into two sets. And you also have to make sure you split the data in a random manner. To help us with this task, the Scikit library provides a tool, called the Model Selection library. There is a class in the library which is, **‘**[**train\_test\_split**](http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html)**.’** Using this we can easily split the dataset into the training and the testing datasets in various proportions.

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.

In general, you can allocate 80% of the dataset to training set and the remaining 20% to test set.

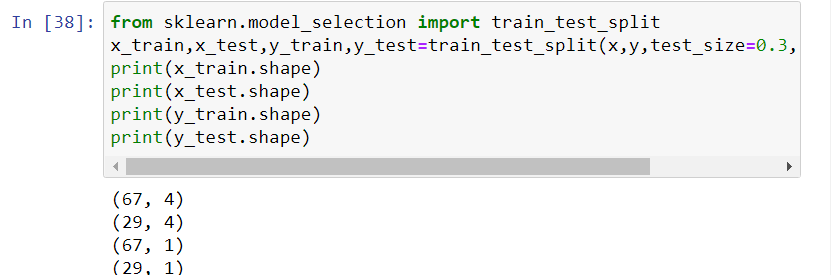
We will create 4 sets— X\_train (training part of the matrix of features), X\_test (test part of the matrix of features), Y\_train (training part of the dependent variables associated with the X train sets, and therefore also the same indices), Y\_test (test part of the dependent variables associated with the X test sets, and therefore also the same indices.

There are a few other parameters that we need to understand before we use the class:

* **test size** — this parameter decides the size of the data that has to be split as

the test dataset. This is given as a fraction. For example, if you pass 0.5 as the value, the dataset will be split 50% as the test dataset

* **train size** — you have to specify this parameter only if you’re not specifying the test size. This is the same as test size, but instead you tell the class what percent of the dataset you want to split as the training set.
* **random state** — here you pass an integer, which will act as the seed for the random number generator during the split. Or, you can also pass an instance of the Random state class, which will become the number generator. If you don’t pass anything, the Random state instance used by np.random will be used instead.
* Now split our dataset into train set and test using train\_test\_split class from scikit learn library.



