STAT441 Project Report

Meng Dong* 20502751 | Weijie Wang 20505037 | Danlin Chen 20536559

November 2016

Summary

We are team Meng and is currently ranked 7th on the Kaggle. This report presents the team work on STAT441 group project, "Predicting surveys' 'interesting' rating" in-class Kaggle Contest. The data for the contest come from LISS Panel, which is the core element of the project Measurement and Experimentation in the Social Sciences (MESS) by research institute CentERdata.

The target of the project is to predict interesting scores of surveys rated by survey participants in scale 1-5 using given information and machine learning algorithms. The dataset contains 9720 participants' demographic information such as age, marriage status, education and income level. The training set has 80297 entries and the test set has 8779 entries where each entry is a record of taken survey. The evaluation metric utilized in the contest is Logarithmic Loss score,

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log(p_i) + (1-y_i)\log(1-p_i)) \tag{1}$$

This metric punishes confident but wrong predictions severely.

The topics discussed in the report are data cleaning, feature engineering, model fitting, model tuning, model selection and reflections on the whole analysis process. After cleaning data, the importance of features is inspected and multiple machine learning algorithms including logistic regression, KNN, random forest, gradient boosting and neural network are tried. We have also investigated the ensemble results from multiple algorithms. We will focus on the models performing better in the analysis part, which are random forest and gradient boosting.

1 Data Cleaning

The combined.csv, which is the training set, contain 30 features in total aside from id. Among those 30 features, 18 are categorical ones and 12 are numerical ones. We use different methods to fill in missing values for different features.

1.1 Repeated Income Features

There are 9 variables related to participants' income, which are gross monthly income, gross monthly income imputed, gross monthly income in categories, 2 net monthly income, net monthly income imputed, net monthly income in categories, gross household income imputed and net household income imputed.

Obviously, many of the 9 features are giving the same fact. Therefore, at the initial step, we kept the four imputed ones because we observed that the missing values in the imputed variables are less. Also, if there is actual value available, the imputed value is equal to the actual value. Therefore, the accuracy is not affected.

1.2 Missing Values

We have tried the following strategies when fitting the missing values.

1. Imputing all missing values for both training and test sets and then fit the models.
2. Imputing missing values for testing set and dropping out all entries in the training set without complete participants information then fit the model.
3. Dropping out all entries in the training set without complete participants information the fit model. To predict the scores of records in the test set with complete participants' information, use the model. For the rest of the records in the test set, use just duration and survey type which are available to all the records.

The way we imputed the missing features depends on whether the feature is categorical or numerical. If the feature is categorical, the missed values were filled with the majority category. If the feature is numerical, the missed values were filled with the mean of all values.

The first strategy gave the worst results and was not investigate any more. We think the reason was that there are more than 10,000 records in the training sets with no participants' information. Too much imputation for these records will hurt the training effect of the models. The third one will improve the model sometimes. However, when the models of gradient boosting and random forest are fully tuned, it will not provide better results. Therefore, we mainly focused on the second strategy.

2 Feature Selection

Generally speaking, there are three types of feature selection methods, wrappers, filters and embedded. The filters select subset of variables indepent of predictors using. The wrappers utilize specific algorithms to score subset of features and require a relatively thorough method to guide the search, which we finally used greedy search with backward selection. The embedded methods select features in the traning process using a specific algorithm such as random forest.

In our case, we tried both wrappers and embedded methods in the feature selection process.

2.1 Feature Importance Ranking

2.1.1 Embeded Method

We used random forest model with default setting in scikit learn package with $n_estimator = 400$ to rank the feature importance. The result are as follows. The x-axis is the cutting threshold of importance values. The y-axis is the logarithmic loss scores.

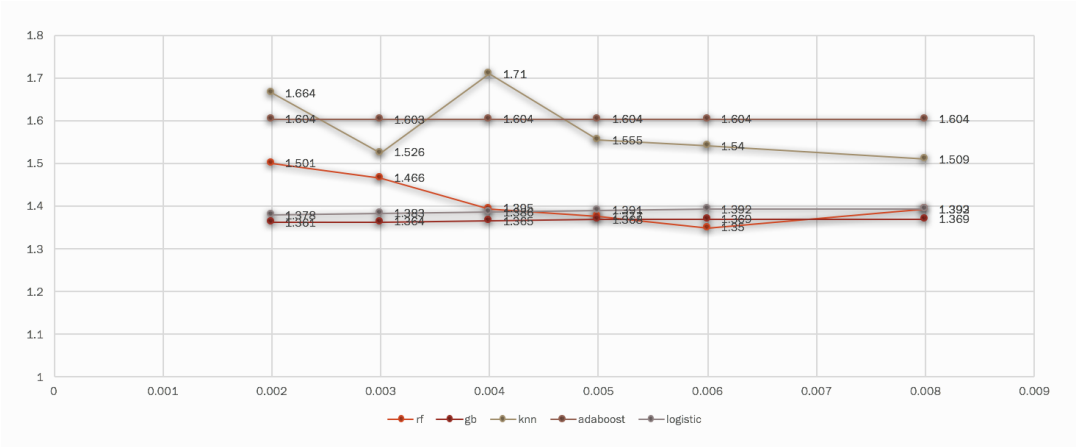

Figure 1: algorithms vs. different cutting values of importance

From the graph we can see that with the number of features decreasing, The result of KNN and random forests are continuously improved. Random Forest hits its minimum and then start to increase. Logistic regression and gradient boosting is increasing but in a tiny amount. The result of Adaboost is almost not affected by the features determined by random forest importance ranking.

