Prediction of interesting rate

Frank Wang

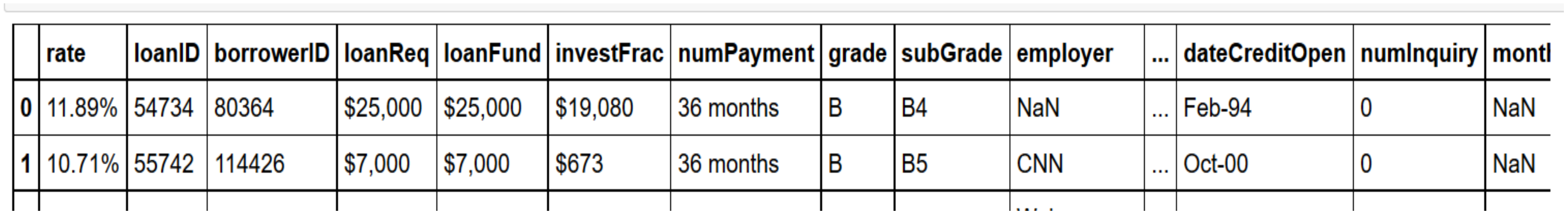
7/12/2016

Frankwanglf@gmail.com

Feature Engineer: 1/5

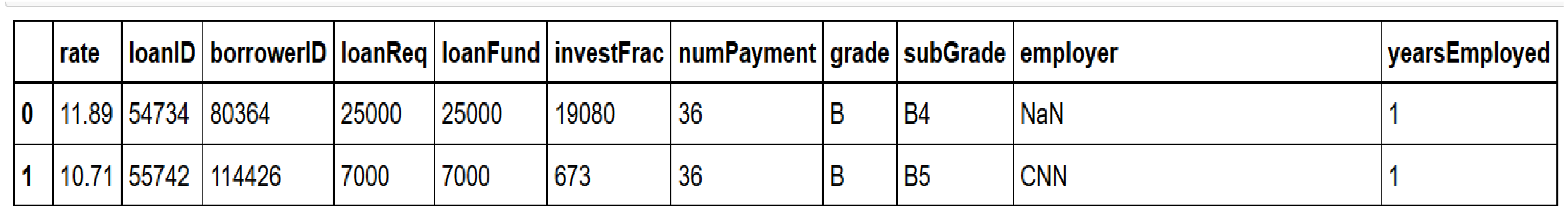
- **Data convert**

  The raw data have different format, including symbols like %, $, > ,+, month. Those data is convert to float number.



	rate	loanID	borrowerID	loanReq	loanFund	investFrac	numPayment	grade	subGrade	employer	...	dateCreditOpen	numInquiry	montl
0	11.89%	54734	80364	$25,000	$25,000	$19,080	36 months	B	B4	NaN	...	Feb-94	0	NaN
1	10.71%	55742	114426	$7,000	$7,000	$673	36 months	B	B5	CNN	...	Oct-00	0	NaN

Raw Data



	rate	loanID	borrowerID	loanReq	loanFund	investFrac	numPayment	grade	subGrade	employer	yearsEmployed
0	11.89	54734	80364	25000	25000	19080	36	B	B4	NaN	1
1	10.71	55742	114426	7000	7000	673	36	B	B5	CNN	1

After data type transformation

Feature Engineer: 2/5

•Drop some features which are not important (importance of features) or not related to prediction: ‘loanID','borrowerID','employer','zipcode','incomeVeri','loanTitle', 'reason','listStatus’,’state’. “State” in overall is non-important based in the feature-importance. We drop it to speed-up computation.

•Check whether there are repeated data. No repeating found.
•Drop rows with missing rate(response)
•Drop features with 50% missing: “reason” and “monthsRecord”Drop rows in each feature with very small number of missing data since we have enough data to use.

•homeowner: keep only [‘MORTGAGE’,’OWN’,’RENT’], drop [‘ANY’,’NONE’,’OTHER’] because of small number sample, and they are not in test.

Feature Engineer: 3/5

- state: drop row where state=[‘IA’,’ID’,’ME’], because small number of samples with very low rate and not in test
- the mean rate does depend on ‘State’. After we drop some state as above, the rate variation with state becomes smaller. ‘State’ is not important according to the feature importance
- Drop “grade” because we use “subgrade”
- Special care about loan issued in early years:
- The interesting rate vary largely at early years. This complicates the prediction. Study shows the number of early year data is small, we simply drop those early data to improve the model performance. The test data is for 2015. Therefore, we believe this treatment is proper. In early-year data can be treated in a proper time series method, we are not going to try here.

Feature Engineer: 4/5

New features

- Loan amount requested = Loan amount funded / Loan amount requested
- Loan amount invested =   Loan amount invested/Loan amount funded
- **Open credit line ratio** = Number of open credit lines in the borrower's credit file / The total number of credit lines currently in the borrower's credit file
- **Amount of payment per month** = Load amount requested / Number of payments (36 or 60)
- **Length of history** = Date loan was issued - Date the borrower's earliest reported credit line was opened
- **Loan amount vs revolving balance** = Loan amount requested / Total credit revolving balance
- **Loan amount vs annual income** = Loan amount funded / Annual income of borrower

Feature Engineer: 5/5

Missing data

We tried two different transformations on missing data:

- drop missing data
- imputation(random imputation for categorical features and mean imputation for continuous features)

In overall drop missing works fine. Since we have enough data to use, drop-missing can speed up the computation.

