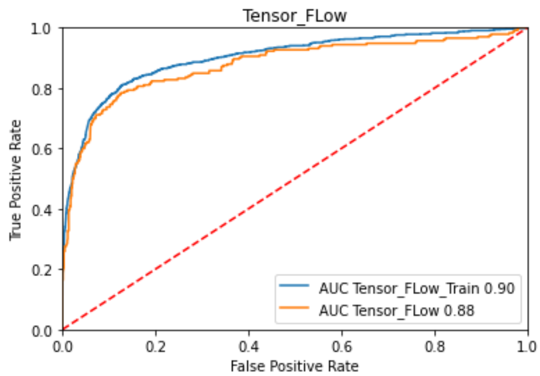
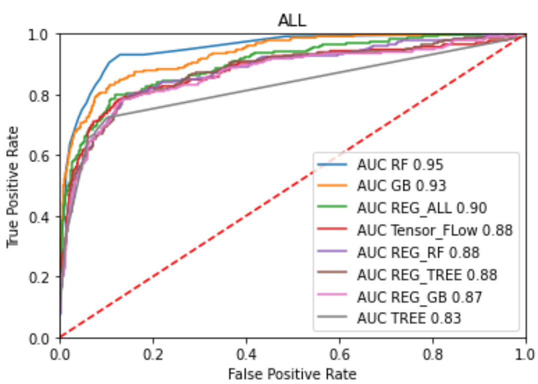
MSDS 422 Machine Learning: Assignment 4 – Neural Networks

In addition to our models that we had created in prior assignments (decision tree, random forest, gradient boosting, logistic regression models using different set of variables), we have created an additional model using neural networks. We created an ROC curve of our neural network model to give a sense for how well it performs at predicting loan default. We limited our model to use the variables that our Gradient Boosting model found to be predictive. Below is the ROC curve for our neural network model:




```
Tensor_FLow CLASSIFICATION ACCURACY Activation fn = relu
======
Tensor_FLow_Train  =  0.8611577181208053
Tensor_FLow  =  0.8540268456375839
------
```

Based on these metrics, this appears to be a very strong model for predicting loan defaults as the accuracy against the training dataset is not that far off from the accuracy of the test dataset. Here is how our neural network model (Tensor_Flow) compares to the other loan default predicting models:




```
ALL CLASSIFICATION ACCURACY
======
RF  =  0.912751677852349
GB  =  0.910234899328859
REG_ALL  =  0.8901006711409396
REG_TREE  =  0.8766778523489933
REG_GB  =  0.8766778523489933
TREE  =  0.8766778523489933
REG_RF  =  0.875
REG_STEPWISE  =  0.8741610738255033
Tensor_FLow  =  0.8540268456375839
------
```

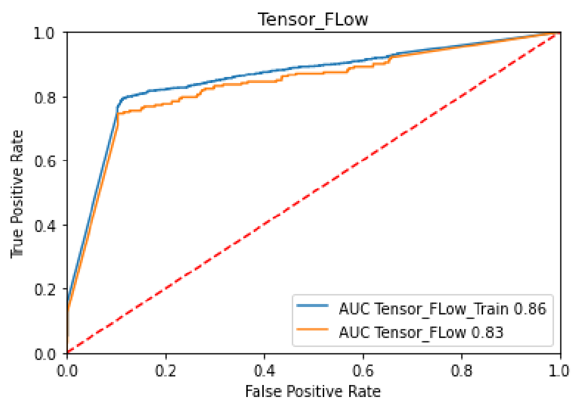
While the Tensor Flow model rates high in terms of AUC (.88) and Accuracy (.85), it is not amongst the top models in these rankings. We would recommend going with the Regression Model with Tree-based variables. Even though it did not have the best AUC or accuracy, regression models tend to be easy to implement. Additionally, the fact that it has a lesser number of variables (than say the Regression Model with ALL variables) makes it easy to work with.

We also built a model for predicting the loss amount using neural networks. Below are our model rankings:

```
ALL LOSSES MODEL ACCURACY
======
GB  =  2422.2769707799052
RF  =  2951.0813707195125
REG_ALL  =  3634.676632630516
REG_TREE  =  4239.965700169976
REG_RF  =  4358.06472084949
REG_GB  =  4358.06472084949
REG_STEPWISE  =  4796.838013306385
TREE  =  5722.46895603711
Tensor_FLow  =  9467.696005753261
------
```

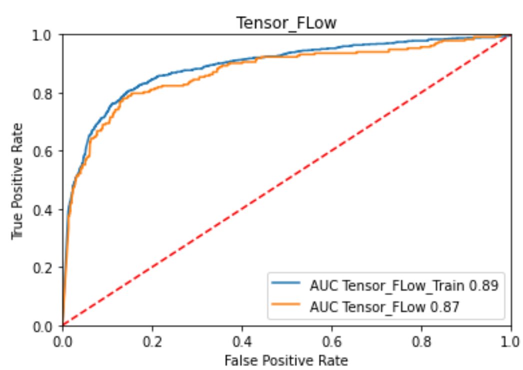
With its Root Mean Square Error of 2422.28, we would elect to go with the Gradient Boosting method to predict loss amounts. Even though regression models can be easier to work with, the difference in RMSE between Gradient Boosting and the closest regression model (Regression with all variables) is fairly large. Our Tensor Flow model did not do a great job at predicting loss amounts based on its accuracy.

