Capstone Project

Report

Machine Learning Nanodgree

Minxia Ji

-Defination-

>> Background

When a customer accept a credit card, he or she needs to agree to certain terms, such as make minimum payment by the due date listed on theircredit card statement. If the customer missed the minimum credit card payment six months in a row, his or her credit card will be in default. The credit card issuer will likely close the customer's account and report the default to the credit bureaus. But to the credit card issuer(usually a bank), the lost might be irretrievable.Eventually, usually
after a period of 90 days of nonpayment, the loan/payment is written off. Banks are required by law to maintain an account for loan loss reserves to cover these losses. Banks could reduce credit risk by conducting a credit risk analysis on credit card
applicants/holders. Banks can substantially reduce their credit risk by lending to their customers, since they have much more information about them than about others, which helps to reduce adverse selection. Checking and savings accounts can reveal how well the customer handles money, their minimum income and monthly expenses, and the amount of their reserves to hold them over financially stressful times. Banks will also verify incomes and employment history, and get credit reports and credit scores from credit reporting agencies.

>> Reference

Bank Risks
http://thismatter.com/money/banking/bank-risks.htm
Banks that make the most money, and the least, on credit card loans

http://www.creditcards.com/credit-card-news/bank-yields-loans-1276.php

>> Problem statement

Inputs:

3000 instances with 23 features and one label are included in the dataset. Most of features are numerical features, i.e. bill statement. Some of features, which represent customer information, such as gender and marriage, are categorical.

Output:

a trained model which performed well on predicting whether a customer would default or not given his/her information. Learning task:

Binary classification: detecting the clients who are likely to default or not(default:1;not default:0)

>> Solution statement

Data:

-Deal with class imbalance: try both random undersampling the majority class and oversampling minority class

-Splitting data: training set: 80% testing set 20%. In order to maintain class balance in training and testing sets, use train_test_split function in sklearn and specify a param ‘stratify’.

-Check the skewness of features and deal with it.(i.e.log transformation.)

Models:

-Classification models: employ RandomForestClassifier, GradientBoostingClassifier and LogisticRegressionClassifier.

-Train and find a best classifier and optimize it using grid search method

-Conduct predicting default based on new dataset using the classifier

>> A set of evaluation metrics

-F0.5score =$(1+0.5^2)$* (precision*recall/$(0.5^2$* precision)+recall)

Why F0.5score: our aim is to find out who will conduct default payment in the future.We hope that we could improve the precision(true positive/classified positive). Thus, we can introduce F-beta score, choose beta = 0.5 so that more emphasis is placed on precision.

-False negative rate = false negative/(true positive + false nagetive)

In our problem, if the false negative rate is low, then our model is good. Otherwise, if we have high false negative rate, which means among the clients who are going to default, we classified a lot of them as ‘not going to default’. That is extremely bad and we don’t want to see that happens.

>> Benchmark model

Detect credit card fraud by decision trees and svm:

http://www.iaeng.org/publication/IMECS2011/IMECS2011_pp442-447.pdf

-Analysis-

>> Data Explorition and Visualization

Data source:

Data:default of credit card clients Data Set

Data Source:UCI https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients

Dataset dimension:

30000 instances and 23 features are included in the dataset.

Most of features are numerical features, i.e. bill statement, and we need to check the skewness and may need to do some transformation. Some of features, which represent customer information, i.e.gender, are categorical. We need to transfer variables under these features to dummies variables.

Missing value:

```
data.isnull().values.ravel().sum()
```

```
0
```

No missing value in the dataset.

A sample of data and explainations:

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14	X15	X16	X17	X18	X19	X20	X21	X22	X23	Y
ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	PAY_5	PAY_6	BILL_AMT1	BILL_AMT2	BILL_AMT3	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT1	PAY_AMT2	PAY_AMT3	PAY_AMT4	PAY_AMT5	PAY_AMT6	default payment next month
1	20000	2	2	1	24	2	2	-1	-1	-2	-2	3913	3102	689	0	0	0	0	689	0	0	0	0	1

Features:

X1: Amount of the given credit (NT dollar): it includes both the individual consumer credit and his/her family (supplementary) credit.

X2: Gender (1 = male; 2 = female).

X3: Education (1 = graduate school; 2 = university; 3 = high school; 4 = others).

X4: Marital status (1 = married; 2 = single; 3 = others).

X5: Age (year).

X6 - X11: History of past payment. We tracked the past monthly payment records (from April to September, 2005) as follows: X6 = the repayment status in September, 2005; X7 = the repayment status in August, 2005; . . .;X11 = the repayment status in April, 2005. The measurement scale for the repayment status is: -1 = pay duly; 1 = payment delay for one month; 2 = payment delay for two months; . . .; 8 = payment delay for eight months; 9 = payment delay for nine months and above.

