Documentation/Reporting

Initial exploration led to the conclusion that data is randomly generated with specified conditions with all uniform distributions and numerical columns that do not affect the target column.

I conducted statistical tests for the columns (both numerical and categories) to check their impact on churn target.

Hence before jumping on creating new columns, I confirmed one last time if the data was even separable and compressed data to 3d and plotted. That plot confirmed my doubt.

I created new columns:

Age group (category) from age bins

Average Usage Per Month and created its bins to use as categories. Because if the numerical column itself does not have an impact on the target, derived numerical columns wouldn’t either. (in my humble opinion). This column assumes that total usage was uniform across all subscribed months.

Bill per GB was created by dividing monthly bill by total usage in GB and its categories were used.

Bill Subscription Product by multiplying total months to monthly bill.

Surprisingly these four categories did not have much significant differences in churn rates BUT when combined all these with gender and location categories, the churn rates varied much.

We interpret this in a way that there are combinations of these categories where the mean churn is 1 so in future if a customer comes from that particular group which historically has a record of customers churning, it is likely that the new customer will churn.

With the assumption that model would be able to catch the patterns, I trained based on these 6 categories after applying one hot encodings.

Then I tested four models on a fraction of data, picked two of two to train on whole data. Later finalised neural network (single feedforward) with ReLU and Sigmoid activation functions, with adam optimiser as it converges faster, and loss was calculated by binary\_crossentropy as it is binary classification.

Performance metrices: Confusion matrix

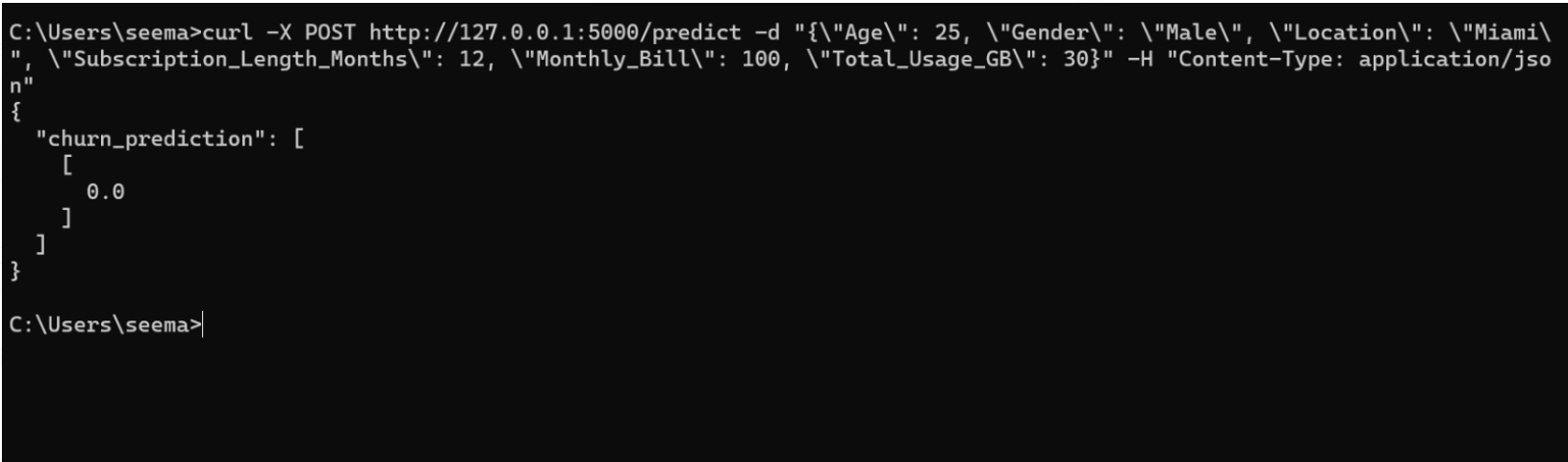
[[4077 6002]

[3935 5986]]

Accuracy (train) = 53% and (test accuracy) = 50% which means model still works on chance level given that is it random.

After that pickle files of model and encoder were used and deployed locally using flask.

Example of testing using CURL



In flask we defined a preprocessor function which takes raw input, creates the new columns, encodes them and the model predicts based on the input.

Index.html page is for the front page when using GET method.