Credit Default Prediction

Part of the recruitment process at Billie.io

by Hoang Luong

Purpose

- Introduction
- Data
  a. data manipulation
  b. dealing with missing values
- Introduction to Model Used
- Model Tuning
- Performance comparison
- Summary

Introduction

In this project, my goal is to predict credit default, while using results from Suspiciously Datalicious as a benchmark.

Main Outcome:

- Performance improved mainly by choosing right features

Difference from the benchmark:

- Default definition
- Using more powerful algorithms

Data

- Loan data available from Lending Loan Club
- 144 features
- Loan Status as the outcome variable
- 2,260,066.8 observations

Manipulation, Data Wrangling

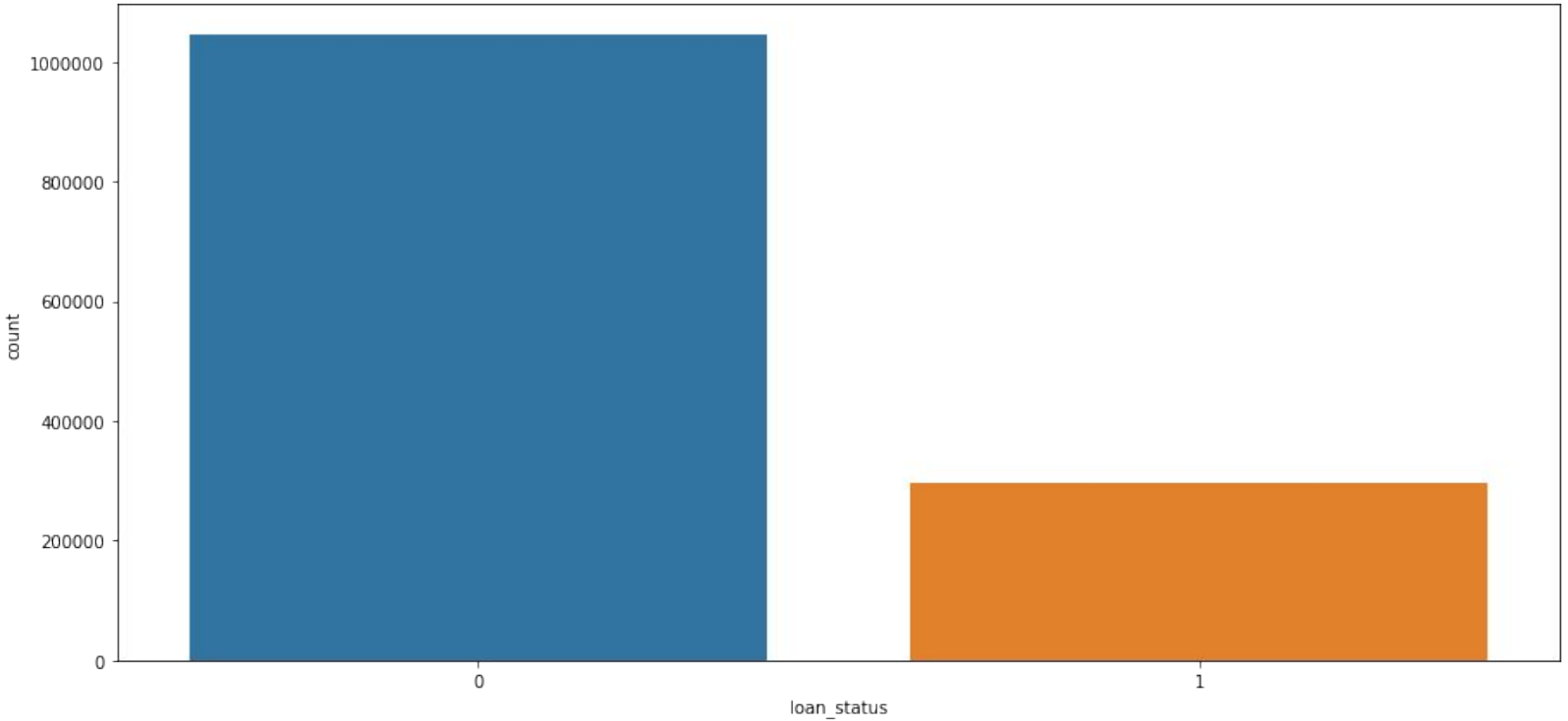
- Removed about 30 variables that are found to be less useful, hard to deal with missing data, or computationally heavy to process
- Took a logarithm of some variables to make them look “normal”
- Imputed missing values, mainly with 0 or mean (not ideal)
- Created the outcome variable
  - Dropped Current Loans. Interested default or non default. Current Loans should be a third category.
  - Defined Churn: { **'Default'**: ['Late (31-120 days)','Late (16-30 days)' ,'In Grace Period', 'Does not meet the credit policy. Status:Charged Off', 'Charged Off', 'Default'], **'Not Default'** : ['Fully Paid']}
  - 1,340,968 observations left. Still a very big amount of data
  - Other option might be just too consider Fully Paid and Default

Manipulation, Data Wrangling

- Transformed some date value to numerical
  - For example hardship_length by using the amount of days between hardshi_start, hardship_end, or experience by calculating day between today and earliest_credit_line
- Created dummies for categorical variables

Due to the time constraint and fact that the project is an extension of other results, most visualisations and description of data can be already found on Suspiciously Datalicious. Here only some “interesting” variables are provided.

