# Classifier Comparisons On Credit Approval Prediction

Zhoutong Fu; Zhedi Liu

## I. Introduction

Inspired by the paper of *Simplifying Decision Trees* by J.R. Quinlan and the book *C4.5: programs for machine learning* by J.R. Quinlan and Morgan Kaufmann where they test the very traditional pruned decision tree models on credit approval data set, we want to re-exam the data set used in *Simplifying Decision Trees* and build advanced models to increase the accuracy. No surprise that all of the models we built beat the bench mark (12.9%, the best result from Quinlans paper), among which SVM with polynomial kernel and random forest achieve the best result (9.18%).

## II. Data Exploration

### A. Data Format

Data concerns with credit card applications and is collected from http://archive.ics.uci.edu/ml/datasets/Credit+Approval. There are 690 observations (37 of which have missing values ) with 15 features (among which 6 are real valued and the rest are categorical) and 1 class attributes. For confidential reasons, all attribute names and (categorical) values have been transformed to meaningless symbols.

### B. Data Imputation

Several methods have been applied for data imputation:
1. Deletion: delete any observations with missing values.
2. Simple imputation: impute missing value of numerical variable with its mean and missing value of categorical variable with any of its categories.
3. Random regression imputation: run a multiple regression of the missing variable using the remaining variables as predictors.
4. Iterative multiple imputation: keep running random regression imputations on all missing variables until their values converge.
Conclusion: although advanced methods such as random regression and iterative multiple imputation usually work better in most cases than simple imputation methods, in this case however, we haven't observed any significant differences between them when building models to get estimated error. Hence, deletions are used for observations with any missing values as to reduce bias.

### C. Data Visualization

It would be extremely useful and could bring in more insights about modeling if data could be visualized before applying any further operations. Fig. 1 shows the scatter plot of numerical variables from the data set where no particular patterns could be observed between any pairs which suggests no apparent correlations and hence we need further investigations on the data set.

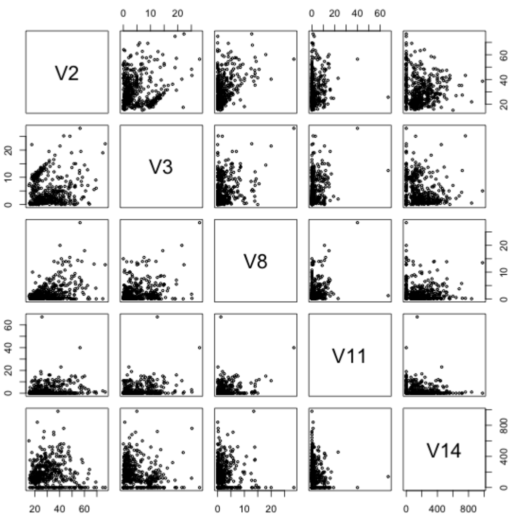

Fig. 1. Scatter plot of numerical variables

Moreover, principal component analysis (PCA) is applied to detect the main directions of data variance. Before performing PCA, we first transform all categorical variables into indicator variables (0-1 valued) because PCA could only work on numerical inputs. Fig. 2 is the biplot of first and second principle components and their associations with each variable and data point. Upper axis represents the loading scores of input variable while lower axis shows the score of data points on the first two components.

From PCA no dominating directions could be used to explain the variance because the first three principle components have merely explained about 30 percent of total variance and in the biplot the directions of variables vary quite a lot and have very little information in common. Opposite variable directions are also observed which may suggest that some noises exist in the data set.

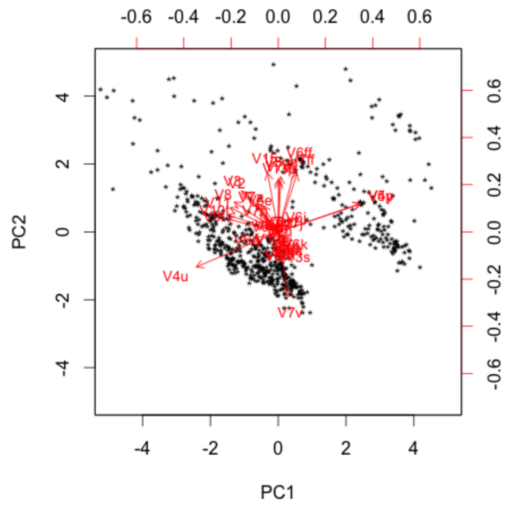

Fig. 2. biplot of principal component analysis

## III. Features and Models

### A. Features Extraction and Selection

Linear dependency and multi-collinearity are detected among original features via variance inflation factors (VIF). For linear models we delete highly correlated variables and combine several related variables into one compounded variable. We have tried to add top principle components to expand the feature space but it turns out to have no improvement on the model accuracy. In support vector machines, features are further expanded via linear kernel, polynomial kernel, radial kernel and sigmoid kernel.