X12-X17: Amount of bill statement (NT dollar). X12 = amount of bill statement in September, 2005; X13 = amount of bill statement in August, 2005; . . .; X17 = amount of bill statement in April, 2005.

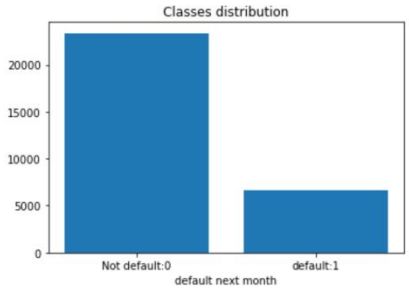
X18-X23: Amount of previous payment (NT dollar). X18 = amount paid in September, 2005; X19 = amount paid in August, 2005; . . .;X23 = amount paid in April, 2005.

Classes:

default payment next month:

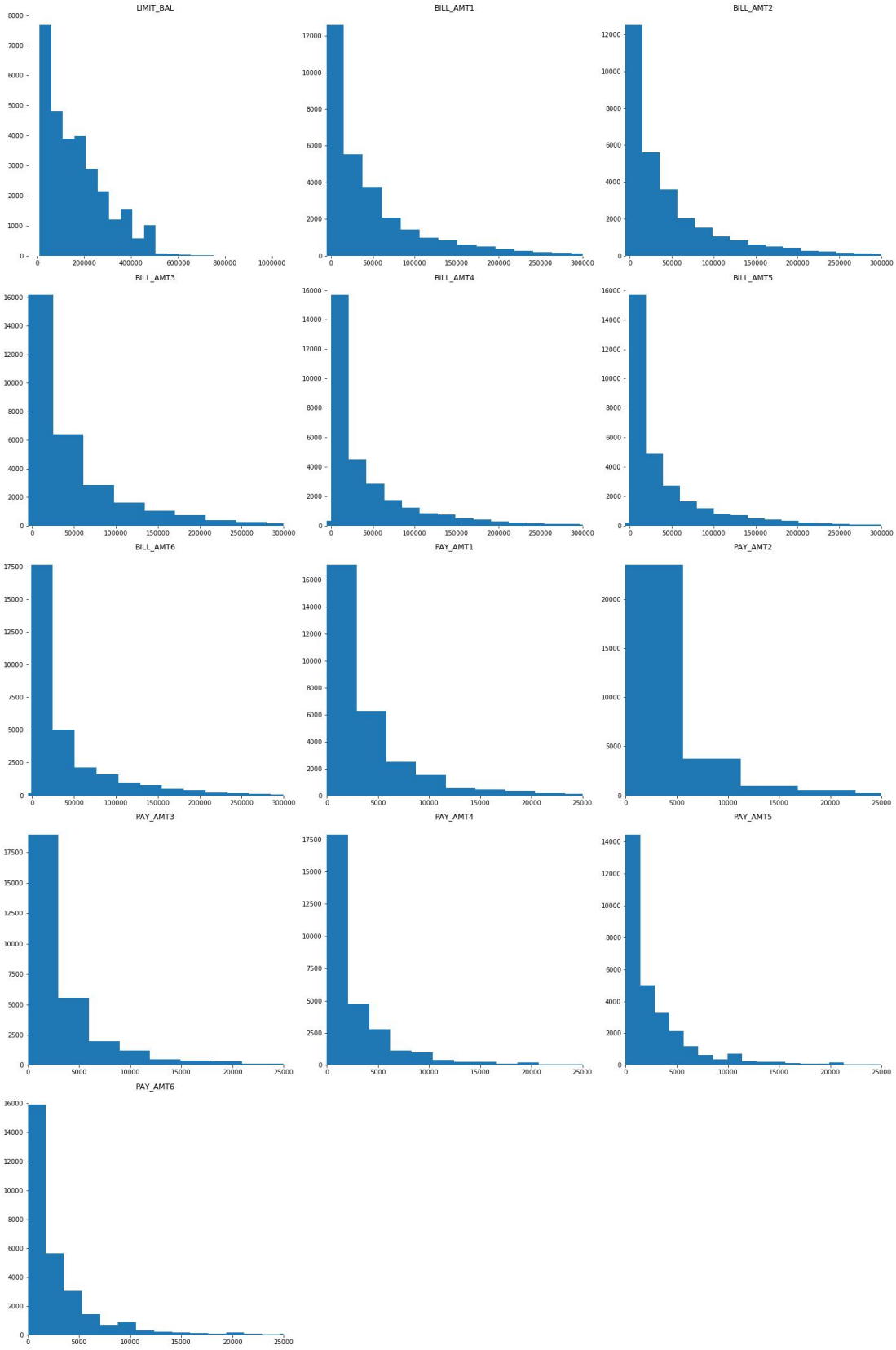
1 stands for the customer who will conduct default payment next month.

