Statistical Learning and Bankruptcy Prediction

STAT432 Final Project

Group Stepanov

4/1/2019

Prepare packages

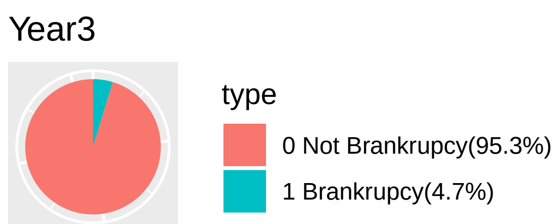
Missing Values

```
## [1] 5618
```

We first conduct basic data preprocessing. Missing values for each dataset are shown in the graph below. Due to the large number of missing values in each dataset, completely delete missing values will result to a large amount of data loss. Thus, we use variable means to replace missing values. We also drop the first variable `id` and factorize variable `class`.

Imbalance Data

Pie Charts to show the imbalance in response variable



The pie charts above show that the data is imbalanced. It has `0` with above 95.3%. The we use the SMOTE method to oversample the minority group and achieve a more balanced dataset.

SMOTE Algorithm For Unbalanced Classification

```
##
##     0     1
## 10008   495
```

```
##
##    0    1
## 7425 5445
```

By applying SMOTE, the new data set is more balanced.

Finally, we test 1. NA values, 2. Data Imbalance

```
## [1] TRUE
```

```
##
##    0    1
## 7425 5445
```

Data Modeling

Split data

Logistic Regression

The `glm` function fits generalized linear models, a class of models that includes logistic regression. Since the dependent variable `class` in our dataset contains only two categories, we will pass the argument `family = binomial` in order to tell R to run a logistic regression rather than some other type of generalized linear model. The function, logit model, we will use is

$$Pr(Y = 1|X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}}$$

We use the modified data set `year3.oversampled` to fit a logistic model `glm.fit` and use the `coef` and `summary` functions in order to access just the coefficients for this fitted model. The table of coefficients are listed below.

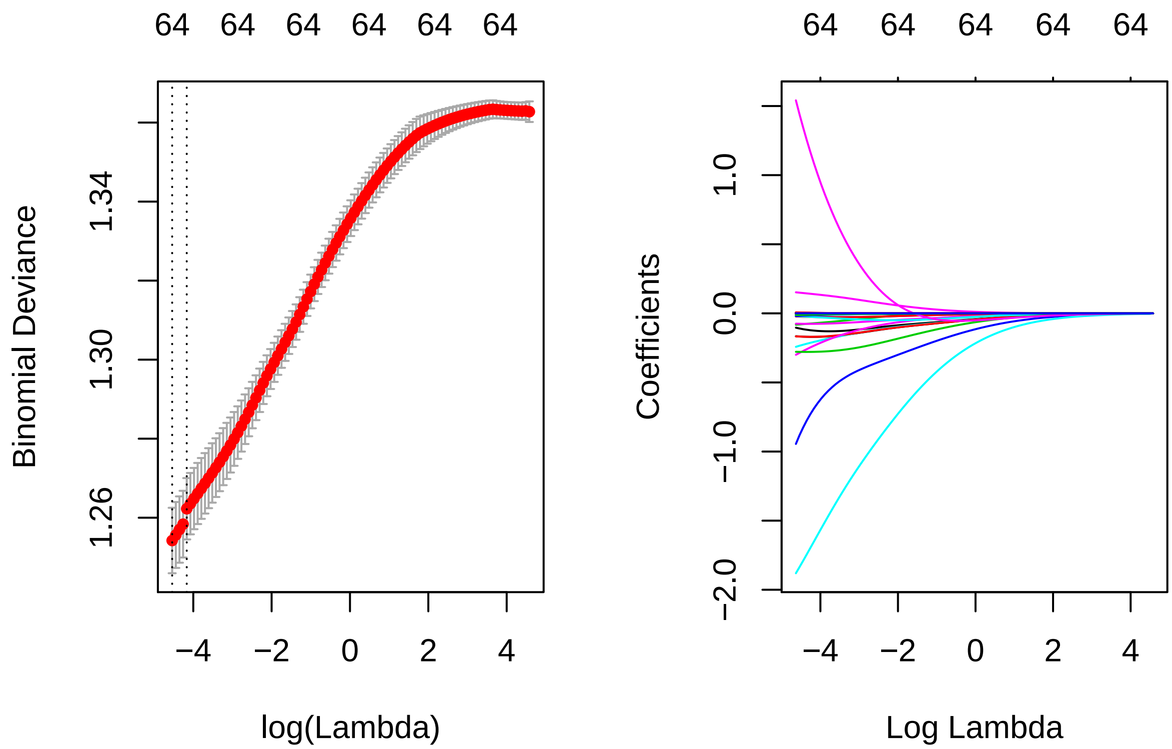
Then, we use the funtion `predict` to predict the probability of `class` and use 'type='response' option to tells R to output the probabilities of the form P(Y=1|X), as opposed to other information such as the logit.Then, we create a class predictions based on whether the predicted probability of bankruptcy is greater than or less than 0.5.

