STAT 718 Project: High Dimensional Data Analysis

Shijie Wang, Xin Zhi, Quan Lin

University of South Carolina

Abstract

In this project, we apply penalized likelihood method and feature importance variable selection method to deal with overfitting problem in high dimension dataset. Thereafter, we fit fifteen model on the train data by cross validation to select several prospective model candidates. Then, by means of boosting and stacking method, we find out boosted SVM and 2 layer stacked model with final layer to be LDA. Our final model is a voting ensemble of LDA, boosted SVM and stacked model, which has an accuracy of 0.782 on the test dataset.

Key words: Penalized Likelihood, Boosting, Stacking, Voting Classifier

1 Introduction

In more and more fields, we may deal high dimensional dataset where features(p) outnumbers observations(n). Under this condition, most classifiers would suffer from severe overfitting problem. Therefore, in this project, we aim to explore an approach or build a robust model to this problem.

Firstly, in order to reduce model dimension, we consider variable selection and implement penalized likelihood method, such as Lasso, SCAD and MCP penalty. Based on the criteria of smaller loss, loss penalty works best on train data by cross validation. Thereafter, we also consider using feature importance from tree model to do variable selection for future model development.

Secondly, before model selection, we fit the train data with fifth classifiers with cross validation and by using out of fold accuracy, we select several prospective candidates such as SGD Classifier, LDA, Gaussian Process, Random Forest and Xgboost.

In the following model development, we first apply the boosting method to those weak classifier and find out boosted SVM is improved well to 0.8 accuracy. Then, by means of 2 layer stacking method, from prospective model candidates, we obtain a stacked model which also reaches 0.8 out of fold accuracy.

Finally, our final model is a volting ensemble of LDA, boosted SVM and stacked model, which has a out of fold accuracy of 0.844 on train data and 0.782 on the test data.

2 literature Review

Cook (2004)[1] and Li et al. (2012)[2] talked about the sufficient variable selection concept. Many methods have been used for selecting the variable. For instance, the absolute shrinkage and selection operator (LASSO, Tibshirani, 1996)[3], the smoothly clipped absolute deviation (SCAD, Fan and Li, 2001)[4]. These methods show appealing results for high dimensional data.

Boosting is a general approach of improving accuracy of any given learning algorithm. The Adaboost algorithm is introduced by Freund and Schapire (1995)[5], who solved many difficulties in practice such as the ability to reduce train error and generalization error. The pseudo Adaboost algorithms is given by the following graph,

```
Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X$, $y_i \in Y = \{-1, +1\}$
Initialize $D_1(i) = 1/m$.
For $t = 1, \ldots, T$:
    • Train base learner using distribution $D_t$.
    • Get base classifier $h_t : X \to \mathbb{R}$.
    • Choose $\alpha_t \in \mathbb{R}$.
    • Update:
          $D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$
      where $Z_t$ is a normalization factor (chosen so that $D_{t+1}$ will be a distribution).
Output the final classifier:
    $H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right).$
```

Figure 1: The Adaboost learning algorithm

In addition to ensemble method such as Boosting, Stacking is another ensemble approch, which is to combine many classifiers generated by different learning algorithm. Stacking method is first introduced by Wolpert (1992)[6]. Facing the problem of choosing features and learning algorithm for stacking, Ting and Witten (1999)[7] proposes to stack base classifiers whose predictions are probability distributions over the set of class values.

3 Dataset

3.1 Data Description

In this project, the dataset contains 200 randomly generated features along with 20,000 observations, and contains the response variable which is either 0 or 1. The dataset is clean with no missing or extreme values and distributed between 0 and 1. By checking distribution of the features, we found that the data is randomly distributed between 0 and 1, and by checking the multicollinearity in an ordinary least square regression analysis, We found that the maximum VIF value is less than 10, which showed that there is no multicollinearity issue. The dataset is available at https://www.kaggle.com/c/overfitting/overview.

