Final Project

Predict credit score using multi-algorithm bagging

Statistical Learning|張致語|108021186

Overview

- Inspiration
- Data and Algorithm
- Sub-model1: SVM classifier
- Sub-model2: Tree classifier
- Sub-model3: Logistic regression
- Sub-model4: QDA
- Aggregate Process and comparison
- Conclusion

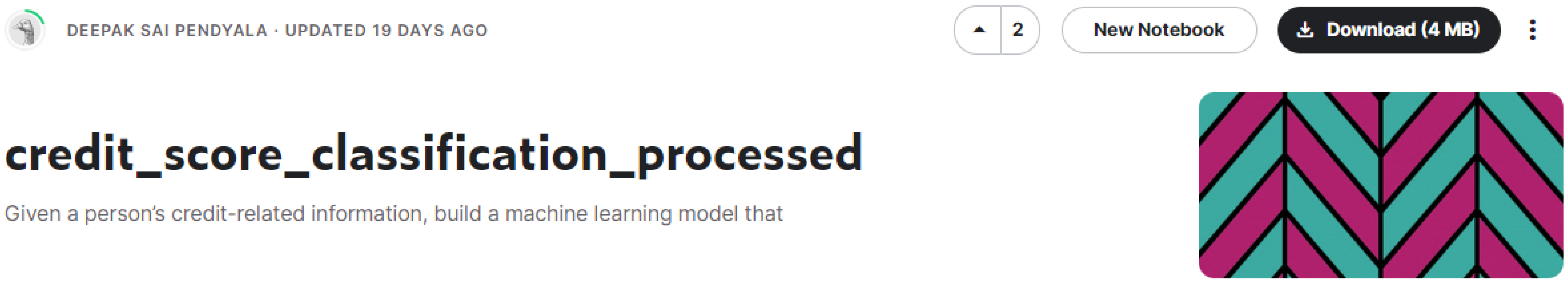
Inspiration

- Paper 'Inproving Bagging Performance through Multi-Algorithm Ensembles' connects the diversity with correlation and further shows that with higher diversity will improve accuracy in classification problems.

diversity → correlation → accuracy

Data and algorithm

- Source of data



- Link: https://www.kaggle.com/datasets/deepaksaipendyala/credit-score-classification-processed

Data and algorithm

- Source of data

Data update: 2022/12/18

Dimension of data: 25 variables with 79805 rows

- Variables

X <int>	Month <int>	Age <dbl>	Occupation <int>	Annual_Income <dbl>	Monthly_Inhand_Salary <dbl>	Num_Bank_Accounts <dbl>
1	2	23.00000	13	19114.12	4194.171	3
2	6	34.42982	13	19114.12	4194.171	3
3	0	23.00000	13	19114.12	4194.171	3
4	7	23.00000	13	19114.12	1824.843	3
5	5	23.00000	13	19114.12	4194.171	3
6	4	23.00000	13	19114.12	1824.843	3

Categorical variable

Data and algorithm

- Variables

Num_Credit_Card <dbl>	Interest_Rate <dbl>	Num_of_Loan <dbl>	Type_of_Loan <int>	Delay_from_due_date <int>
4	3	4	128	-1
4	3	4	128	3
4	3	4	128	5
4	3	4	128	6
4	3	4	128	8
4	3	4	128	3

Categorical variable

Categorical variable

Data and algorithm

- Variables

Num_of_Delayed_Payment <dbl>	Changed_Credit_Limit <int>	Num_Credit_Inquiries <dbl>	Credit_Mix <int>	Outstanding_Debt <dbl>
30.92334	1112	4	1	809.98
7.00000	4238	4	1	809.98
4.00000	3764	4	1	809.98
30.92334	1112	4	1	809.98
4.00000	4163	4	1	809.98
8.00000	1112	4	1	809.98