Then, we output the confusion matrix and calculate the error rate

```
##
## glm.pred    0    1
##        0 7307 5156
##        1  118  289
```

```
## [1] 0.4097902
```

Ridge Regression and Cross Validation

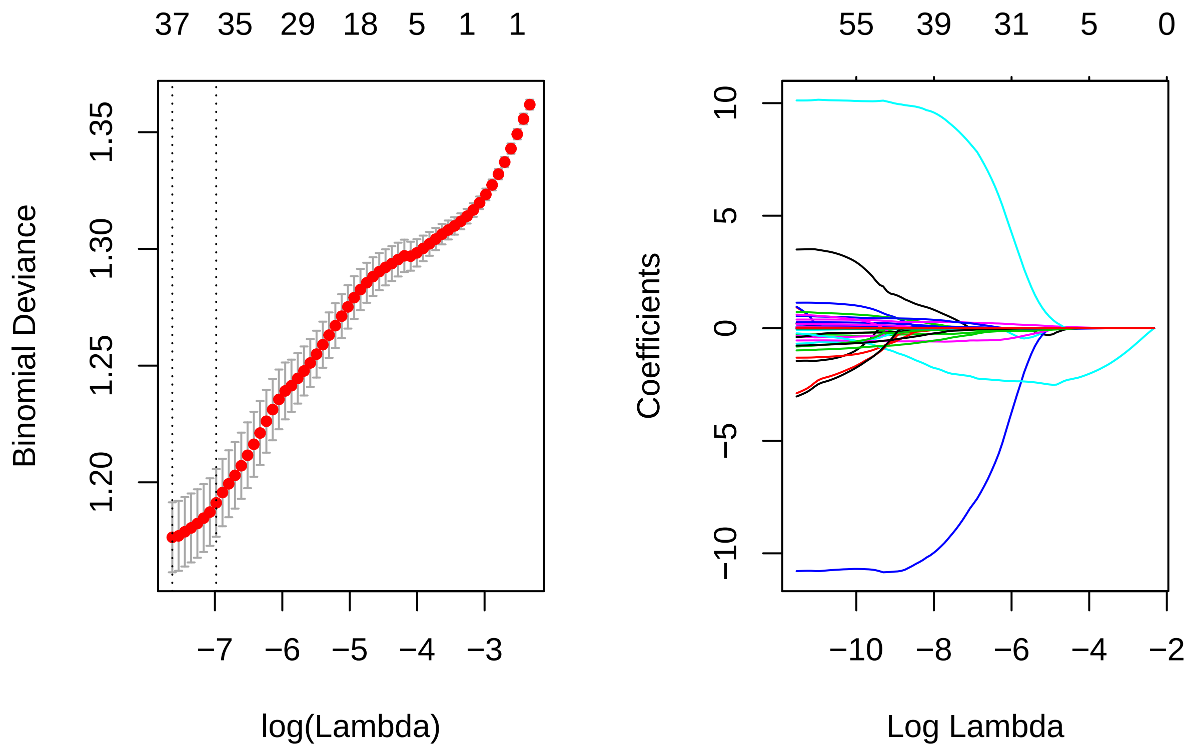



```
##
## pred    0    1
##    0 1947 1127
##    1  280  506
```

The error rate is

```
## [1] 0.3645078
```

Lasso Regression and Cross Validation




```
##
## pred    0    1
##    0 1838  811
##    1  389  822
```

