Supervised learning report

Class: Artificial Neural Network

Name: Yu-Hsuan TING

Date: 2019/10/20

Contents

1. Introduction

Here I introduce a dataset "Churn_Modeling.csv" from Kaggle "Bank Customer Churn Modeling" (https://www.kaggle.com/barelydedicated/bank-customer-churn-modeling). Since I am sure that I am predicting a category "Exited" and "Stay" with value 1 and 0 in the exited column, it is a supervised learning for classification problem.

This project would only discuss the supervised learning method for classification. First of all, I would analyze and understand my data overall. Secondly, I would preprocess my data. In this data preparation stage, I would decide whether I would change my columns (group categorical columns, drop some columns) also to encode and scale my data. I would already in the stage split my data into training (70%), validation (15%) and test (15%) set for the decision for this stage. The final decision for this stage for me is to check the simple models (with default setting from the library) and see the performance, here I would use only the training dataset for evaluation.

In the third step I have already my chosen data sets (training, validation and test). I would use the training set to tune the parameters of all the models I've chosen to use. In this report I choose, logistic regression, gradient bosting classifier, random forest, SVM and ANN models. Using grid search alone with K-fold cross validation I can choose the best parameters for all models.

Forth step the validation (encoded and scaled on the second step) dataset would get involve to help me choose the best performance model. Finally the encoded and scaled test dataset would be use for the final testing to see if my model is consider a good model and make a conclusion for my project.

Before we jump into my project, there's two points I need to think before starting. How do I measure the performance of my model?

1.1. Performance criteria

I wrote a function for my performance criteria. There's three terms in confusion matrix I would need to specified before I check the performance of my model: Accuracy rate, Recall rate and Precision rate. The result of my data is shown below. With 1 means that the client would leave and 0 means the client would stay.

		PREDICTION	
		0	1
TRUE	0	True Negative	False Positive
	1	False Negative	True Positive

- Accuracy rate: (TP+TN)/(all data)
- Recall rate: TP/(TP+FN)
- Precision rate: TP

Here true positive is when we determinate the client is leaving when they are actually leaving. The problem of only looking at the accuracy rate is that I might tend to correctly predict only one class then other class if my data is imbalanced. That's why I need to look into Recall and Precision rate. I would only focus on one of them. So what's the pros and cons of having a higher recall or precision rate?

Higher Recall rate here means that I correctly classified most churners, it might be good when a customer leave would affect more the revenue. Whereas higher precision rate means that I would have fewer false alarm, so that it would reduce the marketing and communication cost on the wrong target. Here I assume the company would be affect more is the lost of revenue. That is I would prefer to improve my recall rate of the model.

1.2. Potential problems to be solved

There are many problems might need to be careful about my project:

a) Be careful about the imbalanced data, not just check the accuracy rate.
b) Should I group some categorical data?
c) Can I drop some data without losing the information from the variables?
d) How would scaling approach affect my data?

All those questions I would be solving during the whole project.

2. Dataset information

After dropping the columns that we would not need for modeling. (RowNumber, CustomerId and Surname.) So that we have a full data of 10,000 rows and 11 variables.

Numerical: need to be scaled

Columns	Mean	Std	Min	Max	25%	50%	75%
CreditScore	650.53	96.65	350	850	584	652	718
Age	38.92	10.49	18	92	32	37	44
Balance	76485.89	62397.41	0	250898.09	0	97198.54	127644.24
EstimatedSalary	100090.24	57510.49	11.58	199992.48	51002.11	100193.92	149388.25

Categorical: might need to encode

Columns	Unique value	Comment
Geography	France, Spain, Germany	Encoding needed
Gender	Femal, Male	Encoding needed
Tenure	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10	Might be good to group them
NumOfProducts	1, 2, 3, 4	Might be good to group them
HasCrCard	1, 0	1: has credit care
IsActiveMember	1, 0	1: Is active member
Exited	1, 0	1: exited (target value)

Target percentage is imbalance, almost 80% of the exited column value is 0 which is dangerous for the model to predict most of the result to be 0.