Categorical variable

Data and algorithm

- Variables

Categorical variable

Credit_Utilization_Ratio <dbl>	Credit_History_Age <int>	Payment_of_Min_Amount <int>	Total_EMI_per_month <dbl>
31.94496	0	1	49.57495
28.60935	267	1	49.57495
31.37786	268	1	49.57495
24.79735	269	1	49.57495
27.26226	270	1	49.57495
22.53759	271	1	49.57495

Total_EMI_per_month <dbl>	Amount_invested_monthly <dbl>	Payment_Behaviour <int>	Monthly_Balance <dbl>	Credit_Score <int>
49.57495	118.28022	3	284.6292	2
49.57495	81.69952	4	331.2099	2
49.57495	199.45807	5	223.4513	2
49.57495	41.42015	1	341.4892	2
49.57495	62.43017	6	340.4792	2
49.57495	178.34407	5	244.5653	2

Categorical variable

Dependent variable

Data and algorithm

- Pre-processing

```
age_out=which(data$Age%%1≠0)
card_out=which(data$Num_Credit_Card%%1≠0)
account_out=which(data$Num_Bank_Accounts%%1≠0)
inquire_out=which(data$Num_Credit_Inquiries%%1≠0)
delay_out=which(data$Num_of_Delayed_Payment%%1≠0 | data$Num_of_Delayed_Payment<0)
date_out=which(data$Delay_from_due_date<0)
data_out=Reduce(union,list(age_out,card_out,account_out,inquire_out,delay_out,date_out))
data=data[-data_out,]

data$Occupation=factor(data$Occupation)
data$Type_of_Loan=factor(data$Type_of_Loan)
data$Interest_Rate=factor(data$Interest_Rate)
data$Credit_Score=factor(data$Credit_Score)
data$Payment_of_Min_Amount=factor(data$Payment_of_Min_Amount)
data$Payment_Behaviour=factor(data$Payment_Behaviour)
```

Remove NA's and wrong collected data

Change categorical variables into factor

Data and algorithm

- EDA: summary of data and special findings

1. The response is very unbalanced

```
Credit_Score
0:17902
1:33964
2:11554
```

We need to balanced the training data to avoid the overfitting.

2. The range of month balance and income are very large

```
 Annual_Income        Monthly_Balance
 Min.   :    7006    Min.   :   0.0886
 1st Qu.:   19645    1st Qu.: 272.0805
 Median :   38107    Median : 341.6435
 Mean   :  179116    Mean   : 406.2164
 3rd Qu.:   73850    3rd Qu.: 474.1676
 Max.   :24198062    Max.   :1602.0405
```

Data and algorithm

- EDA: summary of data and special findings

3. There are some categorical variables that exist more than 32 categories

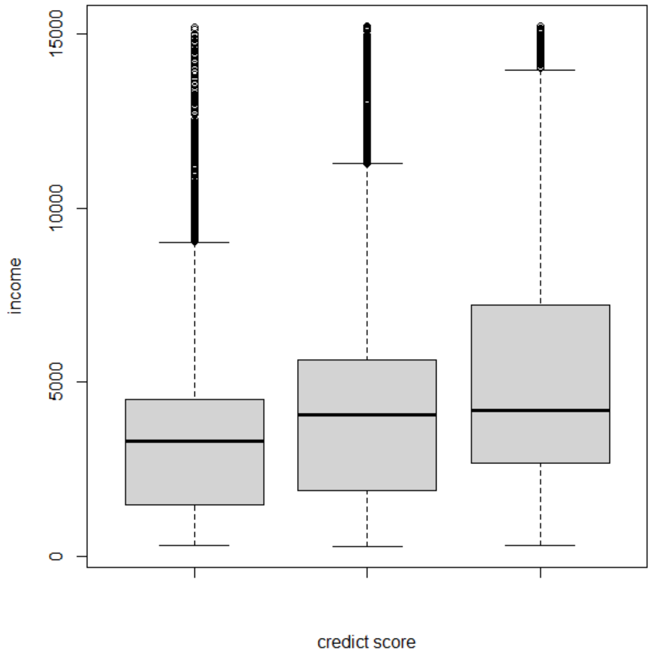
We need to use modules other then tree when fitting classification tree.