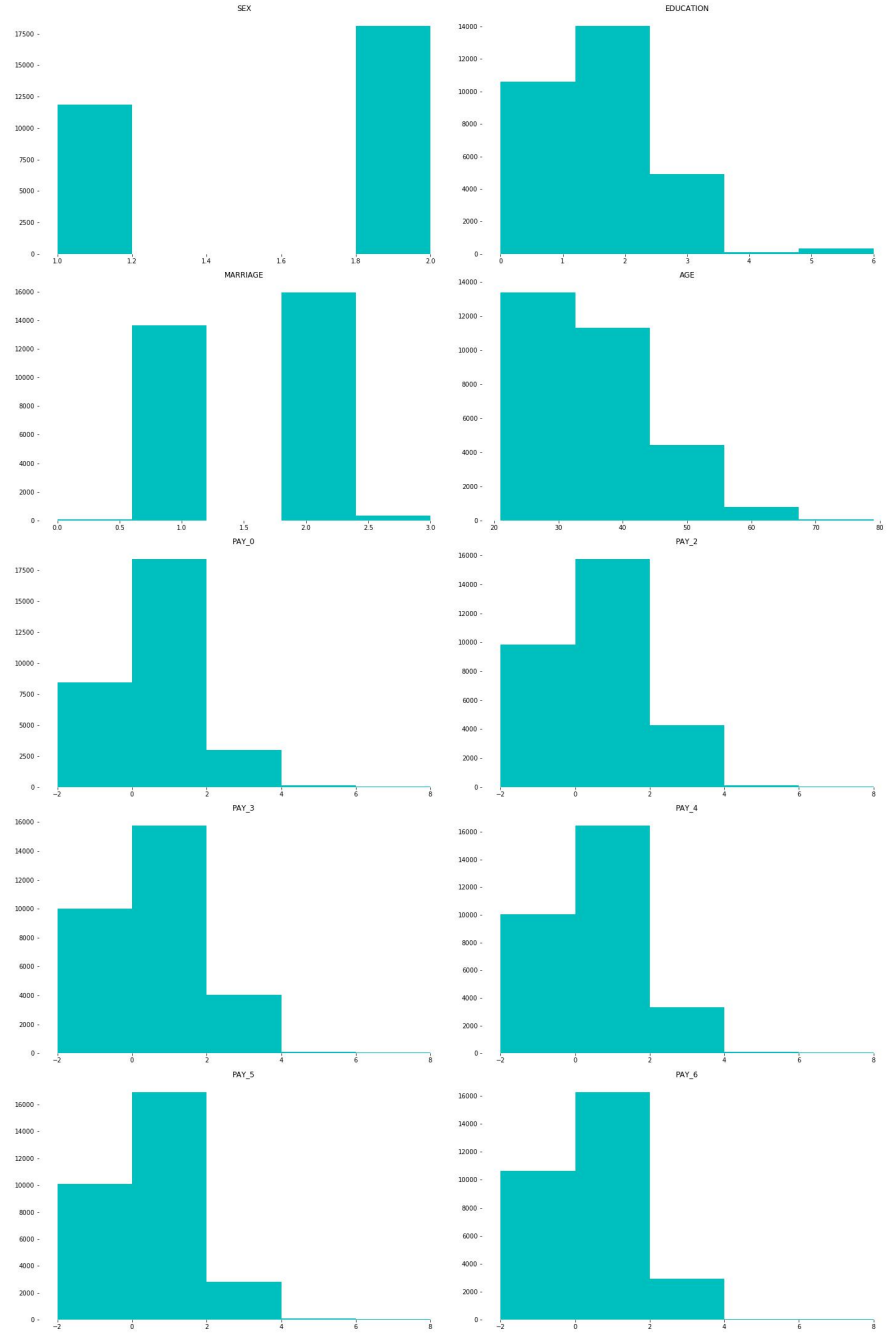
0 stands for customer who will not conduct default payment next month.



From the bar plot above, it is clear that we need to deal with class **imbalancement**: try both random undersampling the majority class and oversampling minority class.