To perform the classification model, an ‘equation’ based on this dataset was created to generate a response be to predicted. Therefore, this Kaggle dataset consists of two parts: the train data and the test data. The train data contains 150 observations along with 200 features, while the test data contains 19850 observations along with 200 features.

Therefore, the problem is clear, given only 150 cases, we are going to build a model that gives the best predictions on the remaining 19850 cases.

3.2 Challenge of the dataset

In the train dataset, the number of features is much greater than that of the observations, thus the inevitable overfitting problem is considerable. Under the circumstance of overfitting, model fits quite well on the training dataset but performs worse on the test dataset. The essence of overfitting is to have unknowingly extracted some of the residual variation as if the variation represented underlying model structure. To be specific, if we apply the usual classification approach on this project, the model can perfectly predict the training data simply by memorizing the data in its entirety, but will typically fail severely when making prediction.

Therefore, the purpose and challenge of this project is to simulate research and highlight existing algorithms, techniques or strategies that can be used to guard against overfitting.

4 Feature Selection

4.1 Logistic regression with three different penalties

Because the number of predictors is less than the number of observations in the training dataset, in order to reduce the overfitting problem and improve the accuracy of classification, we need to consider variable selection first. Variable selection, a special case of model selection, is to select variable which should be included in our model. It is a method to carry out these kind of high dimensional data. In our case, we consider three variable selection methods:

logistic regression with least absolute shrinkage and selection operator (LASSO) regularisation, smoothly clipped absolute deviation (SCAD) penalty, and minimax concave penalty (MCP).

On the one hand, let's assume π_i is the probability of $Y = 1$, thus $0 < \pi_i < 1$. In the logistic regression setting, the most widely used nonlinear function to model probabilities is the logit link

$$\begin{aligned} \operatorname{logit}(\pi_i) &= \log\left(\frac{\pi_i}{1-\pi_i}\right) \\ &= \boldsymbol{x}_i^T \boldsymbol{\beta} \end{aligned}$$

Then we get logistic regression

$$\begin{aligned} \pi_i &= \frac{\exp\left(\boldsymbol{x}_i^T \boldsymbol{\beta}\right)}{1+\exp\left(\boldsymbol{x}_i^T \boldsymbol{\beta}\right)} \\ &= \left[1+\exp\left(-\boldsymbol{x}_i^T \boldsymbol{\beta}\right)\right]^{-1} \end{aligned}$$

On the other hand, lasso is a regression analysis method that performs both variable selection and regularization in order to enhance the prediction accuracy and interpretability of the statistical model (wikipedia definition). By combining the logistic regression with lasso penalty, we can get the following function that we want to minize

$$L + \lambda \sum_j |\beta_j|$$

where

$$L = -\log\left(\prod_{i:Y_i=1} \pi_i \prod_{j:Y_j=0} (1-\pi_j)\right)$$

Logistic regression with SCAD penalty or MCP are similar, but applying different constraints on coefficients. In R, we can implement these three methods within "ncvreg" package. We apply these three methods on our training dataset, and use 10-fold cross validation to tune the best parameter. Theoretically, the SCAD penalty or MCP could select less variables than lasso penalty. However, in our case, lasso penalty would result in smallest loss. In terms of loss, we are going to use lasso to select the variable first, and use the reduced training data set to build the classification model.

4.2 Feature importance

In our case, we consider ensemble tree specific feature importance, which is the local model-specific feature importance measure. It concentrate on the contribution of variables of a specific model, and it is usually can be applied for any models. The tree-specific feature importance calculates the average reduction in impurity across all trees due to each feature. If the features could split nodes closer to the root of a tree, then these features would lead to the larger importance value. In our case, we also consider the variable selection by using feature importance values from some tree-based models. To be specific, we obtain the feature importance values from the random forest model, extra tree model, adaptive boosting (adaboost) model, eXtreme gradient boosting (XGBoost) model, light gradient boosting machine (LightGBM) model, and gradient boosting decision tree (GBDT) model, respectively. Then we calculate the average feature importance values of them, and use the average as our criterion for variable selection.