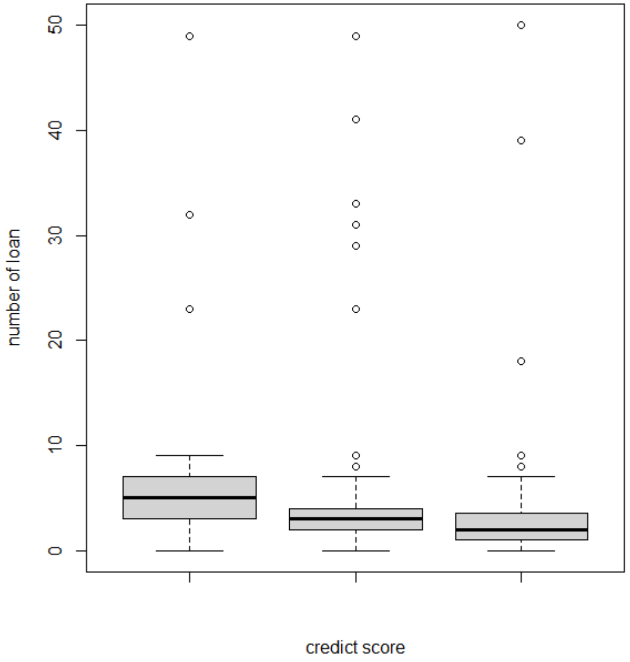
```
  Occupation
 12       : 4456
 7        : 4192
 4        : 4036
 13       : 4024
 1        : 4013
 0        : 4000
 (Other):38682
```

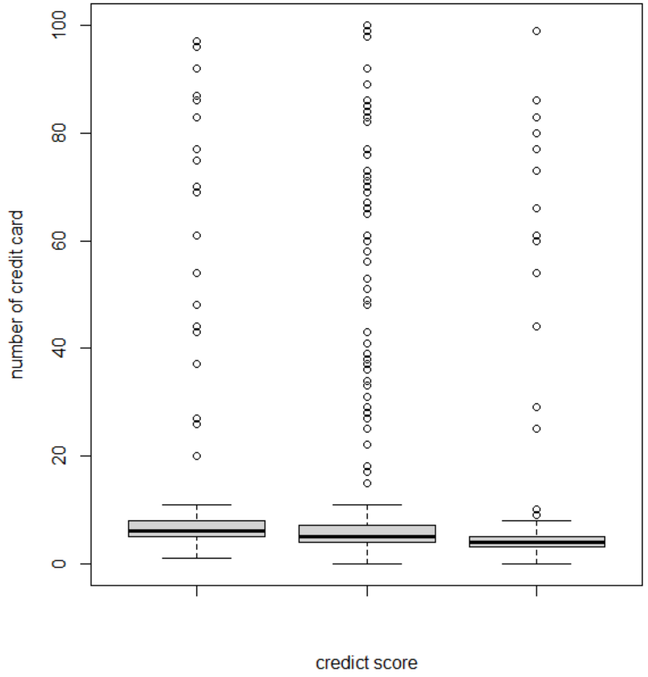
```
  Type_of_Loan
 6260    : 7397
 3463    :  919
 4878    :  859
 5591    :  851
 684     :  845
 1410    :  838
 (Other):51694
```