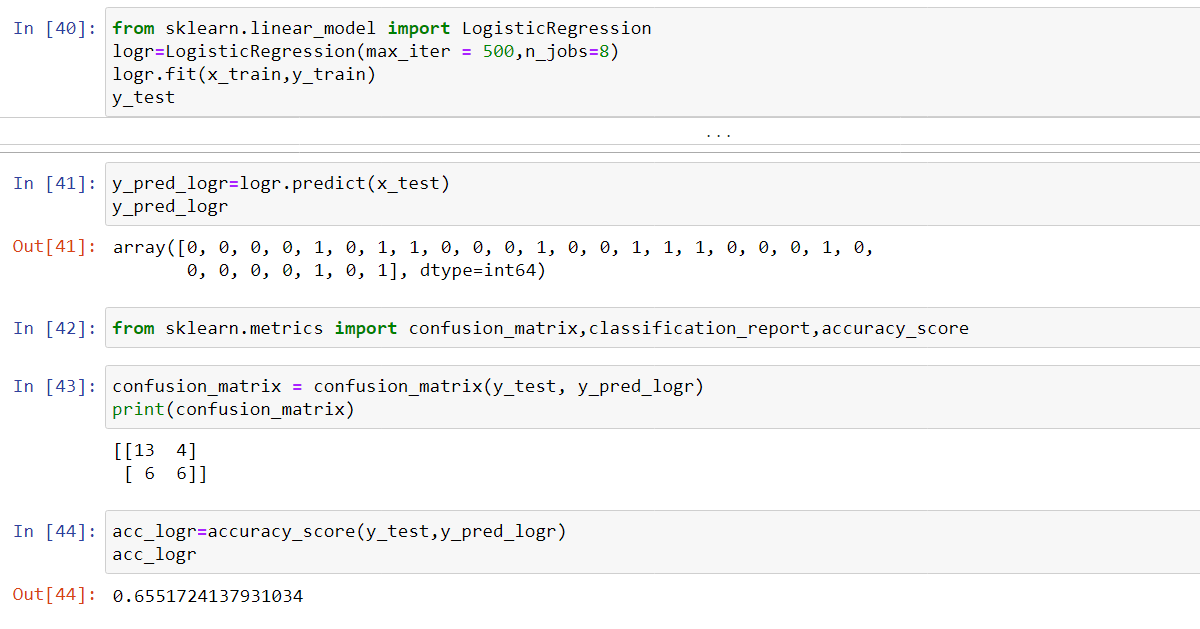
**Model Building**

Training and testing the model:

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 it may be Classification algorithms are Regression algorithms.

**#Logistic regression**

**Logistic Regression** is a Machine Learning classification algorithm that is used to predict the probability of a categorical dependent variable. In **logistic regression**, the dependent variable is a binary variable that contains data coded as 1 (yes, success, etc.) or 0 (no, failure, etc.).



**#Decision Classifier**

Decision Trees are a type of Supervised Machine Learning (that is you explain what the input is and what the corresponding output is in the training data) where the data is continuously split according to a certain parameter. The tree can be explained by two entities, namely decision nodes and leaves. The leaves are the decisions or the final outcomes. And the decision nodes are where the data is split.