BINGO BONUS: We'll explore how these settings impacted the neural network results. As a baseline, here how the model performed with 200 epochs, relu activation function, and only 1 layer and no dropout:



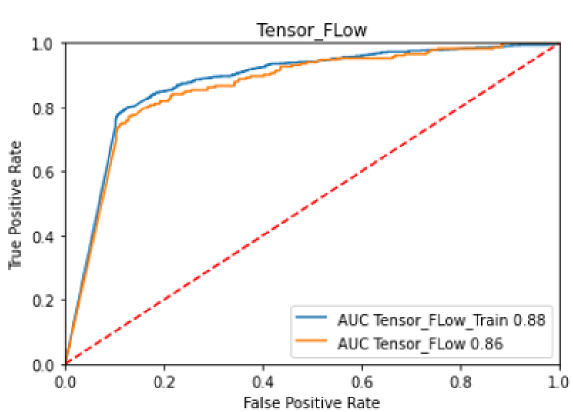

```
Tensor_FLow CLASSIFICATION ACCURACY Activation fn = relu
======
Tensor_FLow_Train  =  0.8280201342281879
Tensor_FLow  =  0.8271812080536913
```

<u>Adding hidden layer</u> – this seemed to smoothen our curve somewhat while also increasing the AUC for both datasets. Also saw significant increases in the accuracy:



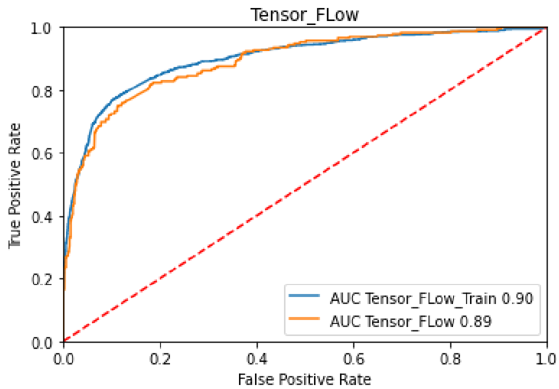

```
Tensor_FLow CLASSIFICATION ACCURACY Activation fn = relu
======
Tensor_FLow_Train  =  0.8781459731543624
Tensor_FLow  =  0.8733221476510067
------
```

<u>Adding dropout</u>: saw a very slight decrease in AUC for both datasets and small reduction in accuracy:



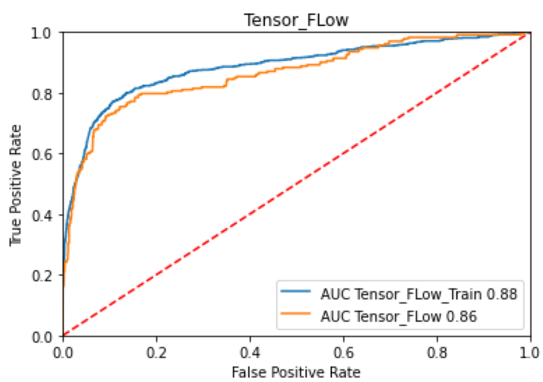

```
Tensor_FLow CLASSIFICATION ACCURACY Activation fn = relu
======
Tensor_FLow_Train  =  0.8687080536912751
Tensor_FLow  =  0.8624161073825504
------
```

<u>Increase epochs from 200 -> 800:</u> jump in AUC as well as accuracy.



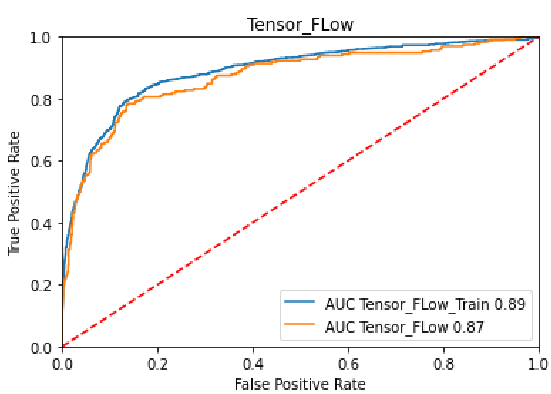

```
Tensor_FLow CLASSIFICATION ACCURACY Activation fn = relu
======
Tensor_FLow_Train  =  0.8873741610738255
Tensor_FLow  =  0.8791946308724832
------
```

<u>Selu Activation function:</u> dip in AUC as compared to relu function




```
Tensor_FLow CLASSIFICATION ACCURACY Activation fn = selu
======
Tensor_FLow_Train  =  0.881501677852349
Tensor_FLow  =  0.8825503355704698
------
```

<u>Sigmoid Activation function:</u> slightly less AUC and accuracy then relu:




```
Tensor_FLow CLASSIFICATION ACCURACY Activation fn = sigmoid
======
Tensor_FLow_Train  =  0.8733221476510067
Tensor_FLow  =  0.8741610738255033
------
```