The error rate is

```
## [1] 0.3108808
```

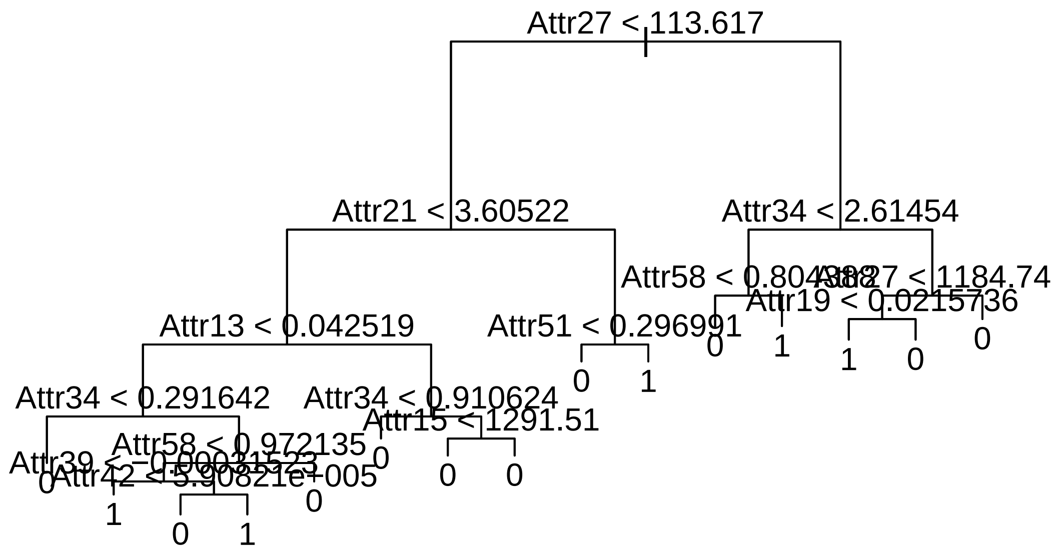
KNN

We will use cross validation on knn method to fit a knn model. We will use `createDataPartition`, `trainControl` and `train` functions from package `caret` to fit the model `knn.fit`.

Tree and Cross Validation

```
##
## Classification tree:
## tree(formula = class ~ ., data = train)
## Variables actually used in tree construction:
##  [1] "Attr27" "Attr21" "Attr13" "Attr34" "Attr58" "Attr39" "Attr42"
##  [8] "Attr15" "Attr51" "Attr19"
```

```
## Number of terminal nodes:  15
## Residual mean deviance:  0.66 = 5937 / 8995
## Misclassification error rate: 0.1454 = 1310 / 9010
```

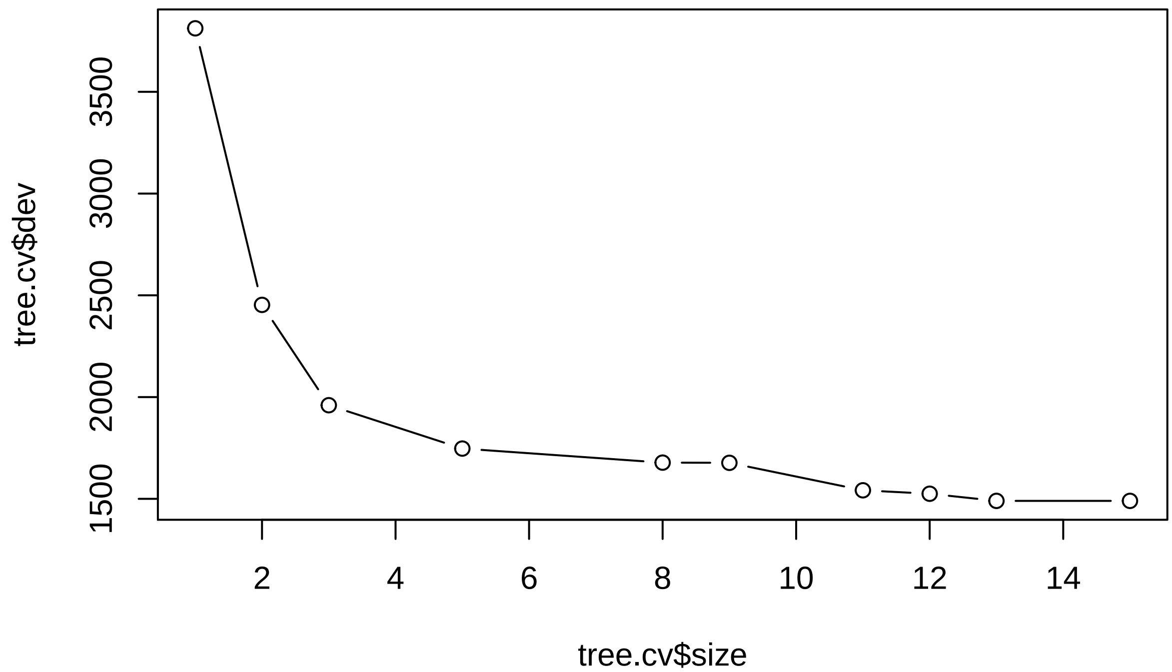



```
##
## tree.pred    0    1
##         0 2105  485
##         1  122 1148
```