Data and algorithm

4. There are a lot difference in income,credit card and loan of different credit score







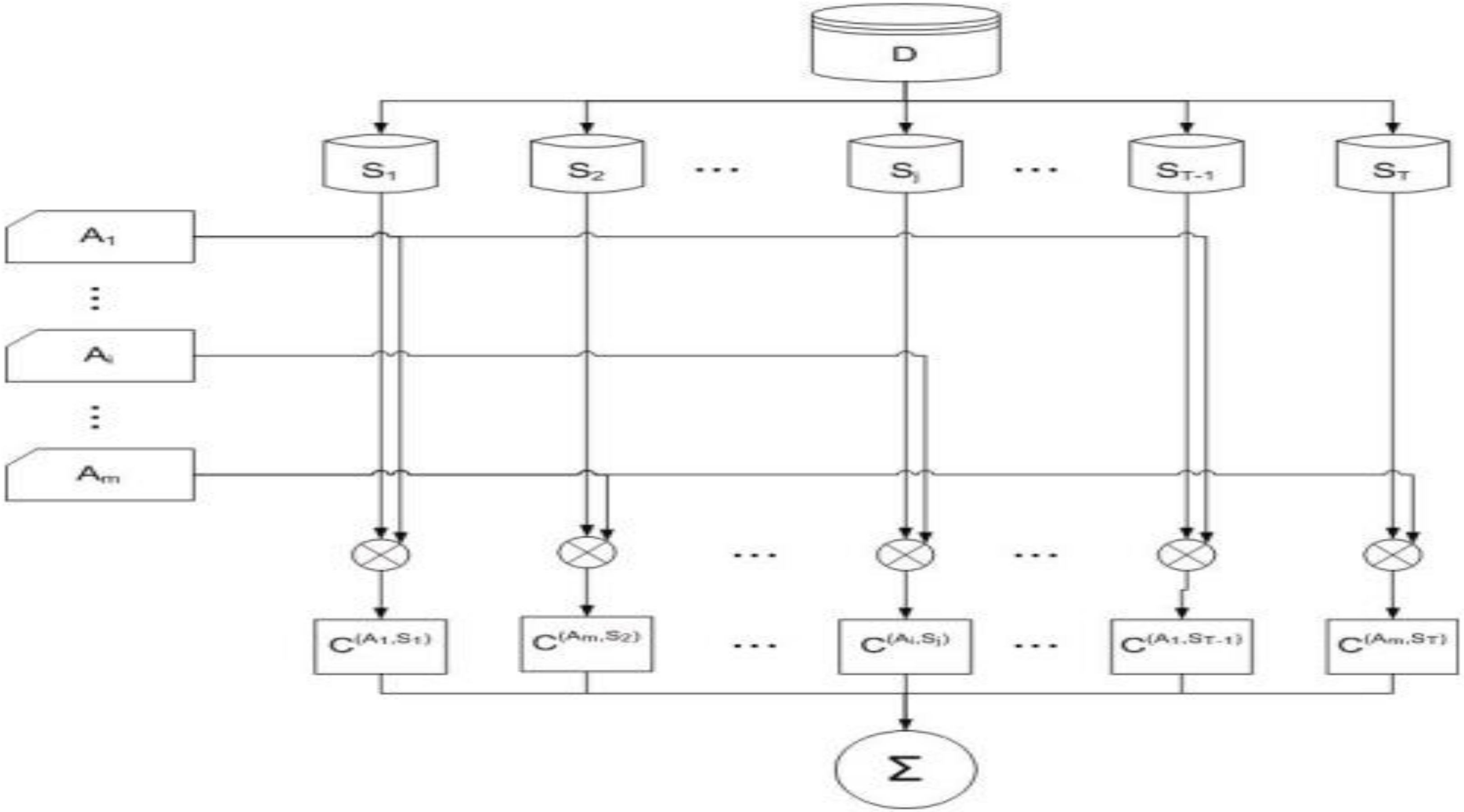
Data and algorithm

- Algorithm

1. Train data: contains 2400 data with each label are equality counted
2. Test data: 61003 rows of data
3. Variables: 22 independent variables and one dependent variables
4. Algorithm bag:

SVM model, Classification tree, Logistic regression, LDA

Data and algorithm



Data and algorithm

- Algorithm

Algorithm 1 (Framework for bagging with multi-algorithm ensembles)

