Jumio: ML/AI Engineer
Take-Home Assignment

The following assessment will help us assess your proficiency at coding and your coding style, as well as your approach to machine learning problems. As such we expect a report detailing your approach to the ML problem, motivating your choices, and discussing your results.

You are welcome to use frameworks such as TensorFlow, PyTorch, etc.

Task

The objective is to build a classifier on MNIST dataset that meets certain performance criteria.

For this task, you will need to build a test set with at least 1,000 samples per class.

Train the model of your choice and compute the following metrics:

1. Confusion matrix
2. Per class FPR
3. Per class TPR

The goal now for your classifier is to reach a maximum false positive rate of **0.00015** for each class in the test set while maintaining the recall as high as possible, by carefully designing a decision strategy. Your classifier will output **'not_sure'** for cases where the model prefers to avoid making a decision.

With your updated decision strategy recompute the following metrics on your test set:

1. Confusion matrix
2. Per class FPR
3. Per class TPR
4. Coverage => 1 - (num predicted 'not_sure' examples) / (total examples)

Once the task is complete, please return us your code implementation (ipython notebook or python scripts) and make sure we can run it on our side as is.

Data description

The data file mnist.csv contains gray-scale images of hand-drawn digits, from zero through nine.

Each image is 28 pixels in height and 28 pixels in width, for a total of 784 pixels in total. Each pixel has a single pixel-value associated with it, indicating the lightness or darkness of that pixel, with higher numbers meaning darker. This pixel-value is an integer between 0 and 255, inclusive.

The data set (mnist.csv), has 785 columns. The first column, called "label", is the digit that was drawn by the user. The rest of the columns contain the pixel-values of the associated image.

Each pixel column in the training set has a name like pixelx, where x is an integer between 0 and 783, inclusive. To locate this pixel on the image, suppose that we have decomposed x as x = i * 28 + j, where i and j are integers between 0 and 27, inclusive. Then pixelx is located on row i and column j of a 28 x 28 matrix, (indexing by zero).

For example, pixel31 indicates the pixel that is in the fourth column from the left, and the second row from the top, as in the ascii-diagram below.

Visually, if we omit the "pixel" prefix, the pixels make up the image like this:

```
000 001 002 003 ... 026 027
028 029 030 031 ... 054 055
056 057 058 059 ... 082 083
 |   |   |   |  ...  |   |
728 729 730 731 ... 754 755
756 757 758 759 ... 782 783
```

Acknowledgements
More details about the dataset, including algorithms that have been tried on it and their levels of success, can be found at http://yann.lecun.com/exdb/mnist/index.html. The dataset is made available under a Creative Commons Attribution-Share Alike 3.0 license.