INFO 4150E Mini Project 1
Process Parameter Prediction

Domain information

Manufacturing processes are monitored by different types of sensors like pressure, temperature, flow, etc. Tracking the trend of changes in one parameter due to changes in another is a critical aspect of process monitoring. In this application we have a main sensor value (sensor 1) which determines what the value of the other sensor should be in normal conditions. This WebApp predicts what the trend should look like, if all is well, given the main sensor readings and the operators monitor the real trend versus these predicted values to detect deviations and then check for causes.

Task

1. Build a Flask based WebApp which will predict the sensor2 values based on the model that you will train and make available. **All your work will be done in Visual Studio Code and Jupyter Notebook will not be used.** Please recollect that Flask based applications require a certain directory structure to work properly.

2. The Flask based App will have three routes and 3 associated HTML pages for them.

    a. The first route will be called "**home**" and will render the landing / home page where the user's and administrators will have options to select from.
        i. For the user, then there should be an input option called - "**Input Sensor 1 Value**" available to click upon.
        ii. For an administrator, then there should be a menu option which is called "**Train Model**" available to click upon when password is validated.
        iii. Feel free to add a picture or a logo of your business.

    b. The second route will be called "**model**" and it will train on a dataset which is available in the database called '**process_values.db**' in a table called '**sensor_data**'. Build a **Linear regression** model, and then **save it** in the same directory as the WebApp. Save the model with the name "**model.pkl**". <u>The training can only be initiated by an administrator upon entering a valid password after clicking the 'Train Model' button.</u> The password you will use for the administrator shall be "**password**". When the training is done, publish the RMSE score on the webpage. Training can be done any number of times by the administrator and will use all the data in the database (no splitting into test and train required). The RMSE score of the trained model along with a message saying "Model Trained" will be displayed on a separate HTML page.

    c. The third route will be called "**predict**" and shall take an input from the user, which is done by entering a number and clicking the 'Submit' button and then use the saved model to predict the Sensor 2 value and then render it to the user as a prediction. **Please note that the user should only be allowed to enter sensor 1**

values ranging between -10 to 10. The HTML interface should control this. **This route shall also save the user inputs and the predicted price as a new row in the database.**

3. While making the site presentable **using CSS is encouraged, it is not a must**. However, the **HTML pages rendered should make it easy for the user to provide the inputs and see the results. For example, the User should be able to navigate from any page back to the home page.**

Guidelines for building the model:

1. In the route where you are training the model, read in all the data from the table in the database into a pandas dataframe. Use "pd.read_sql_query(…) as shown in the examples.
2. If you have to reshape X after separating into X and y, use X.values.reshape(-1,1) as X is a pandas series object.
3. The data (sensor 1 versus sensor 2) may not have a linear trend and hence the RMSE values when using Linear regression may be high. There would be a new feature that you need to introduce which will either have a polynomial degree of 2 or 3. You can plot sensor1 versus sensor2 separately for your own consumption, independent of the project.
4. Remember the input provided by the user has to be fed to the model as a numpy array for prediction.

Answer the following questions in a word document:

1. What is the accuracy of your model in terms of RMSE?
2. Describe the design of the complete WebApp you built - the flow, the user interface and anything unique that you did in the design?
3. What did you enjoy the most about this project and what were the challenges?

Tip:

1. Use the '***PolynomialFeatures***' function **to generate additional feature while training.**
2. Make sure that the **input provided by the client** is also **preprocessed to generate the same number of polynomial features** otherwise the model will not be able to predict.

The MP1 is due on July 8th, 2022, by midnight.