1. *Initialize* $C^* = \emptyset$
2. *For* $j = 1$ *to* T
3. $\quad S_j \leftarrow Bootstrap(D, p = 1)$
4. $\quad$ *Select an algorithm* A_i *from the given bag of algorithms* M
5. $\quad C_i \leftarrow \mathbf{C}(S_i, A_i)$, *where* $\mathbf{C}$ *is a procedure, e.g. API, to create a classifier*
6. $\quad C^* \leftarrow C^* \cup C_i$
7. *End for*

Sub-model1: SVM classifier

Choose 400 data as validation set to choose best cost and gamma

↓

Fitting the model

↓

Perform 10 classifiers to see the divergence

```r
library(e1071)
set.seed(108021186)
tune.out <- tune(svm, Credit_Score ~ ., data = train[1:400,], kernel = "linear",
    ranges = list(cost=c(0.1,1,10,100,1000,10000,100000)))
summary(tune.out)
```

R Console

data.frame 1 x 1

data.frame 7 x 3

cost <dbl>	error <dbl>	dispersion <dbl>
1e-01	0.3600	0.06687468
1e+00	0.4350	0.06032320
1e+01	0.4850	0.07564537
1e+02	0.4975	0.06503204
1e+03	0.4950	0.06324555
1e+04	0.4950	0.06324555
1e+05	0.4950	0.06324555

Sub-model1: SVM classifier

Choose 400 data as validation set to choose best cost and gamma

Fitting the model

Perform 10 classifiers to see the divergence

```r
library(e1071)
set.seed(108021186)
svm_result=1:61003
for (i in 1:20){
    boot=train[sample(1:2400,2400,replace=T),]
    svmfit<-svm(Credit_Score~.,data=boot,kernel="linear",
    cost=0.1,scale = FALSE)
    svm_result=cbind(svm_result,predict(svmfit,test[,-23]))
    print(i)
}
svm_result_mod=svm_result[,-1]-1
```

Sub-model1: SVM classifier

Choose 400 data as validation set to choose best cost and gamma

Fitting the model

Perform 10 classifiers to see the divergence

Aggregate 20 classifiers using simple mean

Accruacy:48.27%

```
result=function(mat){
  resul=NULL
  for(i in 1:nrow(mat)){
    a0=sum(mat[i,]==0)
    a1=sum(mat[i,]==1)
    a2=sum(mat[i,]==2)
    res=which.max(c(a0,a1,a2))
    resul[i]=res-1
  }
  return(resul)
}
```
```

```{r}
res=result(svm_result_mod)
table(res,test$Credit_Score)
```
```

res	0	1	2
0	5578	5381	531
1	9837	19397	5744
2	1680	8382	4473

Sub-model1: SVM classifier

Choose 400 data as validation set to choose best cost and gamma

Fitting the model

Perform 10 classifiers to see the divergence

Correlation matrix with first 10 classifiers

1.000	0.506	0.199	0.460	-0.235	0.220	-0.096	-0.378	0.193	0.288
0.506	1.000	0.278	0.614	-0.560	0.446	-0.337	-0.648	0.158	0.374
0.199	0.278	1.000	0.062	-0.418	-0.027	0.236	-0.488	0.539	-0.091
0.460	0.614	0.062	1.000	-0.316	0.490	-0.421	-0.319	-0.154	0.471
-0.235	-0.560	-0.418	-0.316	1.000	-0.297	0.003	0.703	-0.213	-0.162
0.220	0.446	-0.027	0.490	-0.297	1.000	-0.452	-0.215	-0.193	0.372
-0.096	-0.337	0.236	-0.421	0.003	-0.452	1.000	-0.290	0.501	-0.430
-0.378	-0.648	-0.488	-0.319	0.703	-0.215	-0.290	1.000	-0.453	-0.019
0.193	0.158	0.539	-0.154	-0.213	-0.193	0.501	-0.453	1.000	-0.170
0.288	0.374	-0.091	0.471	-0.162	0.372	-0.430	-0.019	-0.170	1.000