The graph gives an overall view of the performance of different algorithms how they respond to the number of features. We later chose to focus on random forest and gradient boosting because they both have a great room of improvement and we can see from the experiments that gradient boosting gives stable and strong results.

2.1.2 Wrappers Method

When training neural network models, backward selection with fitting logistic regression was used to select features. The reason of using logistic regression was that the activation functions of the neural network are logit transformation.

In the backward selection, the first step was to use all the main effects variables. The largest p-value variable was deleted. If the AIC is decreased by the deletion, then the similar iterations were inducted until the AIC could not be decreased any more. Otherwise, delete the second largest p-value variable, and conduct the same steps as above. In the end, 12 variables were deleted.

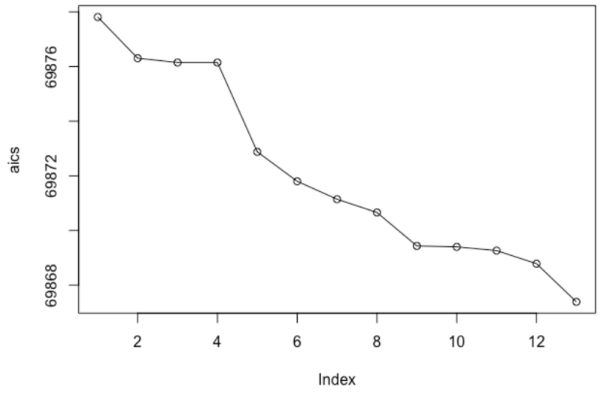

Figure 2: backward feature selection

3 Model Fitting

3.1 Gradient Boosting

The first model we tried out was the Gradient Boosting model. We divided the parameters that need tuning into 2 categories: the model specific parameters and the tree specific parameters. We started with trying out varieties of model specific parameter combinations to find the fastest one with not too low performance. Then, based on this combination, we ran a grid search for all possible tree specific parameter combinations and then go back and tune the model specific parameters. In this way, we could get a pretty high quality result. The thought behind this is actually the model related parameters have little effect on tree specific parameters. Clearly, within the model specific category, n_estimator and learning_rate depend on each other closely, and with both of them fixed, the result of the model will mostly depend on the performance of each tree, which is tuned by tree-specific parameters. We can therefore reduce the number of models need to try from

$$O(n^{\#ofmodelspecificparameters+\#oftreespecificparameters}) \tag{2}$$

to

$$O(n^{max(\#ofmodelspecificparameters,\#oftreespecificparameters)}) \tag{3}$$

where n is the number of different values we need to try for each parameter. As a result, time was saved for digging further into feature filtering.

3.2 Randrom Forest

The second model we tried out is the Random Forest model. The method we used to tune the model parameters here is similar to the one we used for Gradient Boosting model. Compared to Gradient Boosting model, it has several positive characteristics . The first one is stability. Minor adjustment to the model parameters will lead to a more stable and smooth result. Also, Random Forest models often take less time to fit and test, which yields possibilities of more attempts on parameter combinations.

As shown in the plot where the parameters n_estimators (a model specific parameter) and max_depth (a tree specific parameter) are tuned together, the log_loss goes better with the increase of n_estimators and converges when n_estimator reaches around 500. This trend is not affected by the value of max_depth. (Note: We used GridSearchCV in python sklearn package for grid search. The log_loss it printed out was negative because of the way way the package was developed)

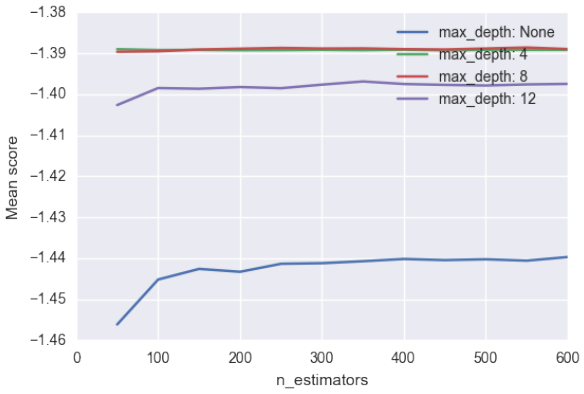

Figure 3: tree estimator and depth

In addition, as introduced in feature selection section, Random Forest model can also be used as a feature selector. After obtaining a relatively good Random Forest model, we could check the feature importance of this model and filter out features whose importance is low. In this data set particularly the boundary of the importance is easy to find, since if you sort the features by importance and print it out, there will be a cliff in most cases which can be set as the cut off boundary.