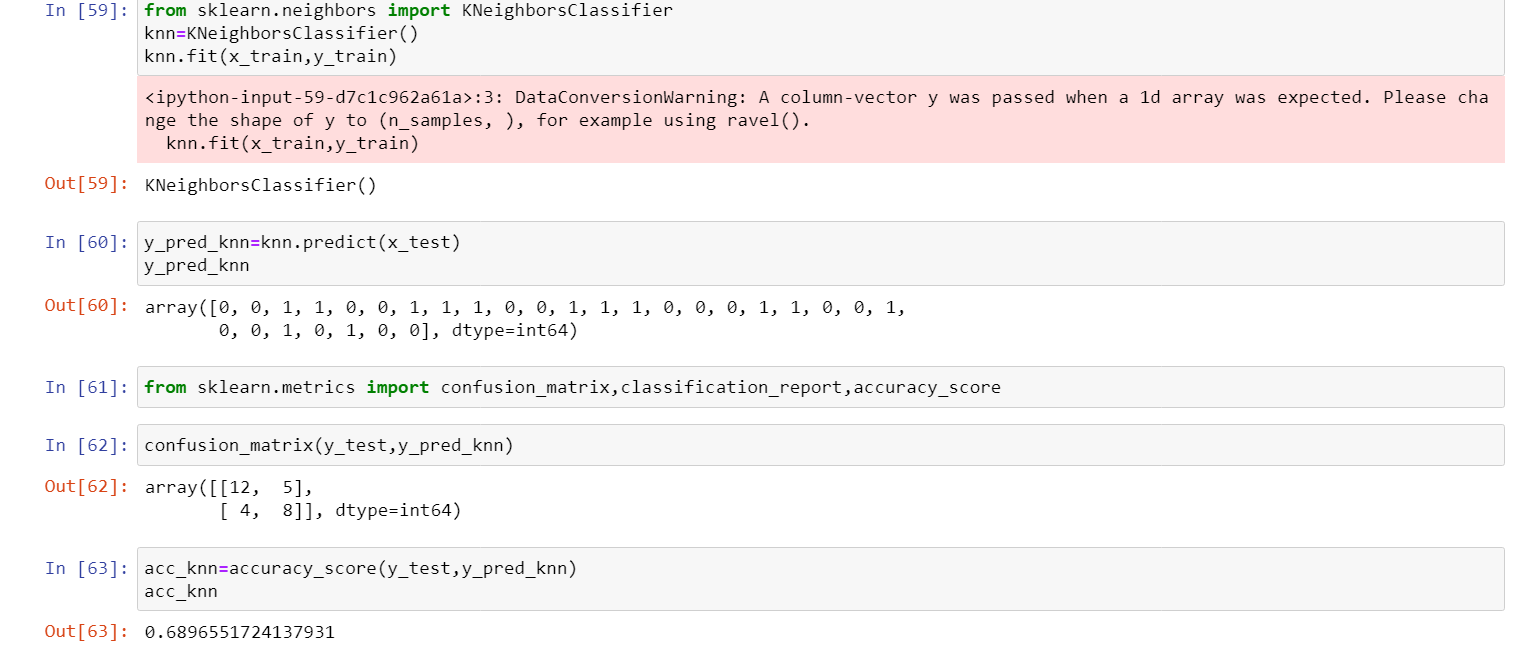
**#SVM**

Support vector machines (SVMs) are powerful yet flexible supervised machine learning algorithms which are used both for classification and regression. But generally, they are used in classification problems. An SVM model is basically a representation of different classes in a hyperplane in multidimensional space. The hyperplane will be generated in an iterative manner by SVM so that the error can be minimized.



**#K Nearest Neighbour**

K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique. KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new da ta.

.

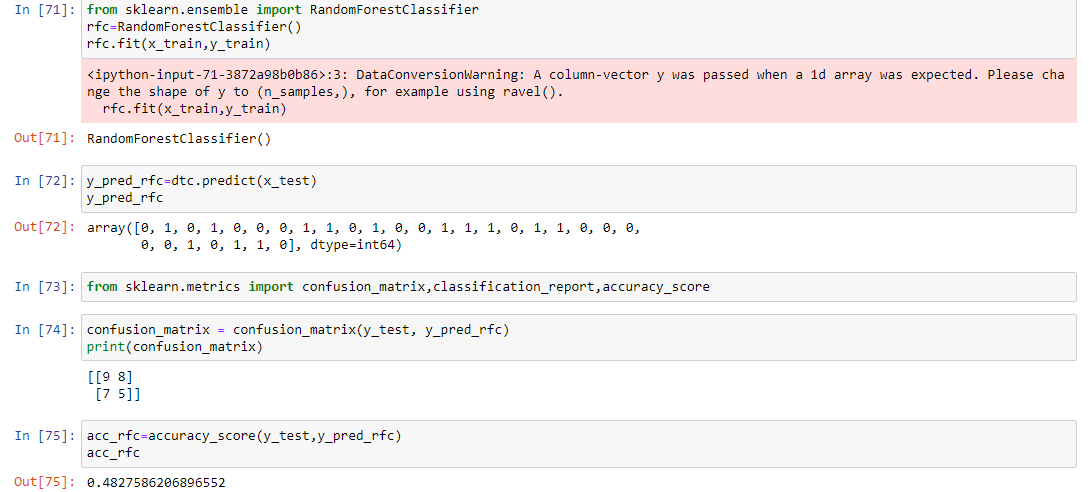
**#Naive Bayes**

Naïve Bayes algorithm is a supervised learning algorithm, which is based on **Bayes theorem** and used for solving classification problems.It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.



**#Random Forest**

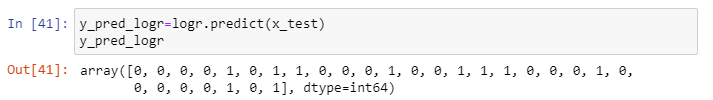
Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. ***"Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset."***



**Predict the values**

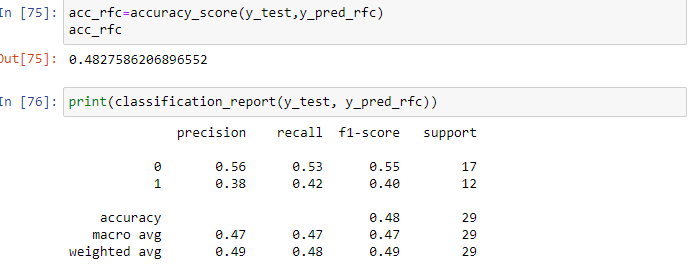
Once the model is trained, it’s ready to make predictions. We can use the **predict** method on the model and pass **x\_test** as a parameter to get the output as **pred.**

