Predicting defaulting on credit card applications

When customers come in financial difficulties, it usually does not happen at once. There are indicators which can be used to anticipate the final outcome, such as late payments, calls to the customer services, enquiries about the products, a different browsing pattern on the web or mobile app. By using such patterns it is possible to prevent, or at least guide the process and provide a better service for the customer as well as reduced risks for the bank.

Synopsis

- Getting the Data
- Data Preparation
- Descriptive analytics
- Feature Engineering
- Modeling

Modeling

- Deep Learning (keras/tensorflow)

Convert the data

We use pandas to read the data from its original excel format into a dataframe

Clean up

We lowercase the column name, and rename the column names when required, In particular, remarkably this dataset misses a column PAY_1. In the analysis here below we assume that PAY_0 is actually pay_1, to be consider the repayment of the month prior to the month where we calculate the defaulting

Attributes description

This study uses 23 variables as explanatory variables, extracted/interpreted from :

```
-----------------------------------------------------------------------------------------
Name             Explanation
-------------------- -------------------------------------------------------------------
limit_bal        Amount of the given credit (NT dollar):
                 it includes both the individual consumer credit
                 and his/her family (supplementary) credit.

sex              Gender
                 (1 = male; 2 = female)

education        Education
                 (1 = graduate school; 2 = university; 3 = high school; 4 = others)

marriage         Marital status
                 (1 = married; 2 = single; 3 = others)

age              Age (years)

pay_1 - pay_6    History of past payment. Past monthly payment records
                 From April to September, 2005 as follows:

                 pay_1 = the repayment status in September, 2005
                 pay_2 = the repayment status in August, 2005
                 ...
                 pay_6 = 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

bill_amt1-bill_amt5 Amount of bill statement (NT dollar).

bill_amt1 = amount of bill statement in September, 2005
bill_amt2 = amount of bill statement in August, 2005
...
bill_amt6= amount of bill statement in April, 2005

pay_amt1-pay_amt6 Amount of previous payment (NT dollar)