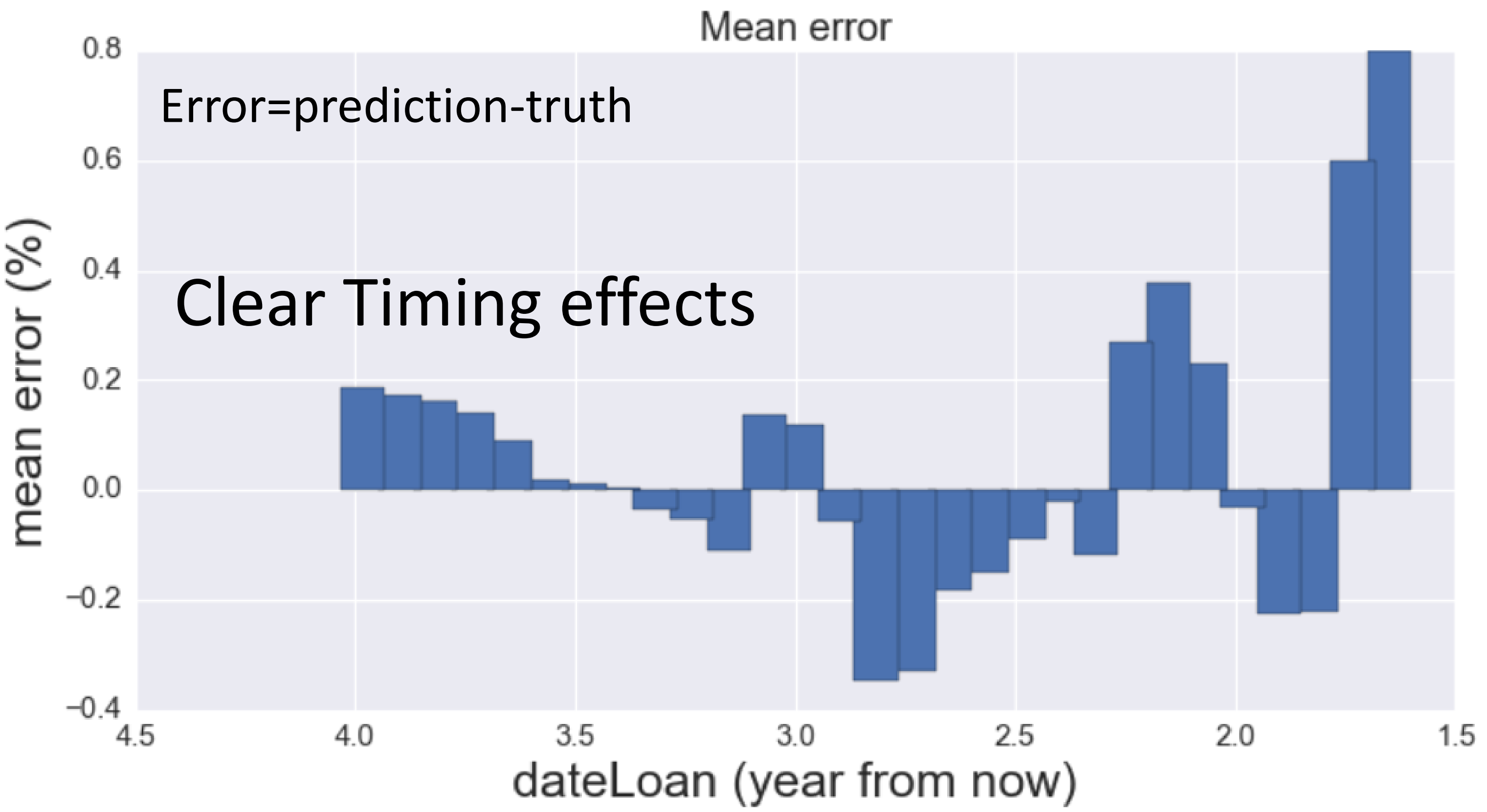
Pre-estimation using Linear model

R^2 : 0.990

RMSE: 0.42%

Pros: Fast, easy explain

Cons: difficult for nonlinear effects

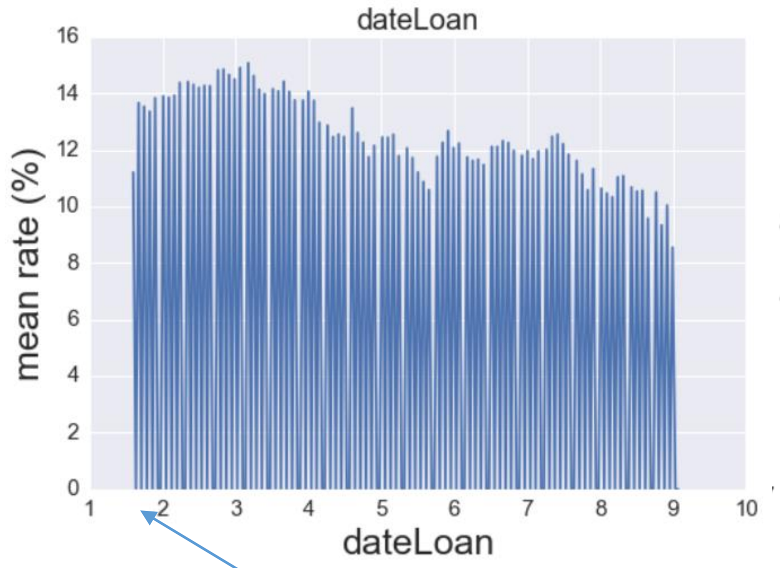


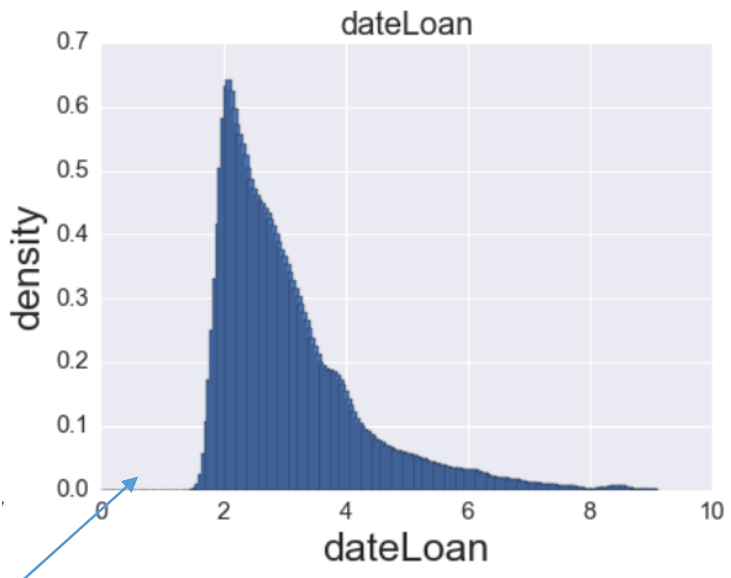
Model I : GBM

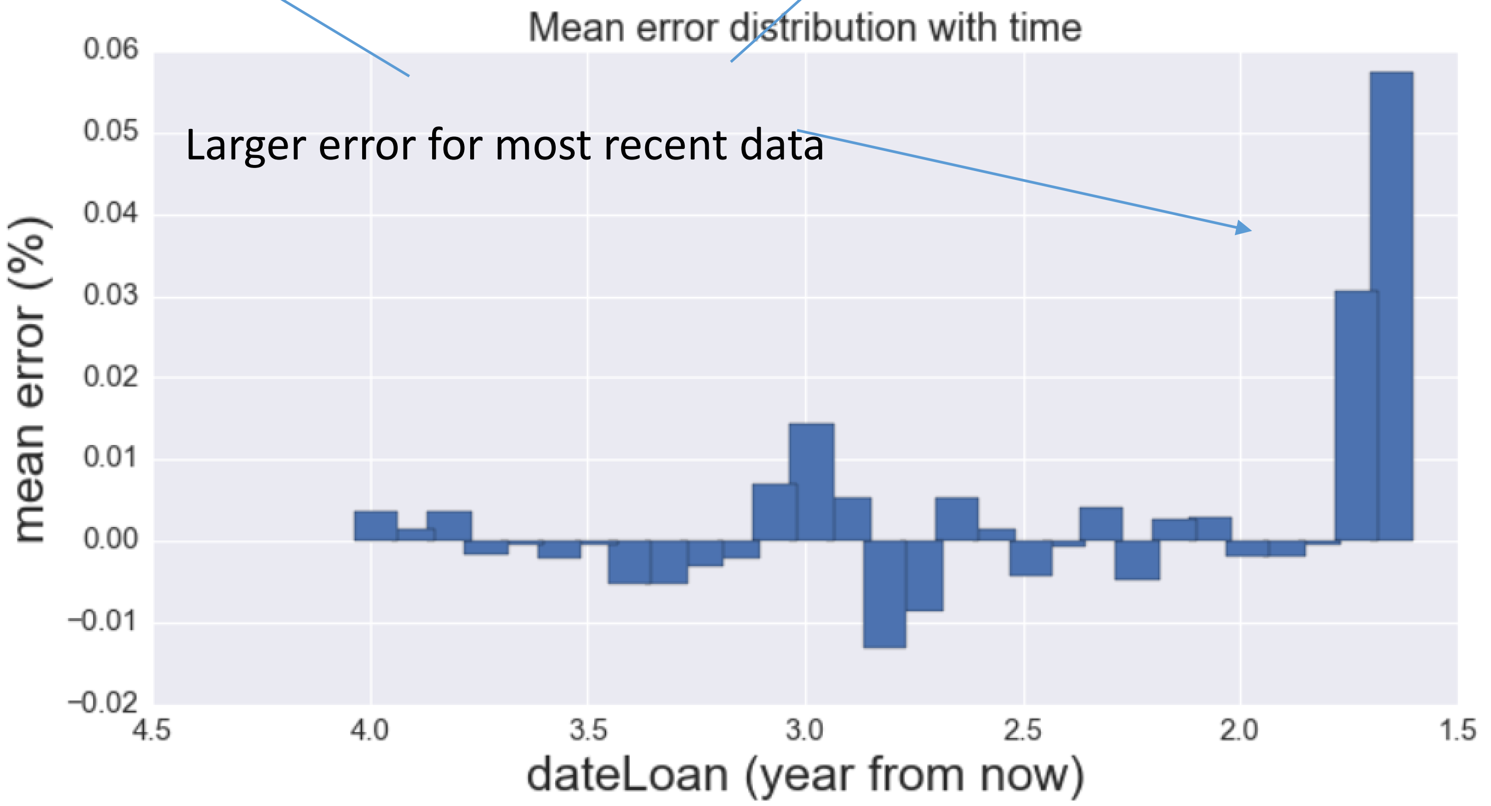
R^2 : 0.999
