Machine Learning Engineer Nanodegree

Capstone Project

Marvin Oeben July 11 2017

I. Definition

Project Overview

Banks struggle with many issues these days. One of these is a thorough understanding of their clients. Not only their background information is important but also their behaviour. Interest arises in whether they are involved in possibly unwanted behaviour (e.g. money laundering or terrorism financing) or increasing their risk of default (e.g. missing monthly payments).
Machine learning is starting to help banks investigate their clients, provide new and better credit scores and predict which clients may run into payment issues.
In this project we look at a bank's data. We use an old data set (from 1999) which includes almost all properties real banks have. Even though this data is not as rich as that of a bank nowadays, it will provide a nice introductions into the techniques used by data analysts working for large banks. The dataset is from the PKDD'99 and can be found here:
http://lisp.vse.cz/pkdd99/berka.htm (http://lisp.vse.cz/pkdd99/berka.htm).
The competition had the following original description:
The bank wants to improve their services. For instance, the bank managers have only vague idea, who is a good client (whom to offer some additional services) and who is a bad client (whom to watch carefully to minimize the bank loses). Fortunately, the bank stores data about their clients, the accounts (transactions within several months), the loans already granted, the credit cards issued. The bank managers hope to improve their understanding of customers and seek specific actions to improve services. A mere application of a discovery tool will not be convincing for them.

Problem Statement

Our research will try to answer the following question:
Using a customer's properties and transactional behaviour, can we predict whether or not he/she is able to pay future loan payments?
To answer this question, we need to make some assumptions before we can attempt to answer this question. Some of these will not directly be clear but will once we dive deeper into the data.

- We make a clear seperation between running and finished loans. In this, we see finsihed loans as those of which we have full information and running loans as the ones we want to have classification on.
- All loans are equal in time. We look at all loans from start to finish. In this, we do not take into conern the seasonality between different years. Using these assumptions, we will manipulate the data to create features which we hope can be predictive for running and new loans.
  This will require a large amount of preprocessing. In this usecase, we will also try to use network features to predict whether we have enough interaction to build network-based features.

Metrics

In this project, we will use an F1-score to evaluate how well our model will be able to predict a future defaults. We will use the finished loans for both traning and testing data. As we will see that we have a relatively small amount of defaults and F1 score will give us a nice balance over accuracy and false positive ratio. The F1 score is defined as:

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

where

$$Precision = \frac{tp}{tp + fp}$$

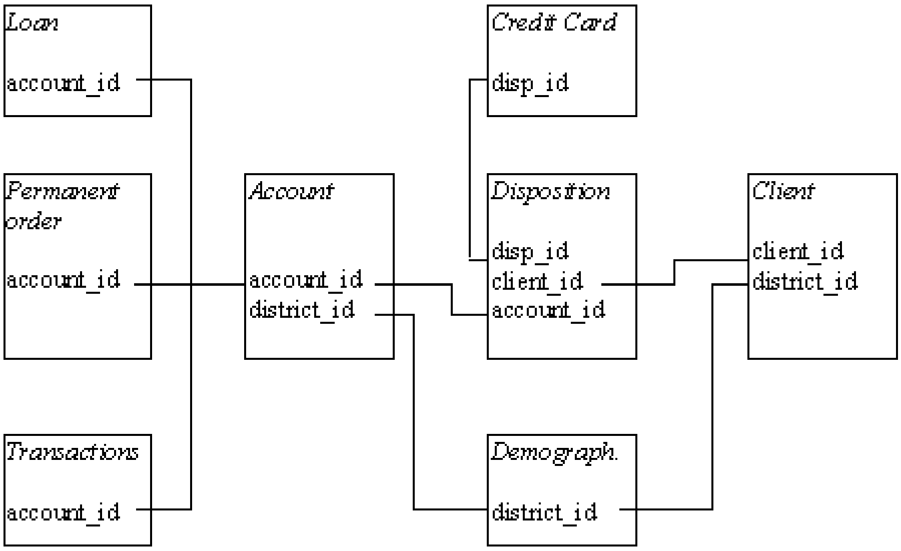
and

$$Recall = \frac{tp}{tp + fn}$$

II. Analysis

Data Exploration

The data for this project is provided on the competition website. In this project, it is downloaded and handled in the `Get_data.py` file. The data has the following structure:

