Author : Emily Webber

1. Executive Summary

Delinquent loans can be difficult to predict. For a bank with a large set of customers, estimating the likelihood that someone will default is critical to a healthy business model. Because there are so many accounts to keep track of, it's more important for a bank to accurately predict a case that's likely to become delinquent, rather than detecting every possible case showing any signs of delinquency. This means that precision in the top 5% of the sample is more important than overall model performance, so we chose to pick our model based on precision.

Using data from Kaggle we've been able to accurately predict 60% of the most-likely cases, which represents a significant improvement from a baseline fifty percent estimation. This means that we were able to detect 60% of the cases that were found to be delinquent after the fact. A number of models were able to reach precision levels in the high 50%, but only AdaBoost and RandomForest did this consistently.

- Best models for AUC
  - AdaBoost, precision = 0.604930, AUC = 0.8945275
- Best model for Precision at top 5%
  - Random Forest, precision = 0.6082611, AUC = 0.89055
- Faster Models
  - Logistic Regression
  - Decision Tree
  - Gaussian Naive Bayes
  - K Nearest Neighbors
- Slower Models
  - Random Forests
  - Extra Trees
  - AdaBoost
  - SVM

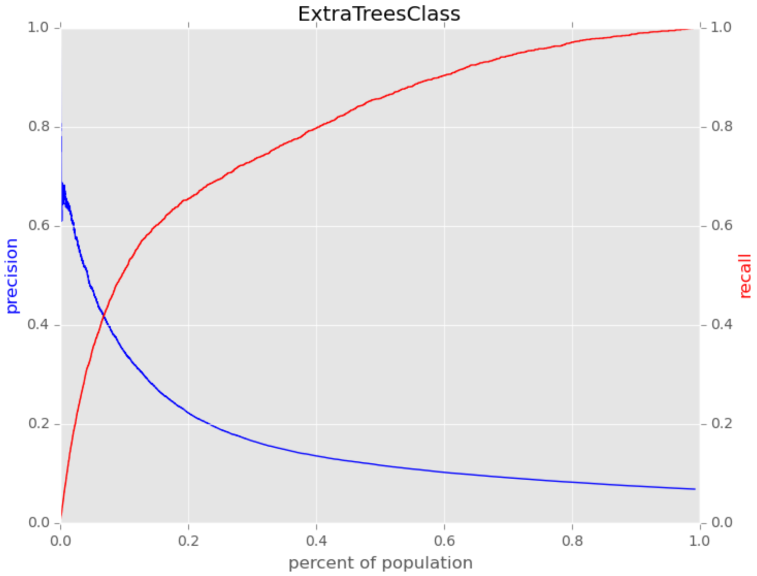
2. Discussion

The models with the most consistently well-perfomring precision-recall curves are AdaBoost, Random Forest, and Extra Trees. While for the first two models also indicates high precision at the top 5%, in the case of Extra Trees this almost never reaches a very high level of precision.

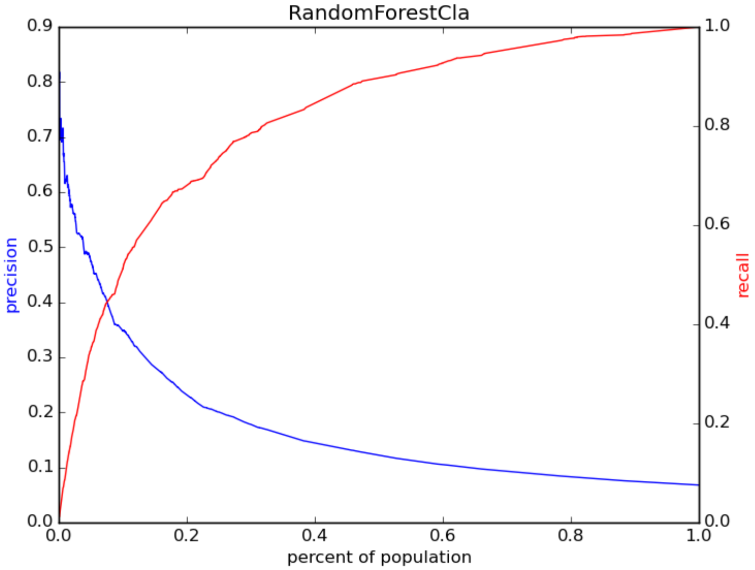
A few models regularly have very poorly performing precision-recall curves, such as K Nearest Neighbor, Decision Tree, and Logistic Regression. The plain Naive Bayes classifier that was set without any changed parameters did surprisingly well; it's possible that further specification could yield even better results.