- From the individual columns information: (see appendix)

  In this dataset we have more people(50%) from France, more people (70%) has credit card. Gender and IsActiveMember are more or less balance. Numeric variables has a various range that should be scaled.

  Assumption for the next step to check:

a) Grouping Tenure might be a good idea, since there are not extreme values in the data.
b) For the NumOfProducts since 3 and 4 are very low, I can group them together or even drop them.

- From the comparison of the columns comparing to exited column. (see appendix)

  Here we get the assumption that the exited clients are: (will be check in the final model)

  a) Older (in average and medium)
  b) Higher balance (according to the box chart we are not sure for the assumption)
  c) Less an active member
  d) Higher salary (according to the box chart we are not sure for the assumption)

3. Data preprocessing

In this stage I would have my final datasets by deciding whether I should group my columns. What encoder should I use? If I am using one hot encoding should I drop some columns? What Scaler should I use for scaling the data?

There's 3 main step for my data preparation stage: group, encode and scale. In each step I use simple models to check the performance. Choice for each stage is shown as the below table:

Group (see appendix)	Encode (see appendix)	Scaler (see appendix)
For Tenure and NumOfProduct • Group • Not group	• One hot encoding • Label encoding • Target encoding	• MinMaxScaler • RobustScaler • StandardScaler

Eventually I've chosen to group both columns, use one hot encoding and standard scaler for my final dataset. With this dataset I can perform model parameter tuning.

4. Model parameter tuning

In this stage I'll use the training set to tune the parameters of each methods below. Then use the validation set to choose the model

- Logistic regression
- SVM (maybe I should use the original data set)
- Gradient Boosting
- Random forest
- KNN
- ANN

Use param_grid search to find the best parameter for the models except for ANN.

Logistic regression	Changed parameters: (multi_class = 'ovr', penalty= 'l1', solver= 'liblinear'). Slightly improvement in performance (0.02% for accuracy, 0.2% for recall rate and 0.13% for precision rate).
SVM	SVC, LinearSVC: keep the default settings.

Gradient Boosting Classifier	With the new model (loss='exponential',max_depth=5,max_features= None), although the accuracy drop a little (0.2%) but the recall rate increase 1.7%.
Random Forest	Only the parameter criterion change to entropy. Recall rate increase 7%.
KNN	Leaf size change to 1 with p change to one. The accuracy rate increase vy 0.13% where as the
ANN	For ANN the decision start from one stage to another.

5. Model selection and changes

With the models of what I've chosen, I use the validation set to fit in and see the prediction result, the accuracy rate seems high but my initial assumption is to get a high recall rate as well. I see that all my model's recall rate are lower then 50% that means that I am not performing well in my model due to the imbalance data.

Therefore I've chosen an easy way to resample my data, that is oversampling. This method would help me get a better balance in my data. Just by doing oversampling to my data, my overall recall rate on my validation set has improve significantly without changing the model.

```
LR
validation Acc:  84.53
validation Recall:  42.44
validation percision:  71.35
SVC
validation Acc:  84.27
validation Recall:  36.01
validation percision:  75.17
LSVC
validation Acc:  84.27
validation Recall:  36.66
validation percision:  74.51
GB
validation Acc:  84.27
validation Recall:  45.98
validation percision:  67.77
KNN
validation Acc:  82.93
validation Recall:  39.23
validation percision:  64.55
ANN
 Avg Acc:  85.87
 Avg Recall:  45.98
 Avg Percision:  76.47
```

```
LR
validation Acc:  77.94
validation Recall:  77.98
validation percision:  78.05
SVC
validation Acc:  85.18
validation Recall:  82.15
validation percision:  87.56
LSVC
validation Acc:  78.02
validation Recall:  78.57
validation percision:  77.85
GB
validation Acc:  89.66
validation Recall:  86.82
validation percision:  92.12
KNN
validation Acc:  78.95
validation Recall:  70.39
validation percision:  85.08
ANN
 Avg Acc:  83.63
 Avg Recall:  81.15
 Avg Percision:  85.5
```

Eventually, I've decided to retune all my models with the new oversampling dataset. With the new data set I've also retune all the model using the new training set then selected ANN with 2 layer and classification boost classifier.

