MA751 Statistical Machine Learning Final Project

Improving Credit Card Fraudulent Detection Using a Neural Network Stack

Robert Xu, brx@bu.edu

I. Introduction

Ensemble methods, such as bagging, boosting, and stacking are designed to increase model performance through the process of combining predictions from a multitude of machine learning techniques. Different ensemble methods achieve different goals, bagging, or bootstrap aggregating, decreases model variance by sampling with replacement and averaging over many estimates. Boosting, decreases model bias by sequentially training weak learners and converting to a stronger learner. Stacking, on the other hand, takes advantage of the fact that some models perform better under certain situations or data samples. Stacking techniques aggregate the predictions of multiple regression or classification models, and feeds them as input to a final model, this final model is said to be "stacked" on top of the others.

In this project, I explore stacking techniques to better detect anomalous fraudulent credit card transactions. Building credit card fraud models is a commonplace task for most banks. In today's world, where there is a huge influx of online transactions, it is crucial that credit card companies detect fraudulent transactions so that they do not mistakenly charge their own customers, or else they need to reimburse the customer out of pocket, and have to spend legal resources to track down the fraudulent user. The challenge of building these models is that fraudulent transactions happen very infrequently, so a naïve model that predicts nothing is fraudulent will still have extremely high accuracy. Instead, we are more focused on reducing the number of false negative predictions, and so we will be trying to optimize our

model's recall rate instead. We aren't as focused on precision since making a false positive prediction just means the credit card customer would have to call the bank to clear things up, and no real losses are suffered in between. However, precision is still important since human resources still need to be spent to clear up these problems.

II. Data

To explore this problem, I gather a credit card transaction dataset from Kaggle. This dataset consists of European credit card transactions made in September 2013. Due to confidentiality issues, it contains only numerical input variables which are the result of a PCA transformation. These are features V1~V28. In addition, time and amount are provided. Feature 'Class' is our response variable of interest, it takes value in 1 if the transactionw as fraudulent, and 0 otherwise. Below we can see that the original dataset is highly unbalanced.

Out of the total 284, 807 transactions, only 492, or 0.172% of them are actually fraudulent. This means a baseline model of predicting all non fraudulent, is expected to have a 99.82% accuracy. However, as previously mentioned, optimizing model accuracy is not the main goal

here, since false negative predictions are going to be highly costly. To deal with this unbalanced dataset, I apply a simple under sampling technique, where I generate a balanced subset by including all fraudulent transactions, and proportional amount of normal transactions. For this project I under sampled so that 10% of transactions in my dataset are fraudulent. Finally, I shuffle then split the balanced dataset so that 80% is used for training, and 20% for evaluating the final model.

III. Model

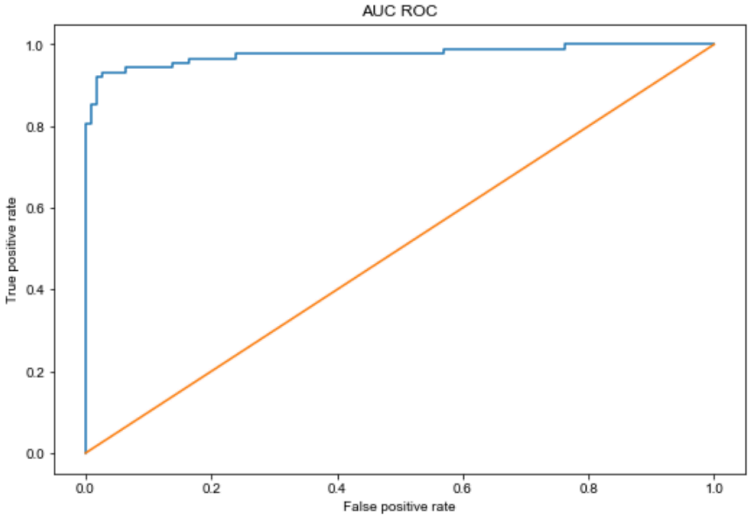
First, I fit a simple logistic regression model over the training data to get a sense of how a single model would perform. Its results can be seen below:

```
Training accuracy: 98.5264 %
Training AUC: 94.8535 %

##############################

Testing accuracy: 98.0691 %
Testing AUC: 94.5455 %
Testing precision score: 92.1925 %
Testing recall score: 89.4151 %
```

We see that a simple logistic regression model already yields extremely high training and testing accuracy. AUC is the area under the ROC curve. It is a measure of how good a model distinguishes false positives and false negatives. Precision is the fraction of total fraudulent transactions actually predicted as fraudulent. Recall is the fraction of all the fraudulent predictions that are actually fraudulent. ROC curve is given on the next page. The ROC curve plots true positive rate against false positive rate, it measures a binary classifier's ability to distinguish false positives and false negatives.

