**Avant’s Default Loan Prediction Challenge Write-up**

**Part 1**

I decided to make this a binary classification problem. From the problem prompt, I interpreted this project as trying to predict if a loan applicant is going to fully pay off a loan or default. So, I considered the positive label, i.e. dependent variable, for this project as “Default” in the loan\_status column in data.csv and “Fully Paid” as the negative label. For the examples in data.csv where loan\_status is “Current”, I did not consider these examples when constructing model labels and features because we do not know yet if these loans will result in defaulting or be fully paid yet. Because of this filtering, the data went from 80,000 to 17,571 examples.

**Part 2**

Excluded these

For

**Bonus Part**

**Part 3**

***Your choice of an appropriate classification method that you think would perform best on this dataset and explain your choice.***

Because of the diversity of features available in terms of real value, binary, integers, etc. and that this is an unbalanced dataset (34% positive / 66% negative), an ensemble train classifier is what I would use. Ensemble My default is to use a random forest model in these situations. However, it should be possible to train a boosted tree method in a reasonable amount of time because the dataset is relatively small.

***Please describe any additional data challenges you may have to deal with for each choice of different modeling methodologies.***

Parameter tuning is

Overfitting is possible here. If I chose a logistic regression or neural network approach, I would need to worry about parameter regar

Normaliz

The dataset is unbalanced

Might not have expressive features

***Please list the inputs/variables that will go into your model after any variable selection that you deem necessary.***

The features extracted from data.csv and stored in model\_features.csv are ready to be inputted straight into a model. I would consider doing PCA or other variable elimination techniques to reduce the dimensionality of the problem.

I would also look for the most meaningful features out the extracted features in model\_features.csv. To do this, I would train classifier models on each single feature and evaluate these single feature model’s performances. When a low performing model resulted from a particular feature, I would remove it from the feature set.

Data normalization is not typically needed for tree models. However, if I switched methods to a logistic regression method for example, I would normalize the data so all features have a zero mean and 1 std-dev.

***And please discuss how would you validate the model that you would build.***

A lot of model tuning will be required to train a decent classifying model. Because of this, avoiding over fitting here is key. Given that I extracted 21 features here and have 17,571 examples to train an ensemble tree classifier model, I am not too worried about running out examples so I would probably not used a k-fold cross-validation approach. Rather, I would separate the data into a training set, validation set, and test set. I would train off the training set and then evaluate on the validation set. Based off the results, I would readjust my model’s parameters and continue until I maximize my evaluation metrics using validation data. At which point, I would evaluate using the testing set of data to get a final assessment on my trained model.

The model evaluation metrics I would use in this case are accuracy, recall, and precision. I would consider the ROC area metric if I thought my model was not actually learning anything other than picking the negative label for all examples because this is an unbalanced dataset (34% positive / 66% negative).

**Additional Note**

I actually had time to quickly train a classifier. Once you have the features and labels extracted, it is not much more work to train a classifier if you skip variable selection. So, I just wanted to see how things stand with this dataset using a model with default parameters. I went with a GradientBoostingClassifier model classifier, Sklearn’s equivalent to XGBoost. Using a 20% validation set, the evaluation metrics were:

- Validation Accuracy: 0.684210526316

- Validation Precision: 0.582627118644

- Validation Recall: 0.231481481481

The null hypothesis point for accuracy is around 66% (34% of the data is positive) for this dataset. My current accuracy is only slightly better than. So, I have a lot of modeling parameter tuning and variable selection needed to increase performance. Might be a simple fixe of dropping the likelihood of a positive label. I can evaluate this idea using a gain or lift chart.