6. Summary for final model testing

Finally I've tested the final 2 model with the test set and see the result. Eventually I realized that if I split the data after oversampling reshaping technique the result would be great, but if I take the original test set to see the result. The accuracy rate is very low for both especially for gradient boost classifier. (as shown below)

```
ANN2
oversampling test result (accuracy rate, recall rate, precision rate):
(0.8694014231896191, 0.8344709897610921, 0.8923357664233577)
Real test result (accuracy rate, recall rate, precision rate):
(0.574, 0.8945578231292517, 0.3019517795637199)
GB
oversampling test result (accuracy rate, recall rate, precision rate):
(0.9133528673084973, 0.8694539249146758, 0.9496738117427772)
Real test result (accuracy rate, recall rate, precision rate):
(0.196, 1.0, 0.196)
```

In summary for this project, I've first understand my data, did the data processing decisions (Whether I should group my data? Which encoder to use? Which Scaler to use?) then to get a final dataset for tune the model parameters (Logistic regression, SVM, Random forest, Gradient boosting classifier).

After realizing my data is facing imbalanced data problem, I've chosen to use oversampling to solve this problem and reprocess all the stage for data processing and model tuning. The performance seems to be great on the oversampling test dataset but not the original data set. That is if today we have a new imbalanced dataset my model of gradient boost tend to predict wrongly to 1 (the client would exit).

Eventually, if I would still choose for my model with ANN both with oversampling and without fit similarly to my data even before oversampling is better (as shown below). After all the problem is not fully solved. Next step for the learning process is to dig into imbalance data for the model training.

```
ANN without oversampling
Real test result (accuracy rate, recall rate, precision rate):
(0.8746666666666667, 0.5578231292517006, 0.7387387387387387)
```

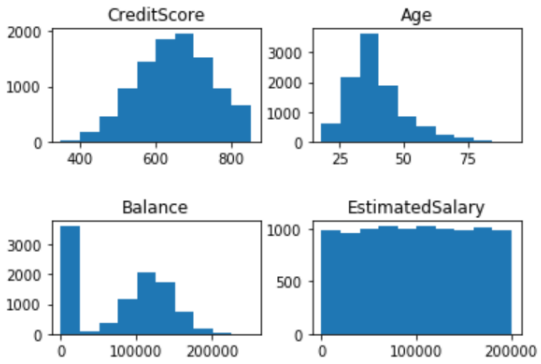
7. Appendix

7.1. 2nd Part

7.1.1. Individual columns info

```
col_0      percentage
Geography
France          50.14
Germany         25.09
Spain           24.77
col_0   percentage
Gender
Female       45.43
Male         54.57
col_0   percentage
Tenure
0             4.13
1            10.35
2            10.48
3            10.09
4             9.89
5            10.12
6             9.67
7            10.28
8            10.25
9             9.84
10            4.90
```

```
col_0          percentage
NumOfProducts
1                   50.84
2                   45.90
3                    2.66
4                    0.60
col_0      percentage
HasCrCard
0               29.45
1               70.55
col_0           percentage
IsActiveMember
0                   48.49
1                   51.51
col_0   percentage
Exited
0            79.63
1            20.37
```



In this dataset we have more people(50%) from France, more people (70%) has credit card. Gender and IsActiveMember are more or less balance. Numeric variables has a various range that should be scaled.

Assumption for the next step to check:

a) Grouping Tenure might be a good idea, since there are not extreme values in the data.
b) For the NumOfProducts since 3 and 4 are very low, I can group them together or even drop them.