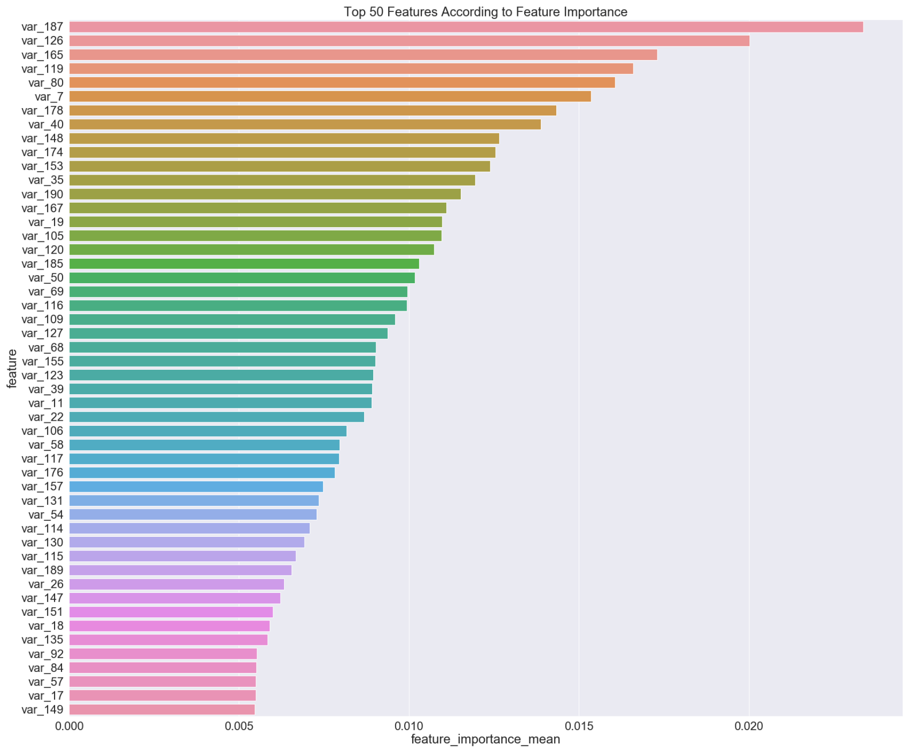

Figure 2: Classifiers Performance with different Feature Selection Method

5 Model Building Approach

After applying penalized likelihood method for linear model and feature importance for tree model, we then consider several binary classifiers and check their performance on the train dataset. By using 10 fold cross validation on train dataset, we compare different models' AUC score/accuracy with different feature selection methods. Thereafter, we consider using **Boosting** Method for those weak classifiers and **Stacking** method for those relatively strong classifiers. Finally, our final model will be linear combination of the boost model and stacked model.

5.1 Initial Model Fitting

Firstly, we considered fifteen binary classifiers to see their performance on the train dataset with 10 fold cross validation. Then, we calculated the average out of fold accuracy to be a measure for this model's generalization ability.

The fifteen binary classifier we considered are: Random Forest, Xgboost, Extra Tree, Decision Tree, Adaboost Classifier(base is Indicator function), Lightgbm, GrandientBoostingTree Model, Multilayer Perceptron model, Support vector Machine, Logistic Regression, Gaussian Process, Stochastic Gradient Descent Classifier K-nearest nerighbour, Linear Discriminant Analysis and Quadratic Discriminant Analysis. The feature selection approach for these model are: feature importance selection, MCP penalty and Lasso penalty. The results of each model are shown in the following table:

Table 1: Classifiers Performance with different Feature Selection Method

Model Type	Raw Dataset	FI 50 vars	FI 30 vars	MCP Penalty	Lasso Penalty
Random Forest	0.51	**0.77**	**0.75**	0.53	0.51
Extra Tree	0.49	0.69	0.68	0.51	0.49
Adaboost	0.51	0.57	**0.73**	0.58	0.55
Xgboost	0.57	**0.79**	**0.75**	0.55	0.54
Lightgbm	0.53	0.53	0.47	0.47	0.47
GBDT	0.61	0.67	0.75	0.52	0.49
Logistic	0.47	0.47	0.47	0.47	0.47
SVM	0.53	0.53	0.47	0.47	0.47
SGD	0.57	0.72	0.77	0.57	0.57
QDA	0.59	0.52	0.55	0.57	0.56
LDA	0.60	**0.82**	0.73	0.55	0.56
KNN	0.57	0.59	0.61	0.53	0.54
Decision Tree	0.45	0.47	0.59	0.47	0.51
Gaussian Process	0.47	**0.79**	**0.77**	0.48	0.47
Mlp	0.38	0.38	0.50	0.53	0.55

Note: FI 30 vars refers to **Feature Importance Selection** with 30 variables

The above table shows the accuracy of model performance on the train dataset. Noticeably, both linear models and tree models with penalized likelihood method, namely MCP and Lasso penalty, results in bad accuracy, less than 0.6, on the out of fold samples. However, the feature importance selection method yields several relatively strong classifiers both for linear models such as SGD(0.72, 0.77), LDA(0.82, 0.73) and Gaussian Process(0.79, 0.77), and tree models such as Random Forest(0.77, 0.75), Xgboost(0.79, 0.75) and GBDT(0.67, 0.75).

Therefore, in terms of feature selection method, we determined to use feature importance selection with 50 variables. Though there is no obvious improvement when reducing 50 variables to 30 variables, choice of 50 variables gives two classifiers Xgboost and LDA close to 0.8(relatively stronger than others). A clearer way to see the difference between different feature selection methods for different models can be through with the following graph,