3.3 Support Vector Machine

The third model we tried is Support Vector Machine. We tried different possible settings of the kernel parameter but only the default one converged on this dataset. Other parameters we tried were 'C' and 'gamma'. What data scientists often do before kicking start the fitting process is standardizing the training data. Unfortunately, it is not the case here as the outcome turned worse after standardizing. Since fitting a Support Vector Machine is often extremely time consuming (energy consuming) and the outcome is not promising enough, we gave up after several trials and spent more time on other models.

3.4 Ensemble Method

Besides models mentioned above, we also tried ensemble models. Model ensemble takes the output of several models and combines them together in some way to obtain a better result. It is also known as a powerful tool of avoiding overfitting. In addition to our best Random Forest Model and Gradient Boosting Model, we also take the prior model where each row is the percentage of each class of the training set. We tried to combine each 2 models among these 3 options with the formula

$$new\_result\_row = model\_row1 * ensemble\_ratio + model\_row2 * (1 - ensemble\_ratio) \quad (4)$$

with ensemble ratio sampled from [0, 1]. The best ratio for the three combinations are shown in the table. Normally, the best ensemble ratio locates toward the better model and this effect increases with the log loss difference of the 2 models. (i.e. GB is better than Prior, so the ensemble ratio for RF vs. GB is larger)

Ensemble Type	Ratio
RF vs. Prior	0.01
GB vs. Prior	0.02
RF vs. GB	0.09

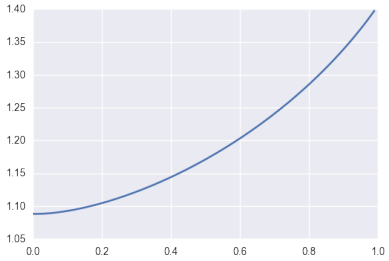

Figure 4: RF vs. Prior

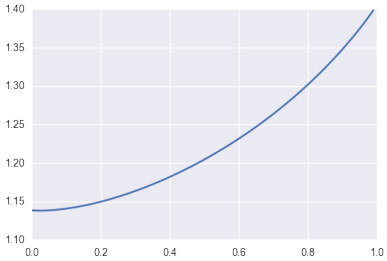

Figure 5: GB vs. Prior

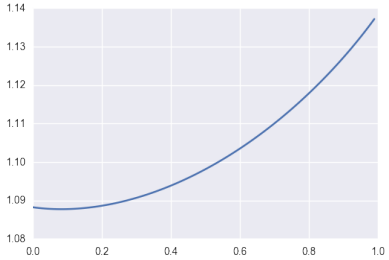

Figure 6: RF vs. GB

3.5 Neural Network

The neural network can have multiple layers, and then using input variables to compute weights in the layers. For the ext step, use a transformation function of the weights to predict the response variable. The visualization of the method is as follows.

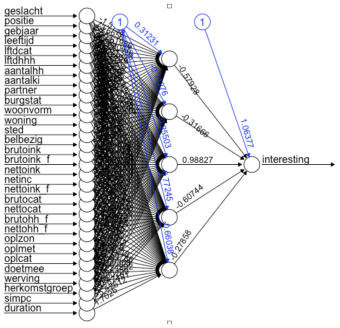

Figure 7: Plot of Squared Residuals

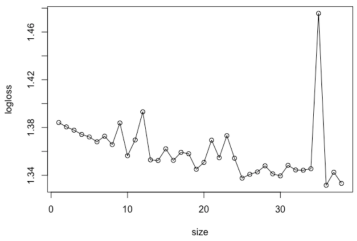

Figure 8: RF vs. Prior

At first, the variables need to be standardized. All the continuous variables were scaled by the Z score method. The min-max method has been applied to all the categorical variables.

$$Z = \frac{X - \mathrm{E}[X]}{\sigma(X)} \qquad X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Then, the r package nnet has been used and the model of this package could only have one layer and limit weights (< 1000). In this case, the largest number of units that nnet can handle is 38.

Thefirst tuning parameter is the maxit, which represents the number of iterations the model will run. For all the cases that have been tested, all of them could be converged before reaching the 10000th iteration. The second tuning parameter is the size, which represents the number of units in the layer. The best

size is figured out by the following plot. The model is obtained by applying the training dataset and predicting with the test dataset. As the plot of size and logloss show, the best size is 36. The logloss is 1.3304.

4 Conclusion

The performance of different models in log_loss score is summarized in the following graph. It turns out that random forest is our best model measured by log_loss score. Later, we also tried ensembled model and the combination of RF and GB gives the best result. On kaggle, our current kaggle score is 1.119 which is a dramatic improve against the benchmark.

Figure 9: Model Performance

By finishing this project, we gained the experience of how to complete a machine learning project. We also learned that selecting the right sets of parameters and the correct features can have dramatic impacts on the outcome.