Sub-model2: Classification tree

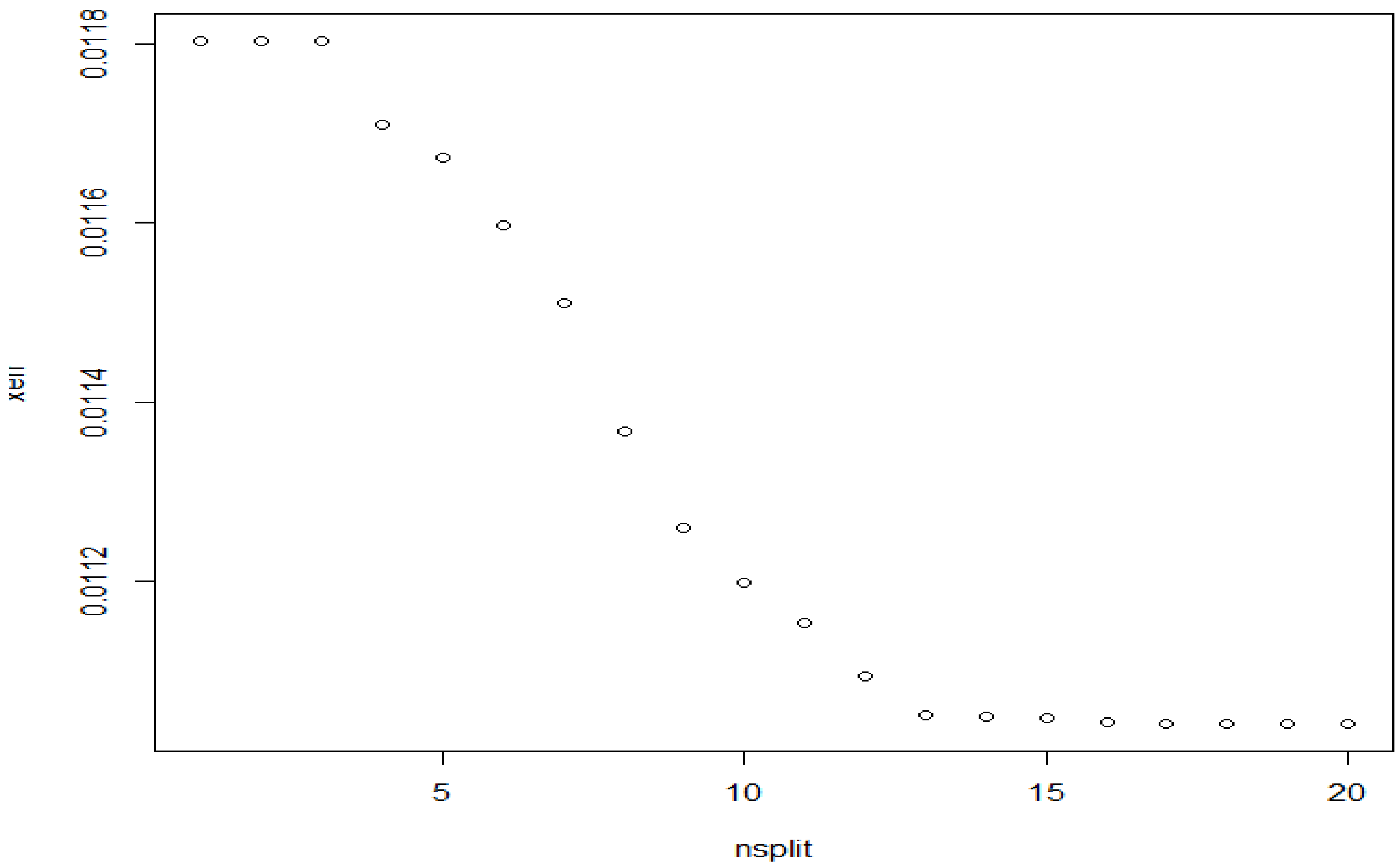
Choose 400 data as validation set to choose best cost and gamma