Loan Status Histogram



Outcome Variable Discussion

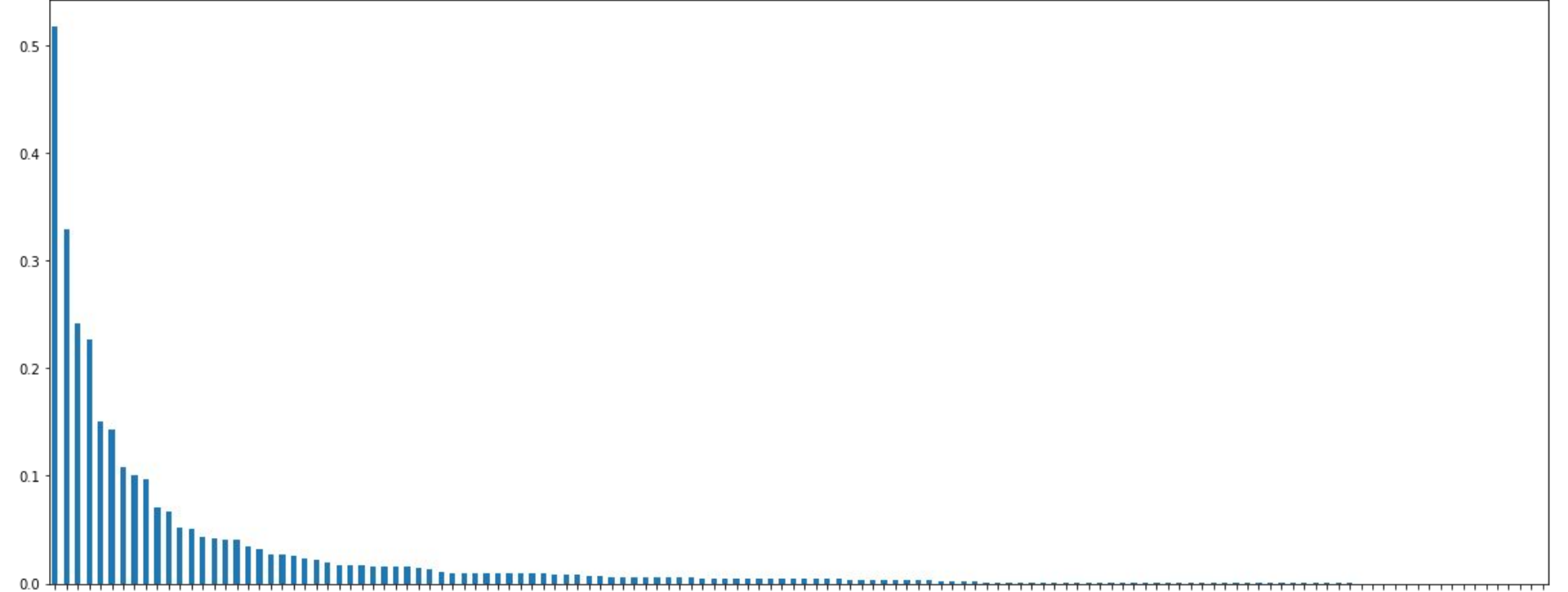
The rate of default is much smaller than the rate of non-default. Accuracy of the model is important but it is just a small part of the story. The emphasize should be put on “Recall,” namely the number of true positives divided by the number of true positives plus the number of false negatives.

Features Picking

- Using Mutual Information (MI) criteria is used to pick get the “best” 40 features
- With the left 40 features, Forward Search Algorithm to pick up 10 features
  - Reason: MI doesn’t consider that some variables might be highly correlated
  - Algorithm:
- 10 variables should be enough to build a good prediction model that does not overfit

Mutual Information (MI)

Intuitively, if x_i is a feature and y is an outcome variable, MI is a measures of how the distribution of p(x_i, y) is different from the p(x_i)p(y). If x_i and y are independent, then we would have p(xi , y) = p(xi)p(y), and the MI between the two distributions would be zero.


