# Machine Learning Report

By Huan Xu

## Overall Structure

Titanic Machine Learning from Disaster is a binary classification problem with limited features (only 10) and limited data (about 900 training data). For this kind of problem, single ensemble method such as Random Forest and Gradient Boosting Trees is enough to get an effective and accurate model. However, for a coursework, it’s necessary to try different models and a little more complex framework. The overall structure of my model is shown in Figure 1.



Figure 1, Overall Structure of Model.

This kind of 3-layer framework is a widely used model for Machine learning tasks (especially for competition)[1-3]. It firstly preprocesses data and send data to multiple classifiers, then using cross-validation to search for best parameters. The final result will be given by majority vote, average or blending by those fine-turned classifiers in layer 2. This framework is very robust to noise data and usually not overfitting. This report will discuss the details of each part of this model.

## Feature Engineering and Preprocess

Feature engineering is to use humans’ domain knowledge to analyze and understand data[4]. Based on the information we got from features, we can estimate and fill missing data or generate new related features.

For Titanic survivor predicting problem, we are given 11 different features, I will generally discuss how I analyzing and preprocess each of features.

#### Ticket and Cabin

Ticket feature is a random generated letters, it rarely contains information for predicting survive, so I directly drop this feature. Cabin may describe the space relationship between different passengers, but nearly 75% of Cabin information are missing, I also delete this feature.

#### SibSp and Parch

Those two features describe the family relationship of passengers. By analyzing the mean and number of them, I found passengers who broad with families have a better chance to survive. In this case, I add a statistic new features Alone to distribute family relationship.

#### Name and Age

There are about 200 missing age information. For such amount of data, it’s not a good idea to set all of them to average valve. I analyzing the existing data, and it shows there is a high correlation between the title in names and age. So I extract the Title from name and calculate the average age for a title, then set the missing age by his or her title.

In addition, the training data also shows children has a better chance to survive. To clearly illustrate this feature, I use Pandas Cut function to separate data into different group, and found the best edge for defining ‘Child’ is 16. Then I generate a child feature.

## Stratified K-fold Cross Validation

Different classifiers have different parameters, and the selection of those parameters is highly related with training data. It hard to find general accepted parameters for different training data, so we need some methods to fine turn those parameters.

A straightforward way is to split training data into training set and validation set, then we can use training set for training and evaluate the selected parameters with validation set. The drawback of this method is it may overfit for the selected training set. To solve this problem, K-fold cross validation is introduced. K-fold means to split training data into K part with same size, when evaluating a model, each one of K part will be selected as validation set to evaluate the classifier. The final accurate are calculated by averaging the K result[5].

K-fold can avoid overfit to specific selected training data, but when some features in training data is not balanced (e.g. P class in Titanic). The random generated K part may miss some category, to avoid this problem, we use Stratified K-Fold instead. The stratified K-fold can preserve the percentage of samples for each class. In implementation, because the training data is limited, I choose to use stratified 5-fold cross validation for turning parameters.

## One hot encoding and Standard scaling

There are generally two different features, category features and numeric features. The categories just mean the different types which the target belonging to, the distances between categories are meaningless. For example, in Titanic Pclass feature, the number only refers to different class types, the comparison class 2 -class 0 > class 1-class0 makes no sense. On the contrary, the distances between Numeric features are useful, for example, in Titanic Fare feature, the equation 200 – 10> 30 -10 contains information for classification.

Decision Tree based algorithm (Decision Tree, Random Forest, Gradient boosting trees) can handle category and numeric features at the same time, because the splitting point for each node is selected separately by different features. However, for other methods, such as SVM or Logistic Regression, features are used to calculated distances and adding all the distances multiply weight together. So for category features, those distances calculate makes no sense. To solve this problem, I used one hot encoding to separate multiple category features into multiple binary columns which avoid the meaningless distances calculation[6]. Another advantage of one hot encoding is it makes linearly non-separable features to be linearly separable by adding more dimensions. In this case, we can directly use linear classifier which is much more quickly than kernel method.

Another problem of combine category and numeric features is the different scale, e.g. the scale of Fares in Titanic is about 15 to 300, but categories are either 1 or 0. The different scale my influence the optimization speed, so I apply standard scaling to numerical data.

## Single Classifier and Its Performance

I selected 5 different classifiers and using Stratified 5-fold cross validation to search for the best parameters. In implement, I use Sklearn GridSearchcv[7] to automatic this progress, which makes the training as a pipeline progress.

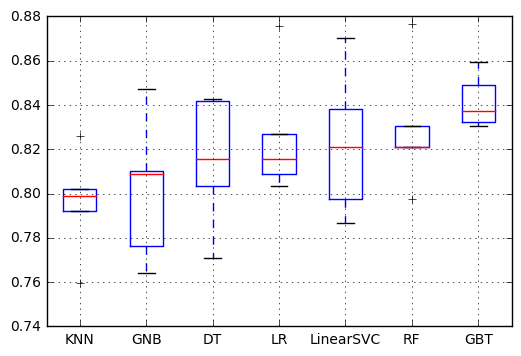
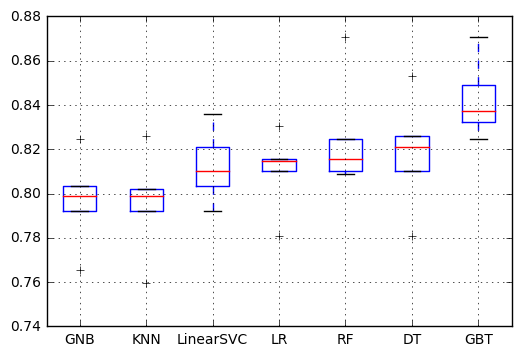


