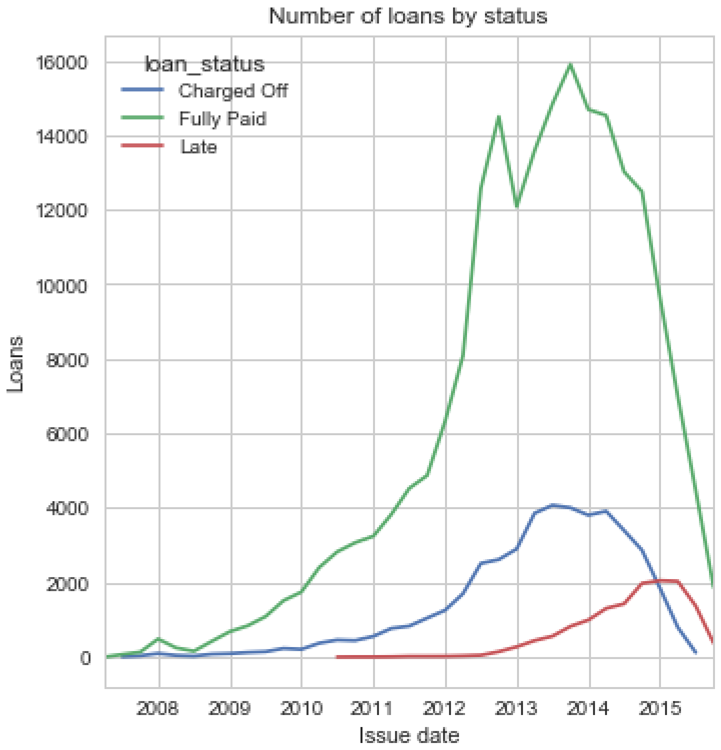
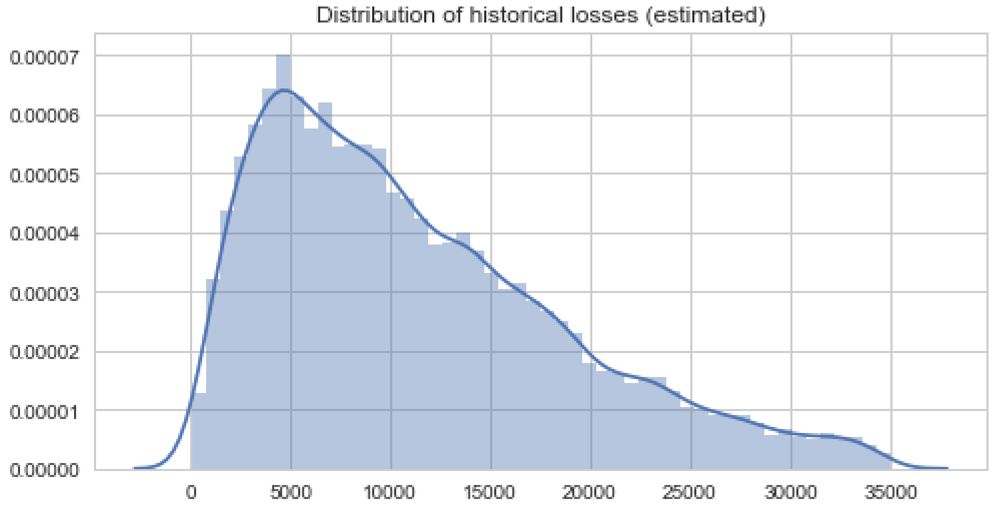
Predicting Credit Defaults

A prototype model using advanced analytics

Outline

1. An overview of the data: Historical defaults





- ➢ The historical default rate (of terminated loans) has been around 18%
- ➢ Using median recovery rate this amount to approximately 0.5bn in losses

2.1 Credit default modeling: A supervised learning problem

Training data

trained model

$$y = f(x)$$

default info from training data

features (data on borrower, loan)

