Tianyi Yang
tyan227

COMPSCI 761 Assignment 1

Kaggle Challenge: Will they default?

The objective of the Kaggle challenge is to predict whether a customer will default on their credit card payments by training a model on the dataset. In order to do so, the first thing to do is exploratory data analysis There are 24 data fields and one target class (default payment) in the training dataset. For the target class, it's obviously a binary value, with 0 or 1 stand for Yes or No. Similarly, we can identify in the fields attributes like gender, which also should be classified as binary. There is also some fields should be modified, for example, education and Age could be recategorized as nominal or ordinal. But before that, we should do the feature processing.

Feature processing is vital for data mining, it refers to the process of removing irrelevant features and retaining related ones. It can also be considered as selecting one of the best feature subsets from all features. Feature selection can essentially be considered as a process of dimensionality reduction. In our case, if we can reduce the scale of the fields for analyzing, the process of data mining would be more efficient and more likely to find one ideal model for the prediction.

For the Kaggle challenge, Attributes like ID actually would have no relationship with data, but it may cause overfitting later on, and make the prediction less accurate. By using WEKA, we can easily apply feature processing by removing meaningless attributes like ID and other potential ones. By applying "Attribute Selection" under the "Filters", we can remove redundant attributes determined by the algorithm. Then we get a dramatic result that retains only 5 variables. By comparing the chosen 5 variables and other removed variables, we can find one pattern: For the selected attributes such as the history of past payment (Figure 1 left), the proportion of target 1 (Red) in different values is increasing. At the same time, for other attributes, like region (Figure 1 right), we cannot locate similar trend or distingue differences, as their distribution is nearly the same. This situation appears frequently in some extreme values, so we may keep them in this prediction.

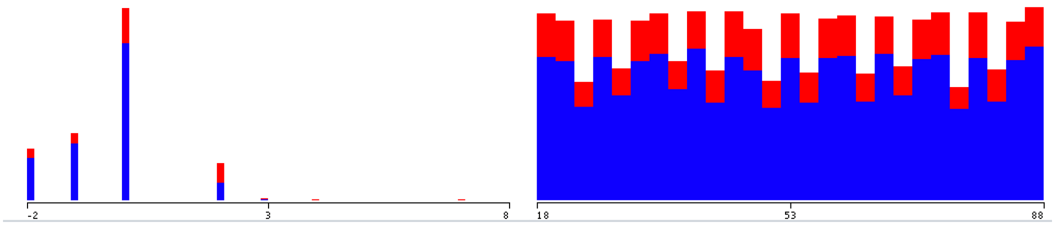

Figure 1: Proportion change in selected attribute comparing to unselected attribute

After removing unnecessary attributes, the cross-validation performance increases from 81.22% to 81.375%. During this prediction I didn't standardize the data because the selected data have similar scales.

Similarly, we can remove outliers from the data by applying "Interquartile Range" and "Remove With Values", to keep the prediction from overfitting to these values. Then, the performance will further increase to 81.5795%. Now, the data preparation process could be considered finished, and we could take this as a start point and apply it to different algorithm.

As mentioned before, we use cross-validation to test the performance of the prediction. For the following algorithms, I used the default set for their settings, and set the folds of the cross-validation to 3 from 10, to balance the prediction accuracy with the computer load and time.

The previous result are all calculated by using Random forest. I've also tried other algorithms, and here's the result:

Random Forest	81.5795%
Random Tree	81.6081%
J48	82.1815%
Ensemble (multiple algorithms)	81.8881%

Table 1: Model Performance on local cross-validation result

As we all know, J48 (C4.5) is an algorithm of decision tree, which is built by using the whole dataset. After getting the output, you can see its model shown in Figure 2 (left). Random tree is a collection or ensemble of decision trees. During the data mining, only a range of randomly selected features is actually training the data, and the final result is built on this subset (Figure 2 right). In this way, considering Random tree as only part of the J48, its lower accuracy is acceptable.