RMSE: 0.14%

Pros: slight better prediction than GBM
Cons: lack of interpretation; slow than linear model

learning_rate=0.15,
max_depth=6,
min_samples_leaf=1,
min_samples_split=4,
n_estimators=300,
subsample=0.9,

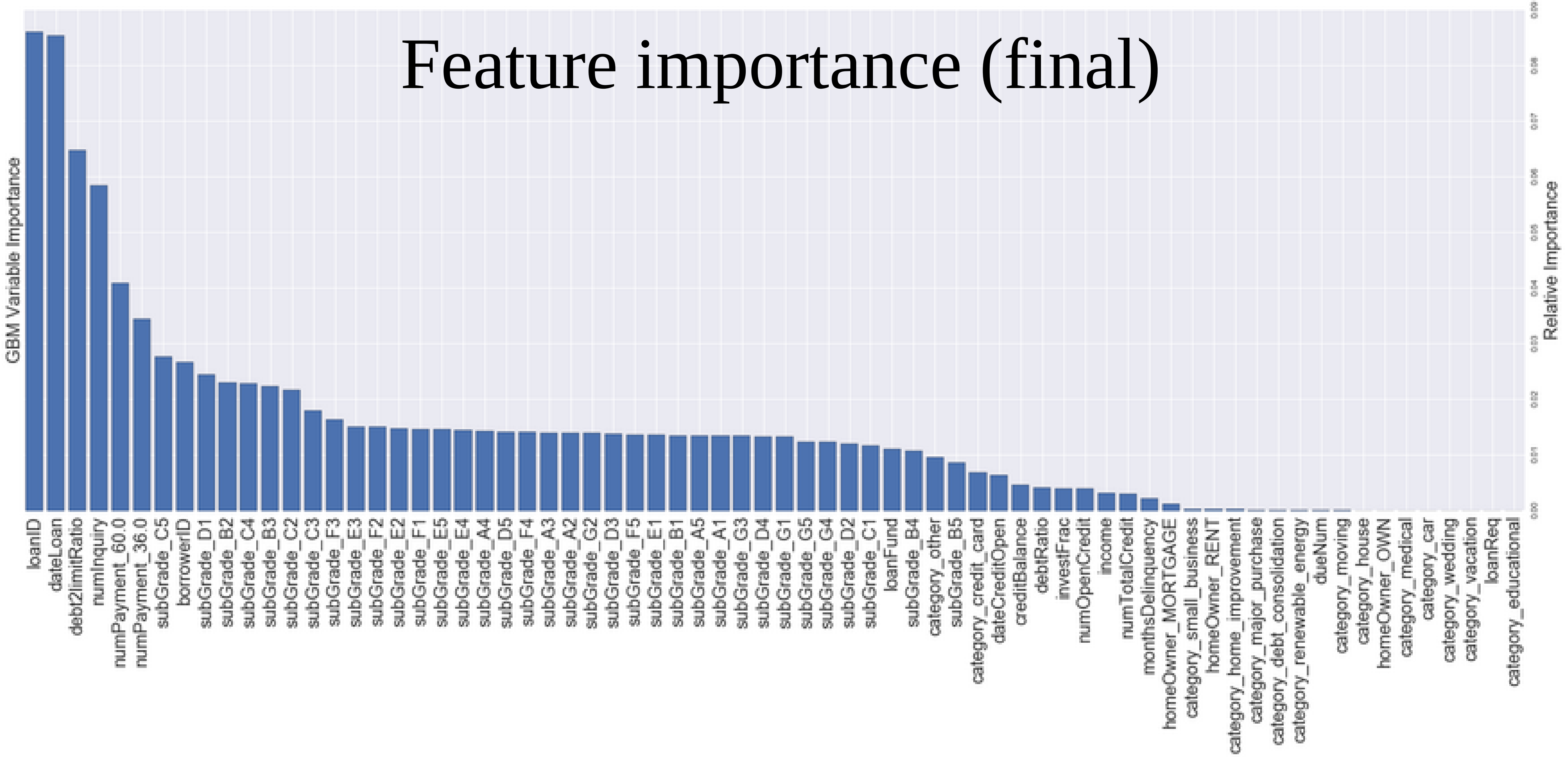






Larger error for most recent data

Feature importance (final)