Test data

trained model

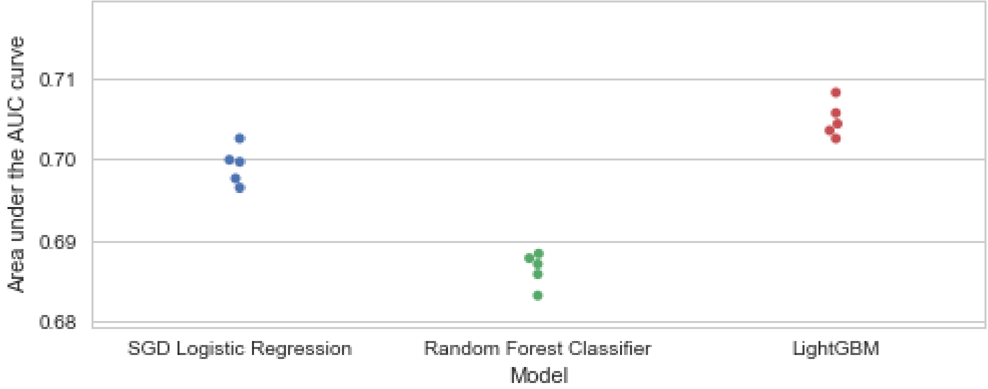
$$\hat{y} = f(x)$$

model prediction

new unseen data (data on borrower, loan)

2.2 Credit default modeling: Finding the ‘f()’

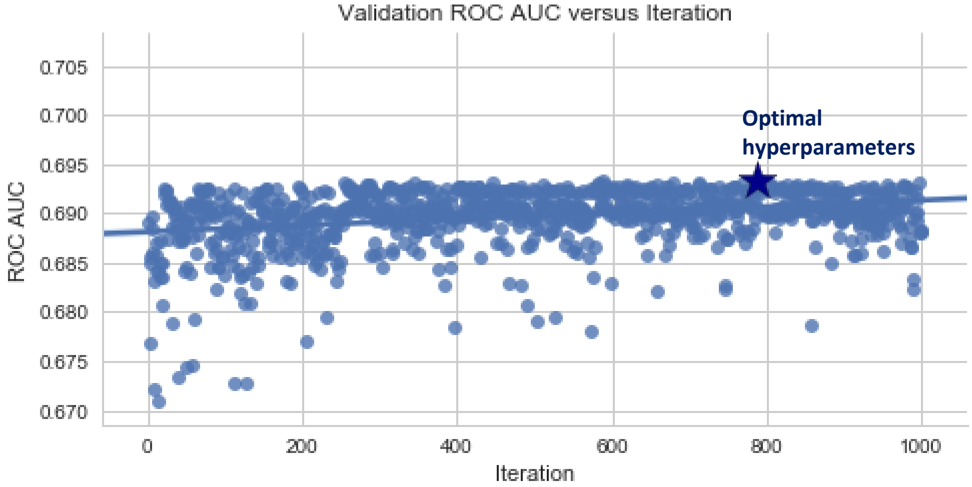
Step 1: GridSearchCV with 3 different algorithms



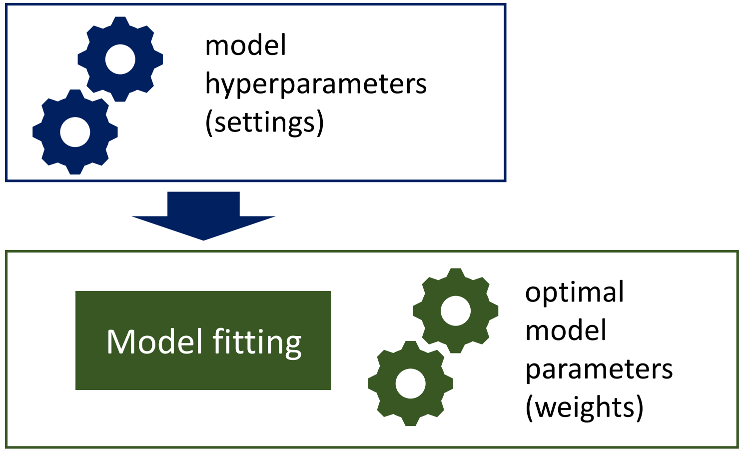
Cross-validation = 5-kfold

iteration 1	Test	train	train	train	train
iteration 2	train	Test	train	train	train
iteration 3	train	train	Test	train	train
iteration 4	train	train	train	Test	train
iteration 5	train	train	train	train	Test