pay_amt1 = amount paid in September, 2005
pay_amt2 = amount paid in August, 2005
...
pay_amt6 = amount paid in April, 2005

Descriptive Analytics

Payment Delays

	pay_1	pay_2	pay_3	pay_4	pay_5	pay_6
0	2	2	-1	-1	-2	-2
1	-1	2	0	0	0	2
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	-1	0	-1	0	0	0
5	0	0	0	0	0	0
6	0	0	0	0	0	0
7	0	-1	-1	0	0	-1
8	0	0	2	0	0	0
9	-2	-2	-2	-2	-1	-1

As you can see some people pay 2 month upfront, others one month upfront, most of them are on par. a few are running behind payments. One thing worth of notice is that the textual information provided about this variables and the actual values are not exactly the same. So always look and explore the data, before proceeding with any analysis, explore and verify the actual data and the textual info about the data itself.

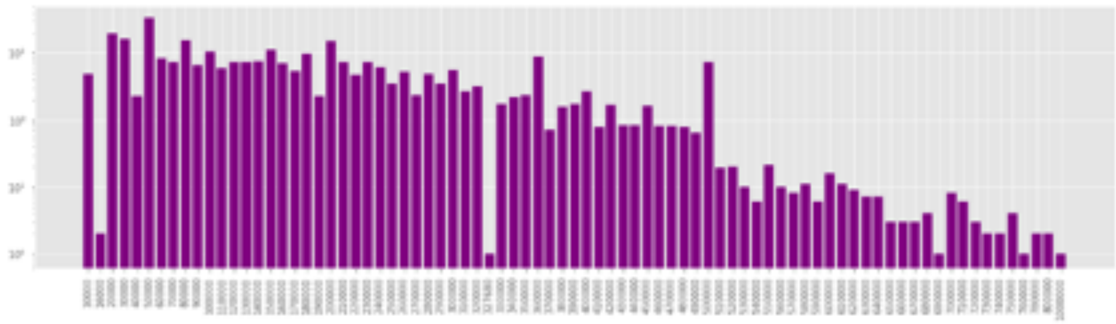
Standing credit

Let's look now at how the debts/credit is accumulating over the months, credit to be repaid is a positive number here.

	bill_amt1	bill_amt2	bill_amt3	bill_amt4	bill_amt5	bill_amt6
count	30000.000000	30000.000000	3.000000e+04	30000.000000	30000.000000	30000.000000
mean	51223.330900	49179.075167	4.701315e+04	43262.948967	40311.400967	38871.760400
std	73635.860576	71173.768783	6.934939e+04	64332.856134	60797.155770	59554.107537
min	-165580.000000	-69777.000000	-1.572640e+05	-170000.000000	-81334.000000	-339603.000000
25%	3558.750000	2984.750000	2.666250e+03	2326.750000	1763.000000	1256.000000
50%	22381.500000	21200.000000	2.008850e+04	19052.000000	18104.500000	17071.000000
75%	67091.000000	64006.250000	6.016475e+04	54506.000000	50190.500000	49198.250000
max	964511.000000	983931.000000	1.664089e+06	891586.000000	927171.000000	961664.000000

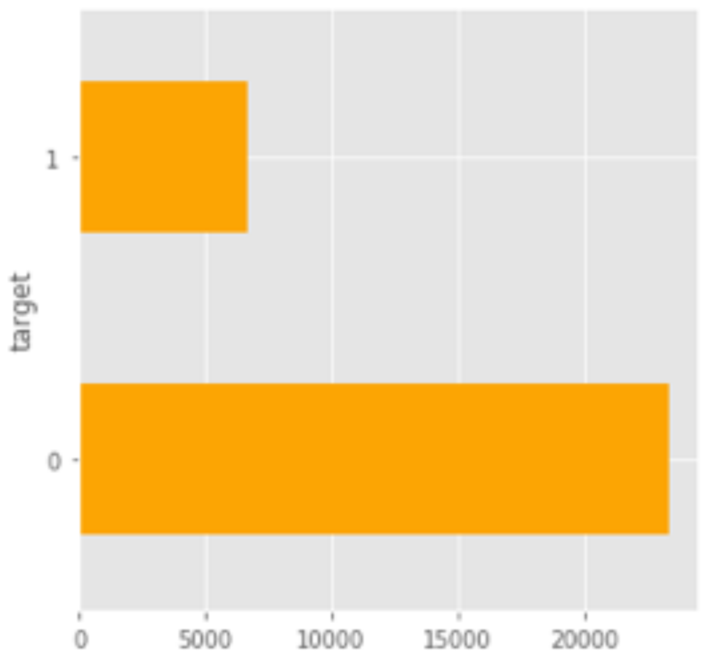
Payments in the previous months

	pay_amt1	pay_amt2	pay_amt3	pay_amt4	pay_amt5	pay_amt6
count	30000.000000	3.000000e+04	30000.00000	30000.000000	30000.000000	30000.000000
mean	5663.580500	5.921163e+03	5225.68150	4826.076867	4799.387633	5215.502567
std	16563.280354	2.304087e+04	17606.96147	15666.159744	15278.305679	17777.465775
min	0.000000	0.000000e+00	0.00000	0.000000	0.000000	0.000000
25%	1000.000000	8.330000e+02	390.00000	296.000000	252.500000	117.750000
50%	2100.000000	2.009000e+03	1800.00000	1500.000000	1500.000000	1500.000000
75%	5006.000000	5.000000e+03	4505.00000	4013.250000	4031.500000	4000.000000
max	873552.000000	1.684259e+06	896040.00000	621000.000000	426529.000000	528666.000000



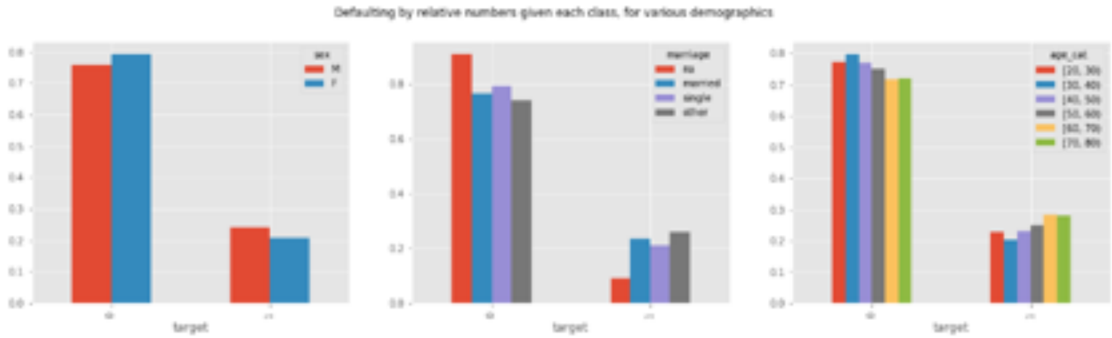
Explore Defaulting

