DANA LITOVSKY ALCALA Machine Learning, report 3

In the previous report, I provided a summary analysis of the loan and debt delinquency data. In this report, I will discuss more advanced machine learning classifiers I applied to the data to understand what factors make individuals more likely to not be able to pay debt.

Recall the attributes of the data:

```
"SeriousDlqin2yrs', 'RevolvingUtilizationOfUnsecuredLines',
  'age', 'NumberOfTime30-59DaysPastDueNotWorse', 'DebtRatio',
  'MonthlyIncome', 'NumberOfOpenCreditLinesAndLoans',
  'NumberOfTimes90DaysLate', 'NumberRealEstateLoansOrLines',
  'NumberOfTime60-89DaysPastDueNotWorse', 'NumberOfDependents"
```

Where `SeriousDlqin2yrs` is 1 if an individual experienced 90 days past due delinquency, or worse, and 0 otherwise..

Recall that missing values pose a challenge to work with data. Monthly Income and Number of dependents have many missing values (other features have no missing values):

```
MonthlyIncome                                 29731
NumberOfDependents                             3924
```

I improved my previous imputation methodology by imputing training Income data with its median and training dependents data with the mode. I applied these training statistics to impute test data.

Creating and Evaluating Models to predict loan default

To predict which individuals would be more likely to default on debt, I employ six models (including SVM, Logistic Regression, Decision Trees, etc) and various parameters to pick the best model-parameter mix. I use the Area Under the Curve metric to determine which is the best model. The AUC is a metric that calculates the area under the precision-recall curve, providing a metric for overall performance.

AUC is a good metric for this data because I care about overall precision-recall performance. It's important to remember that the data is unbalanced in the prediction attribute class. Most of the data points are classified as serious delinquency in the last two years = 0, with only 10% of data = 1. So, using accuracy is not a good metric because if I assigned all data points to class = 0, I'd already get 90% accuracy or more. This data's base metric is about 93%.

I also considered using F1 score as a metric. The challenge with F1 score is that it is calculated with one precision-recall data point, and does not use thresholds. Thus, it would be challenging to compare F1 scores across models since I would only be picking precision-recall on one point of the curve. If I cared about precision at a certain point of recall, or recall at a certain precision level, then F1 would be an appropriate metric. Yet, AUC is a metric that takes into account precision-recall points at all thresholds (from 0 to 1 threshold along the curve) and is appropriate when looking for overall performance.

Selecting a Best Model

I make the model selection process and the estimation of the AUC more robust by employing k-fold cross validation, which splits the data into a number of folds for train and test sets. At each fold, I impute missing data, create features (in this case, age and income bins, the log of monthly income, and a scaled monthly income), and run the model with its parameters. I compute the average AUC over all the folds to get a more robust AUC metric for the model-parameter combination. I chose n = 3 folds due to the complexity and time-constraint of running a dataset with more than 100,000 observations. Running k fold cross validation across all models and parameters took many hours!

Here's a sample table of some of the 198 results for various model-parameter combinations.

MODEL	**PARAMETERS**	**AVERAGE AUC**	**AUC STD DEV**
LOGISTIC REGRESSION	C=5, class_weight=None, dual=False, fit_intercept=True, intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=-1, penalty='l2', random_state=0, solver='liblinear', tol=0.0001, verbose=0, warm_start=False	0.19372	0.001149
DECISION TREES	DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=1, max_features='sqrt', max_leaf_nodes=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0resort=False, random_state=0, splitter='best')	0.06016	0.06230
RANDOM FORESTS	bootstrap=True, class_weight=None, criterion='gini', max_depth=75, max_features='log2', max_leaf_nodes=None, min_samples_leaf=1, min_samples_split=10, min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=-1,oob_score=False, random_state=0, verbose=0, warm_start=False	0.253487	0.022690

SVM	C=1, class_weight=None, dual=False, fit_intercept=True, intercept_scaling=1, loss='squared_hinge', max_iter=1000, multi_class='ovr', penalty='l1', random_state=0, tol=0.0001, verbose=0	0.1165454	0.0140220

After looping over all model-parameter combination, the model with the best average AUC is the Random Forest with parameters:

bootstrap=True, class_weight=None, criterion='gini', max_depth=75, max_features='log2', max_leaf_nodes=None, min_samples_leaf=1, min_samples_split=10, min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=-1, oob_score=False, random_state=0, verbose=0, warm_start=False

These parameters stop the random forest from creating trees that have more depth than 75, preventing overfitting, and make internal nodes split at minimum 10 samples.

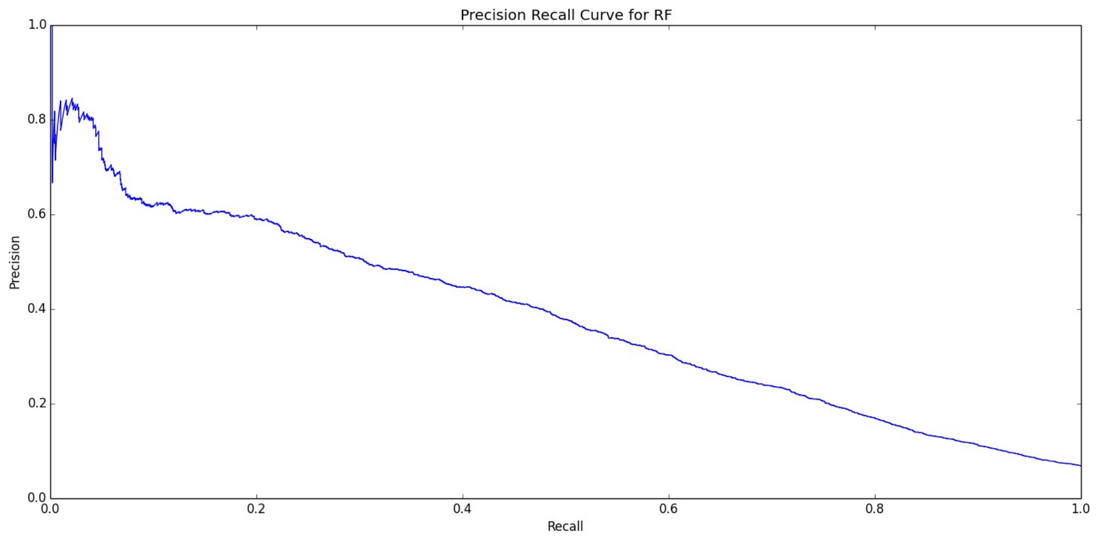
I am now interested in how this best model would perform on the entire dataset. I split the data into 80-20 train-test set to calculate precision, accuracy, recall, AUC and plot precision-recall curve for the Random Forest.

In particular, I wanted to see if I could improve the accuracy score I attained previously by running logistic regression (PA2).
The logistic regression returned an accuracy score of: **0.931866666667**

By running the Random Forest on the same train-test split, I attained an accuracy score of **0.9353000**. This result is not much of an improvement, but recall that due to the data's imbalance in label class, base accuracy is about 93%, and accuracy is not a great measure to use in this case.

Instead, I also calculated precision, recall, and AUC on this final train-test split.

PRECISION:	0.606007
RECALL:	0.16642
AUC:	0.380523



PRECISION RECALL CURVE FOR RANDOM FOREST

Recall that precision and recall are calculated only at a certain threshold, such as 50%. However, looking at AUC shows that this Random Forest is able to have an overall performance across all precision-recall threshold of about 38%.