COMPSCI 367 ASSIGNMENT 1

Analysis on credit card data to predicate default

Haozhou SHEN, 731260510, hshe507

August 10, 2018

Introduction

The report is mainly about using different techniques in machine learning to analyze given data of credit card holders and use algorithms to predict if they will default in the upcoming payment.

1 Feature processing

The data acquired is the information about the current credit card holders.

- ID: an anonymous id unique to a given customer
- X1: Amount of the given credit
- X2: Region
- X3: Gender
- X4: Education
- X5: Marital status
- X6: Age (year)
- X7 - X12: History of past payment
- X13-X18: Amount of bill statement
- X19-X24: Amount of previous payment
- Y: default payment (Yes = 1, No = 0) - target class

1.1 Relevant/irrelevant features

With initial inspections on the given training data, some attributes which will cause noise or which will need further modifications are shown below:

- ID:
  Data in id are continuous integers from 20001 to 30000 which are used only to identify customers from each other. Thus this features should be removed.
- X2(Region):
  Region data is 71 distinct integers with in range from 18 to 88. We can find that this is only part of the all data as first 17 regions' data are missing. Also, region is likely to have a small influence on the outcome and may have noise on the modeling process. It should be considered to be removed.
- X3,X4,X5(Gender,Education,Marital Status):
  Each of these attributes only have several integers with in them to show different categories the customers falls in. So it should be modified to be nominal type.
- X6(Age):
  When processing X3,X4,X5, I initially considered age should be set to a nominal type. But with cross-validation process, we can find that the age data show a normal-distribution pattern. So it should be considered as a numeric data. Additionally, if we take age as nominal may cause loss of accuracy when model encounter samples with ages not in the train set.
- X7-X12(History of past payment):
  Initially I think this is same as other payment in the from of money paid, but with inspections, these attributes contains only integer -2 to 8. So I believe they indicates the payment status rather the actual amount paid, and should be seen as categorical.
- Y(if default):
  The ratio of two class in Y attributes is obviously imbalance, which could cause the trained model to be imbalance.

1.2 Techniques skills

Several methods have been used to pre-process data to get more accuracy, and they will bring different effect to the final result when using different algorithms.

- Remove outlier/extreme value:
  Remove outlier/extreme values could be helpful when dealing with enormous numeric data. However, in this case, if we use the filter to remove data that seems to be incorrect, about 23% of data will be removed. So this process technique will cause a big loss on the raw data, and therefore decrease model accuracy.

- Balance the weight of two outcome:
  When using classBalancer, it should be help improving the accuracy of Y=1. However after applied with filter, the accuracy dropped to around 70%, and also prediction for Y=0 worse off.

- Remove irrelevant attributes:
  In modeling process, I removed ID as it is an irrelevant feature. Also, sample which removed x2(region) also be tested. Result shows that removing region will have different effects when applying various algorithms.

- Change attributes type:
  As attributes can be recognized as numeric or nominal. Some of attributes like(region,gender,education,martial status and age) can be seen as both types. Lots of combinations are tested. Result shows that when change the type of an attribute from numeric to nominal will help improve the accuracy at most time.

- Normalization/Standardization:
  As tested on multiple algorithms, normalization
  standardization shows no obvious effect on the result. However normalization is required when processed with Multilayer Perceptron.

2 Cross-validation performance

Within Weka, many of algorithms are tried on the different set of data to see the cross-validation performance. Algorithms used includes J48 decision tree, RandomForest, DecisionTable, Hoeffding and Decision Stump.

2.1 General performance

The cross-validation performance of different algorithms with different data sets is shown with figure below:

(e.g All data acquired are based on the given training data with different setup, all models built with 10 cross-validations.)

all model built with 10 coree-validition			
	J48	Random Forest	Decision Table
all attributes as numeric	79.665	81.545	81.9
no id all numeric	79.915	81.57	81.9
no id all numeric normalize	79.915	81.4	81.9
no id all numeric remove value(15260)	78.2438	80.197	80.5832
no id with class balancer	65.6589	69.3978	70.4121
no id region,gender,marital status and education as nominal	81.425	80.9	81.91
no id no region all numeric	80.155	81.485	81.9
no id no region GME as nominal	80.22	81.735	81.91
no id no region GME as nominal remove value(15282)	78.53	80.14	80.6439
no id no region GME as nominal normalized	80.315	81.57	81.91
Only X7-X24	81.73	81.115	81.89
Onlu X1,X7-X24	81.655	81.15	81.89
Only X7-12 as nominal	81.575	80.905	81.205
No id RGME 7-12 as nominal	81.715	80.75	81.685
no id no region GME 7-12 as nominal	80.82	81.56	81.685
no id no region GME 7-12 as nominal nomalized	80.775	81.525	81.685

Figure 1: General performance

With the pre-processing of the raw data going deeper, the accuracy increases.