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



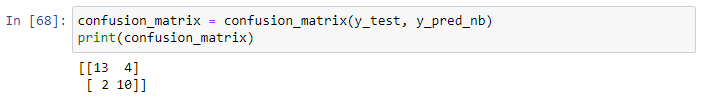
**Evaluation**:

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



* + We also plot **Confusion Matrix** for the same to evaluate.

A **confusion matrix** is a tabular summary of the number of correct and incorrect predictions made by a classifier. It can be used to evaluate the performance of a classification model through the calculation of performance metrics like accuracy, precision, recall, and F1-score.



**Saving A Model**



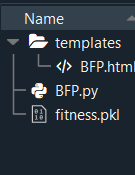
**Application Building**

Creating a HTML File, flask application.

* Build python code
* Importing Libraries
* Routing to the html Page
* Showcasing prediction on UI
* Run The app in local browser

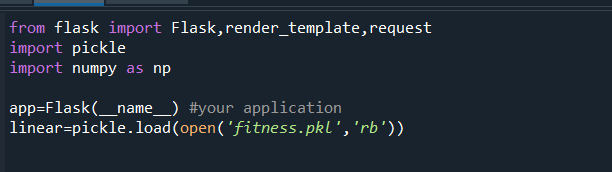
**Project Structure**

Create a Project folder that contains files as shown below



* We are building a Flask Application that needs HTML pages stored in the templates folder
* Template's folder contains index.html
* Static folder contains CSS and image files.

**Task 1: Importing Libraries**



Importing the flask module in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of the current module (\_\_name\_\_) as argument Pickle library to load the model file.

**Task 2: Routing to the html Page**

Here, declared constructor is used to route to the HTML page created earlier.

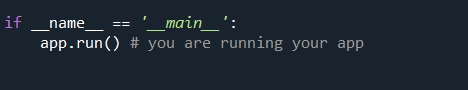
In the above example, ‘/’ URL is bound with home.html function. Hence, when the home page of the web server is opened in browser, the html page is rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.

Here, “index.html” is rendered when home button is clicked on the UI



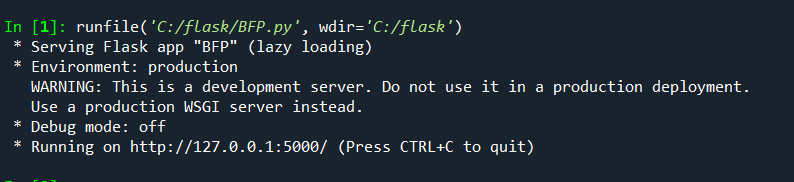
**Task 3: Main Function**

T used to run the application in a local host.