The error rate is

```
## [1] 0.1572539
```

Now, consider using cross validation to prun the large tree



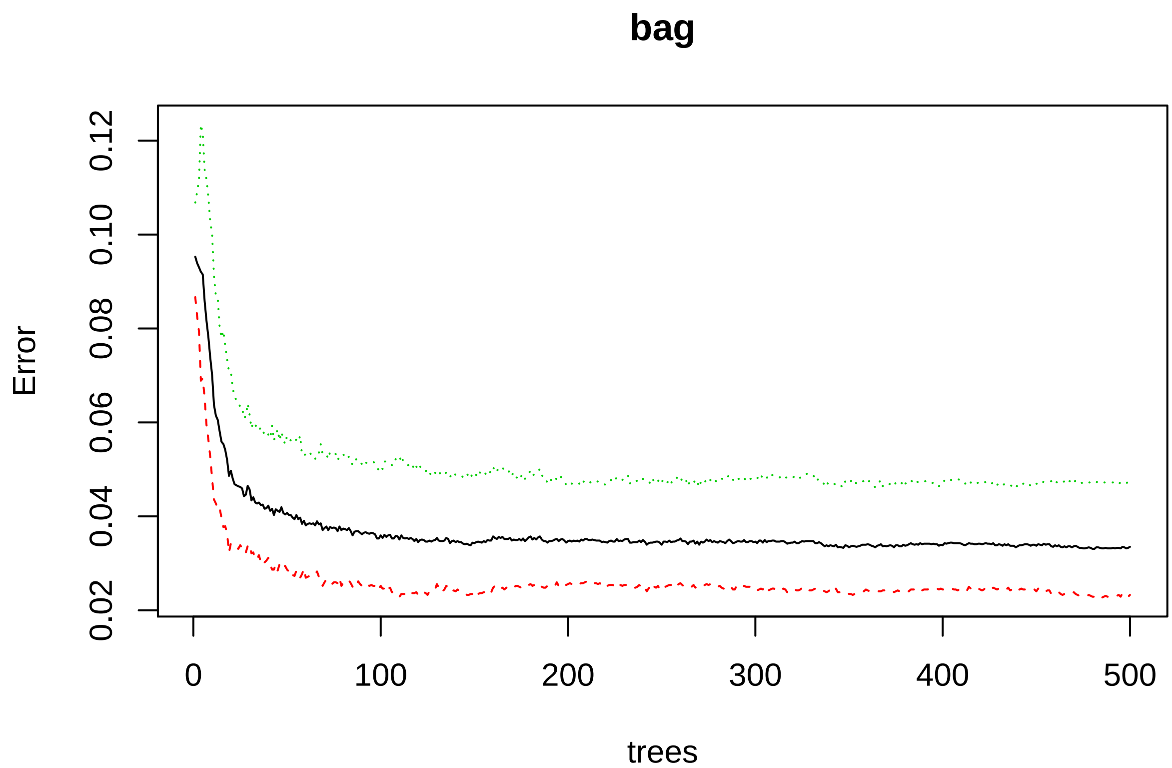

```
##
## tree.pred    0    1
##         0 2105  485
##         1  122 1148
```

```
## [1] 0.1572539
```

There is no much improvement when we prune the full tree model. But the error rate for tree is much smaller than the error rates for previous methods.