Abnormalities:


LIMIT_BAL
BILL_AMT1
BILL_AMT2
BILL_AMT3
BILL_AMT4
BILL_AMT5
BILL_AMT6
PAY_AMT1
PAY_AMT2
PAY_AMT3
PAY_AMT4
PAY_AMT5
PAY_AMT6



Blue: numerical data

Light blue: catergorical data

From the plot above, we found these features are **right-skewed**:

'LIMIT_BAL','AGE','PAY_AMT1','PAY_AMT2','PAY_AMT3','PAY_AMT4','PAY_AMT5','PAY_AMT6'.

>> Algorithms and techniques

Classifiers:

GradientBoostingClassifier: https://en.wikipedia.org/wiki/Gradient_boosting

Strengths:

build trees one at a time, where each new tree helps to correct errors made by previously trained tree.

With each tree added, the model becomes even more expressive.

Weaknesses:

Harder to tune than other models, because you have so many hyperparameters and you can easily overfit.

Lack of interpretability, compared to linear classifiers.

Not very speedy to train or score.

Why a good candidate for the problem: Our dataset is cleaned and not very large. By employing gradient boosting model, we can conduct quick training iterations and maximize our overall f0.5beta score on the unseen testing data.

RandomForestClassifier: https://en.wikipedia.org/wiki/Random_forest

Strengths:

non parametric modelling techniques.

No assumptions to be followed in particular.

Efficient for very large datasets.

bagging technique.

Weaknesses:

very slow to create predictions once be trained

Why a good candidate for the problem: Should be very efficient to train on the dataset.

Logistic Regression: https://en.wikipedia.org/wiki/Logistic_regression

Strengths:

In Logistic regression, the explanatory variables can take any form;Linear combination of parameters and the input vectors are easy to compute. Very quick to train a model.

Weaknesses:

Can not solve non-linear problems with logistic regression since it's decision surface is linear.

Why a good candidate for the problem: Given that our prediction would be binary and the data should be linear seperateble, we could run the logistic regression quite well.

>> Benchmark Model

I found the publication for Credit Frauds Detection from Microsoft Academic.

The benchmark gives some insights to the structure of credit card data and a summary of the classification methods used to develop the classifier models of the 'credit card fraud detection', which is basically the same as what I did: default clients detection. The benchmark include the details of their approach, the results, a short discussion about the results:

TABLE III
PERFORMANCE OF CLASSIFIERS W.R.T. ACCURACY OVER TRAINING & TEST SETS

Classifier Model		Set-1F-To-1N				Set-1F-To-4N				Set-1F-To-9N			
		Train	Test	Build Time	Frauds	Train	Test	Build Time	Frauds	Train	Test	Build Time	Frauds
DT Methods	C&RT	90,01%	86,79%	<3m	254	92,13%	92,53%	<3m	242	93,85%	94,69%	<3m	216
	C5.0	92,51%	91,08%	<1m	260	97,44%	92,81%	<1m	227	99,15%	94,52%	<1m	176
	CHAID	92,51%	89,37%	<1m	252	92,92%	92,53%	<1m	233	94,32%	94,76%	<1m	165
SVM Methods	SVM with RBF Kernel	99,78%	83,02%	<74m	228	98,00%	88,97%	<74m	184	98,75%	93,08%	<74m	158
	SVM with Polynomial Kern.	99,78%	83,02%	<2m	228	98,00%	88,97%	<2m	184	98,75%	93,08%	<2m	158
	SVM with Sigmoid Kernel	99,78%	83,02%	<2m	228	98,00%	88,97%	<2m	184	98,75%	93,08%	<2m	158
	SVM with Linear Kernel	99,78%	83,02%	<2m	228	98,00%	89,18%	<2m	191	98,75%	93,08%	<2m	158

From the table above, which from the benchmark model, clearly, the evaluation metric is accuracy, which is different from mine. However, I prefer my evaluation metrics because they make more sense compared to accuracy. Nonetheless, I will compute accuracy in my final model to compare my results with the benchmark model.