7.1.2. Other columns compare to exit in

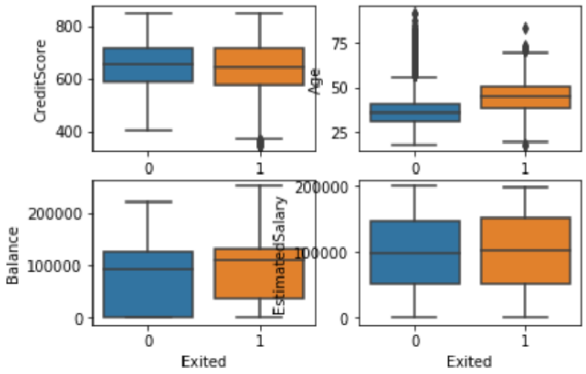
I check the average of each columns with regarding to exited:

Exited	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
0	651.853196	37.408389	5.033279	72745.296779	1.544267	0.707146	0.554565	99738.391772
1	645.351497	44.837997	4.932744	91108.539337	1.475209	0.699067	0.360825	101465.677531

More detail:

```
Geography      France    Germany      Spain
Exited
0           83.845233  67.556796  83.326605
1           16.154767  32.443204  16.673395
Gender      Female       Male
Exited
0        74.928461  83.544072
1        25.071539  16.455928
Tenure          0          1          2          3          4          5   \
Exited
0        76.997579  77.584541  80.820611  78.88999  79.474216  79.347826
1        23.002421  22.415459  19.179389  21.11001  20.525784  20.652174

Tenure          6          7          8          9          10
Exited
0        79.731127  82.782101  80.780488  78.353659  79.387755
1        20.268873  17.217899  19.219512  21.646341  20.612245
NumOfProducts           1          2          3      4
Exited
0               72.285602  92.418301  17.293233    0.0
1               27.714398   7.581699  82.706767  100.0
NumOfProducts           1          2          3      4
Exited
0               72.285602  92.418301  17.293233    0.0
1               27.714398   7.581699  82.706767  100.0
HasCrCard           0          1
Exited
0           79.185059  79.815734
1           20.814941  20.184266
IsActiveMember           0          1
Exited
0                73.149103  85.730926
1                26.850897  14.269074
Exited      0      1
Exited
0        100.0    0.0
1          0.0  100.0
```



Here we get the assumption that the exited clients are: (will be check in the final model)

- Older (in average and medium)
- Higher balance (according to the box chart we are not sure for the assumption)
- Less an active member
- Higher salary (according to the box chart we are not sure for the assumption)

7.2. 3rd Part

7.2.1. Grouping the data

I created 2 dataset df_non_group and df_grouping data for two columns: Tenure and NumOfProducts. Tenure Class 1 (0, 1), Class 2 (2, 3, 4), Class 3 (5, 6, 7) and Class 4 (8, 9, 10) and NumverOfProducts merge 3 and 4 to “More products”.

The changes is shown in the graph below:

```
Tenure
0            4.13
1           10.35
2           10.48
3           10.09
4            9.89
5           10.12
6            9.67
7           10.28
8           10.25
9            9.84
10           4.90
```

```
col_0       percentage
Tenure
Class 1         0.1448
Class 2         0.3046
Class 3         0.3007
Class 4         0.2499
```

```
col_0          percentage
NumOfProducts
1                  0.5084
2                  0.4590
3                  0.0266
4                  0.0060
```

```
NumOfProducts
More products      0.0326
OneProduct         0.5084
TwoProduct         0.4590
```

7.2.2. Data encoding for both group and non-group data

I need to encode the labeled data and try 3 ways to encode the data: one hot, label and target. I would split the data into train(70%) validate(15%) and test(15%) dataset. Note that for the one hot and label encoder the output would not affect if we first split or we first encode the data, so I chose to split the data first for one hot and label encoding for easier implementation.

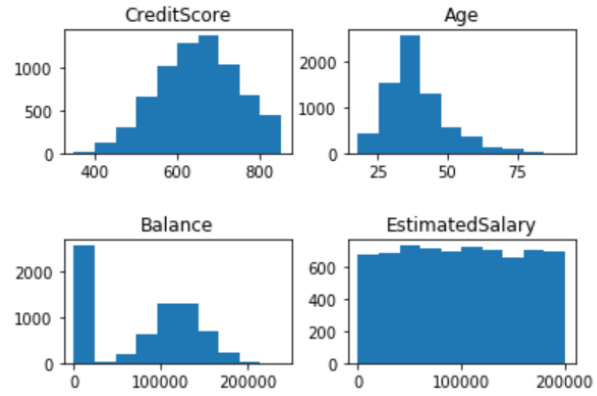
A. One hot encoding

One hot encoding is to make every category in the categorical columns an individual columns with value 1 and 0. Generally people prefer one hot encoding for ANN models, I would make a test on the different encoder.