```
defaulting accounts are 22.12% out of 30000 observations
```



Absolute statistics

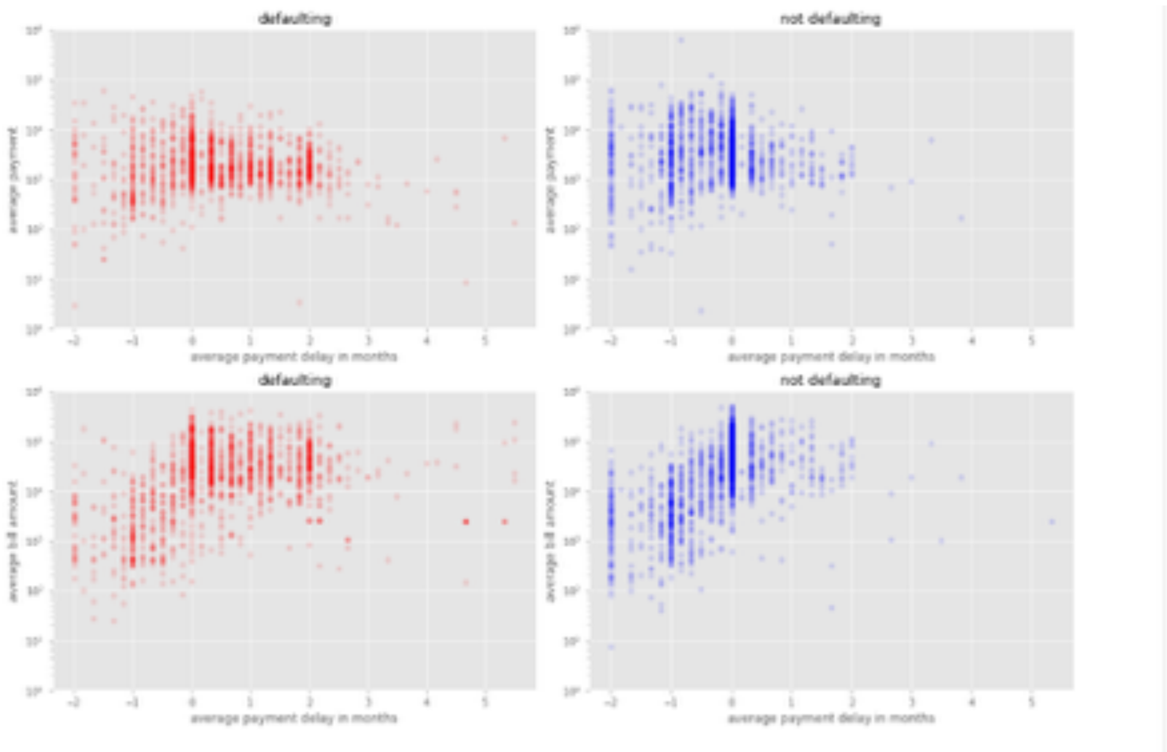
Statistics relative to the population



Feature engineering

It's not about blind feature conversion to values between 0 and 1, it's about understanding data. In this case we see that payment they exhibits a log/log distribution, so first off, we are going to take the log of the payments.

Intuition: if the credit is much larger than the bill, being behind might not be a problem. Therefore this contracted feature might turn up useful when predicting defaulting



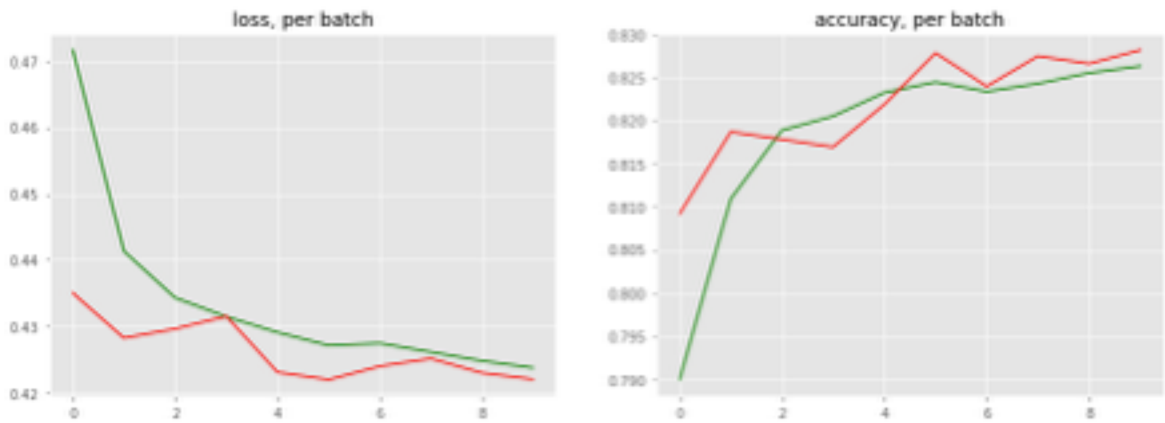
Feature selection

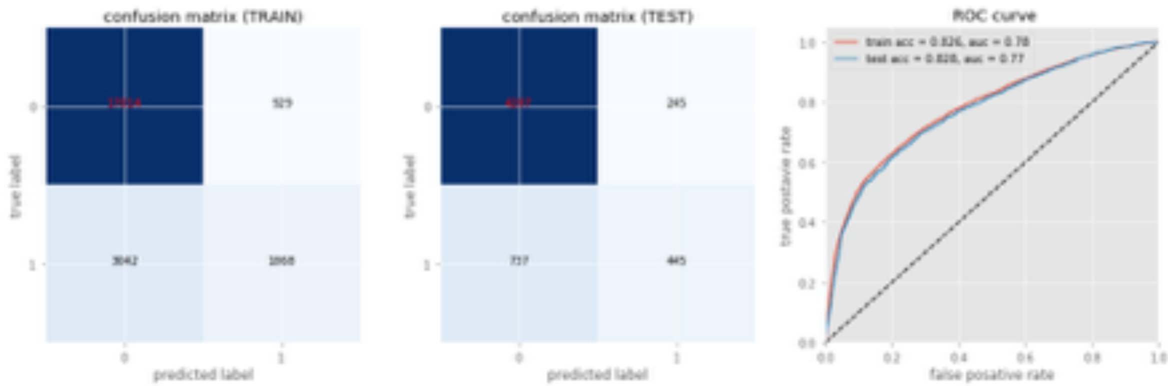
some of the constructed features are indeed beneficial. Also it seems that demographics are only marginally influencing the prediction. paid amounts, delays, and bill relative to credit issued are top indicators. Interestingly education score quite high as a a predictive feature.

Models

Feed forward deep neural nets:-