3. Figures

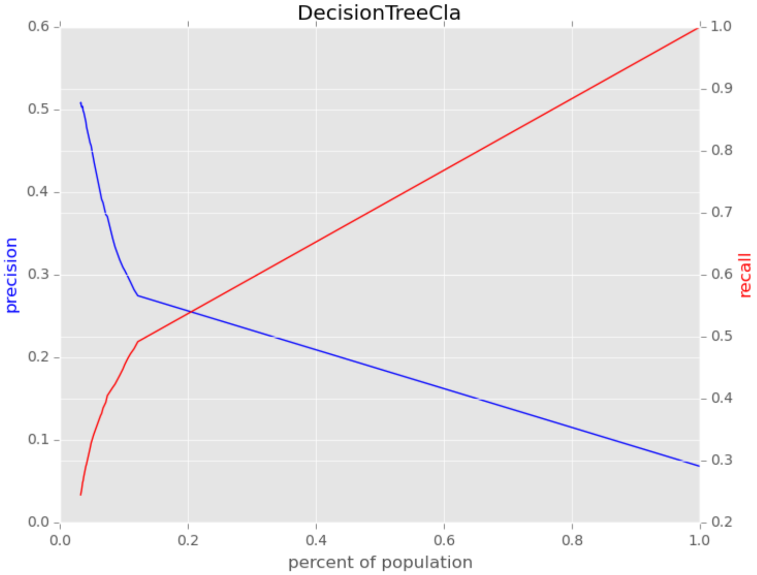
- A note on the following graphs: the figures themselves are estimations, but the written models are exact.



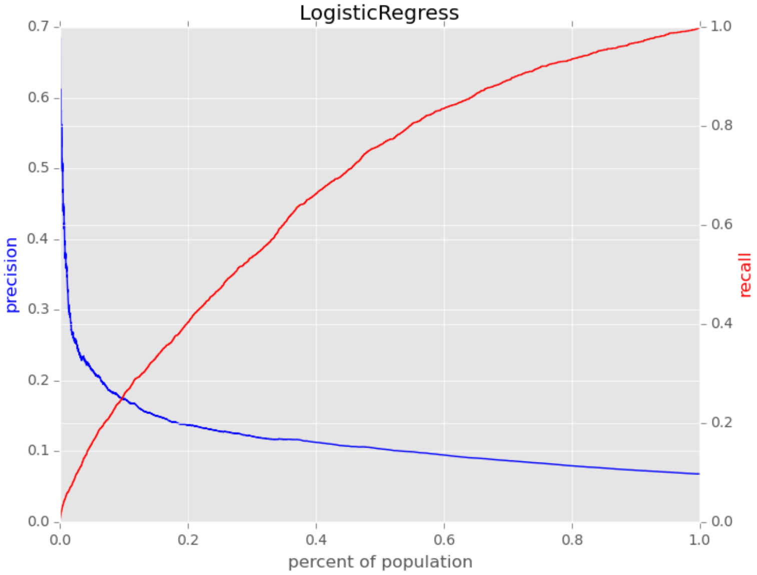
```
"ExtraTreesClassifier(bootstrap=False, class_weight=None, criterion='gini',
 max_depth=10, max_features='sqrt', max_leaf_nodes=None,
 min_samples_leaf=1, min_samples_split=5,
 min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=-1,
 oob_score=False, random_state=None, verbose=0, warm_start=False)":
 {'auc': 0.8133091023588066, 'precision': 0.47501665556295802}
```