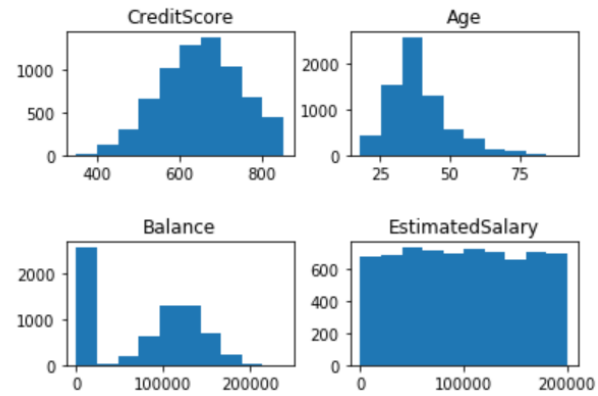
Instead of using the OneHotEncoder from sklearn, I use the panda get_dummies to keep the column names as different category for better chasing each category. I only drop the Female column after encoding.

After one hot encoding, I make sure that the distribution after encoding stays the same after splitting. What has change is the shape of the dataframe. Originally I have 10 columns for X, after one hot encoding Non-group dataset it becomes 25 columns Whereas the grouped one is 16 columns.

```
Non Group train x:
(7000, 25)
```




```
Group train x:
(7000, 17)
```



B. Label encoding

In Label encoding the shape would still be the same, only the category columns would be transfer from string to numeric, (0, 1, 2) for example, but here the value meaning would not be the same as the real numeric (0, 1, 2) therefore this approach is less appreciated.

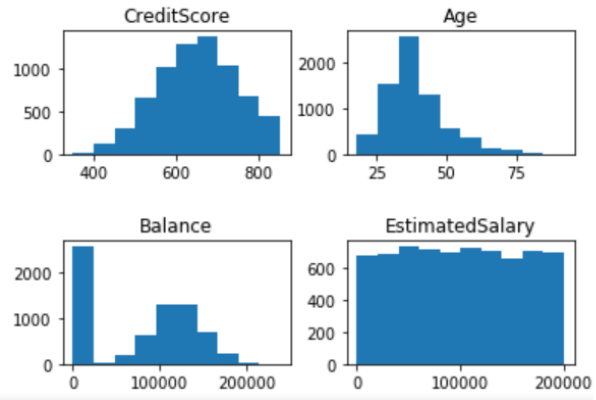
I use directly the label encoding within sklearn. The problem for this is the value is sorted by alphabet order, that is if the data has order, this should be take into account. The shape and distribution is also checked to be unchanged after splitting and encoding.

The changes for both group and non-group data is shown as below:

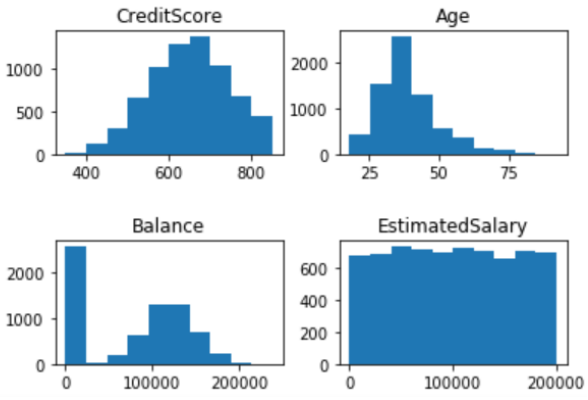
Group	Non-group
• Geography: (France, Germany, Spain) to (0,1,2) • Gender:	• Geography: (France, Germany, Spain) to (0,1,2) • Gender:

(Female, Male) to (0,1) • Tenure: class(T1,T2,T3,T4) to (0,1,2,3) • NumOfProducts: (OneProduct,TwoProducts, MoreProducts) to (0,1,2)	(Female, Male) to (0,1)

```
Non Group train x:
(7000, 10)
```




```
Group train x:
(7000, 10)
```



C. Target encoding

For target encoding, I would only use the training set and give each value a percentage in order to also apply to the validation and test set. In this case the value compare to label encoding has more meaning Therefore I split the dataset first. After encoding and splitting I as well make sure the distribution and the shape of the dataset stays the same.