Model II: XGB

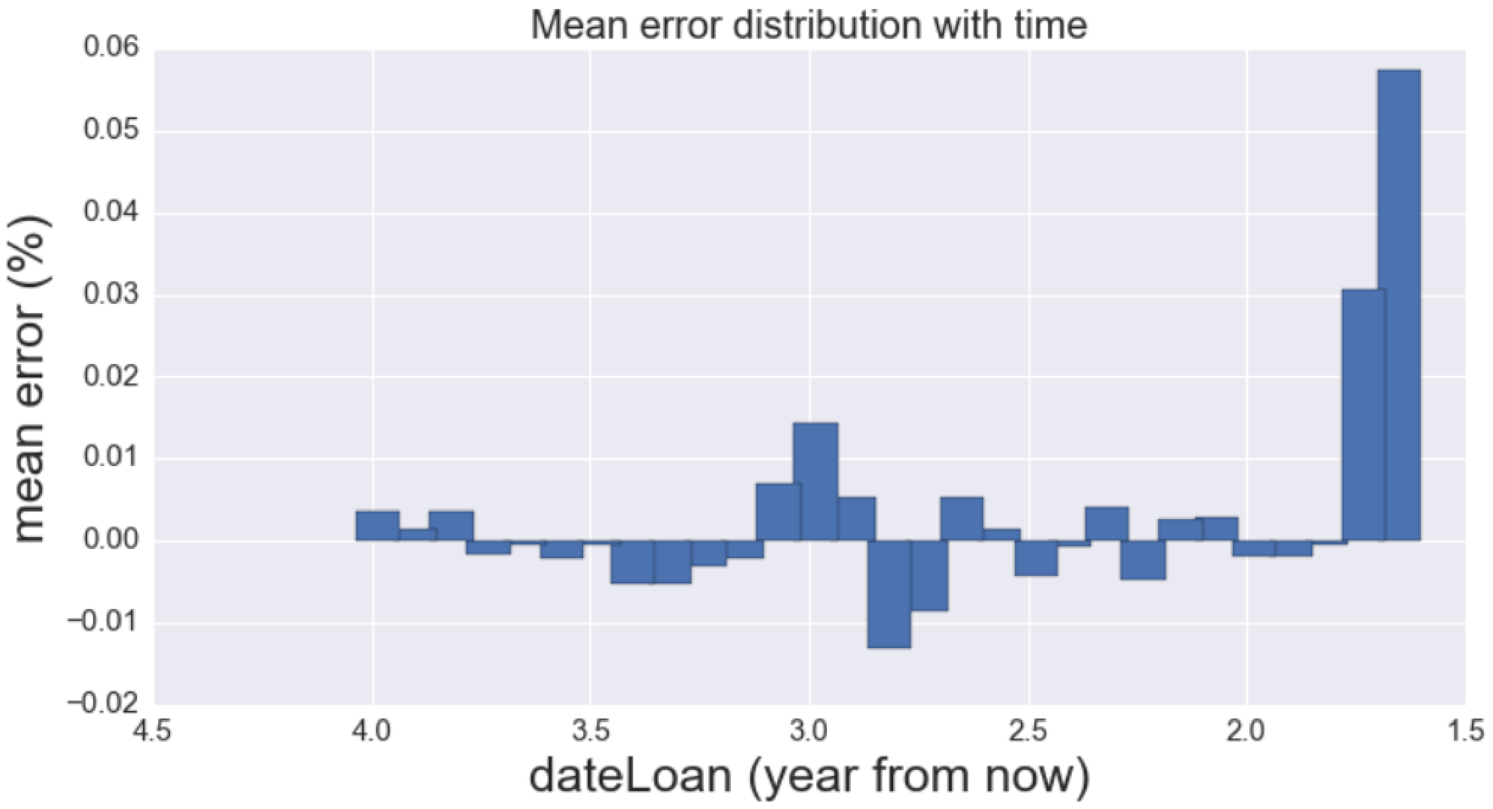
R^2 : 0.998

RMSE: 0.18%

Pros: Faster than GBM

Cons: lack of interpretation

'eta': 0.12,
'eval_metric':'rmse',
'subsample': 0.7,
'max_depth': 7,
'min_child_weight': 2,
'colsample_bytree': 0.5,
'gamma': 5,
num_round = 280