Step 2: Bayesian hyperparameter optimization

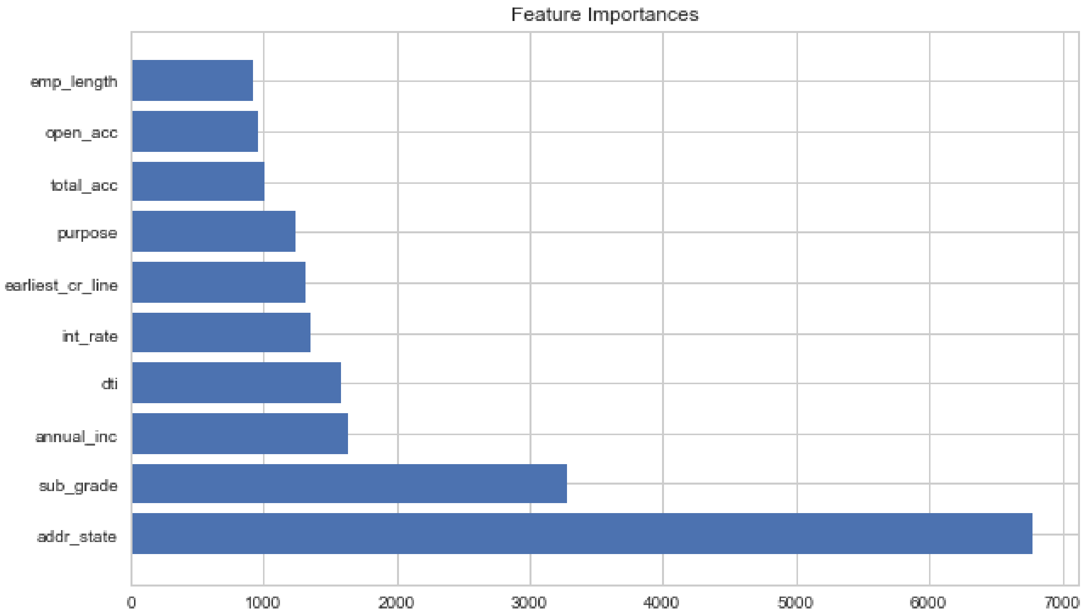


Machine learning process

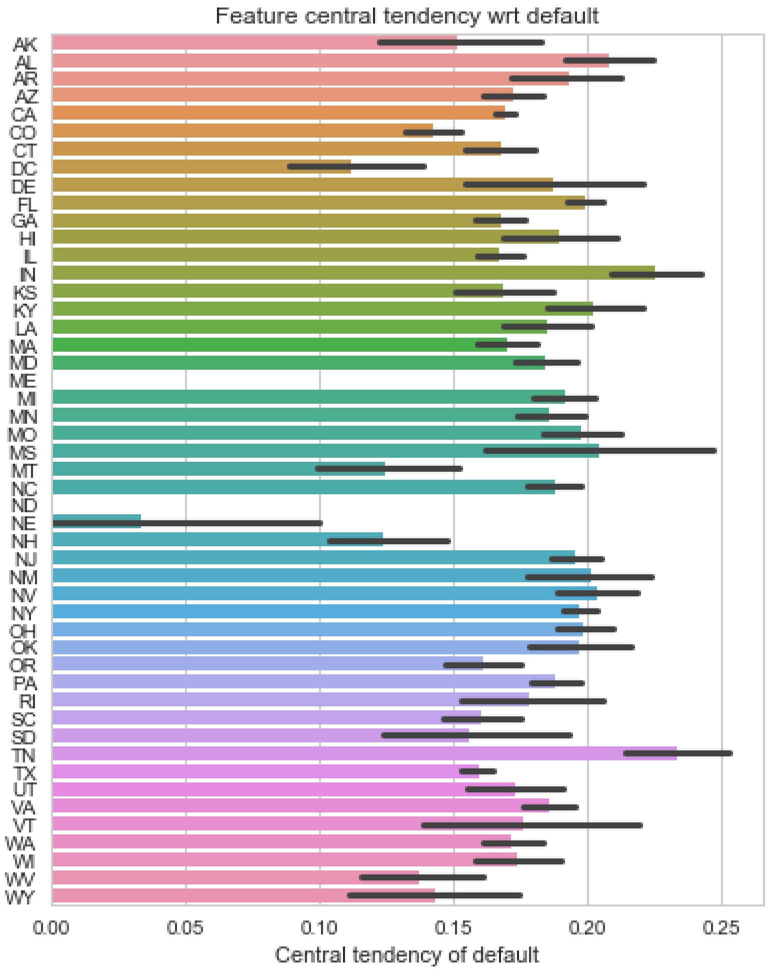


2.3 Credit default modeling: Factors driving default

Feature importances

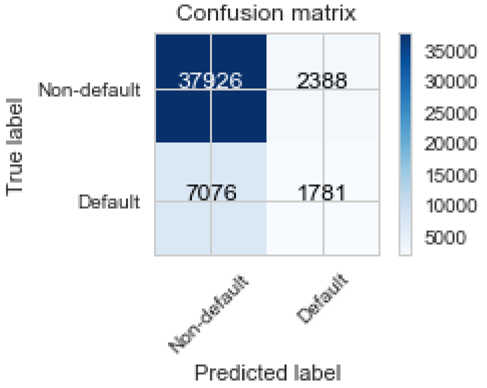


Default by US state ('addr_state')

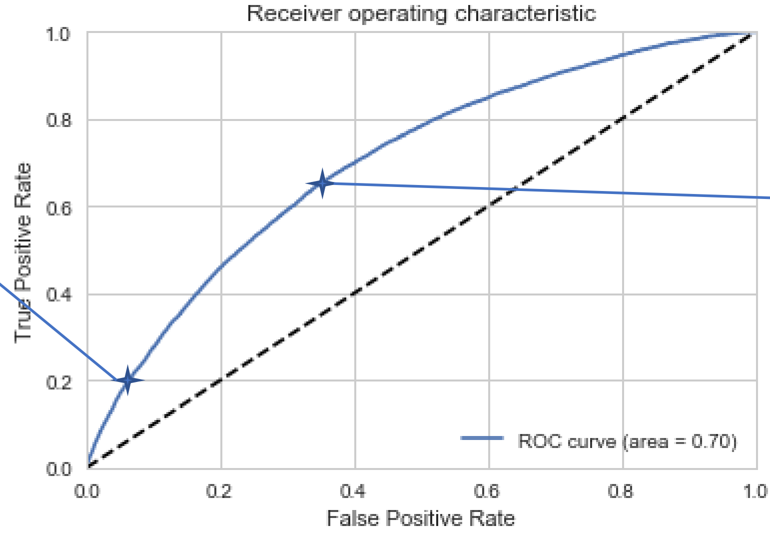