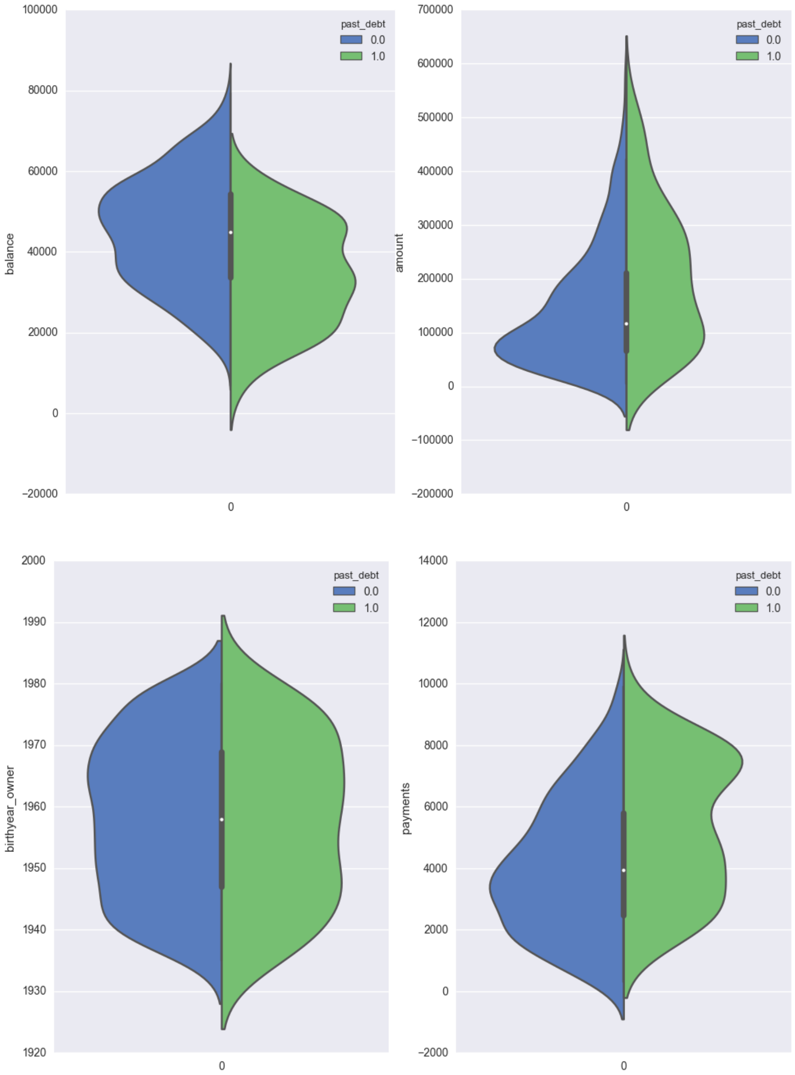


In the `Get_data.py` file we deal with the merging and checking of the quality of the data. Any reader interested in downloading and checking the data for themselves can follow the steps in that file.
The data is provided to us in a flat text file with `;` delimiter and `\n` EOL statements. Many of the headers/cells in the dataset are provided in Czech and translated to English for further use. The data is then loaded into pandas dataframes and merged into five files:

- `client_info.cv` containing all the info on the client
- `demographic_info` containing all the info on the demographics of the Czech republic
- `Transaction_info` A file containing all transactions made for the accounts in the dataset
- `order_info` A file containing info on the ordered products per client
- `loan_info` A file containing the loan information