```
Train on 22853 samples, validate on 5714 samples
Epoch 1/10
22853/22853 [==============================] - 5s 204us/step - loss: 0.4716 - acc: 0.7900 - val_loss: 0.4351 - val_ac
c: 0.8092
Epoch 2/10
22853/22853 [==============================] - 4s 180us/step - loss: 0.4413 - acc: 0.8110 - val_loss: 0.4283 - val_ac
c: 0.8187
Epoch 3/10
22853/22853 [==============================] - 4s 183us/step - loss: 0.4343 - acc: 0.8189 - val_loss: 0.4297 - val_ac
c: 0.8178
Epoch 4/10
22853/22853 [==============================] - 4s 160us/step - loss: 0.4315 - acc: 0.8205 - val_loss: 0.4315 - val_ac
c: 0.8169
Epoch 5/10
22853/22853 [==============================] - 4s 155us/step - loss: 0.4291 - acc: 0.8232 - val_loss: 0.4231 - val_ac
c: 0.8218
Epoch 6/10
22853/22853 [==============================] - 4s 156us/step - loss: 0.4272 - acc: 0.8244 - val_loss: 0.4220 - val_ac
c: 0.8278
Epoch 7/10
22853/22853 [==============================] - 4s 158us/step - loss: 0.4275 - acc: 0.8233 - val_loss: 0.4241 - val_ac
c: 0.8239
Epoch 8/10
22853/22853 [==============================] - 4s 156us/step - loss: 0.4262 - acc: 0.8243 - val_loss: 0.4252 - val_ac
c: 0.8274
Epoch 9/10
22853/22853 [==============================] - 4s 155us/step - loss: 0.4249 - acc: 0.8255 - val_loss: 0.4230 - val_ac
c: 0.8266
Epoch 10/10
22853/22853 [==============================] - 4s 165us/step - loss: 0.4238 - acc: 0.8263 - val_loss: 0.4221 - val_ac
c: 0.8281
```





this are result which is given by the model getting prediction

Test log loss 0.422080701053
Test accuracy 0.828141407008
about 82.8% the model is getting accuracy

PYTHON LIBRARIES REQUIRED

* Pandas
  * Numpy
  * Scikit-Learn

* * Pylab
  * Matplotlib

* * tqdm

* keras

*

USEFUL FRAMEWORKS

* SPYDER
* JUPYTER NOTEBOOK

Code

code is given on def.py file