MetLife

ML Engineering Challenge

The problem

MetLife is an international insurance company that pays its customers thousands of health insurance claims per year. Given this, it is important to estimate the potential amount of money to be payed to our clients at any given point in time, accounting for different future scenarios to make better decisions.

The challenge

Create a machine learning model to predict medical insurance costs for MetLife clients. To do this, you will use the provided dataset (*dataset.csv*), which contains the relationship between personal attributes (age, gender, BMI, family size, smoking habits), geographic factors, and their impact on medical insurance charges:

Age: The insured person's age.
Sex: Gender (male or female) of the insured.
BMI (Body Mass Index): A measure of body fat based on height and weight.
Children: The number of dependents covered.
Smoker: Whether the insured is a smoker (yes or no).
Region: The geographic area of coverage.
Charges: The medical insurance costs incurred by the insured person.

Considering the problem to be solved, you will:

1) Define a modelling approach (what type of algorithm) and give a brief argumentation (about 1 paragraph) for your choice.
2) Within your code, create an instance of MySQL or PostgreSQL database, where you'll create a table named training dataset and insert the *dataset.csv* contents.
3) Build a training pipeline (*training.py*) where you will read this table, prepare it for modelling, run a hyperparameters search, store the best trained model in any desired format, and store a text or pdf file with some evaluation metrics to describe the model.
4) Build a scoring pipeline (*scoring.py*) where you will:
    a. Read a new table that you will have to create via randomly sampling 10 rows from *dataset.csv* (this is just for testing purposes).
    b. Load the previously trained model and predict the above table.
    c. Append the prediction results to this table (or a different one, with the appropriate logic).
    d. Report a final performance metric of your choice.
5) Build a script that will package the whole end-to-end solution into a Docker image (training and scoring should execute sequentially and without errors). If possible, publish such an image in a public repo; else make sure this script is self-explanatory for evaluation.

Final considerations:

1. Resulting code needs to be posted into a public github repo and shared as part of the solution for this challenge.
2. You can use jupyter notebooks for development and training purposes, but final pipelines need to be under the form of a python script.
3. If you think any definition is missing, make an educated guess and properly communicate it in your solution.
4. Always remember that there's not a single solution: simplicity, robustness, innovation and python coding skills will be valued.
5. If something seems difficult or you don't have clear technical directions on how to solve it, sketch a pseudo-code solution on how you think you should move forward. Problem solving under partial information scenarios is a skill we value a lot.

Good luck!