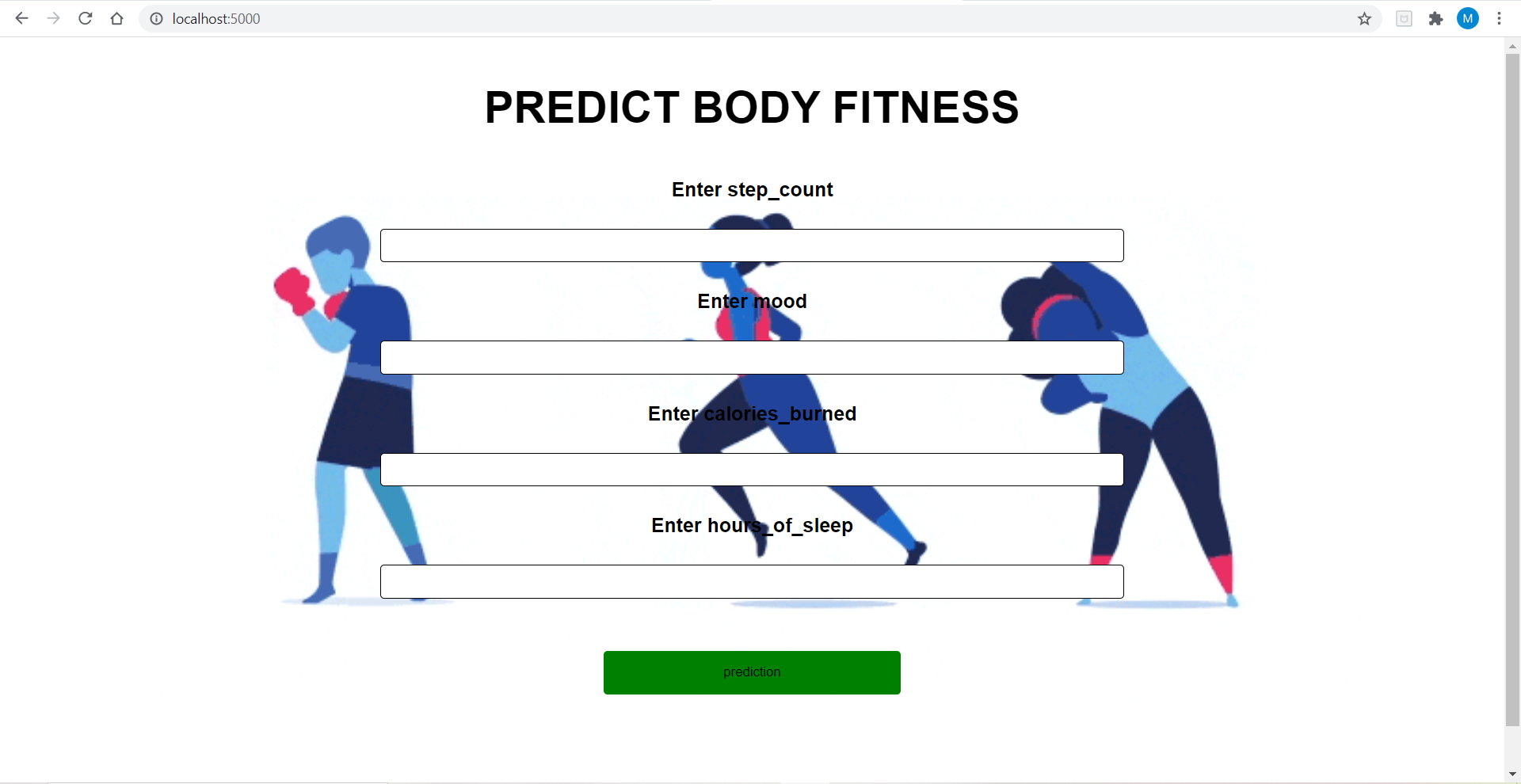


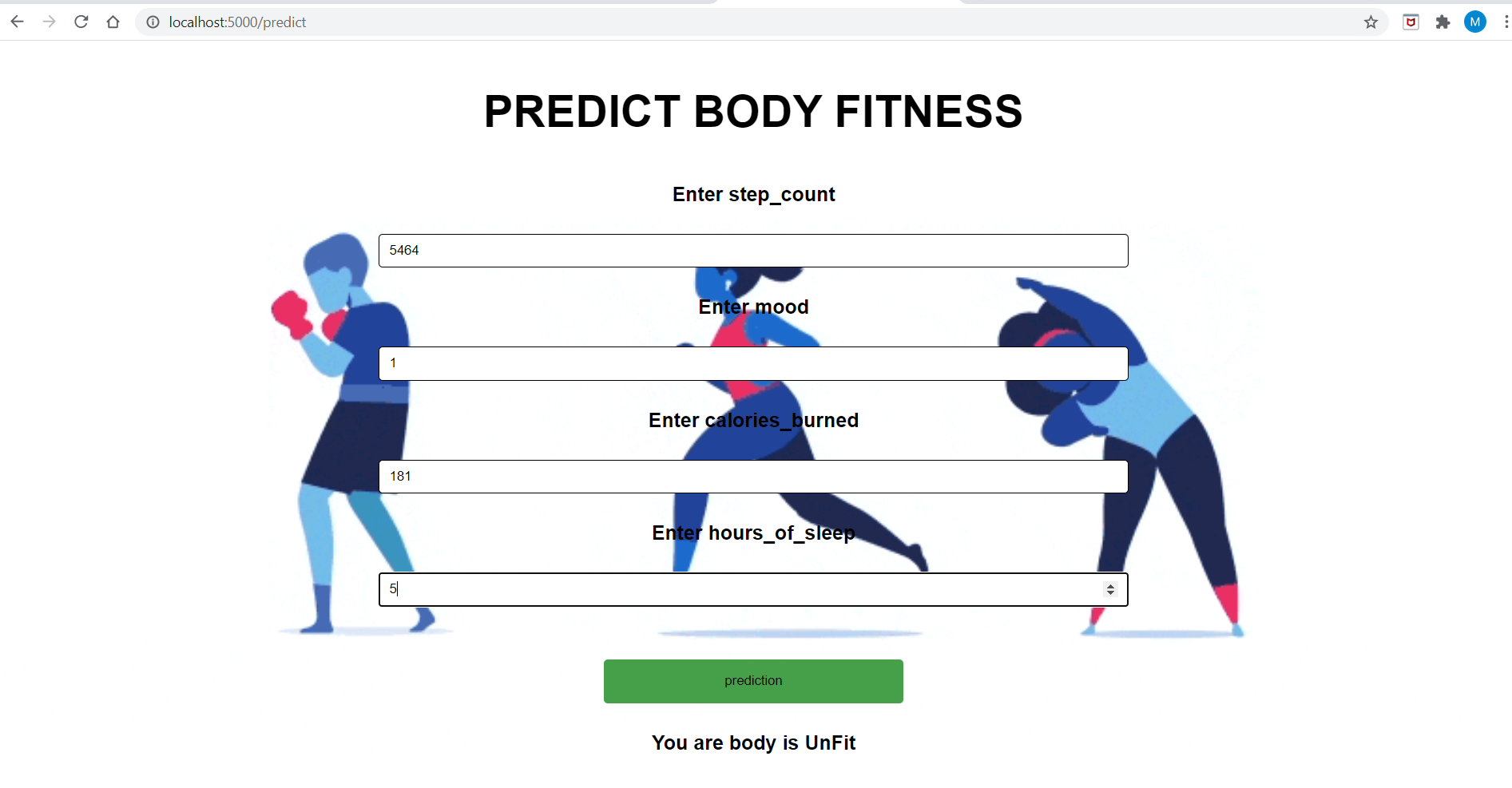
**Run the application:**

* + Open the anaconda prompt from the start menu.
  + Navigate to the folder where your app.py resides.
  + Now type “python app.py” command.
  + It will show the local host where your app is running on [**http://127.0.0.1:5000/**](http://127.0.0.1:5000/)
  + Copy that local host URL and open that URL in the browser. It does navigate me to where you can view your web page.
  + Enter the values, click on the predict button and see the result/prediction on the web page



**Output Screen:**





**Conclusion**

This paper presented a machine learning model developing a stack of regressors predict the Bool of activeness. The model relies on four predictors: step count, mood, calories burned, hours of sleep. An experiment was completed using ensemble stacking of the regressors and thereby proving that the combination of models will give high accuracy in the prediction.

**Reference**

* 1. www.kaggle.com
* 2. www.quora.com
* 3. www.wikkipedia.com