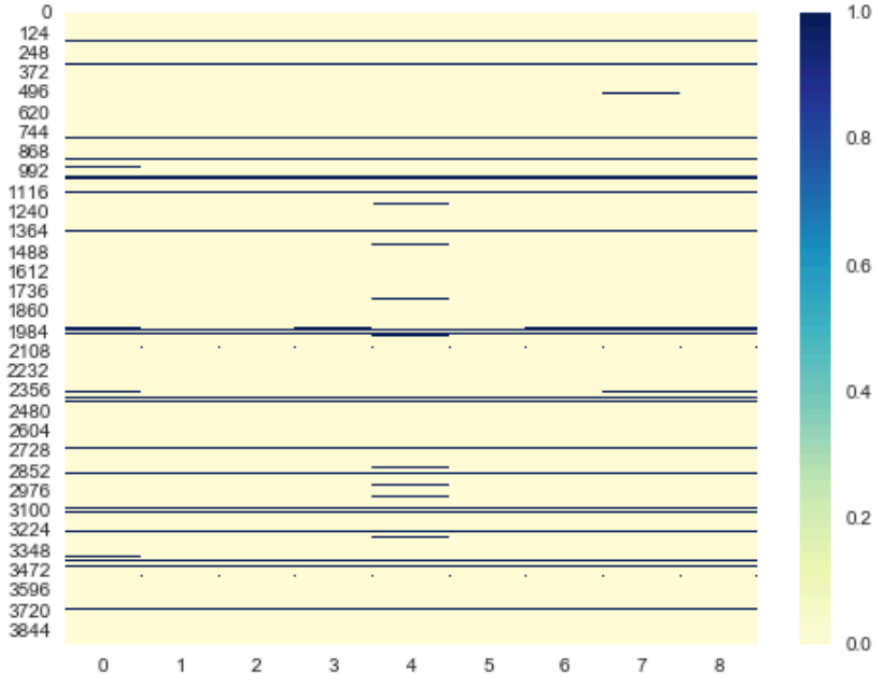


Next, we will construct our stacked model and see if it is able to improve upon the simple logistic regression model's recall score. To calculate the stacking features, I do the following steps:

1. Split the training data into 10 folds
2. Fit a level 1 base model on 9 folds, and make prediction for the remaining test fold.
3. Repeat this for all 10 folds, generating a prediction for each training instance.
4. Fit the base model on the entire training set, make predictions of the test set.
5. Repeat steps 2-4 for every base model you want to add to the final stacked model.

After we have generated all the stacking features (the base model predictions for training, testing points), combine them and use it as a final training set to train the stacked model. In this project, I consider the following baseline models: Naïve Bayes, logistic regression, LDA, KNN, decision trees, KMeans, SVC, RandomForest classifier, Gradient Boosting classifier, and AdaBoost classifier. Training all these models on the entire original dataset was extremely costly, especially since I use 10-folds to generate the stacking features (some models took 20 minutes to train). Using our balanced dataset to train these base models greatly reduced

computational cost. Below is a heatmap of our base model predictions: (dark blue indicates fraud)



In the heatmap above, if a single row is a solid color, it means that all baseline models make the same prediction. We can see from above that not all base models generate the same prediction. In these base models, some of them have low precision but high recall, others have high accuracy but lower recall and precision etc., and thus the motivation to use a second layer model to aggregate all these models. Initially, I fit a logistic regression model onto the stacking features. Results are as follows:

```
Training accuracy: 98.5264 %
Training AUC: 94.8535 %
Training precision: 86.2069 %
Training recall: 93.2496 %

##############################

Testing accuracy: 98.0691 %
Testing AUC: 94.5455 %
Testing precision: 86.0126 %
Testing recall: 92.4145 %
```