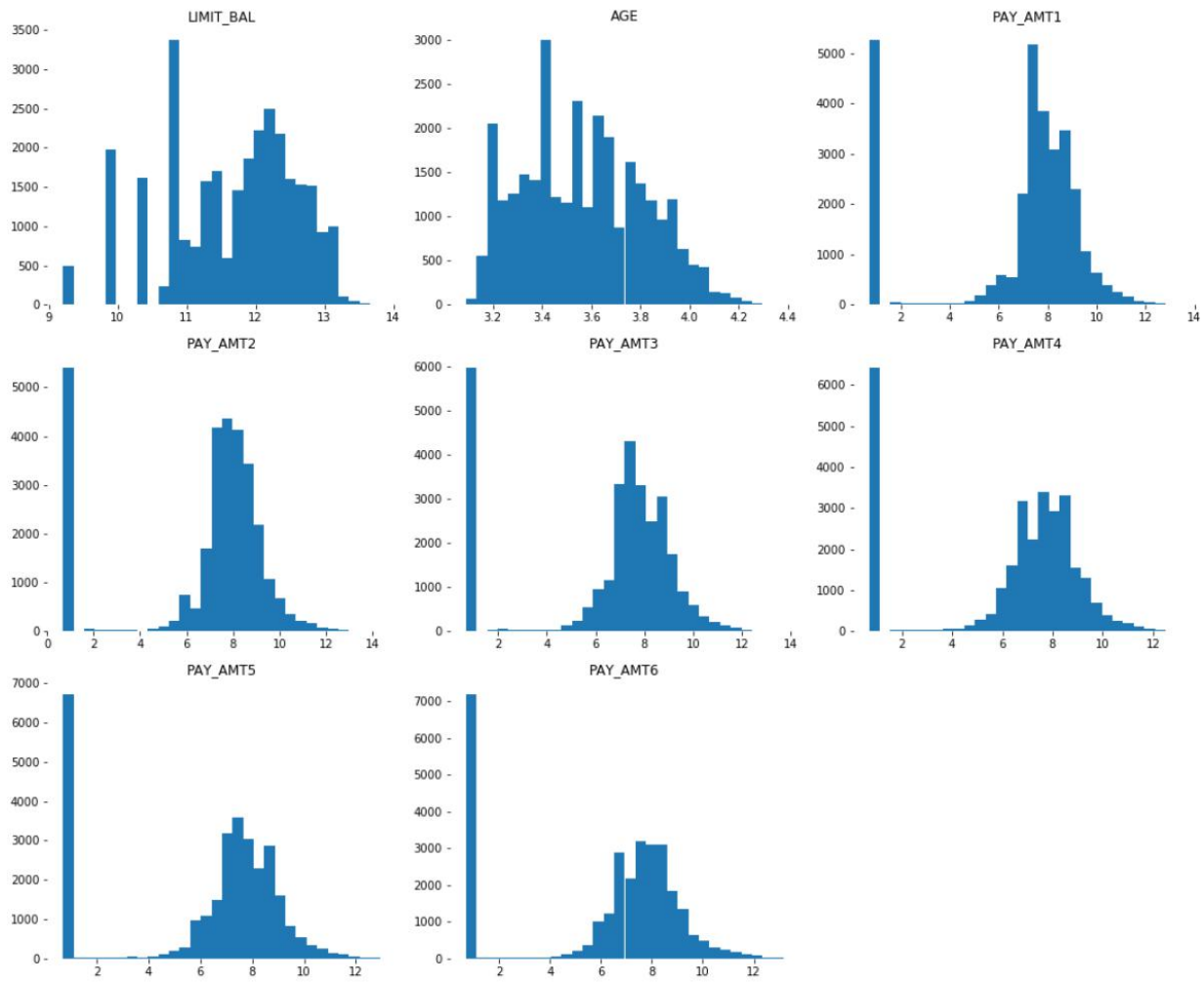
In addition, the benchmark includes a conclusion of the study and shows directions for future work. Overall, the benchmark can guide me both in the methodology part and in the report structure part.

-Methodology-

>> Data preprocessing

These features are **right-skewed**:
'LIMIT_BAL','AGE','PAY_AMT1','PAY_AMT2','PAY_AMT3','PAY_AMT4','PAY_AMT5','PAY_AMT6'.
Thus, we need to perform log transformation on them.Noticed that the min of some features I want to transform is 0, in order to conduct log transformation, I **add 1** to those features of each data point.

Skewed feature after log transformation:



From the plot above, we can see that after log transformation, the distribution of these skewed features became **normally distributed**.

Feature scaling:

Employed MinMaxScaler to do Feature scaling on numerical features.

One hot coding:

Convert catergorical variables to dummies.

>> Implementation

Libraries:

Pandas, scikit-learn, matplotlib, imblearn, etc..