All of these files are in the further steps reduced to clients and accounts which have actual loans. These steps are described in `Loan_payment_feature_engineering.ipynb`.
For some relevant features, we can provide the plots as found in the data. This allows us to check the distribution and whether or not there is a connection to missed payments. In these plots, we compare the distributions for loans which are past debt and those which are not. We can give some quick statistics and plots:

Feature	**mean**	**std**	**min**	**25%**	**50%**	**75%**	**max**
Amount	151410	113372	4980	66732	116928	210654	590820
Balance	44023	13793	6690	33725	44879	54396	79272
Birthyear	1958	13	1935	1947	1958	1969	1980
Payments	4190	2215	304	2477	3934	5813	9910



We can see the following things:

- Clients in debt seem to have higher monthly paymets but less balance on the bank.

With these, we hope to be able to predict which loans will be in default, and which will run along just fine.

Algorithms and Techniques

In this section, we will try to answer the following questions:

- What will be the metric to which we test?
- How will we define our training and test sets?
- Which features will we use?
- Which models will we use?

To tackle the first, remember that we want to predict future debt using past debt. Our data is highly skewed however, as we have 606 accounts without any debt and only 76 with debt. Hence, using our accuracy score may not be the best choice. We will go with the F1 score as it will provide us with a nice balance between precision, recall and accuracy.
Now actually configuring the data to do what we want is quite difficult. As we want to predict 'future missed payments' we need to use data in which we can predict future payments (as our past_debt only shows us if there is any past debt). We do however have the missed counts in all the quarters which we can use. Thus the setup will be as follows:

1. Use the finished data as training set
2. Use the finshed data as test set.

3. For both these datasets, use the first 3 quarters as prediction for missed payment in the last.
4. Use a few running tests as vaildation.
5. Finish with predictions on the full running dataset.

We will use the following features:

- Loan amount
- Loan duration
- Loan payment heigth
- Bank balance
- Birthyear account owner
- Gender
- Startyear of the owner

As for the models, we will use the usual suspects:

- **Naive Bayes**: A model that leverages conditional probability. Using known distributions of data to predict related distributions. It is run by the main rule:
$$p(C_k \mid \mathbf{x}) = \frac{p(C_k)\, p(\mathbf{x} \mid C_k)}{p(\mathbf{x})}$$
or in layman's terms:
$$\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}$$
In our case, we will use a gaussian kernel that makes the assumption that the data is somewhat normally distributed to estimate the probability density function.
- **SVM**: A linear SVM tries to lay a line (or in more dimension a (hyper)plane) inbetween data in order to create a seperation. Each side of the line receives a different classification (in our case payment miss or not). This is then optimized to build the most robust and least error prone line.
- **Logistic regression**: A classifier version of Linear regression. Instead of the normal formula
$$y = \beta_0 + \beta_1 x + \epsilon$$
with normally distributed epsilon we have
$$y = \begin{cases} 1 & \beta_0 + \beta_1 x + \varepsilon > 0 \\ 0 & \text{else} \end{cases}$$
with epsilon distributed as in the logistic distribution.
- **Decision Tree**: An algorithm which in each step slices data based on the feature leaving the least entropy (a measure of impurity of data). It is called a decision tree as one can make decisions based on every step and builds a tree by splitting the data, leaving the leaves as the classified data.
- **Bagging**: An algorithm that subsets the data and trains decision trees on those subsets. The idea is that this procedure improves the stability of the algorithm. The final output is the average classification over all sampled classifiers.
- **AdaBoost**: Adaptive boosting, an algorithm that only selects only features that show predictive power. It then builds a weighted tree based on the weighted predictive power for each feature.
- **Random Forest**: Extension of bagging by also subsetting random features and pruning the tree.

Benchmark

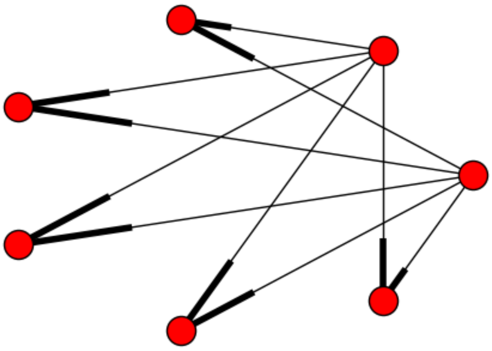
Our benchmark will be the running loans and their history. Based on the finished, are we able to predict past-debt running loans accordingly?