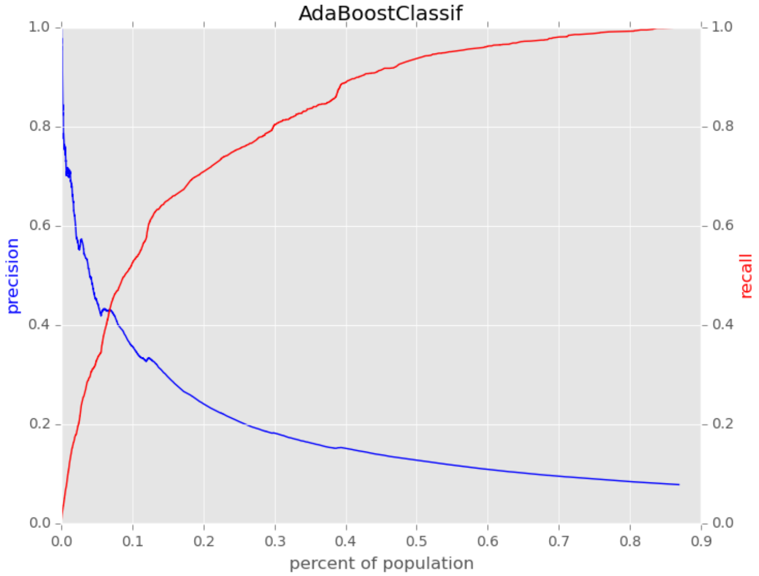
```
( RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=10, max_features='log2', max_leaf_nodes=None,
            min_samples_leaf=1, min_samples_split=5,
            min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=-1,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False))]
```



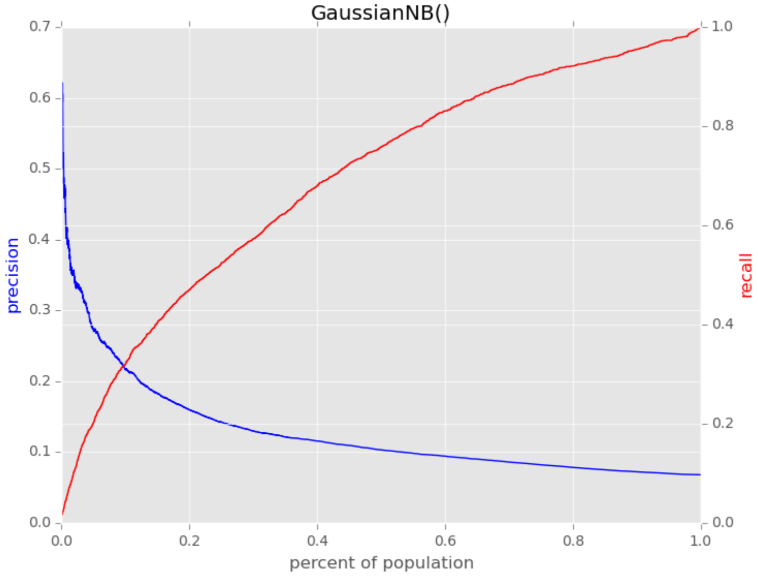
```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=100,
            max_features='log2', max_leaf_nodes=None, min_samples_leaf=1,
            min_samples_split=10, min_weight_fraction_leaf=0.0,
            presort=False, random_state=None, splitter='best'):
             {'auc': 0.86962081881182784, 'precision': 0.53400503778337527}
```



```
LogisticRegression(C=10, class_weight=None, dual=False, fit_intercept=True,
        intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
        penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
        verbose=0, warm_start=False):
         {'auc': 0.69113777467252624, 'precision': 0.21585609593604263}
```



```
"AdaBoostClassifier(algorithm='SAMME.R', base_estimator=DecisionTreeClassifier(class_weigh
max_leaf_nodes=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0
learning_rate=1.0, n_estimators=1000, random_state=None)":
{'auc': 0.89452758292483991, 'precision': 0.60493004663557626}}
```



```
'GaussianNB()': {'auc': 0.70757182149756548, 'precision': 0.27448367754830111}
```