How to get oversampling/undersampling dataset

-3 datasets-

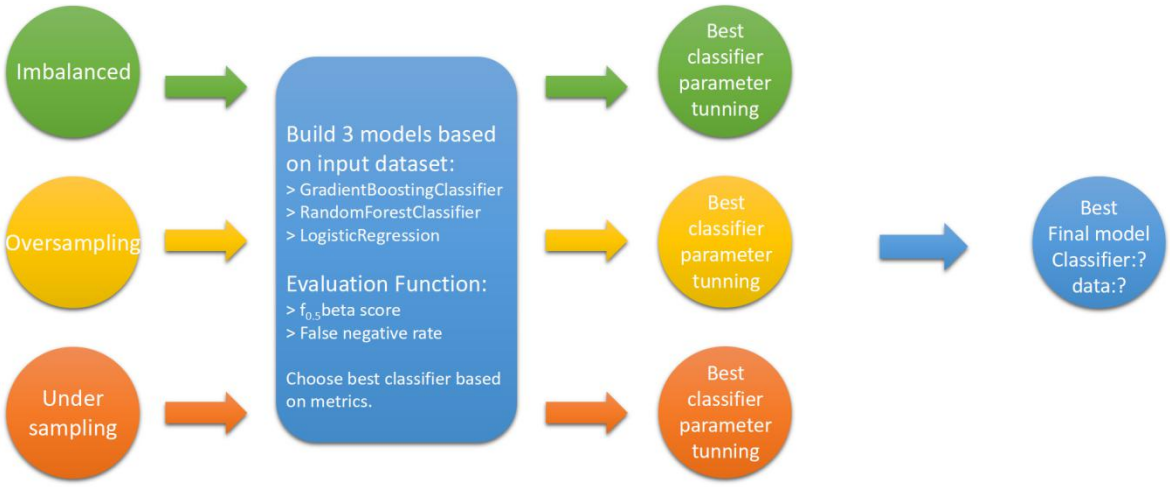
Imbalanced: Data shape: (30000, 92)	Preprocessed dataset
Oversampling Data shape: (29884, 92)	Based on training dataset splited from imbalanced dataset, use SMOTE to generate more data on minority class, which originally has 4258 instances. After oversampling, both classes have the same amount of instances: 14942 instances.
Undersampling Data shape: (8516, 92)	Based on training dataset splited from imbalanced dataset. From the indexes of majority classes, randomly choose 10684 indexes and delete the corresponding instances. After undersampling, both classes have the same amount of instances: 4258 instances.

-3 datasets: traning/testing splitting-

	imbalanced	oversampling	undersampling
training	x_train, y_train	Oversampled data from x_train, y_train	Undersampled data from x_train, y_train
validation	x_v, y_v	x_v, y_v	x_v, y_v
testing	x_test, y_test	x_test, y_test	x_test, y_test

Notice: Because we want to know whether oversampling or undersampling would have influence on the result, we need to control test dataset and only conduct oversampling or undersampling on training dataset.

Steps:



Imbalanced dataset:

Choose classifier for imbalanced dataset:

-Imbalanced dataset: performance on validation dataset-

	F0.5beta score	False Negative Rate
GradientBoostingClassifier Parameters: default	0.565013570195	0.331386861314
RandomForestClassifier Parameters: default	0.516338238962	0.385756676558
LogisticRegression Parameters: default	0.534550135701	0.377656566747

Based on the results above, we should choose GradientBoostingClassifier as our learner since the f0.5beta score is the highest among 3 classifiers, and the false negative rate is also relatively low. However the false negative rate, 0.34, is still very high. It means out of 100 customers who are going to default, we will misclassified approximately 34 customers to be not going to default, which is not good.

Therefore, we should choose **GradientBoostingClassifier** with default parameters for imbalanced dataset.

Choose for undersampling data:

-Undersampling: performance on validation dataset-

	F0.5beta score	False Negative Rate
GradientBoostingClassifier Parameters: default	0.485217391304	0.542122538293
RandomForestClassifier Parameters: default	0.448680351906	0.575707154742
LogisticRegression Parameters: default	0.456847322487	0.574297188755

we should choose **GradientBoostingClassifier** for undersampling.

Choose for oversampling data:

-Oversampling: performance on validation dataset-

	F0.5beta score	False Negative Rate
GradientBoostingClassifier Parameters: default	0.558247526751	0.392307692308
RandomForestClassifier Parameters: default	0.506178192066	0.433939393939
LogisticRegression Parameters: default	0.466704610131	0.561027837259

we should choose **GradientBoostingClassifier** for oversampling.

Notice: all of the 3 datasets share the same best model:

```
GradientBoostingClassifier(criterion='friedman_mse', init=None,
              learning_rate=0.1, loss='deviance', max_depth=3,
              max_features=None, max_leaf_nodes=None,
              min_impurity_decrease=0.0, min_impurity_split=None,
              min_samples_leaf=1, min_samples_split=2,
              min_weight_fraction_leaf=0.0, n_estimators=100,
              presort='auto', random_state=None, subsample=1.0, verbose=0,
              warm_start=False)
```

Hence, in the next part, we only need to figure out what are parameters we should tune for GBC.