III. Methodology

Data Preprocessing

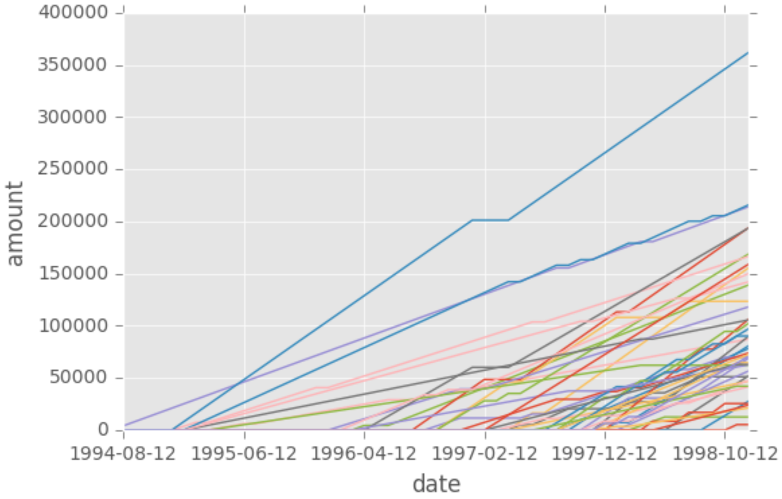
This project took some preprocessing. As described earlier a lot of effort had to be done wrestling the data into a single dataset and translating the Czech to English.

At first the idea was to use the transaction network (with accounts as nodes and transactions as edges) to provide us with a lot of information. The dataset however, does not have the same key for the receiving and sending parties, making this network very sparsely connected. The small exercise in doing this can be found in the file `Network_analysis.ipynb`. As a matter of fact, the largest connected component only had 7 nodes as seen in the picture below:

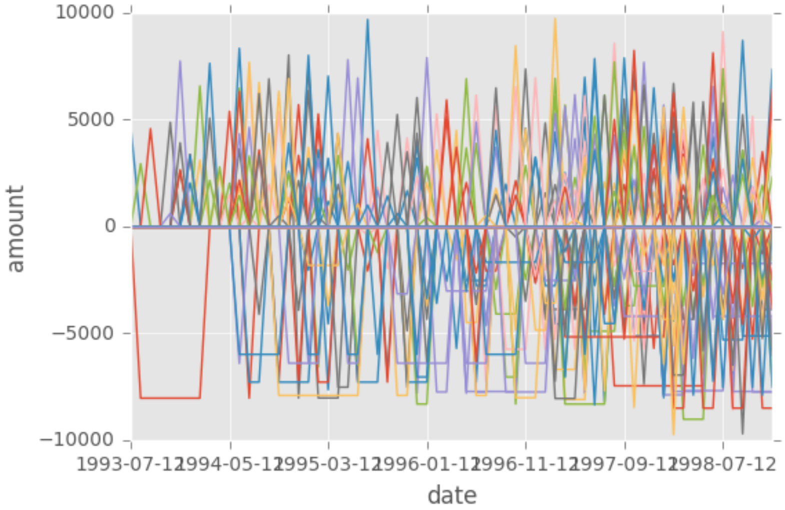


In this picture, the bold parts are the receiving accounts and we see that no features can be derived from the features of the network. Finding datasets in which this is possible seems very difficult. One could try to use the Bitcoin ledger. By property of the crypto ledger, no other information can be derieved from the network, making the analysis quite useless.