2.2 Algorithm analysis

2.2.1 3 algorithms tested on all data sets

With all models tested, decision table has a overall best among all the algorithms. However it is very hard to improve with modifying input and parameters. Decision table is built among certain pattern that will not change much when majority of data unchanged.

On the other hand, random forest performed better than the J48 decision tree, however when dealing with data with majority of numeric data, J48 sometimes outperformed random forest.Additionally, decision table always perform better than the decision tree and random forest, however the accuracy of the algorithm is very hard to improve with different settings and dataset.

2.2.2 Additional algorithms tested

Additional algorithms like Hoeffding, Decision Stump, LMT and REPTree also have been tested on small amount dataset:

	J48	Random Forest	Decision Table	Hoeffding	Decision Stump	LMT	REPTree
no id all numeric	79.915	81.57	81.9	81.62	81.925	81.835	81.47
no id region,gender,marital status and education as nominal	81.425	80.9	81.91	81.11	81.925	81.795	80.995
Only X7-X24	81.73	81.115	81.89	81.64	81.925	81.885	81.085
No id RGME 7-12 as nominal	81.715	80.75	81.685	81.005	81.205	81.76	80.995

Figure 2: Performance of additional algorithms tested

Among these 6 algorithms decision stump always perform best only except data with most types as nominal. Same effect appears on the decision table too, this shows a close relation between decision table and decision stump algorithm.

We can find that these additional algorithms seems to stand still when the input training data changes.So we can say these simplified algorithms are not sensitive to the amount and the property of the input data.

2.3 Neural network

A special algorithm named Multilayer Perceptron was tested for modeling. This algorithms is building a feedforward neural network to classify the data. The result of accuracy are shown below:

	Accuracy for Multilayer Perceptron
no id all numeric	81.63
no id no region all numeric	81.78
no id no region GME as nominal	81.755
no id no region GME as nominal normalized	81.755

Figure 3: Result for Multilayer Perceptron

We can find that, the result is acceptable accurate. However it is very hard to improve. Increase of input nodes and hidden layers shows a minimal effect on the output.
Also building a neural network costs huge amount of time and memory resources. Assigning more attributes to nominal means the increase of the input nodes and therefore increasing the nodes in the hidden layer. More nodes always take more memory and will take exponentially more time to compute the model.

3 Leaderboard performance

Altogether nearly 20 models submitted, the best performance model is built with random forest algorithm from data that deletes the ID attributes and leave all other attributes as numeric.

With submissions of models, we can find that the performance score on leader board is slightly higher than the cross-validation performance, especially for the models built with random forest. I suppose the data used for evaluation is more typical than the training set provided.

Among my submissions mainly 3 types of algorithms are used.Decision table and decision stump got a highest accuracy in the cross-validation test so most submissions are based on these two. The overall score for these algorithms is around 0.816-0.823.

4 Final models

Two models got highest scores on leader board:

1. Randomforest:0.82566
   This model is built with random forest algorithm from data that deletes the ID attributes and leave all other attributes as numeric.

2. DecisionStump:0.82133
   The model is built with Decision Stump algorithm, using data removes ID and region(x2) attributes, all other attributes are numeric.

The best score is achieve by the random forest algorithm with parameters setting:[-I, 7, -K, 4, -depth, 5], based on only numeric data. It get a accuracy of 82.25% in cross-validation. I suppose random forest is one of the best algorithms that fits in the test data and therefore perform best.

On the other hand, decision stump get a higher accuracy than random forest in cross-validation. However it did not rank the first place in the leader board. I believe this is caused by the imbalance of the given training data. Y=0 takes a around 75% part of all data. So the decision model trained always get a very high accuracy on prediction of Y=0 but much lower for Y=1. As the online test data is unknown, majority part of it could be Y=1. I suppose this is the reason why decision stump algorithm loses some accuracy in the leader board.

References

[1] AutoWeka https://www.cs.ubc.ca/labs/beta/Projects/autoweka/>

[2] Credit Card Default Prediction Using TensorFlow (Part-1 Deep Neural Networks) S.Hussain