# Neural Networks For Santander Customer Satisfaction

Team: Sam Schwager, Scott Morris, Nick Steele. {sams95, swmorris, nsteele} @stanford.edu

Predicting:
Customer satisfaction is an important metric for success for large financial institutions such as banks. Unhappy customers don't come back, and are hard to detect since they often don't voice displeasure before leaving. Two years ago Santander Bank launched a Kaggle competition to identify dissatisfied customers, with a $60,000 prize pool. We created a deep learning algorithm that gives 0.794 score on the test data set despite weighting towards a balanced distribution.

Models:
We used fully connected neural networks with varying numbers of hidden layers and Xavier weight initializations.

We started using a standard logistical loss function, which performed very well on the AUC metric that the competition uses.

$$Loss\,(y^{truth},\, y^{pred}) = y^{truth} log(y^{pred}) + (1 - y^{truth})(1 - log(y^{pred}))$$

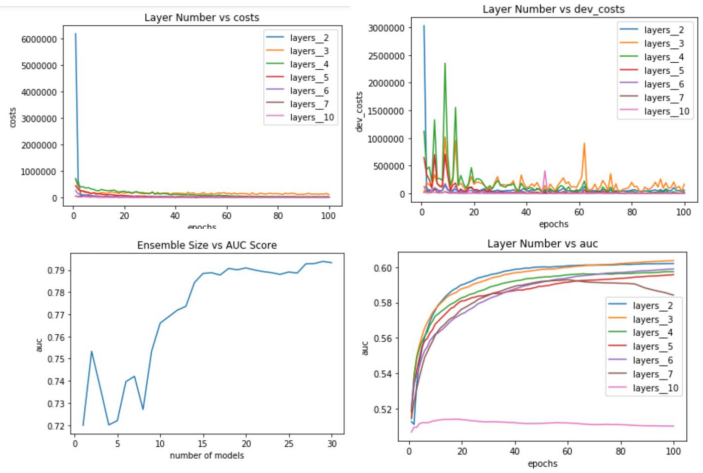
However, when we examined our predictions and found nearly all of them to be zero's (which naturally performs very well given the data imbalance), we updated the loss function to penalize missed ones.

$$Loss\,(y^{truth},\, y^{pred}) = \lambda * y^{truth} log(y^{pred}) + (1 - y^{truth})(1 - log(y^{pred}))$$

We tuned this lambda hyperparameter until our model predicted ones with a similar frequency as represented in the underlying data.

Data:
All data was given to us in the Kaggle competition (there are 68418 training examples). The rows are individual customers, and the columns are anonymized features. Each customer is labeled with a 1 or 0 for satisfied or unsatisfied.

Features:
Each example has 370 numerical anonymized features, all given to us by the Kaggle competition. We did not derive any features, as in theory the network will automatically do so. However, we did mean-variance normalize each feature.




Results:
We used 80% of our data for training 10% for dev, and 10% for test. The charts show various metrics with different hyper-parameter choices. We used an ensemble approach with 30 models on our test data and computed an AUC to 0.794 , not from the competition winning 0.829. This is depite the fact that we intentionally weighted our loss function away from optimization on the AUC to ensure prediction of 1s.

Future Work:
The next few steps would likely include analyzing the features more concretely and attempting to develop some sort of intuition for the data.

Discussion:
We noticed a marked improvement in performance from before tuning the number of hidden layers and using model ensemble (from low .7s accuracy on the dev set), to afterwards (our best model yielded 0.794 AUC on the test set set). Our model of a multi-layer neural network is a reasonable choice for the problem at hand. These tuning techniques allowed us to test a wide variety of networks and ultimately settle on one that preformed best on the test set. The Kaggle competition winner had an AUC score of .829072 on the test set. While our score did not quite match the winner's, we are happy that we were able to create a competitive algorithm, especially given our loss function modification that heavily encouraged the output of ones.