↓

Fitting the model

↓

Perform 10 classifiers to see the divergence



Sub-model2: Classification tree

Choose 400 data as validation set to choose best cost and gamma

Fitting the model

Perform 10 classifiers to see the divergence

Choose max.depth=7

Aggregate 20 classifiers using simple mean

```
library(rpart)
set.seed(108021186)
tree_result=1:61003
for (i in 1:20){
    boot=train1[sample(1:2400,2400,replace=T),]
    treefit<-rpart(Credit_Score~.,data = boot, maxdepth=10, cp=0)
    tree_result=cbind(tree_result,predict(treefit,test1[,-1],type='class'))
    print(i)
}
tree_result_mod=tree_result[,-1]-1
```

sample(x, size, replace = FALSE, prob = NULL)

Accruacy:67.40%

agg_tree	0	1	2
0	12139	6429	508
1	2853	20348	1610
2	2103	6383	8630

Sub-model2: Classification tree

Choose 400 data as validation set to choose best cost and gamma

Fitting the model

Perform 10 classifiers to see the divergence

Correlation matrix with first 10 classifiers

1.000	0.356	0.342	0.330	0.310	0.321	0.330	0.329	0.334	0.357
0.356	1.000	0.404	0.377	0.388	0.324	0.404	0.409	0.407	0.409
0.342	0.404	1.000	0.369	0.387	0.353	0.394	0.375	0.396	0.395
0.330	0.377	0.369	1.000	0.339	0.339	0.388	0.368	0.361	0.347
0.310	0.388	0.387	0.339	1.000	0.359	0.379	0.382	0.364	0.366
0.321	0.324	0.353	0.339	0.359	1.000	0.362	0.319	0.365	0.328
0.330	0.404	0.394	0.388	0.379	0.362	1.000	0.373	0.395	0.399
0.329	0.409	0.375	0.368	0.382	0.319	0.373	1.000	0.381	0.393
0.334	0.407	0.396	0.361	0.364	0.365	0.395	0.381	1.000	0.388
0.357	0.409	0.395	0.347	0.366	0.328	0.399	0.393	0.388	1.000

Sub-model3: Logistic regression

Fitting the model

Perform 10 classifiers to see the divergence

```
library(nnet)
set.seed(108021186)
lr_result=1:61003
for (i in 1:20){
    boot=train[sample(1:2400,2400,replace=T),]
    lrfit<-nnet::multinom(Credit_Score ~., data = train)
    lr_result=cbind(lr_result,predict(lrfit,test[,-20],type='class'))
    print(i)
}
lr_result_mod=lr_result[,-1]-1
```

Accruacy:60.16%

res	0	1	2
0	9363	6213	310
1	4840	18498	1597
2	2892	8449	8841

Sub-model3: Logistic regression

Fitting the model

Perform 10 classifiers to see the divergence

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]	[,9]	[,10]
[1,]	1.000	0.810	0.802	0.830	0.839	0.801	0.839	0.829	0.828	0.828
[2,]	0.810	1.000	0.793	0.824	0.825	0.800	0.814	0.791	0.833	0.827
[3,]	0.802	0.793	1.000	0.824	0.805	0.787	0.814	0.777	0.803	0.801
[4,]	0.830	0.824	0.824	1.000	0.833	0.840	0.834	0.840	0.868	0.833
[5,]	0.839	0.825	0.805	0.833	1.000	0.820	0.839	0.814	0.841	0.867
[6,]	0.801	0.800	0.787	0.840	0.820	1.000	0.839	0.792	0.845	0.837
[7,]	0.839	0.814	0.814	0.834	0.839	0.839	1.000	0.795	0.855	0.842
[8,]	0.829	0.791	0.777	0.840	0.814	0.792	0.795	1.000	0.832	0.802
[9,]	0.828	0.833	0.803	0.868	0.841	0.845	0.855	0.832	1.000	0.849
[10,]	0.828	0.827	0.801	0.833	0.867	0.837	0.842	0.802	0.849	1.000

