NCRI
NETWORK CONTAGION RESEARCH INSTITUTE

# ML Systems Intern: Screening Assessment

In this assessment, you are going to build a text classification model and serve it to the end user through a Flask application. Here are the constraints that you need to keep in mind:

Part 1-

1. The input dataset should be in .csv format containing a text column and a label column. You can use any dataset that you like, as long as it contains more than 10000 samples. You can use any dataset from Kaggle as long as your project uses the data in the format mentioned above. The labels can be binary or multiclass. The file can contain other columns but we will only focus on the text and the label. Beyond basic data consistency checks, do not spend time in preprocessing the data. Previously, candidates have used the Jigsaw comment toxicity dataset or the IMDb movie reviews dataset but you are free to use any other datasets.
2. Use the Datasets library from HuggingFace to process this dataset. Make sure you have train-test splits in place.
3. Use the AutoModel classes from HuggingFace to set up a model. You can use any suitable model for this task.
4. Make sure to use the HuggingFace Trainer class with suitable arguments to train the model on your train set. Make sure to train the system for at least 3 epochs, and that you only save the model from the epoch with the best F1 score on the test set.

Part 2 -

5. Create a simple Flask application that takes a text input, gets the model prediction, and shows the output to the user using your previously trained model.
6. Next, we want to do benchmarking in terms of prediction speed. For demonstration purposes, use your entire dataset for this task. Note that we are focusing on speed, not accuracy. Pass the dataset to the model and check the total time it took to get predictions for all the samples. As a reminder, make sure that your dataset has more than 10000 samples.
7. **Open-ended question:** Now, make one improvement in your prediction pipeline to speed up the performance. Potential improvements could include optimization in the model architecture, multiprocessing, FP quantization, or the use of a framework that allows you to serve multiple copies of the model in real time. Record the total prediction time in this case for the same dataset. Highlight the speedup factor.

Submission instructions -
Create a GitHub repository containing the following resources:

1. CSV file of your dataset
2. Notebook/Python script used to load the dataset, train, and save the model.
3. Flask Application folder.
4. Notebook/Python script used to benchmark and improve prediction pipeline.
5. README.md containing instructions to run the code.