3. Results on test data – threshold to set the trade-off between rightly predicting defaults vs. wrongly predicting non-defaults

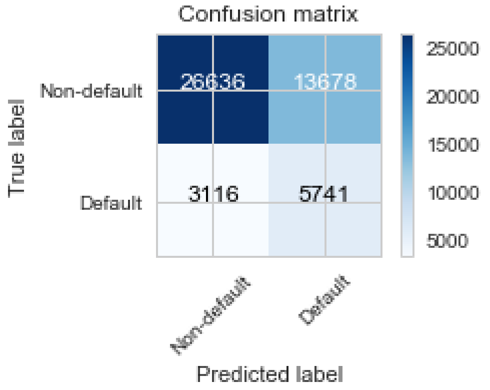
Threshold @ 0.70



TPR = 20%
FPR = 6%



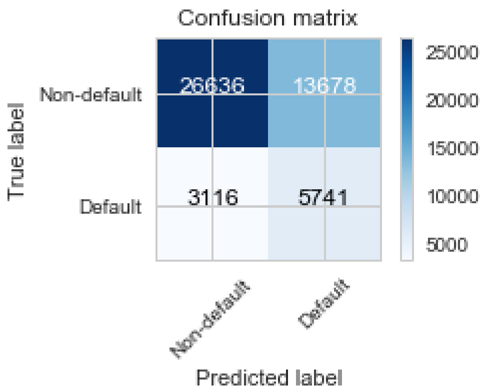
Threshold @ 0.50



TPR = 65%
FPR = 34%

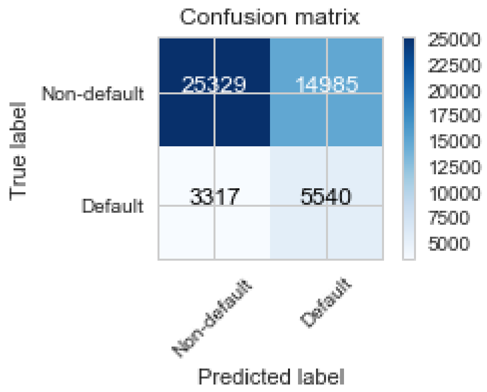
4. Comparing machine learning model with a model based on sub-grades shows that advanced analytics adds business value