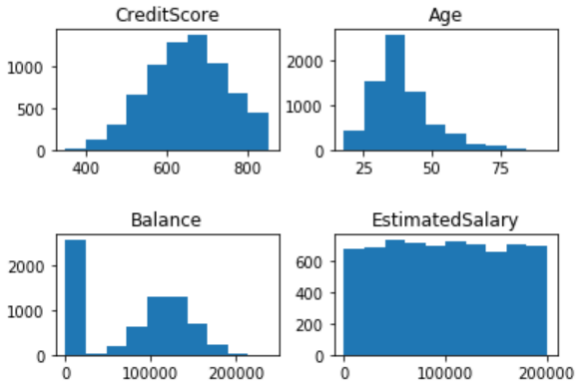
Non-group target encoding train dataset :

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
3144	648	24.928571	54.757143	55	10.342857	81370.07	50.485714	29.457143	51.528571	181534.04
9939	693	24.928571	45.242857	57	9.728571	0.00	46.200000	70.542857	51.528571	135502.77
7925	586	24.928571	45.242857	33	10.128571	0.00	46.200000	70.542857	51.528571	168261.40
309	438	24.900000	54.757143	31	10.300000	78398.69	50.485714	70.542857	48.471429	44937.01
9415	768	24.900000	45.242857	43	9.971429	129264.05	46.200000	29.457143	48.471429	19150.14

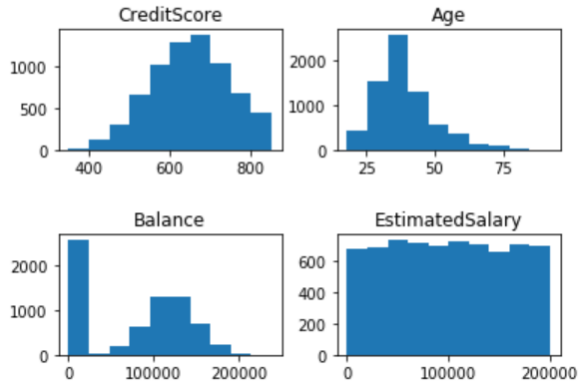
Group target encoding train dataset :

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
3144	648	24.928571	54.757143	55	14.657143	81370.07	50.485714	29.457143	51.528571	181534.04
9939	693	24.928571	45.242857	57	25.071429	0.00	46.200000	70.542857	51.528571	135502.77
7925	586	24.928571	45.242857	33	30.085714	0.00	46.200000	70.542857	51.528571	168261.40
309	438	24.900000	54.757143	31	25.071429	78398.69	50.485714	70.542857	48.471429	44937.01
9415	768	24.900000	45.242857	43	30.185714	129264.05	46.200000	29.457143	48.471429	19150.14

```
Non Group train x:
(7000, 10)
```




```
Group train x:
(7000, 10)
```



D. Decision for grouping and encoder.

I use the basic models (default Logistic regression, Gradient Boosting Classifier, Random Forest and 2 layer ANN) alone with K-fold validation to check the performance function I've defined. (Accuracy, Recall and Precision rate). From comparing the result I decided to use grouped dataset alone with Target encoding. With the targeting encoding we can really see the recall rate improved. Also note that it is normal for ANN to perform bad since we haven't yet apply scaling method to it.