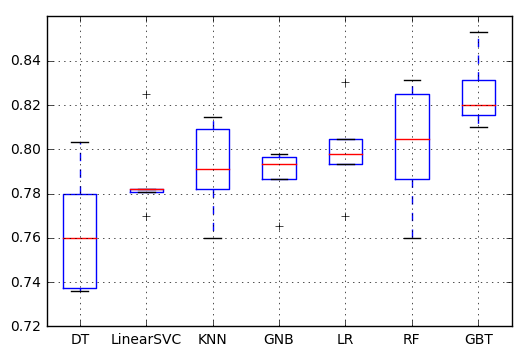
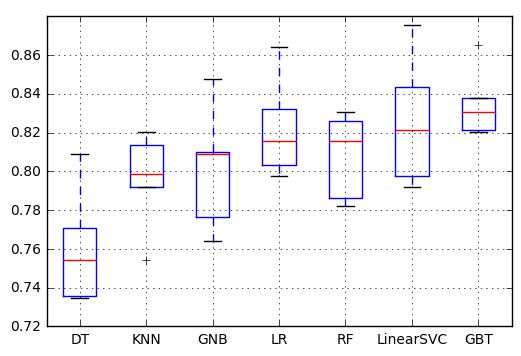
Figure 2a classifier results with one hot encoding. Figure 2b classifier results without one hot encoding

In Figure 2a, we can find the best performance single classifier is Gradient boosting trees which get an average 0.83 accurate. Decision Tree, Logistic Regression, Linear SVC and Random Forest performs similar, but Random Forest, as an ensemble method, is more stable than others.

I also evaluate the effect of one hot encoding. In Figure 2b, all classifiers are trained with origin data without one hot encoding. It’s clear that all decision tree based methods perform better than others.

In addition, In Figure 3a and Figure 3b, I also illustrate the importance of Parameter turning with cross-validation and feature preprocessing.

Figure 3a, classifier results without fine-turning Figure 3b classifier result without feature preprocessing



## Model Ensembles and Evaluations

In part 5, I get seven separated models, because features and training data are quite limit, its hart to get a better result by further feature engineering or parameter turning. In this case, we can combine the seven single models and ensemble to get a final more accurate and robust result.

There are several ways to ensemble models, e.g. majority voting, averaging. In this project, I selected to blending five best performed models with another logistic regression classifier. After fine-turned the five separated model, I regard the results as training data and training a logistic regression classifier as the final result.

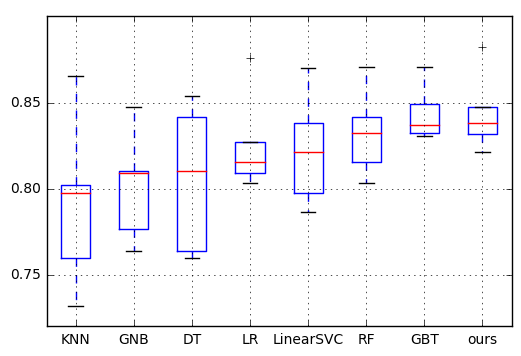


Figure 4

As shown in Figure 4, the result is not as good as I thought. The reason I think is this titanic dataset is not that complex, and a single model can solve it better enough.

I do not analyze the confusion matrix, although it is suggested. Because for binary classification problem, it’s not necessary to check the TF and FT numbers. Besides, the result is roughly balanced (0.383838:0.616162), so it’s not necessary analyses the F1 score either.

## Reference:

1 Jahrer, M., Töscher, A., and Legenstein, R.: ‘Combining predictions for accurate recommender systems’, in Editor (Ed.)^(Eds.): ‘Book Combining predictions for accurate recommender systems’ (ACM, 2010, edn.), pp. 693-702

2 Xie, Z., Singh, A., Uang, J., Narayan, K.S., and Abbeel, P.: ‘Multimodal blending for high-accuracy instance recognition’, in Editor (Ed.)^(Eds.): ‘Book Multimodal blending for high-accuracy instance recognition’ (IEEE, 2013, edn.), pp. 2214-2221

3 Koren, Y.: ‘The bellkor solution to the netflix grand prize’, Netflix prize documentation, 2009, 81, pp. 1-10

4 Domingos, P.: ‘A few useful things to know about machine learning’, Communications of the ACM, 2012, 55, (10), pp. 78-87

5 Moreno-Torres, J.G., Sáez, J.A., and Herrera, F.: ‘Study on the impact of partition-induced dataset shift on-fold cross-validation’, IEEE Transactions on Neural Networks and Learning Systems, 2012, 23, (8), pp. 1304-1312

6 Harris, D., and Harris, S.: ‘Digital design and computer architecture’ (Elsevier, 2012. 2012)

7 Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., and Dubourg, V.: ‘Scikit-learn: Machine learning in Python’, Journal of Machine Learning Research, 2011, 12, (Oct), pp. 2825-2830