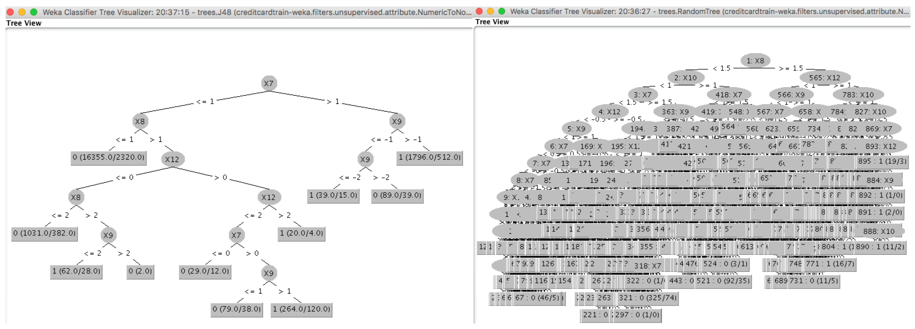
Weka Classifier Tree Visualizer: 20:37:15 - trees.J48 (creditcardtrain-weka.filters.unsupervised.attribute.NumericToNo...
Tree View
X7
<= 1
> 1
X8
X9
0 (16355.0/2320.0)
X12
1 (1796.0/512.0)
1 (39.0/15.0)
0 (89.0/39.0)
0 (1031.0/382.0)
1 (20.0/4.0)
1 (62.0/28.0)
0 (2.0)
0 (29.0/12.0)
0 (79.0/38.0)
1 (264.0/120.0)
Weka Classifier Tree Visualizer: 20:36:27 - trees.RandomTree (creditcardtrain-weka.filters.unsupervised.attribute.N...
Tree View

Figure 2: Visualized decision tree and random tree

However, the performance of Random Forest in this list is a little bit unusual. Random forest is like a collection of random trees which is built on different rows and columns. This algorithm improves the performance by voting or averaging between weak tree classifiers. If the amount of tree is big, we can consider that it almost covers all the data and therefore its accuracy should be higher than the random tree.

Ensemble (stacking) has also been conducted in this prediction by applying multiple algorithms (Random forest, simple logistic, neural network and bagging) at the same time. This would produce a integrated result, however, its performance may be draw back by one or more unsatisfactory algorithms.

The performance of this algorithm showing on Kaggle leaderboard is different from the previous result:

Random Forest	0.82266
Random Tree	0.82133
J48	0.82366
Ensemble (multiple algorithms)	0.81766

Table 2: Model Performance on Kaggle Public Leaderboard

Based on the current list, J48 is still the best algorithm. However, its leading position appears to be inconspicuous, followed closely by Random forest, whose performance is now better than the random tree. In order to further increase the model performance, we can increase the number of trees used in a random forest, in this way, errors caused by noisy data may be simply reduced. Also, we can try to combine algorithms with data processed by different methods, to see if there is potential improvement. Among all these tests, the highest score I got is 0.82933, with a ranking of No.4, in that test, I tried to combine data without outliers to random forest algorithm which uses 400 trees and 10 folds in the calculation.

Now my models seems to be more accurate. But in fact, the public leaderboard of Kaggle only take approximately 30% of the test data into consideration, and the conclusive result for this assignment is based on the other 70%, which means the final standings may be very different. Algorithm with the best performance in the 30% of the data doesn't guarantee its position in the other 70%, as shown below in the Table 3.

Model	1	2	3	4	5	6	7	8	9	10	1-3	4-10	Total
1	**T**	**T**	**T**	F	F	F	F	T	T	T	**100%**	100%	100%
2	**T**	**T**	**T**	T	T	T	T	T	T	T	**100%**	43%	60%
3	**T**	**F**	**T**	F	F	F	F	T	T	T	**66%**	100%	90%

Table 3: Example of difference between partial and total performance

In Table 3, consider the Model 1 as the real situation to be predicted. Model 2 predicts perfectly for the first 3 rounds while model 3 failed once. But from the rest of the table we can see that actually model 3 performs far better than model 2. This means that we could only take the result from the public leaderboard as a reference. The final submission should not only base on the public ranking, but also comprehension to the models and judgment to the situation.

For the final submission, I prefer these two models:

No.	Processed features	Algorithm	Current Score
1	Unnecessary attributes and outliers removed	Random Tree	0.82200
2	Outliers removed	Random Tree	0.82933

Table 4: Final submissions

As you can see, although J48 perform better in the previous two rounds, I still choose random tree algorithm for both of these models, that's because random tree could mitigate the problem of overfitting with increasing error due to bias. On the other hand, removing redundant attributes and outliers are two efficient methods for data mining, for which I applied them during the features processing. What's more, you may have notice that I've kept the model with the highest score, just in case the previous model may accidentally removed some attributes that are actually relevant.

The Kaggle competition is really a meaningful way to learn. Every time after I submitted the model, the short waiting time for the uploading made me nervous and the marks often surprised me because it didn't match the local cross-validation result in most cases. The feedback and the ranking kept pushing me to achieve a better result. During the past few weeks, I kept thinking the possible solutions for a better prediction, which prompted me to look for relevant materials which at the same time enhanced my understanding to the knowledge taught in the lecture.