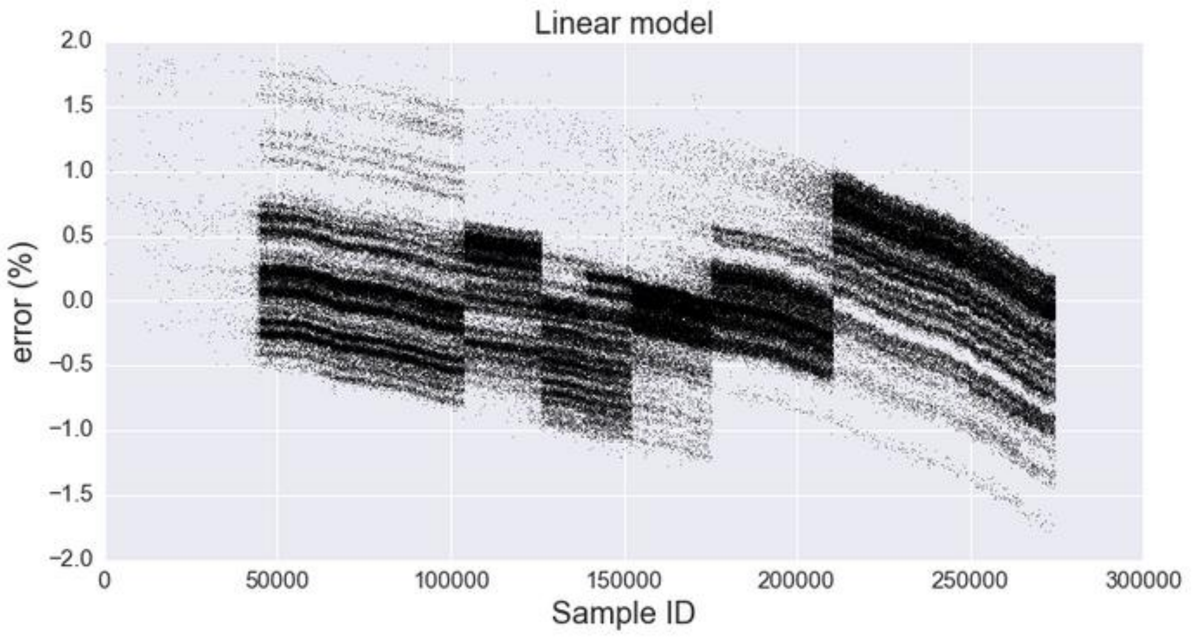
Summary

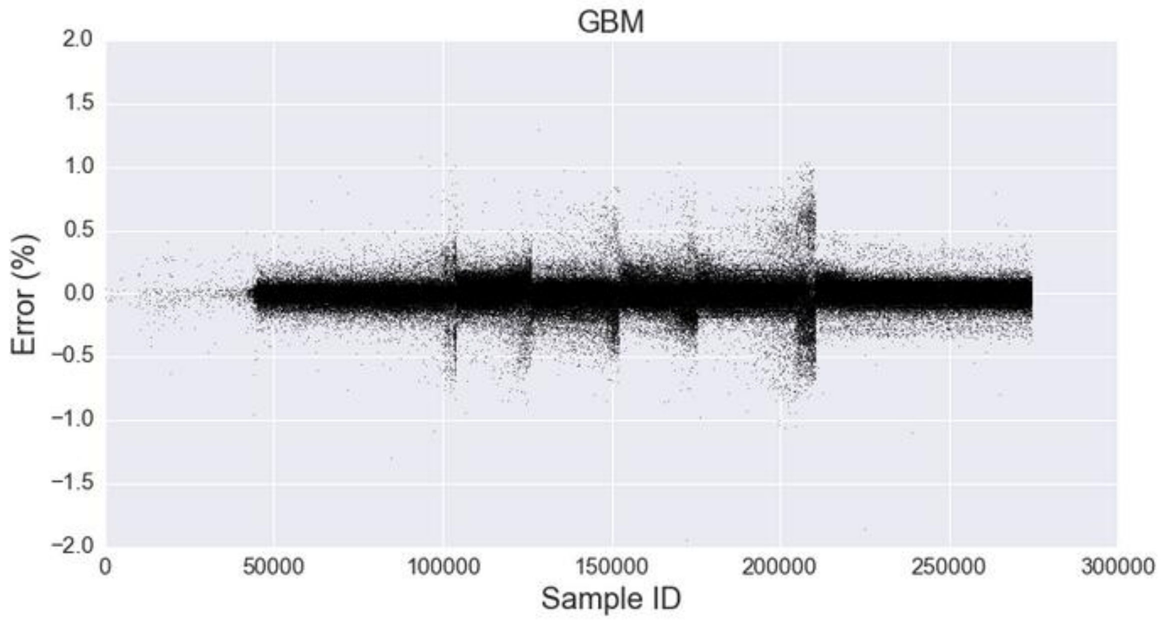
- Detail data clean: variable transform, data missing, data visualization.

  A clear pattern was seen from data.
- Using Linear model for fast estimation, then focus on GBM and XGBoost.
- Linear model is faster and helps to understand the data;
- Gradient Boost can handle nonlinear effects and gives smaller variance. It has more parameters to tune(this can be both pros and cons) . It gives the small RMSE.
- Xgboost also can handle nonlinear effect; easy model. It is faster than GBM, the RMSE is close to GBM.

Interesting pattern!

We noticed that there is a clear pattern in the data; GBM works much better
We have some ideas to investigate it.

