Diabetes Classification

Made By:

1. محمد تامر محمد المصري
2. سيف باسل ابوطالب
3. محمود ناصر الشعراوي

Project Overview

We are classifying patients to see if they have diabetes or not. We started with conducting an exploratory data analysis to study our dataset, we found that:

1. Diabetic patients have Higher glucose rate and higher weight as compared to non-diabetic ones.
2. The Age is not directly related but higher glucose level in oldies can be a cause of Diabetes in them, also the males of age 40 to 80 have higher blood glucose level than females.
3. The BP is not directly related to the diabetes, as patients have highest BP. are Found to be non-diabetic.
4. Diabetic patients have lower HDL-Cholesterol.
5. Females being diabetic are more than the males being diabetic.
6. Higher cholesterol is seen in patients having diabetes.

So, we removed unimportant features from our dataset to get better model results and avoid overfitting.

Then we started developing our machine learning models and studying them, we did 4 models:

1. KNN
2. SVM
3. Neural Network
4. Decision Tree (with post-pruning)

We tested them against different parameters and kernels to get the best output.

Next, we developed a simple user interface to let patients enter their vitals, we used a Flask as our web framework to link our models with user interface, and Threading library to run all the models concurrently to have better performance and used global variables and shared memory to store our results and show them on the interface.

So, our application workflow is as follows:

1. Get user input from the form.

```
# getting patient info from Form
cholesterol_value = request.form["cholesterol"]
glucose_value = request.form["glucose"]
hdl_chol_value = request.form["hdl_chol"]
age_value = request.form["age"]
weight_value = request.form["weight"]
systolic_bp_value = request.form["systolic_bp"]
diastolic_bp_value = request.form["diastolic_bp"]
```

2. Assign each model to a thread.
3. Create global variables to store model results.

```
def run_knn():
    global knn_prediction
    result = predictDiabetes.predict_diabetes_KNN(
        cholesterol_value,
        glucose_value,
        hdl_chol_value,
        age_value,
        weight_value,
        systolic_bp_value,
        diastolic_bp_value,
    )
    knn_prediction = result

def run_svm():
    global svm_prediction
    result = predictDiabetes.predict_diabetes_SVM(
        cholesterol_value,
        glucose_value,
        hdl_chol_value,
        age_value,
        weight_value,
        systolic_bp_value,
        diastolic_bp_value,
    )
    svm_prediction = result
```

```python
def run_nn():
    global nn_prediction
    result = predictDiabetes.predict_diabetes_NN(
        cholesterol_value,
        glucose_value,
        hdl_chol_value,
        age_value,
        weight_value,
        systolic_bp_value,
        diastolic_bp_value,
    )
    nn_prediction = result

def run_dt():
    global dt_prediction
    result = predictDiabetes.predict_diabetes_DT(
        cholesterol_value,
        glucose_value,
        hdl_chol_value,      You, 2 weeks ago • taki
        age_value,
        weight_value,
        systolic_bp_value,
        diastolic_bp_value,
    )
    dt_prediction = result
```

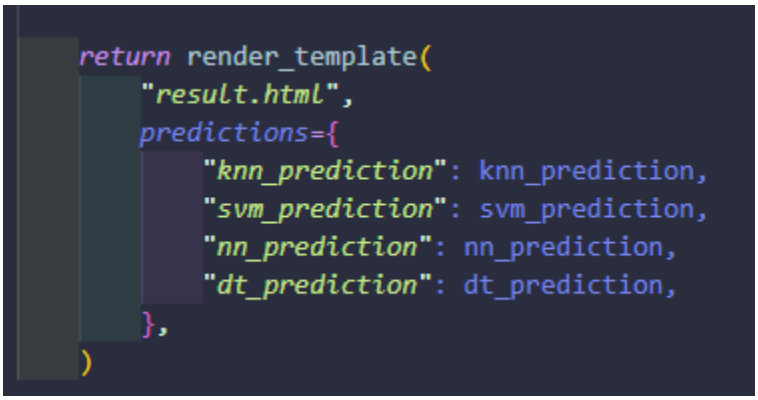
```python
# Assign each a thread to each model
thread_knn = threading.Thread(target=run_knn)
thread_svm = threading.Thread(target=run_svm)
thread_nn = threading.Thread(target=run_nn)
thread_dt = threading.Thread(target=run_dt)
```

4. Run the threads concurrently.

```python
# start the threads concerruntly
thread_knn.start()
thread_svm.start()
thread_nn.start()
thread_dt.start()

# terminate the threads when they're done
thread_knn.join()
thread_svm.join()
thread_nn.join()
thread_dt.join()
```

5. Send the models results to user interface.



How to run the project

To start the project run application.py and ctrl-click on 127.0.0.1:5500 in the terminal.

To see all the models go to predictDiabetes.py.