### B. Modeling

A good variety of models are tested in this paper and for detailed results please refer to Table I where Error1 denote training classification error and Error2 denote test classification error.

(i) Linear Models
We start with logistic regression and linear discriminant analysis. After applying the above feature selection methods, both of them give surprisingly good results.

(ii) Support Vector Machine
A very natural expansion from linear models is support vector machines where kernels have been applied to expand feature space. Here we have tried linear kernel, polynomial kernel, radial kernel and sigmoid kernel. Linear kernel has the form

$$K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij}x_{i'j}$$

which is similar to linear models above and hence we expect their performance to be similar. However, SVM with linear kernel usually beat linear models by faster convergence which is extremely useful when the data size is relatively small. Polynomial kernel

$$K(x_i, x_{i'}) = (1 + \sum_{j=1}^{p} x_{ij}x_{i'j})^d$$

provides a much more flexible decision boundary by introducing polynomials with an intercept. Radial kernel has the form

$$K(x_i, x_{i'}) = exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2)$$

which focus on local behavior of data and only nearby training observations could have effect on modeling. At last we use sigmoid kernel which has the form:

$$K(x_i, x_{i'}) = tanh(\gamma \sum_{j=1}^{p} x_{ij}x_{i'j} + c)$$

(iii) Tree Models
Besides linear, nonlinear and local models, tree models are also tested here to provide a very different view of data splitting and modeling.

Random forest is intended to improve traditional decision tree models by introducing boostrap aggregation and then further improve it by decorrelating the trees. A key parameter is the number of predictors to be chosen as split candidates and we will calibrate this parameter by creating validation set.

Another widely used tree-related model is boosting classification tree where trees are grown sequentially and each new tree is based on information of previously grown trees. The key parameter here is the number of trees to grow and we will apply validation and cross validation to select optimal parameter value.

(iv) Neural Networks
Deep learning methods usually work well on data with hidden complexities and here we tried neural networks with only one hidden layer.

## IV. Results

After data imputation, we collect clean data of 653 observations which is split as 457 (70%) in training set, 196 (30%) in test set. To avoid overfitting, we calibrated parameters for each model in the training set and test them in the test set only once. Table I below represents the overall modeling result and note that Error1 and Error2 represents training and test classification error, respectively. Error1 of boosting classification is not applicable because Bernoulli deviance is used to select best model, instead of classification error.

Among these methods, both support vector machine with polynomial kernel and random forest give the best results which is much lower than 12.9% (the best result from J.R. Quinlan).

TABLE I
RESULT TABLE

| Models | Error1 | Error2 |
|---|---|---|
| Logistic Regression | 0.1312 | 0.1122 |
| Linear Discriminant Analysis | 0.1422 | 0.1071 |
| SVM (linear kernel) | 0.1444 | 0.1122 |
| SVM (polynomial kernel) | 0.1381 | 0.0918 |
| SVM (radial kernel) | 0.1447 | 0.1122 |
| SVM (sigmoid kernel) | 0.1444 | 0.1122 |
| Random forest | 0.1291 | 0.0918 |
| Boosting Classification Tree | not applicable | 0.1071 |
| Neural networks | 0.1317 | 0.1071 |

Since the linear models and support vector machines have been discussed in detail during class, here we focus on results from random forest and boosting.

### A. *Random Forest*

As mentioned in the modeling part, the number of predictors (features) is the key factor in building random forest tree. Fig. 3 illustrates that random forest model chooses 13 parameter to build a classification tree in each bootstrap because it yields lowest estimated error. Fig. 4 represents variable importance from the optimal random forest model and V9 has exceptionally higher importance than others.

### B. *Boosting Classification Tree*

Fig. 5 and Fig. 6 show two different ways to select optimal number of trees in boosting based on Bernoulli deviance. In both figures, the black curve represents the training error while the other colored curve represents validate/cv error and the blue vertical broken line tells the position of optimal value.