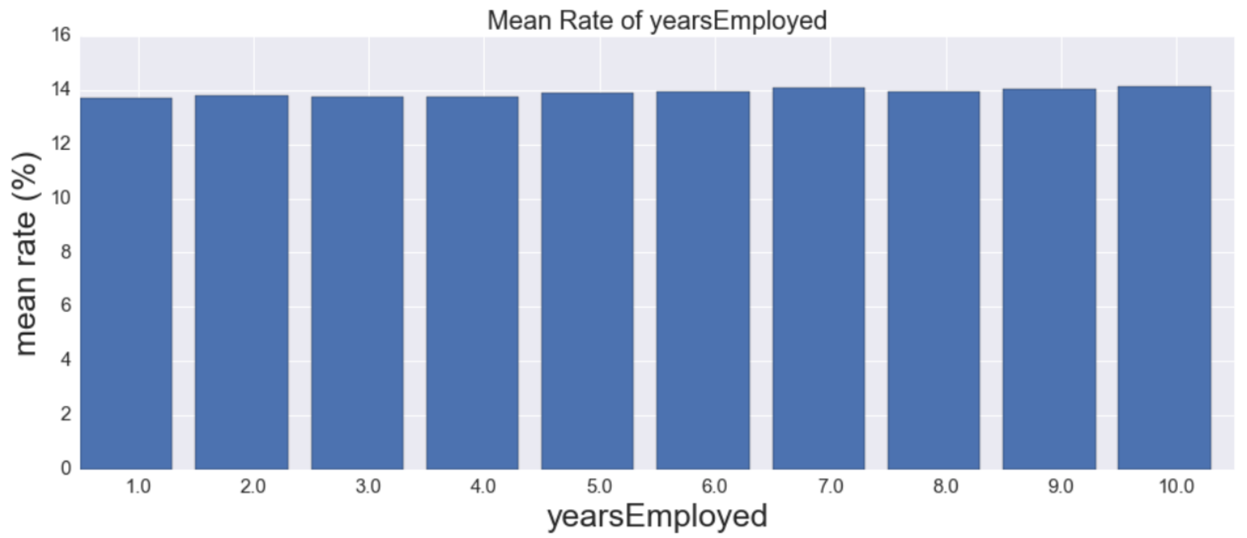


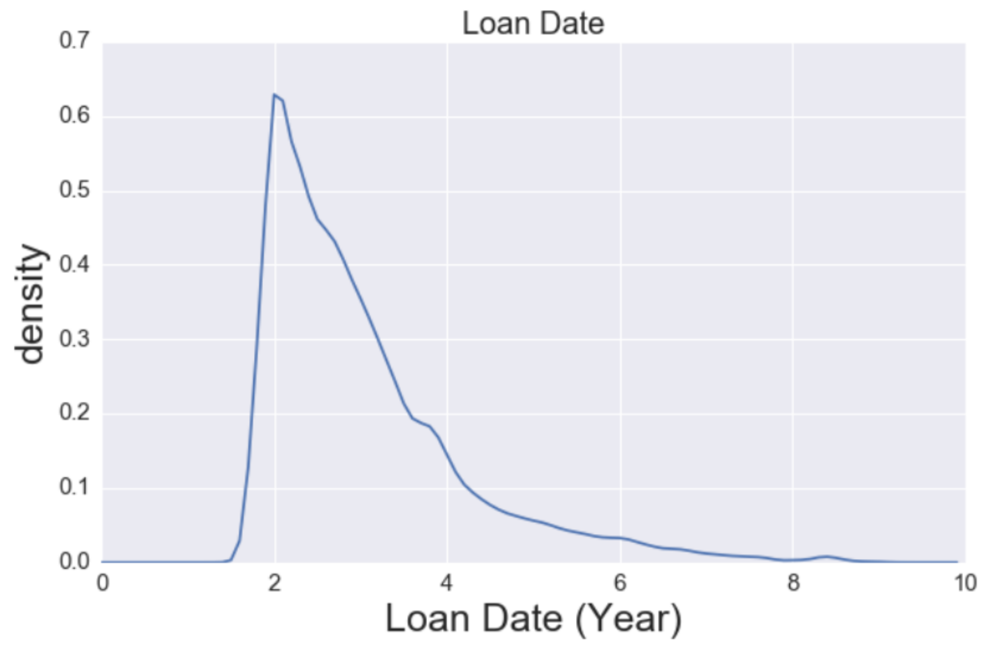


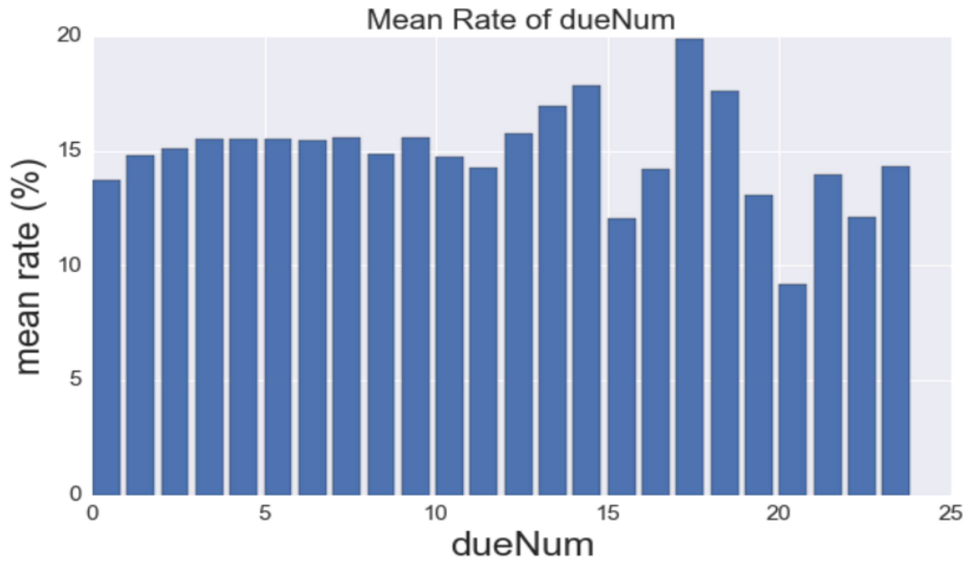
Next

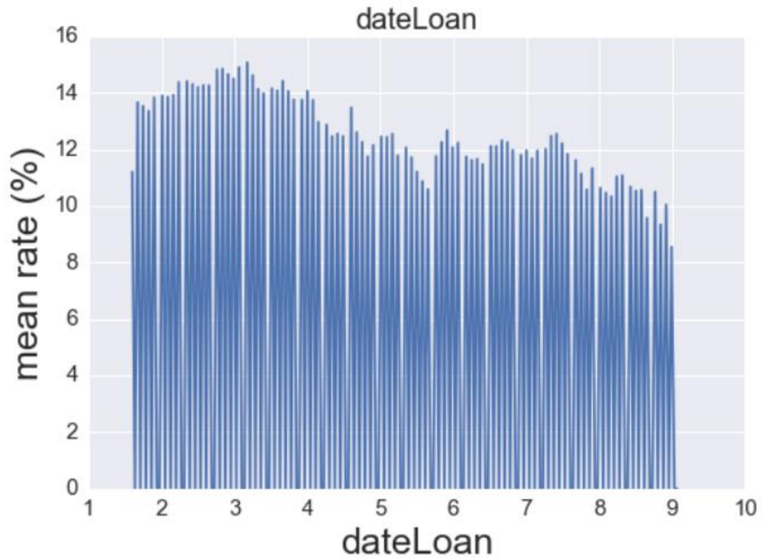
- Try to add time series effect into the model
- Try CNN