Machine learning model with threshold @ 0.50



TPR = 65%
FPR = 34%

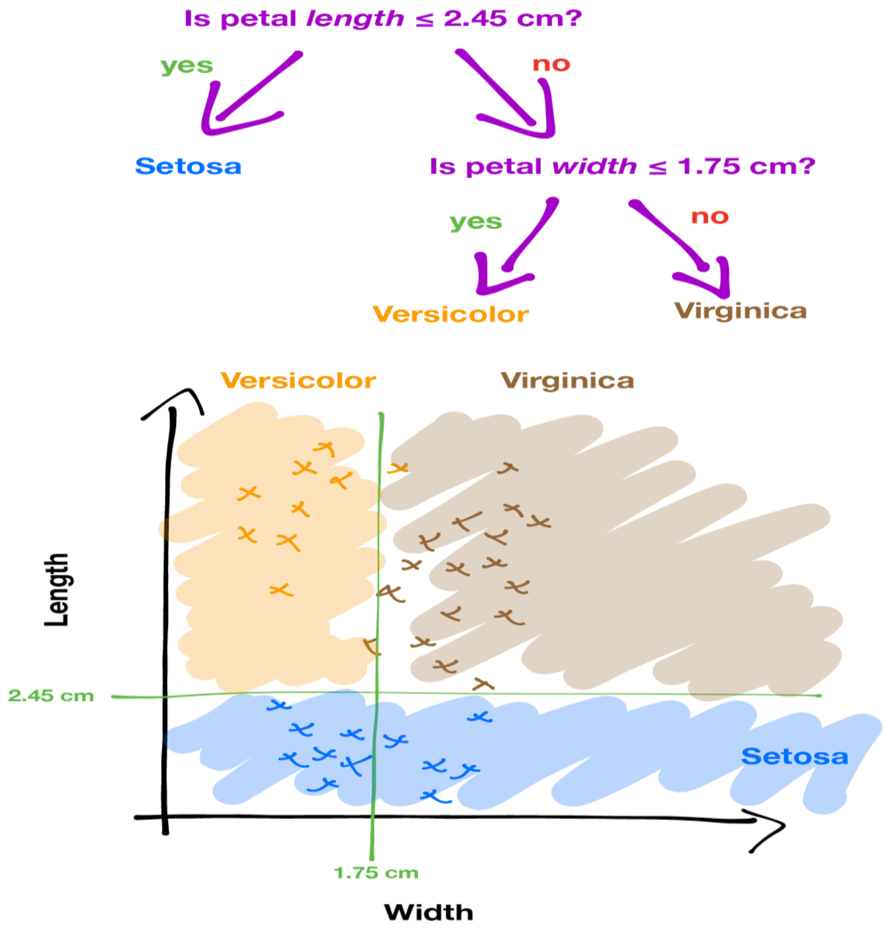
Sub-grades model with threshold @ 0.50



TPR = 62%
FPR = 37%

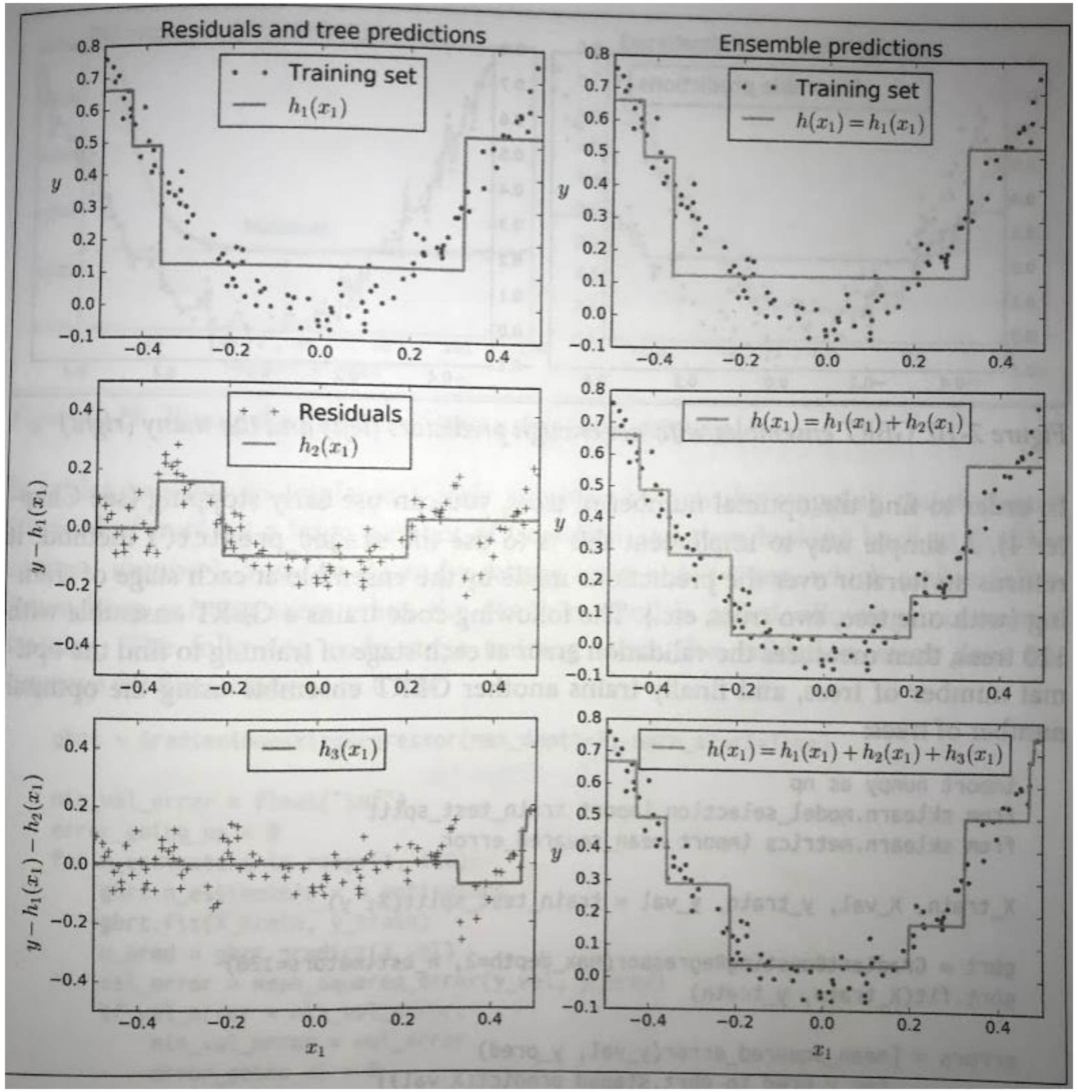
- The sub-grades model has 201 more false negatives than the ML model (losses from 201 loans that could be avoided)
- But equally important, the sub-grade model would wrongly label 1307 more non-defaulters – i.e. foregoing business opportunity from 1307 customers

Appendix I: A quick review of decision trees



- Starts at root node (one feature) and finds the best split to max e.g. gini impurity
- 'Builds the tree' by progressing through each feature, until certain hyperparameters are reached (max depth, minimum samples in leaves)
- **Pros**: easy to interpret and great as building stone in ensembles
- **Cons**: overfits rather easily, and requires data to be orthogonally divisible

Appendix II: (Stochastic) Gradient Boosting



- **Boosting**: ensemble combining several weak learners into a strong learner
- Train learners **sequentially**
- **Gradient boosting**: train next learner on residuals of former learner
- *Learning rate* (or shrinkage factor) to decide how much each weak learner will contribute to the final model
- Optimal number of trees can be found using *early stopping* decided by when the validation score has stopped improving
- To decorrelate the learners, one can do subsampling of the training data – a technique called **Stochastic Gradient Boosting**