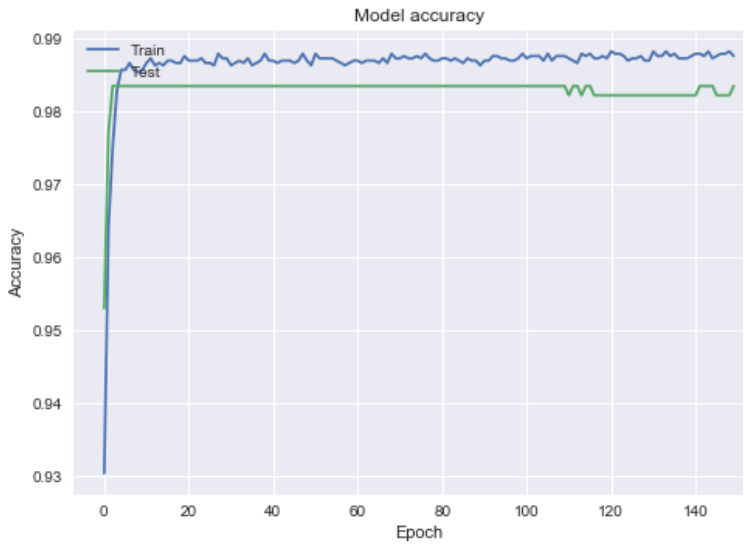
We can see that the stacked model has similar model accuracy to the previous single logistic regression model, slightly improved testing recall, lower precision, but an improvement in recall, our target of interest.

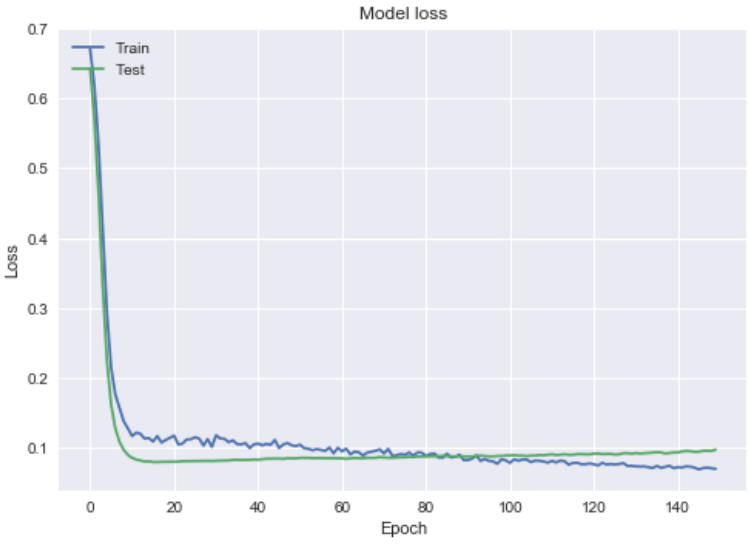
IV. Aggregating with Neural Network

The above stacking model only has slight improvements against our single logistic regression model. Next, I explore aggregating our stacking-features with a neural network to see if we can further improve our model's recall rate. I construct a densely connected 4 layer neural network, where the final layer is activated by a sigmoid function so that it can make a classification prediction. The network is trained over 100 epochs with the Adam optimizer, using 30% of the data for validation. Results for stacking with a neural network are below:

```
This model has Training accuracy: 0.9870 ; and testing accuracy 0.9817
               Training auc: 0.9875 ; and testing auc 0.9769
               Training precision: 0.8753 ; and testing precision 0.8696
               Training recall: 0.9880 ; and testing recall 0.9709
```

The resulting 97% recall is quite an improvement over using a logistic regression aggregated stacking model. We check our training history to make sure the network has an honest fit:





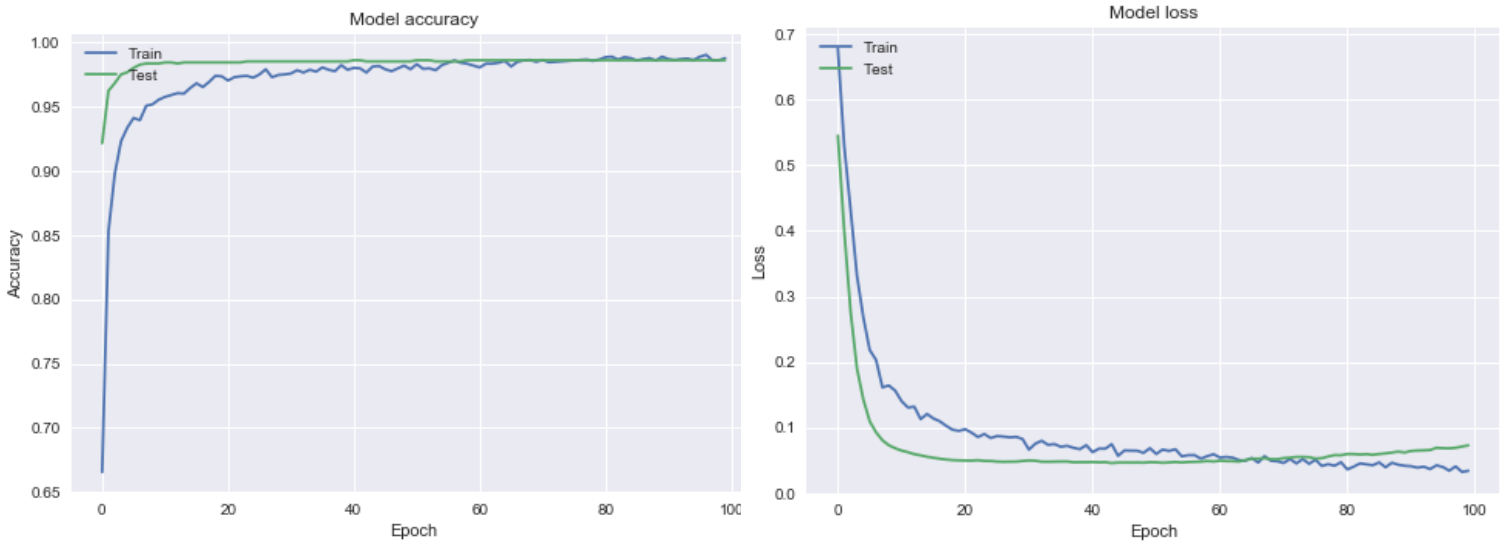
This looks like a reasonable fit, training and validation loss are within a reasonable magnitude throughout the training process. We do see that testing loss initially starts out lower than the training loss, this is due to adding dropout layers in the neural network to control model complexity. Dropout refers to randomly ignoring neurons during the training process, so that their weights don't get updated in a certain epoch, reducing complexity. Dropout is usually activated while trying but deactivated when evaluating the validation set, causing training loss to be higher at start.

Finally, to take advantage of neural networks ability to handle higher-dimensional features, I pass the stacking features along with the original dataset used to generate those features into another neural network. This allows for the network to perhaps learn additional non-linear features in the original dataset that our stacking features were not able to capture.

Results for this model are below:

```
This model has Training accuracy: 0.9898 ; and testing accuracy 0.9827
               Training auc: 0.9918 ; and testing auc 0.9860
               Training precision: 0.8992 ; and testing precision 0.8609
               Training recall: 0.9941 ; and testing recall 0.9900
```

We see this neural network outperforms the previous one trained only on stacking features and achieves 99% testing recall. Again, check training history to make sure we have an honest fit:



Testing accuracy and training loss initially start off larger than their counterparts, again, this is due to the dropout layers in the network to control complexity. Other than that, the network seems to have a reasonable fit. This final model gives us an auc score of 98.46%

V. Conclusion

In this project, I generate 10 baseline models and stack them using a neural network to achieved increased performance in recall when it comes to credit card fraudulent detection. We can see our initial single logistic regression model gave us pretty high accuracy already, but due to the nature of credit card fraudulent transactions, we are more interested in optimizing our model's recall. Recall is the fraction of all transactions predicted fraudulent that are actually true positives. Higher recall means a lower false positive rate, and making false positive predictions are extremely costly for credit card companies. We found that a single logistic regression starts us off with a 89% recall score, while our neural network stack ultimately leads to a 99% recall rate.

VI. Future Work

To focus more on stacking for this project, I simply created a rebalanced dataset in which 10% of transactions are fraudulent. More research can be done on various under/over sampling techniques to deal with these highly unbalanced datasets to increase model performance. In addition, a more robust technique to choose which base models to actually include in the final stacked model can be developed through cross fold validation, although this will be extremely costly.