Here is the return result:

```
LR
Group_onehot Avg Acc: 79.03
Group_onehot Avg Recall: 6.21
Group_onehot Avg percision: 41.32
run time:0.1940004825592041
Non_Group_onehot Avg Acc: 79.03
Non_Group_onehot Avg Recall: 6.21
Non_Group_onehot Avg percision: 41.32
run time:0.16499662399291992
Group_label Avg Acc: 79.03
Group_label Avg Recall: 6.21
Group_label Avg percision: 41.32
run time:0.1289982795715332
Non_Group_label Avg Acc: 79.04
Non_Group_label Avg Recall: 6.21
Non_Group_label Avg percision: 41.5
run time:0.11499977111816406
Group_target Avg Acc: 80.4
Group_target Avg Recall: 17.31
Group_target Avg percision: 56.92
run time:0.15399813652038574
Non_Group_target Avg Acc: 80.47
Non_Group_target Avg Recall: 17.01
Non_Group_target Avg percision: 57.57
run time:0.12400627136230469
```

```
GB
Group_onehot Avg Acc: 86.06
Group_onehot Avg Recall: 46.33
Group_onehot Avg percision: 76.25
run time:1.7689955234527588
Non_Group_onehot Avg Acc: 85.93
Non_Group_onehot Avg Recall: 46.12
Non_Group_onehot Avg percision: 75.69
run time:1.7540020942687988
Group_label Avg Acc: 86.03
Group_label Avg Recall: 45.02
Group_label Avg percision: 77.27
run time:1.1430318355560303
Non_Group_label Avg Acc: 85.9
Non_Group_label Avg Recall: 44.45
Non_Group_label Avg percision: 76.96
run time:1.1289641857147217
Group_target Avg Acc: 86.14
Group_target Avg Recall: 46.41
Group_target Avg percision: 76.7
run time:1.101003646850586
Non_Group_target Avg Acc: 86.19
Non_Group_target Avg Recall: 46.48
Non_Group_target Avg percision: 76.93
run time:1.2399945259094238
```

```
RF
Group_onehot Avg Acc: 85.43
Group_onehot Avg Recall: 45.92
Group_onehot Avg percision: 73.09
run time:2.599518060684204
Non_Group_onehot Avg Acc: 85.47
Non_Group_onehot Avg Recall: 44.46
Non_Group_onehot Avg percision: 74.4
run time:2.176041841506958
Group_label Avg Acc: 85.57
Group_label Avg Recall: 43.6
Group_label Avg percision: 75.68
run time:2.1579647064208984
Non_Group_label Avg Acc: 85.66
Non_Group_label Avg Recall: 43.8
Non_Group_label Avg percision: 75.95
run time:2.1940057277679443
Group_target Avg Acc: 85.67
Group_target Avg Recall: 45.67
Group_target Avg percision: 74.49
run time:2.46498703956604
Non_Group_target Avg Acc: 85.57
Non_Group_target Avg Recall: 44.43
Non_Group_target Avg percision: 74.92
run time:2.394003391265869
```

```
ANN
Group_onehot
Group_onehot Avg Acc: 79.43
Group_onehot Avg Recall: 0.07
Group_onehot Avg percision: 4.17
run time:64.03388381004333
Non_Group_onehot
Non_Group_onehot Avg Acc: 79.33
Non_Group_onehot Avg Recall: 0.57
Non_Group_onehot Avg percision: 17.79
run time:64.79854702949524
```

```
Group_label
Group_label Avg Acc: 79.53
Group_label Avg Recall: 0
Group_label Avg percision: 0
run time:68.29223680496216
Non_Group_label
Non_Group_label Avg Acc: 79.53
Non_Group_label Avg Recall: 0
Non_Group_label Avg percision: 0
run time:72.94671583175659
```

```
Group_target
Group_target Avg Acc: 79.14
Group_target Avg Recall: 1.68
Group_target Avg percision: 10.53
run time:75.28476691246033
Non_Group_target
Non_Group_target Avg Acc: 79.16
Non_Group_target Avg Recall: 0.42
Non_Group_target Avg percision: 11.26
run time:76.56782126426697
```

7.2.3. Choose scaler for the data

Here I used the selected data from the previous stage. My dataset is grouped data and target data. But since it is well known that Onehot encoding work better for ANN model, I would also scale them to see the result. Also note that for the scaler, I only use the information from training set to scale our data for validation and testing set.

Perform 3 different kind of scaler: MinMaxScaler, RobustScaler, StandardScaler. Then to compare all the data from 3 different scaler to the choose whether I should scale my data, if yes, which scaler should I choose. Robust and standard is more preferable then MinMax for the outlier issue, but it can still be a standard for comparation.