Bagging, Random Forest and Boosting

Bagging

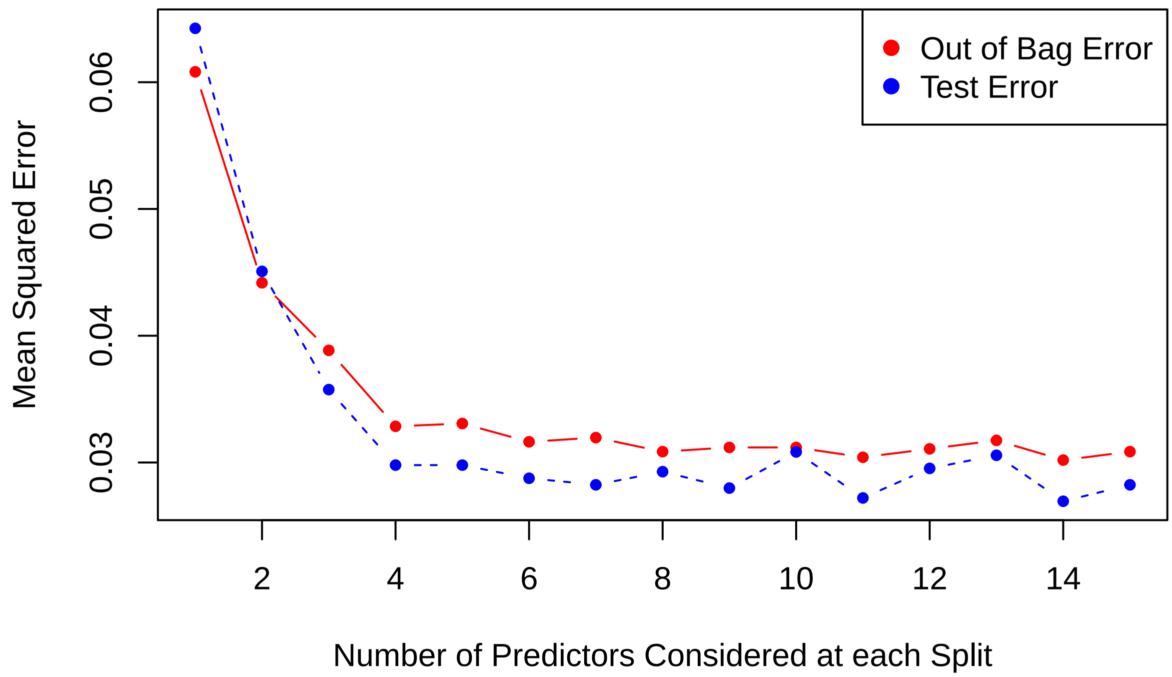



```
##
## bag.pred    0    1
##        0 2176   71
##        1   51 1562
```

The error rate for bagging is

```
## [1] 0.03160622
```

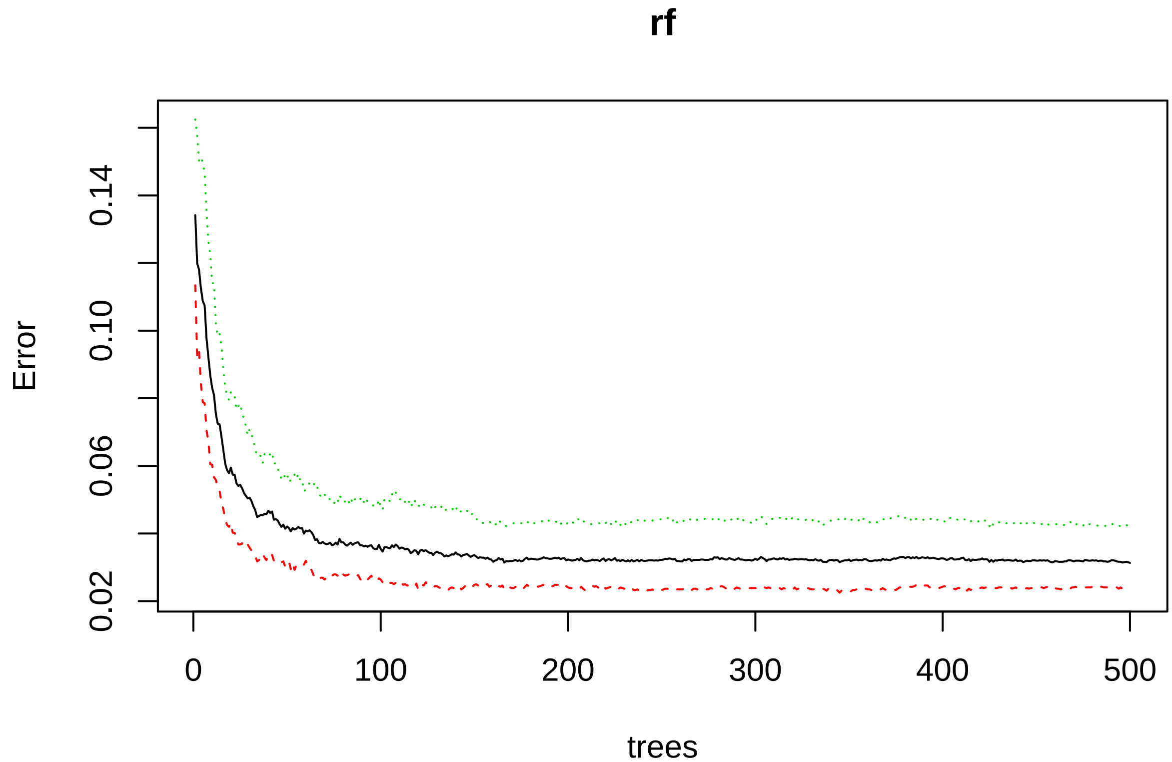
Random Forest




```
##
## rf.pred    0    1
##       0 2177   63
##       1   50 1570
```

The error rate for random forest is

```
## [1] 0.02694301
```



Currently, the random forest method gives us the best performance.