The goal of this exercise is to test your ability to solve Konfer-specific problems of interest involving code analysis. There are three examples that you need to look at.

1. The first example is a time-series model training using an LSTM model. The script is lstm.ipynb
2. The 2nd example is a quasi-time series model training using a KNN model. The script is knn.ipynb
3. The 3rd example is a piece of code that calls a cloud-hosted AI inference service (Konfer-specific fine-tuned GPT-3 model) that takes in a natural language question and generates a SQL statement.

The shared directory contains

1. Python input codes to be analyzed: lstm.ipynb, knn.ipynb and nlq\_to\_sql.ipynb
2. The first two ipynb files use these input files for the models to train on: akm.csv and trainingdata.csv. The third is self-contained but requires a GPT-3/OpenAI key for billing purposes. I can share the Konfer key when the time comes.
3. And of course this readme

Let us assume that we have been given the important locations in the code to do run-time introspection. This in itself is a problem that we need to solve but for the purposes of this exercise, we are not concerned about this. From the Konfer point of view, there are likely to be code pieces that fetch/read data, do model training, set up hyperparameters and run inference because they are important from an AI/ML perspective.

1. Your job is to develop some python code that can take the each of the above code files as input (You should convert the the ipynb to pure Python for this purpose), programmatically does bytecode injection/monkey patching using relevant libraries such as AST and Inspect such that a new modified Python code of the original python code is produced.
2. The programmatically modified Python code should be obviously executable.
3. You don’t need to worry about the modified python code being human-readable. It just needs to do the right thing.
4. If we run the modified Python code it should produce output such as:
   1. Where is the data read from?
   2. Print some rows of the data at different places in the data pipeline including what is being sent to the training API.
   3. What are the training parameters being sent to the training API.
   4. What modeling algorithm is being used.
5. If you are able to monkey-patch into the library itself (sklearn, keras beyond our code) that will be awesome/may be needed.

Some references/keywords of interest: PyCG, AST, Inspect, monkey-patching, unittest.mock, RedBaron and Gorilla.

I am happy to collaborate with you on this over the next few weeks including adjusting/fine-tuning/clarifying the problem statement, bouncing some ideas etc. so this is really a two-way interaction.