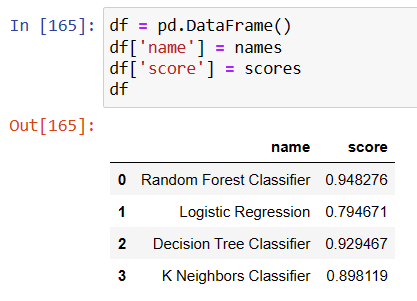
**Testing Model with Multiple Evaluation Metrics**

Multiple evaluation metrics means evaluating the model's performance on a test set using different performance measures. This can provide a more comprehensive understanding of the model's strengths and weaknesses. We are using evaluation metrics for classification tasks including accuracy, precision, recall, support and F1-score.

**Comparing the model**



****

After comparing the model with the help of bar plot. We came to a conclusion that Random Forest is showing the highest accuracy and is performing well.

### Save The Best Model

Saving the best model after comparing its performance using different evaluation metrics means selecting the model with the highest performance and saving its weights and configuration. This can be useful in avoiding the need to retrain the model every time it is needed and also to be able to use it in the future.



**Integrate With Web Framework**

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.

 This section has the following tasks:

* Building HTML Pages
* Building server-side script
* Run the web application

**Building Html Pages**

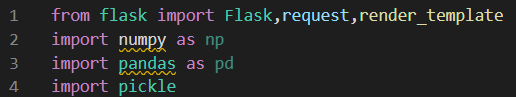
For this project create two HTML files namely

* index.html
* inspect.html
* outputt.html

and save them in the templates folder.

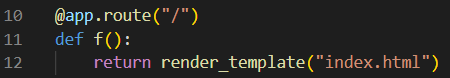
**Build Python code**

Import the libraries



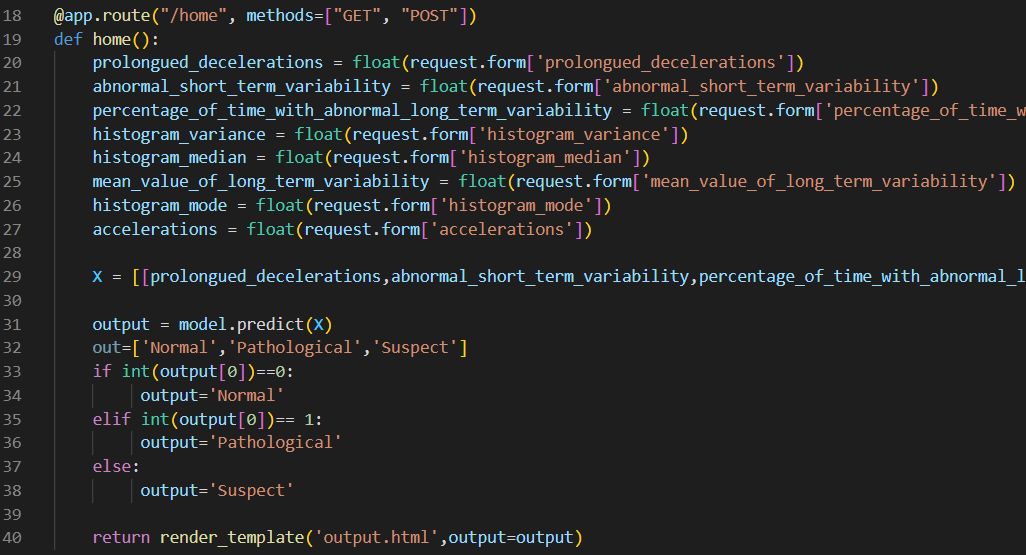
Load the saved model. Importing the flask module in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of the current module (\_\_name\_\_) as argument.

https://lh3.googleusercontent.com/CnIYi6VHUxjay-D3LkahbdVYwtiPTM56QKqOuew7uFslmGpyXdwrPS6WoVjJjVk0eWZtnr4nzqQ6eS98kFXN7E5OMizDPei8lZHKvbRl7pxDCbXaXtZpFrFQkTVhPqXWQdIPqwYFZJS9ozCedq1wRA



Here we will be using a declared constructor to route to the HTML page which we have created earlier.

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



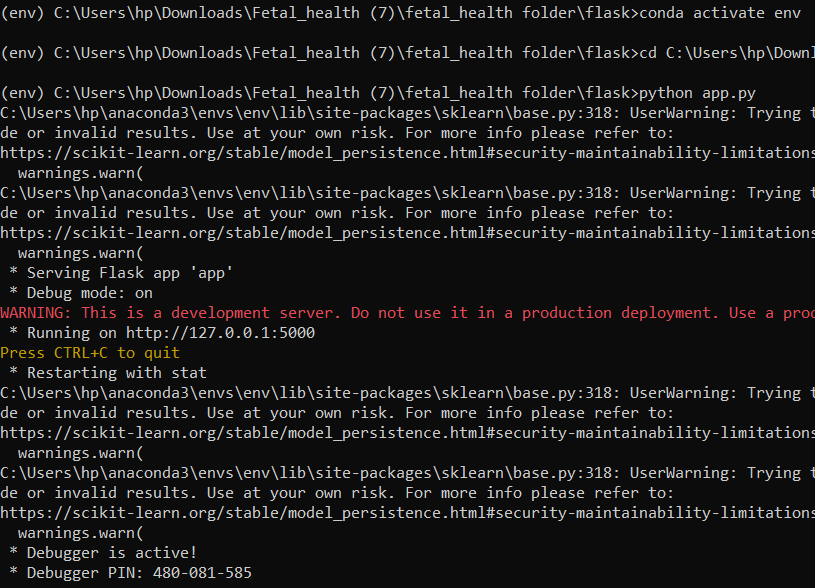
Here we are routing our app to predict() function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the model.predict() function. This function returns the prediction. And this prediction value will be rendered to the text that we have mentioned in the submit.html page earlier.

**Main Function:**

https://lh5.googleusercontent.com/sI6HcCI2bOPGHPSimZMXfbwDIaGb7yfTERSd3sZXSQN4JxfcenZchR8AitaUDdUrJ6ExvJaf5fCVvqSlVu9OzrxJRK_cgltWN7efO9sc1_MUOTBoZyoWkOaZcU53KoLZsBFSLNiQwQw_siqajzoEqQ

**Run the web application**

* 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.
* Click on the predict button from the top left corner, enter the inputs, click on the submit button, and see the result/prediction on the web.



Now,Go the web browser and write the localhost url (http://127.0.0.1:5000) to get the below result.