Sub-model4: QDA classifier

Fitting the model

Perform 10 classifiers to see the divergence

```r
library(MASS)
set.seed(108021186)
qda_result=1:61003
for (i in 1:20){
    boot=train[sample(1:2400,2400,replace=T),]
    qdafit←qda(Credit_S nnet::multinom(...) ata = boot)
    qda_result=cbind(qda_result,predict(qdafit,test[,-20])$class)
    print(i)
}
qda_result_mod=qda_result[,-1]-1
```

Accruacy:51.56%

agg_qda	0	1	2
0	12749	14820	855
1	1454	10043	1228
2	2892	8297	8665

Sub-model4: QDA classifier

Fitting the model

Perform 10 classifiers to see the divergence

1.000	0.679	0.641	0.658	0.672	0.631	0.677	0.670	0.678	0.661
0.679	1.000	0.708	0.742	0.650	0.700	0.675	0.693	0.639	0.684
0.641	0.708	1.000	0.709	0.717	0.691	0.717	0.694	0.712	0.737
0.658	0.742	0.709	1.000	0.648	0.668	0.685	0.691	0.655	0.690
0.672	0.650	0.717	0.648	1.000	0.671	0.736	0.756	0.774	0.717
0.631	0.700	0.691	0.668	0.671	1.000	0.698	0.703	0.642	0.676
0.677	0.675	0.717	0.685	0.736	0.698	1.000	0.769	0.755	0.708
0.670	0.693	0.694	0.691	0.756	0.703	0.769	1.000	0.773	0.710
0.678	0.639	0.712	0.655	0.774	0.642	0.755	0.773	1.000	0.737
0.661	0.684	0.737	0.690	0.717	0.676	0.708	0.710	0.737	1.000

Aggregation process and comparison

- Comparison of different sub-model

model	Single accuracy	Aggregated accuracy	System time
SVM model	54.00%	48.27%	123.09
Tree model	61.67%	67.40%	0.16
Logistic regression	60.16%	60.16%	0.41
QDA model	54.09%	51.56%	0.06

Aggregation process and comparison

- Aggregation process

```
set.seed(108021186)
setting=sample(1:80.20)
mult_result=cbind(svm_result_mod,tree_result_mod,lr_result_mod,qda_result_mod)[,setting]
agg_mult=result(mult_result)
table(agg_mult,test$Credit_Score)
mean(agg_mult==test$Credit_Score)
round(cor_dist(mult_result[,1:10],test$Credit_Score),3)
```
```

```
agg_mult   0   1   2
       0 12267 8220   400
       1 2039 17302 1502
       2 2789 7638 8846
[1] 0.6297231
```
```

Aggregation process and comparison

- Aggregation process

Correlation matrix

1.000	0.055	0.332	0.286	0.021	0.041	0.075	0.053	0.316	0.288
0.055	1.000	0.317	0.168	0.347	-0.090	0.365	0.339	0.319	0.309
0.332	0.317	1.000	0.237	0.320	-0.126	0.360	0.314	0.756	0.686
0.286	0.168	0.237	1.000	0.191	0.003	0.185	0.143	0.277	0.216
0.021	0.347	0.320	0.191	1.000	-0.041	0.398	0.345	0.310	0.313
0.041	-0.090	-0.126	0.003	-0.041	1.000	-0.073	-0.047	-0.117	-0.177
0.075	0.365	0.360	0.185	0.398	-0.073	1.000	0.361	0.353	0.362
0.053	0.339	0.314	0.143	0.345	-0.047	0.361	1.000	0.309	0.317
0.316	0.319	0.756	0.277	0.310	-0.117	0.353	0.309	1.000	0.687
0.288	0.309	0.686	0.216	0.313	-0.177	0.362	0.317	0.687	1.000

Conclusion

- Discussion

1. Lower correlation between classifiers would be more reasonable to aggregate them using simple mean.
2. With multi-algorithm, it seems to lower the performance with just using tree model. But it become more legitimate.
3. We also need to pay attention to time cost while using multi-algorithm bagging model.