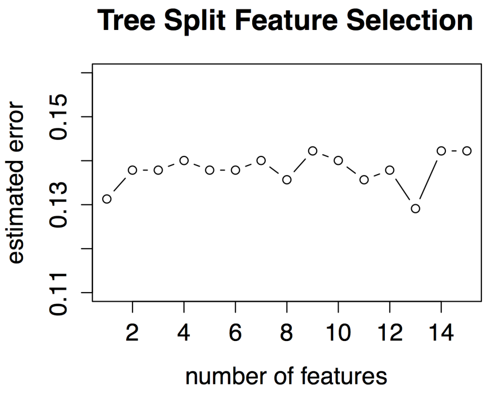

Fig. 3. Random forest: optimal number of feature selection

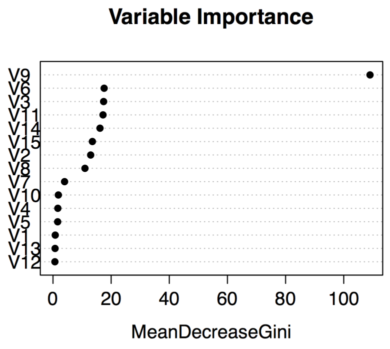

Fig. 4. Random forest: variable importance plot

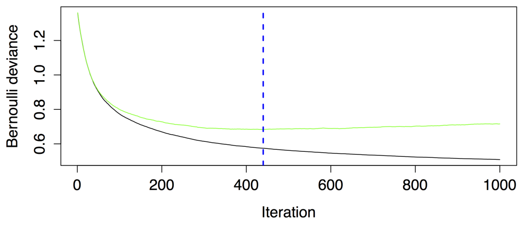

Fig. 5. Boosting: optimal number of trees via cross validation error

## V. DISCUSSION

From the results table we notice that all models beat the bench mark (yay!). This is exactly what we have expected because the bench mark is resulted from pruned decision tree which usually has high variance. Linear models (logistic regression, LDA, SVM with linear kernel) work pretty well because we have relative large p/m rate and hence it will dramatically reduce variance. Similarly, random forest reduce its variance by bootstrap and aggregation. However, there are several issues we have to pay special attention to:

(i) in modeling with boosting classification trees we notice that turning the key parameter is crucial for predicting accuracy. However, the parameter calibrated from cross

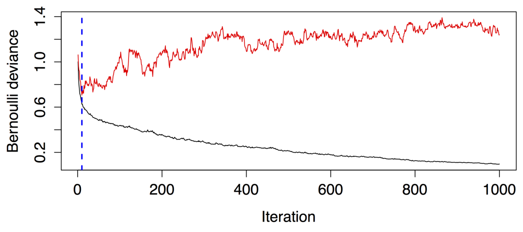

Fig. 6. Boosting: optimal number of trees via test error

validation error shows much better performance than the one from validation/test error. We could also notice that in Fig. 5 the error curve from cross validation is very smooth and the optimal size it chooses is relatively moderate while in Fig. 6 the error curve oscillates so often that it provides much weaker evidence that the resulting parameter is optimal.

(ii) deep learning methods (boosting, neural network) has no apparent advantages over other methods. One problem with boosting classification tree is that boosting model is easily overfitted and hence gives false *good* training error which leads to *poor* test error. The moderate performance of neural networks may also suggest that it is very difficult to discover complicated *hidden* or latent variables associated with the response.

(iii) from the result table we notice that the training error is always higher than test error. We believe this might be caused by noise in our training data and apply some other data splittings may give different training error.

## VI. Future

After exploring the data with methods above, we still believe that the result could be further improved by exploring more on following aspects:

(i) Try boostrap on the original data to get larger data size and perform non-parametric methods such as K-Nearest Neighbors and local regression

(ii) perform clustering on observations and classify them within the cluster to improve accuracy

(iii) consider other metrics such as AUC or better defined score function

## VII. Reference

### References

[1] Quinlan. Simplifying decision trees, Int J Man-Machine Studies 27, pp. 221-234, Dec 1987.

[2] Quinlan. UCI Machine Learning Repository http://archive.ics.uci.edu/ml/datasets/Credit+Approval

[3] Greg Ridgeway with contributions from others (2013). gbm: Generalized Boosted Regression Models. R package version 2.1.

[4] A. Liaw and M. Wiener (2002). Classification and Regression by randomForest. R News 2(3), 1822.

[5] David Meyer, Evgenia Dimitriadou, Kurt Hornik, Andreas Weingessel and Friedrich Leisch (2014). e1071: Misc Functions of the Department of Statistics (e1071), TU Wien. R package version 1.6-4. http://CRAN.R-project.org/package=e1071

[6] Venables, W. N. & Ripley, B. D. (2002) Modern Applied Statistics with S. Fourth Edition. Springer, New York. ISBN 0-387-95457-0

[7] Angelo Canty and Brian Ripley (2014). boot: Bootstrap R (S-Plus) Functions. R package version 1.3-11.

[8] Davison, A. C. & Hinkley, D. V. (1997) Bootstrap Methods and Their Applications. Cambridge University Press, Cambridge. ISBN 0-521-57391-2