After scaling we can see that the performance has increase a lot, especially for ANN model, every value were predicted as 0 before scaling. We can see that for other model both one hot and target encoding are performing well, but for ANN, one hot encoding here perform much better then target. Then for the scaler the standard scaler is then chosen.

Performance shown as below:

LR	GB	RF
ms_onehot Avg Acc: 83.53	ms_onehot Avg Acc: 86.06	ms_onehot Avg Acc: 85.5
ms_onehot Avg Recall: 34.3	ms_onehot Avg Recall: 46.33	ms_onehot Avg Recall: 44.86
ms_onehot Avg percision: 70.1	ms_onehot Avg percision: 76.25	ms_onehot Avg percision: 74.26
run time:0.10499835014343262	run time:1.279165267944336	run time:1.6137876510620117
ms_target Avg Acc: 82.3	ms_target Avg Acc: 86.14	ms_target Avg Acc: 85.8
ms_target Avg Recall: 23.46	ms_target Avg Recall: 46.41	ms_target Avg Recall: 45.63
ms_target Avg percision: 70.16	ms_target Avg percision: 76.7	ms_target Avg percision: 75.48
run time:0.05599832534790039	run time:1.4109961986541748	run time:1.9481277465820312
rs_onehot Avg Acc: 83.39	rs_onehot Avg Acc: 86.07	rs_onehot Avg Acc: 85.49
rs_onehot Avg Recall: 35.14	rs_onehot Avg Recall: 46.4	rs_onehot Avg Recall: 45.37
rs_onehot Avg percision: 68.52	rs_onehot Avg percision: 76.28	rs_onehot Avg percision: 73.84
run time:0.06901764869689941	run time:1.378279685974121	run time:1.9828648567199707
rs_target Avg Acc: 82.3	rs_target Avg Acc: 86.14	rs_target Avg Acc: 85.57
rs_target Avg Recall: 25.21	rs_target Avg Recall: 46.41	rs_target Avg Recall: 45.06
rs_target Avg percision: 68.26	rs_target Avg percision: 76.7	rs_target Avg percision: 74.4
run time:0.06799936294555664	run time:1.1419975757598877	run time:2.56101131439209
ss_onehot Avg Acc: 83.4	ss_onehot Avg Acc: 86.06	ss_onehot Avg Acc: 85.69
ss_onehot Avg Recall: 35.42	ss_onehot Avg Recall: 46.4	ss_onehot Avg Recall: 46.12
ss_onehot Avg percision: 68.41	ss_onehot Avg percision: 76.19	ss_onehot Avg percision: 74.45
run time:0.03799796104431152	run time:1.442892074584961	run time:2.125488758087158
ss_target Avg Acc: 82.31	ss_target Avg Acc: 86.13	ss_target Avg Acc: 85.74
ss_target Avg Recall: 25.35	ss_target Avg Recall: 46.34	ss_target Avg Recall: 46.54
ss_target Avg percision: 68.24	ss_target Avg percision: 76.68	ss_target Avg percision: 74.25
run time:0.029000282287597656	run time:1.492053508758545	run time:2.212996006011963

ANN		
ms_onehot	rs_onehot	ss_onehot
ms_onehot Avg Acc: 83.44	rs_onehot Avg Acc: 85.14	ss_onehot Avg Acc: 84.87
ms_onehot Avg Recall: 31.77	rs_onehot Avg Recall: 42.19	ss_onehot Avg Recall: 39.95
ms_onehot Avg percision: 71.67	rs_onehot Avg percision: 74.17	ss_onehot Avg percision: 74.39
run time:51.830636739730835	run time:52.761894941329956	run time:49.10967993736267
ms_target	rs_target	ss_target
ms_target Avg Acc: 81.31	rs_target Avg Acc: 82.49	ss_target Avg Acc: 84.39
ms_target Avg Recall: 14.54	rs_target Avg Recall: 21.03	ss_target Avg Recall: 32.53
ms_target Avg percision: 47.71	rs_target Avg percision: 79.44	ss_target Avg percision: 78.33
run time:50.848246812820435	run time:54.151406049728394	run time:46.21852350234985