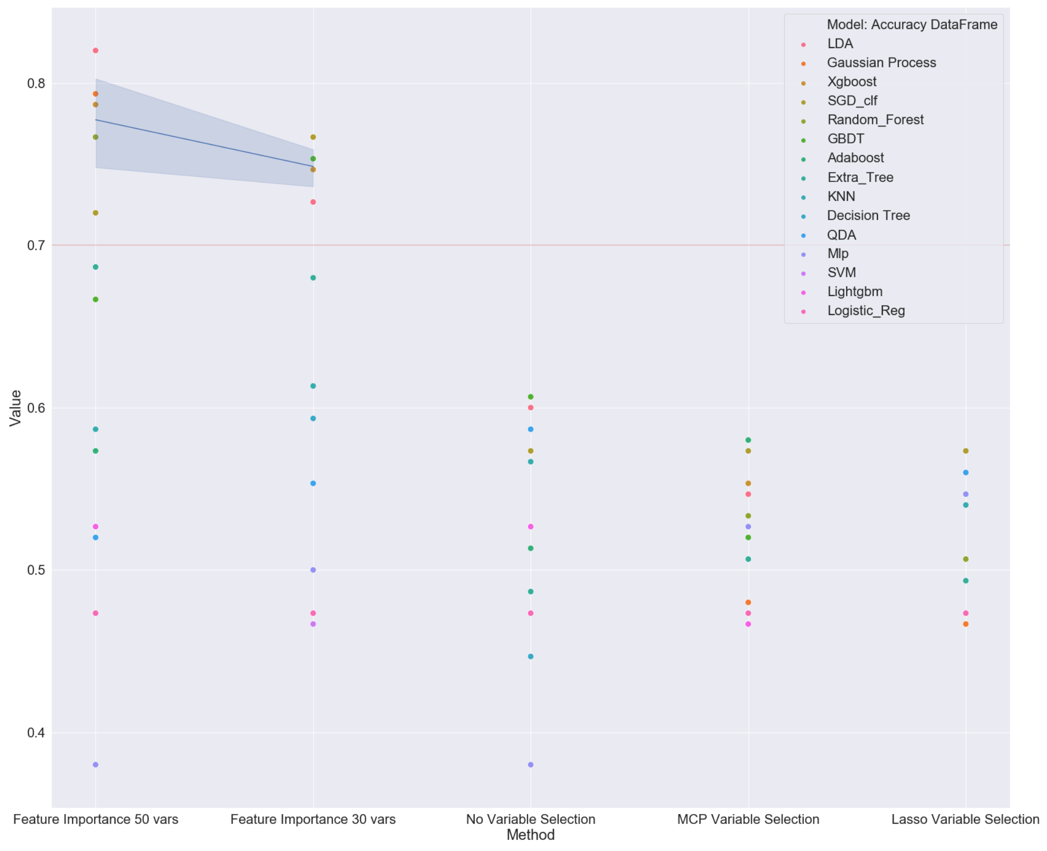

Figure 3: Classifiers Performance with different Feature Selection Method

We mainly focus on the relatively strong classifiers whose out of fold accuracy are above 0.7 and the graph above clearly testifies the choice of 50 variables because the general trend of accuracy is decreasing for those relatively strong classifiers when reducing to 30 variables.

In conclusion, in this step, we show that feature importance selection outperform penalized likelihood method for tree models and some linear models in the train dataset. The initial fitting gives us some insight on the prospective classifier that we can further improve through boosting and stacking method. The potential classifier are: for linear models: SGD(0.72, 0.77), LDA(0.82, 0.73) and Gaussian Process(0.79, 0.77); for tree models, Random Forest(0.77, 0.75), Xgboost(0.79, 0.75) and GBDT(0.67, 0.75).

5.2 Model Boosting

After the initial model fitting, we then consider boosting method for those relatively weaker classifiers. And accutually, from the results above, we find that Adaboost model(with indicator function as base classifier) under feature selection 30 variables has 0.73 out of fold accuracy, which is not so bad. Hence, boosting method can be a good way to improve those weak classifiers.

The core of boosting method is that through iteration, let the base classifier $G(x)$ focus on those wrong classified samples by means of updated sample weights calculated from last interation. Therefore, the final boosted model is going to be a linear combination of many base models $G(x)$ with different weights representing their respectively importance in the final model. Accordingly, since the boosting method relies heavily on the base models $G(x)$, we consider using boosting method to those prospective models in the last section. The result of boosting result is shown in the following table,

Table 2: Boosting Method Performance with different Feature Selection Method

Boosted Model	FI 50 vars	FI 30 vars	MCP Penalty	Lasso Penalty
Logistic	0.5(0.47)	0.5(0.47)	0.5(0.47)	0.5(0.47)
SGD	0.77(0.72)	0.75(0.77)	0.54(0.57)	0.58(0.57)
SVM	**0.79(0.53)**	**0.76(0.47)**	0.48(0.47)	0.54(0.47)
Extra Tree	0.70(0.69)	0.71(0.68)	0.54(0.49)	0.49(0.51)
Decision Tree	**0.63(0.47)**	**0.65(0.59)**	0.57(0.51)	0.55(0.47)
GBDT	0.70(0.67)	0.74(0.75)	0.5(0.54)	0.53(0.55)
Lightgbm	0.5(0.53)	0.5(0.47)	0.5(0.47)	0.5(0.47)
Random Forest	0.78(0.77)	0.76(0.75)	0.55(0.51)	0.53(0.53)

Note: value in () is the accuracy without boosting method

According to the result table above, generally speaking, the boosting method does improve the model a little bit, even though there are few models actually perform worse after boosting. Noticeably, Support Vector Machine(SVM) under boosting method perform quite well, from 0.53(no boosting) to 0.79(boosting method). Decision Tree model also gets some improvement but still far below 0.8. Therefore, Boosted SVM with feature importance selection 50 variables is a good model candidate for final model bagging.