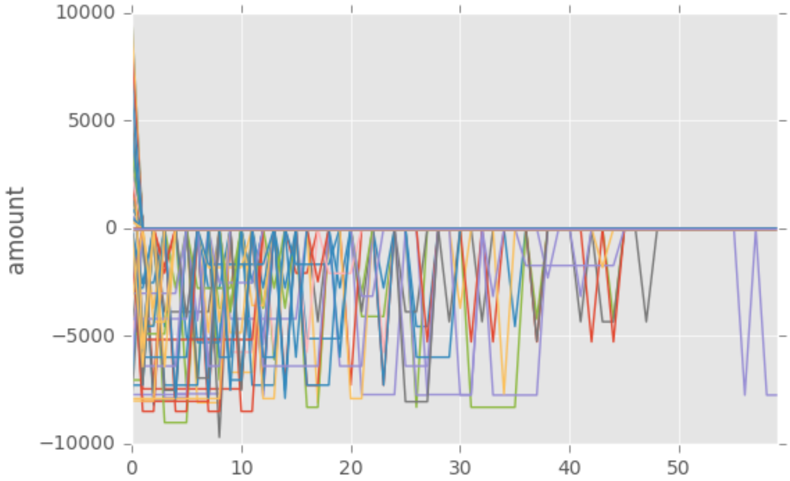
Therefore, I focused on the loan problem at hand. For this I made a lot of steps on the transactions and especially on the missing of payments. I started with all loans and looked at whether those who had missed payments actually had 'flat' months (months in which no loan was paid) to check if the loans were correctly identified. The following picture shows that this is true:



Next, I decided to look at the difference between the ordered and paid amount each month. If this was smaller than zero, this would indicate a missed payment. If it was larger, it was a possible pre-payment (in the first month). This seemed to also happen:



As we assumed that all loans are equal, we push them all back to start-date and see that all prepayments happen in the first month and we can use this data to construct our features:

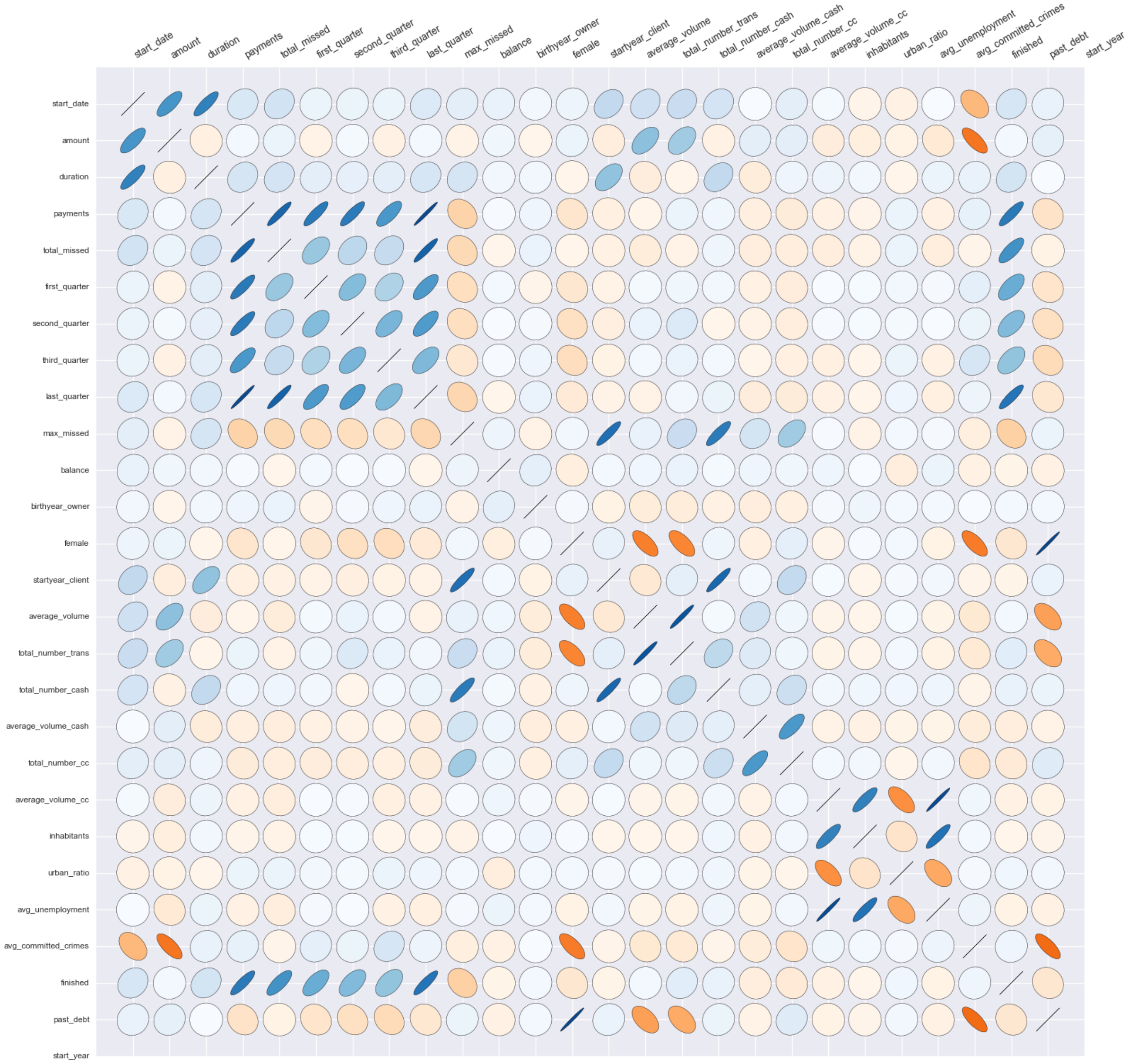


After this, we can take the first 75% of the time for each loan to predict future missed payment, and use the last 25% as the input for the target variable.

In the end, we did choose to not include card/demographic behaviour due to the lack of variance in the data.

Implementation and refinement

After all the feature engineering and exploratory analysis shown above, I moved on to looking at the global correlation structure. I rebuild a plot I often use when working in R to give the first insights.



Based on this, I was able to quickly spot correlation structures and together with the plots based on distributions with the target variable substet the required features.

After this, there was the phase in which we could start doing machine learning. I decided to not go for a CV scheme, as with the low amount of actual delayed payments in the data, there was the risk of creating subsets with no accounts with delayed payments.

To do a grid search I wrote my own little implementation (based on the GridSearchCV in skLearn) which does the job without cross-validation. In the end I did not use it that much though. It can be found in `gridSearch.py`.

As stated above, I will test the following algorithms on the data:

- Naive Bayes
- SVM
- Logistic regression
- Decision Tree
- Bagging
- AdaBoost
- Random Forest

Naive Bayes

I decided to run a standard naive bayes to see how it would perform compared with other algorithms.

SVM

I ran the SVM with both the linear and rbf kernal, a C-value of either 1 or 10 and balanced and unweighted classes.

Logistic regression

With Logistic regression I went with both the L1 and L2 loss functions, a C-value of either 1 or 10 and balanced and unweighted classes.

Decision tree

For the Decision tree, I ran with 0.25, 0.5 and 0.75 max features combined with a max depth of either 1, 2 or 3.

Bagging, Boosting and Random Forest

All these will run on baisc settings with 1000 estimators.

For all algorithms, I will use the skLearn implementations combined with the gridsearch function for those for whom it may help. Running the models gave no complications as my macbook pro was sufficiently fast to run them. It was however hard to find a model which gave a satisfying F1-score, but this will be discussed in more detail in the Results part of this report.

IV. Results

Model Evaluation and Validation

Below is a table which shows both the F1-score for our models. We see that the F1-score is generally low. To deduce these scores, I have done several different tran/test combinations. In all of these combinations, the best score was 0.67 for the F1-score.