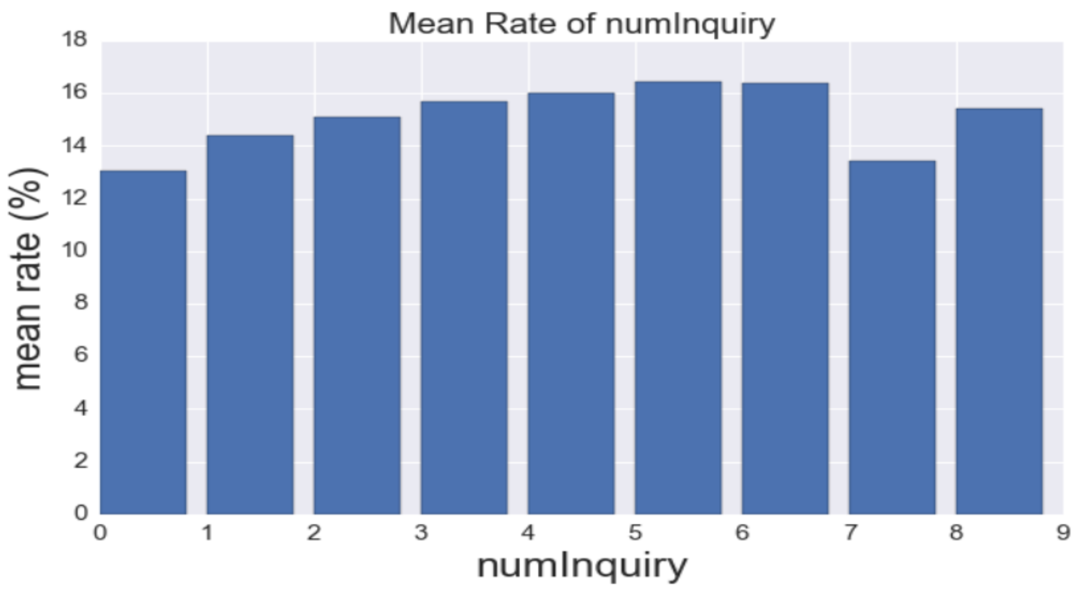
Backup plots

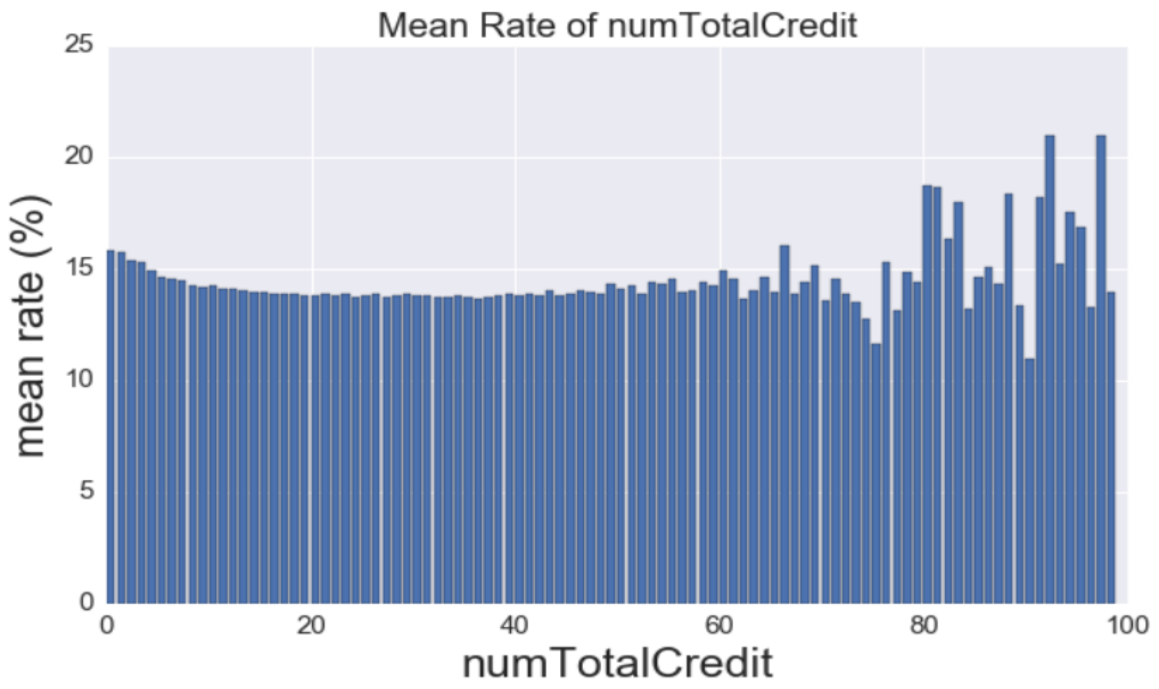


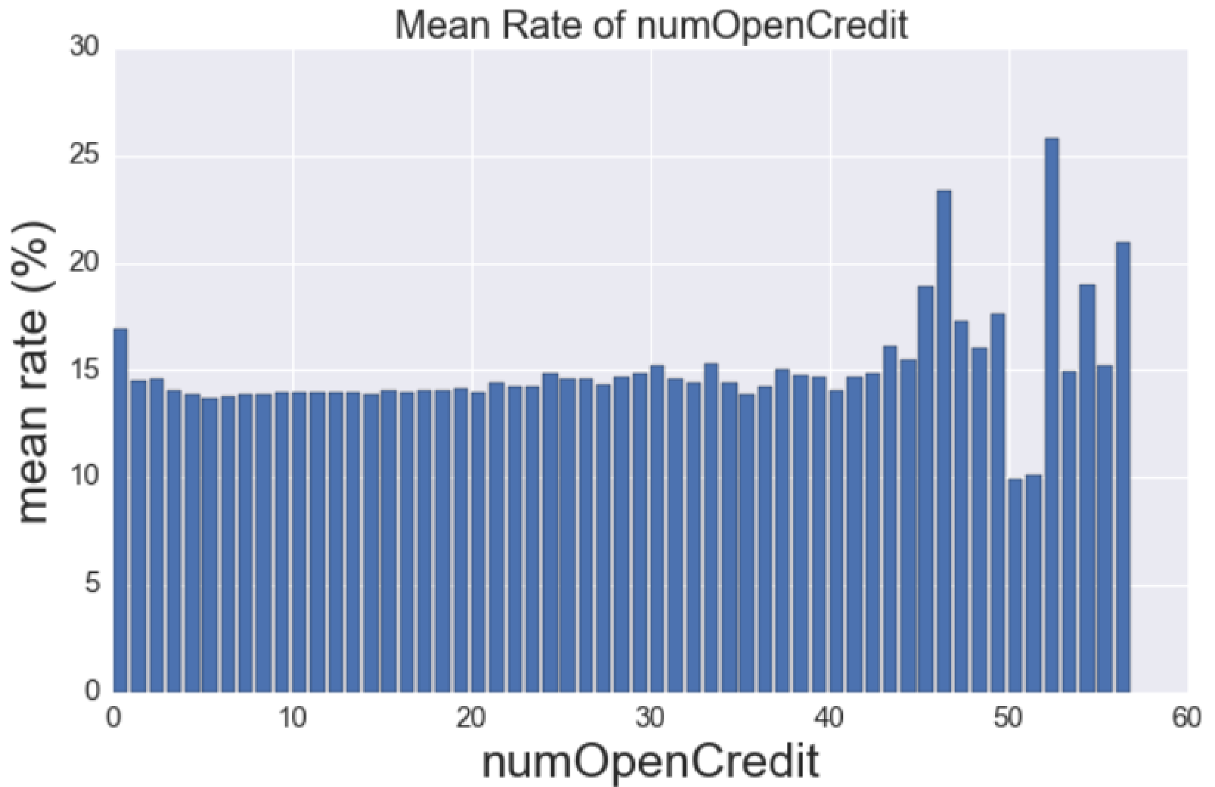


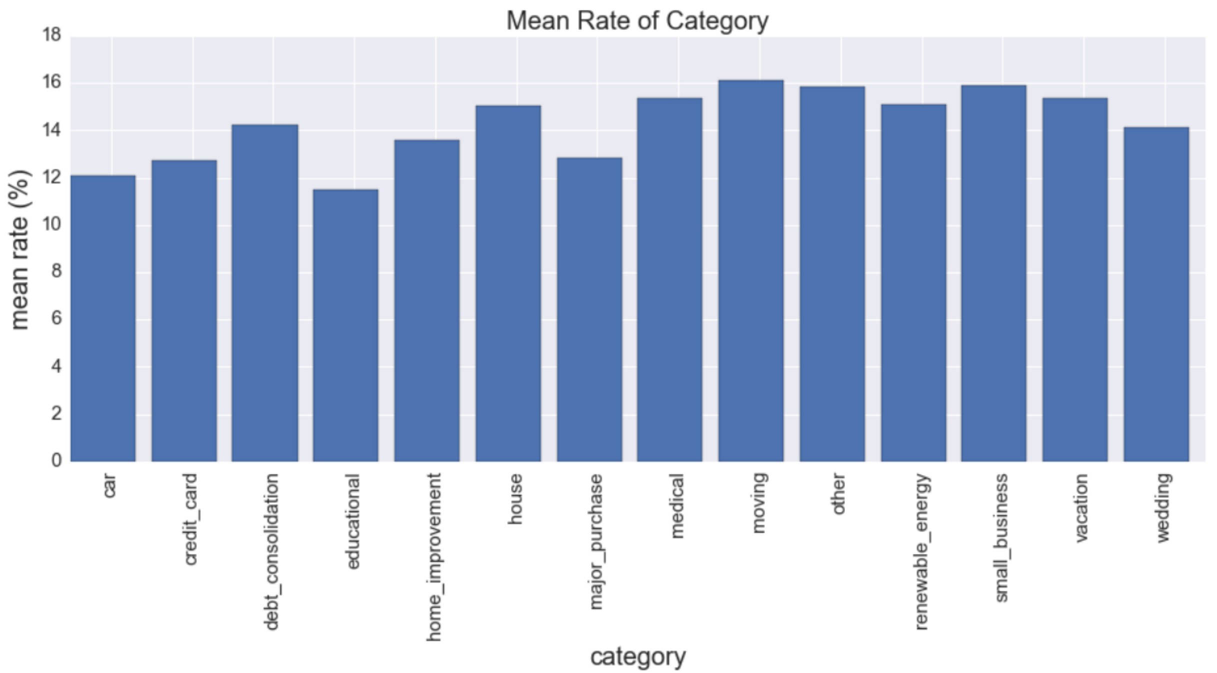


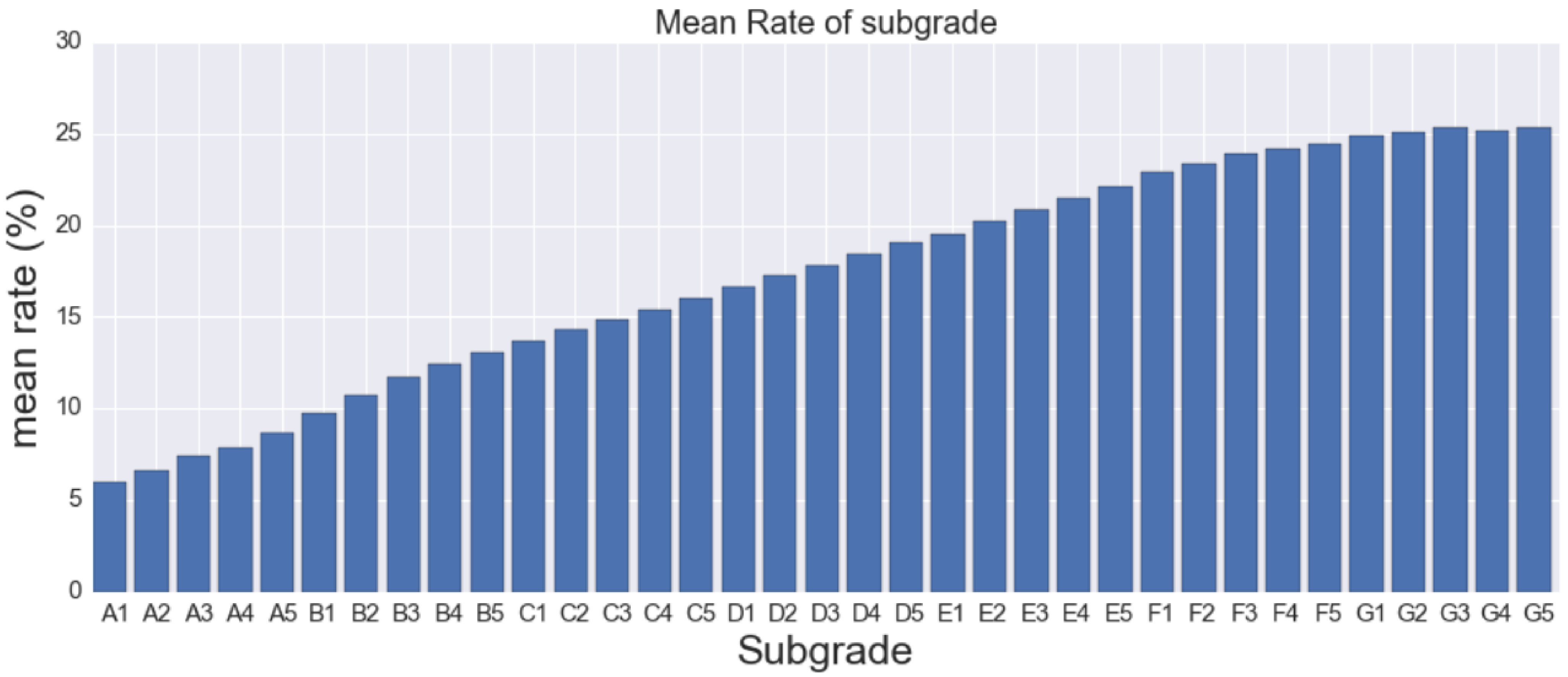



Mean Rate of numInquiry
mean rate (%)
numInquiry


Mean Rate of numTotalCredit
mean rate (%)
numTotalCredit


Mean Rate of numOpenCredit
mean rate (%)
numOpenCredit


Mean Rate of Category
mean rate (%)
car
credit_card
debt_consolidation
educational
home_improvement
house
major_purchase
medical
moving
other
renewable_energy
small_business
vacation
wedding
category


Mean Rate of subgrade
mean rate (%)
Subgrade
0
5
10
15
20
25
30
A1
A2
A3
A4
A5
B1
B2
B3
B4
B5
C1
C2
C3
C4
C5
D1
D2
D3
D4
D5
E1
E2
E3
E4
E5
F1
F2
F3
F4
F5
G1
G2
G3
G4
G5