5.3 Model Stacking

Model stacking also called stacked generalization, combines information from several predictive models to improve predictions. A classic two layer stacked model uses the prediction from first layer models(usually several models) as input for the second layer model or final layer(just one). And therefore, stacked models usually performs better than the single trained models. In this step, we mainly consider the prospective models for model stacking, which are SGD, LDA, Gaussian Process, Random Forest, Xgboost and GBDT.

A more general way to conduct stacking method to the train data is to use k fold cross validation and hence, the input of the second layer is out of fold prediction for each fold and label is the still the same. The accuracy of stacking the propective models is shown below,

Table 3: Stacking Method Performance with different Feature Selection Method

Stacking Method				
First Layer	Second layer	FI 50 vars	FI 30 vars	Lasso Penalty
RF SGD GBDT LDA XGB	Gaussian Process	0.75	0.75	0.49
GP SGD GBDT LDA XGB	Random Forest	0.76	0.77	0.54
RF GP GBDT LDA XGB	SGD Classifier	0.73	0.75	0.43
RF GP SGD LDA XGB	GBDT	0.77	0.71	0.49
RF SGD GBDT GP XGB	LDA	**0.8**	0.75	0.53
RF SGD GBDT LDA GP	XGBoost	0.76	0.76	0.51

Note: **RF**:Random Forest, **XGB**: XGboost Tree, **GP**: Gaussian Process,

LDA: Linear Discriminant Analysis, **SGD**: Stochastic GradientDescent, **GBDT**:Gradient Boosting Decision Tree

Since there are six prospective models, therefore, we considered six different stacked models(different at second layer model). As the results shows, although not so much improvement is seen by stacking method, there's one prospective stacked model which reaches 0.8 standard. That is the fifth model with second layer to be Linear Discriminant Analysis(LDA). Hence, this model is going to be one part of our final model.

6 Final Result and Discussion

Based on the results in section 4, we determined our final model $G(x)$ to be an ensemble model of **Boosted SVM**, **LDA** and **2 layer Stacked Model** with first layer to be Random Forest, SGD Classifier, GBDT, LDA, Gaussian Process and second layer to be Xgboost. Our final ensemble method is **Voting** and our feature selection method is feature importance selection 50 variables.

Our final model $G(x)$'s accuracy on the train dataset consisting of 150 samples is 0.844. Our final model $G(x)$'s accuracy on the test dataset consisting of 19750 samples is 0.782.

There are still many things that can be improved. Firstly, in terms of dimension reduction, we applied penalized likelihood method and feature importance from tree model. However, we haven't tried methods to capture some non-linear patterns or approaches like feature embedding. Secondly, in the process of boosting method, we should be careful of the iterations of boosting method in order to prevent overfitting problem. Also, in the stacking step, we considered 2 layer stacked model for concern of final model being too complicate. Whereas, stacked model has the advantage of transforming the original data into the dimension of number of base models, which is also a way of reducing dimension. Hence, we may try 3 or 4 layer stacked model and even try using boosted model as base model into stacked model.

7 Reference

1. Cook, R. D. (2004). Testing predictor contributions in sufficient dimension reduction. *The Annals of Statistics*, 32(3), 1062-1092.

2. Li, R., Zhong, W., & Zhu, L. (2012). Feature Screening via Distance Correlation Learning. *Journal of the American Statistical Association*, 107(499), 1129-1139.

3. Tibshirani, R. (1996). Regression Shrinkage and Selection Via the Lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1), 267-288.

4. Fan, J., & Li, R. (2001). Variable Selection via Nonconcave Penalized Likelihood and its Oracle Properties. *Journal of the American Statistical Association*, 96(456), 1348-1360.

5. Yoav Freund, & Robert E. Schapire. (1995). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, August 1997.

6. Wolpert, D. (1992). Stacked generalization. *Neural Networks*, 5:2, 241–260.

7. Ting, K. M., & Witten, I. H. (1999). Issues in stacked generalization. *Journal of Artificial Intelligence Research*, 10, 271–289.