Model	**Config**	**Features**	**Score**
Naive Bayes	Gaussian/default	all	0.44
SVM	{'kernel': 'rbf', 'C': 1, 'class_weight': 'balanced'}	Startyear client, Duration, payments	0.67
Logistic Regression	'C': 1, 'class_weight': 'balanced', 'penalty': 'l2'	Amount, Payments, Startyear Client, Balance	0.39
Decision Tree	{'max_depth': 3, 'max_features': 0.75}	Amount, Balance	0.33
Bagging	base_estimatior: Cart, n_estimators: 1000	all	0.4
AdaBoost	n_estimators: 1000	all	0.5
Random Forest	n_estimators: 1000	all	0.4

As our SVM seems to perfom the best, we choose to use it as our final model. In other checks, we found it to be a quite robust method, performing well in combination with the gridsearch function for various subsets.

We chose the SVM with a radial basis function as kernel and a simple C parameter (the misclassification/simplicity tradeoff) of value 1. This combined with balanced class weights (due to skewness) provides a very solid model.

The combination of both training and testing sets gives us a robust model, which I have tested on different splits. The score of only 0.67 makes the results not trustwordy enough to use in production. We do however have a low benchmark (no machine learning based predictions are often done by banks). Therefore, we may even say it is acceptable (but far from optimal).

I would have liked a better model and this sure is work for future improvement. More data and more customers would cetrainly help in this case.

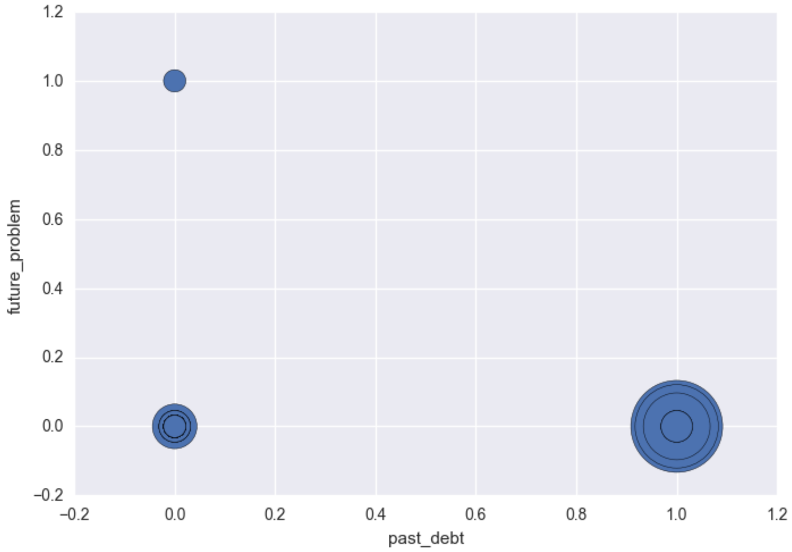
Justification

As we have found a F1-score better than using a decision based method, we can say that it outperforms the benchmark (of traditional expert judgement based methods). This, combined with the straightforward way an SVM works (it can be explained to stakeholders) makes this model a solid basis for further use.

V. Conclusion

Free-Form Visualization

Our final question is whether we can predict if running loans will get into trouble. Remember that we did not use past payment issues as predictor, loans which lack this information (newly formed) can then not be correctly classified. Therefore, we visualize how we classify running loans, based on their past debt. We see that this is not very good. No loans with past debt are classified correctly.



Reflection

I started this project hoping to find payment issues using network properties. I found however that the data (or any data openly available) was not sufficient to provide me with both a loan/transactional dataset and interesting relationships between parties. Therefore I settled with a more traditional project.

The data crunching was maybe on the heavy side, and I could have spent more time in machine learning. But it was good to go through all the steps in Python for a change (our default language is R) and see where it may be more suitable than our default goto tools.

The final model is somewhat dissapointing, but it is a good example of default machine learning not working for all problems.

Improvement

The model can be improved in several ways. The data can be cut up in a different way to generate other features. We can take into account the skewness better by either building a skewed training set or generating more samples (with bootstraps, autoencoders or a GAN). This may improve the model and provide us a better prediction for running loans. In the end, the model would benefit from more personal data, often not disclosed in public datasets which better captivates properties of clients.