>> Refinement

Parameter tuning for GBC

GBC combines a set of weak learners to improve prediction accuracy. The outcomes are weighed based on the outcomes of previous models. For the outcomes predicted correctly, GBM would give a lower weight on it and give a higher weight on the ones miss-classified.

Combine the theory of GBM , the data itself and plot above,

Here are **why these 4 parameters** I choosed to tune:

Parameter	Reason
max_depth	Max_depth is used to control over-fitting. Higher depth will allow model to learn relations

	very specific to a particular sample.
min_samples_leaf	Defines the minimum samples (or observations) required in a terminal node or leaf. Used to control over-fitting. Generally lower values should be chosen for imbalanced class problems. In our problem, the class is slightly imbalanced so by adjusting this parameter, we could prevent overfitting hopefully.
learning_rate	Learning_rate determines the impact of each tree on the final outcome. GBM works by starting with an initial estimate which is updated using the output of each tree. The learning parameter controls the magnitude of this change in the estimates. Lower values are generally preferred as they make the model robust to the specific characteristics of tree and thus allowing it to generalize well. Lower values would require higher number of trees to model all the relations and will be computationally expensive.
n_estimators	The number of sequential trees to be modeled .Usually GBM is fairly robust at higher number of trees.

Prameter tunning method:

Gridsearch: initilizing different values for each parameter. Use Gridsearch to tune parameters and find the optimized model.

Fit GBC models on 3 different datasets: imbalanced dataset, oversampled dataset and undersampled dataset. For each dataset, split it into 80% for training. **Control testing data**: knowing that if we want to evaluate wether using a undersampled/oversampled dataset is good or not, we should control: test data. So we should check the performance on x_test we've already splitted from imbalanced dataset.

3 datasets: Performance on validation

	imbalanced	oversampling	undersampling

F0.5beta score	0.57812966836	0.546504433904	0.471395374571
False negative rate	0.318661971831	0.400452488688	0.558777835318

From the numbers above, we can conclude that the **optimized model feed with imbalanced data** has the best performance.

-Result-

3 datasets: Performance on validation

	imbalanced	oversampling	undersampling
F0.5beta score	0.57812966836	0.546504433904	0.471395374571
False negative rate	0.318661971831	0.400452488688	0.558777835318

From the numbers above, we can conclude that the **optimized model feed with imbalanced data** has the best performance.

Final model:

```
GradientBoostingClassifier(criterion='friedman_mse', init=None,
                           learning_rate=0.05, loss='deviance', max_depth=15,
                           max_features=None, max_leaf_nodes=None,
                           min_impurity_decrease=0.0, min_impurity_split=None,
                           min_samples_leaf=30, min_samples_split=300,
                           min_weight_fraction_leaf=0.0, n_estimators=100,
                           presort='auto', random_state=42, subsample=1.0, verbose=0,
                           warm_start=False)
```

Final model's robustness:

As I mentioned before, I split the dataset into 3 parts:

Training

Validation

Testing

Trained the model on training dataset. Compared the results of applying the model on validation and testing dataset, we got:

	F0.5beta score	False negative rate
Validation	0.57812966836	0.318661971831
Testing	0.580051874558	0.327868852459

From above, the results we got from validation dataset and testing dataset are almost the same, which indicate that the final model is robust.

Compared to benchmark model:

TABLE III
PERFORMANCE OF CLASSIFIERS W.R.T. ACCURACY OVER TRAINING & TEST SETS

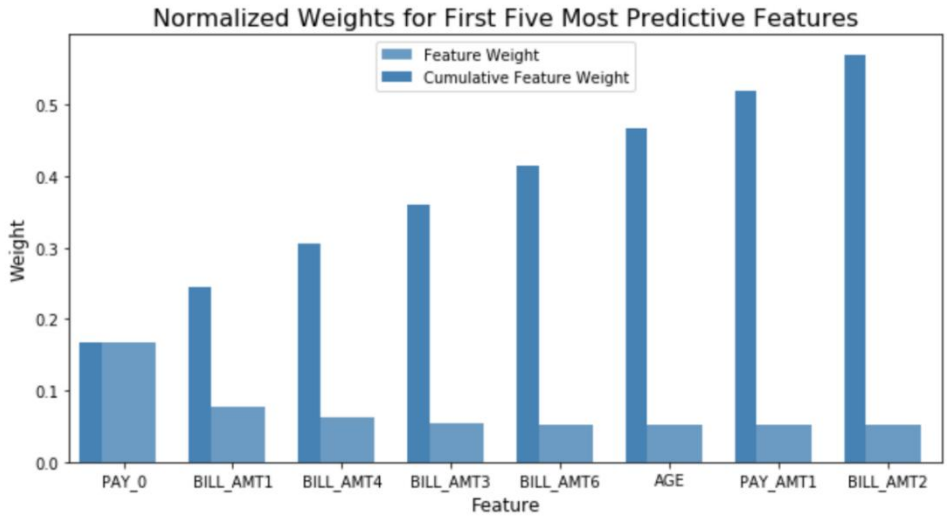
Classifier Model		Set-1F-To-1N				Set-1F-To-4N				Set-1F-To-9N			
		Train	Test	Build Time	Frauds	Train	Test	Build Time	Frauds	Train	Test	Build Time	Frauds
DT Methods	C&RT	90,01%	86,79%	<3m	254	92,13%	92,53%	<3m	242	93,85%	94,69%	<3m	216
	C5.0	92,51%	91,08%	<1m	260	97,44%	92,81%	<1m	227	99,15%	94,52%	<1m	176
	CHAID	92,51%	89,37%	<1m	252	92,92%	92,53%	<1m	233	94,32%	94,76%	<1m	165
SVM Methods	SVM with RBF Kernel	99,78%	83,02%	<74m	228	98,00%	88,97%	<74m	184	98,75%	93,08%	<74m	158
	SVM with Polynomial Kern.	99,78%	83,02%	<2m	228	98,00%	88,97%	<2m	184	98,75%	93,08%	<2m	158
	SVM with Sigmoid Kernel	99,78%	83,02%	<2m	228	98,00%	88,97%	<2m	184	98,75%	93,08%	<2m	158
	SVM with Linear Kernel	99,78%	83,02%	<2m	228	98,00%	89,18%	<2m	191	98,75%	93,08%	<2m	158

In order to compare with benchmark model, I computed accuracy of training and testing dataset for the **optimized model feed with imbalanced data**. I obtained results below:

```
The training accuracy is 0.8521875 and the testing accuracy is 0.823166666667
```

Compared to benchmark model, in terms of both training and testing accuracy, my final model performs as good as the SVM model in benchmark(even better since from the benchmark SVM model, it is obvious that it might be overfitting problem exsited). However, the DT model is much more better than mine when comparing both training and testing accuracy.

Free-foam visulization



Top 8 important features are listed above. The weights are generated by the **final model** and based on **training dataset**.

PAY_0, which is the most important feature, means 'repayment status last month'. Repayment status last month is obviously a very important indicator for repayment this month. If a person paid his or her bill on time, we expected he or she would not default this month.
BILL_AMT means 'Amount of bill statement (NT dollar)' , which is also a important indicator.
AGE is ranked sixth in the feature importance.

Reflection:
Before this project, I only had a vague concept about data manipulation in my head.
As presented in my coding part, I did undersampling by writing my own program: just simply delete instances which labeled with major label randomly. I thought it is the same to do oversampling: just randomly chose some instances which are labeled with minority label and double these instances.
Then I searched some related articles about oversampling and they said the oversampling can not be done by just 'adding' instances arbitrarily; instead, we need to add some random noise to the instances which we want to 'oversampled'. Oversampling is one form of data augmentation, and there are techniques like data mirroring and data cropping, etc..
I realized I still have much to learn. If I knew more techniques, I might have done this project or whatever other projects better.

-Conclusion-

Oversampling or undersampling doesn't improve the f0.5beta score or reduce the false negative rate.
The GBC model feed with imbalanced dataset and performed on validation dataset has the f0.5beta score: 0.565013570195 and after parameter tunning, it slightly improved to 0.57812966836; has the false negative rate: 0.331386861314 and after parameter tunning, it slightly reduced to 0.318661971831. Financial institutions can utilize this credit card default detection model to predict the probability of being default given certain historical information of a customer.

As a **future work**, other algorithms can be used to build new classification models on the same real world dataset, such as DT methods mentioned in benchmark. Also, oversamling or undersampling doesn't work well and that means if we would like to improve performance, we need to gather more real world data and do feature selection.
In addition, we can dive deeper in some fetures, such as PAY_0, which is the most important feature, means 'repayment status last month'. From the instinction, it is a very important

indicator for predicting default: -1 = pay duly; 1 = payment delay for one month; 2 = payment delay for two months; . . .if a client delay his payment for a long time, he or she might default next month. Some other features might be not that useful as these important features. With more in-depth analysis, we might be able to get rid of some features, which both decrease feature dimension and save the cost of collecting data.