0.5
0.4
0.3
0.2
0.1
0.0
total_rec_prncp
last_pymnt_amnt
recoveries
collection_recovery_fee
NO HARDSHIP
application_type
term
total_pymnt
total_pymnt_inv
grade
last_credit_pull_d
debt_consolidation
initial_list_status
int_rate
MORTGAGE
out_prncp
out_prncp_inv
installment
emp_length
total_rec_int
total_rec_late_fee
RENT
Source Verified
num_tl_op_past_12m
Not Verified
num_actv_bc_tl
inq_last_6mths
acc_open_past_24mths
percent_bc_gt_75
mort_acc
num_rev_tl_bal_gt_0
Verified
num_actv_rev_tl
num_bc_sats
open_rv_24m
inq_last_12m
open_act_il
funded_amnt
open_il_24m
loan_amnt
credit_card
open_rv_12m
num_op_rev_tl
num_bc_tl
funded_amnt_inv
num_sats
open_acc
open_acc_6m
all_util
dti_joint
inq_fi
dti
il_util
mths_since_rcnt_il
num_rev_accts
open_il_12m
max_bal_bc
num_il_tl
hardship_length
avg_cur_bal
INTEREST ONLY-3 MONTHS DEFERRAL
hardship_len
total_bal_il
deferral_term
hardship_payoff_balance_amount
annual_inc
hardship_amount
hardship_last_payment_amount
total_acc
total_rev_hi_lim
pub_rec
annual_inc_joint
tot_cur_bal
bc_open_to_buy
num_accts_ever_120_pd
delinq_2yrs
total_cu_tl
revol_util
bc_util
earliest_cr_line
pub_rec_bankruptcies
orig_projected_additional_accrued_interest
OWN
sec_app_num_rev_accts
sec_app_open_act_il
mths_since_last_major_derog
sec_app_open_acc
UNEMPLOYMENT
mths_since_last_delinq
sec_app_revol_util
hardship_flag
revol_bal
mths_since_last_record
num_tl_90g_dpd_24m
sec_app_chargeoff_within_12_mths
revol_bal_joint
DIVORCE
home_ownership
EXCESSIVE_OBLIGATIONS
tax_liens
sec_app_mths_since_last_major_derog
INCOME_CURTAILMENT
tot_coll_amt
num_tl_120dpd_2m
sec_app_inq_last_6mths
sec_app_earliest_cr_line
DISABILITY
mths_since_recent_bc
NONE
pymnt_plan
chargeoff_within_12_mths
acc_now_delinq
wedding
car
MEDICAL
ANY
num_tl_30dpd
medical
NATURAL_DISASTER
educational
house
FAMILY_DEATH
major_purchase
REDUCED_HOURS
OTHER
other
sec_app_collections_12_mths_ex_med
renewable_energy
inq_last_6mths
delinq_amnt
small_business
vacation
collections_12_mths_ex_med

Forward Search Algorithm

1. Initialize an empty set F and a desired amount of features k.
2. Repeat {

(a) For each Feature, if Feature is not in F, add Feature to F and evaluate the model with Features in F via cross validation.

(b) Set F to be the best feature subset

}

3. Stop the algorithm when the amount of elements in F == k

Forward Search Algorithm

After **hours** of computation, I got these variables:

- 'int_rate'
- 'installment'
- 'grade'
- 'out_prncp'
- 'total_pymnt_inv'
- 'total_rec_late_fee'
- 'recoveries'
- 'collection_recovery_fee'
- 'Source Verified'
- 'NO_HARDSHIP'

Modelling

- Since we have a big imbalance between two outcomes (10% default vs. 90 % non-default), just looking at accuracy is a flawed
- Intuition: we don’t want to label/ predict default (True Positive) as non-default (False Negative).
- Recall and Precision
- Extreme case: 90% accuracy and 0% recall would mean that a lender might have used this information and have given loans to all people that would default
- Solution? ROC and F1

ROC and F1

- ROC: compares Specificity vs (1 - Specificity)
- F1: 2*Precision*Recall/(Precision + Recall)
- ROC assumes a balance in the outcome variable, which in our data set is not the case
- F1 might be a better measurement for imbalanced data

Steps

1. Split Data into training and test set (it can also be split into training, test, and validation)
2. Normalize Data
3. Train models on training set
4. Use models on test set
5. Tune parameters if needed
6. Evaluate with metrics

Logistic Regression

We started with a simple Logistic Regression that already gave us a quite nice performance

- F1: 0.94
- Accuracy: 0.97
- Precision: 0.99
- Recall: 0.90
- ROC: 0.94

“Best” Parameters for Random Forest

After running Random Grid Search, I got some improvement of the model.

- F1: 0.92
- Accuracy: 0.96
- Precision: 0.99
- Recall: 0.86
- ROC: 0.93

There is still a room for improvement. Due to a low computational power, I could only test on small grid sizes.

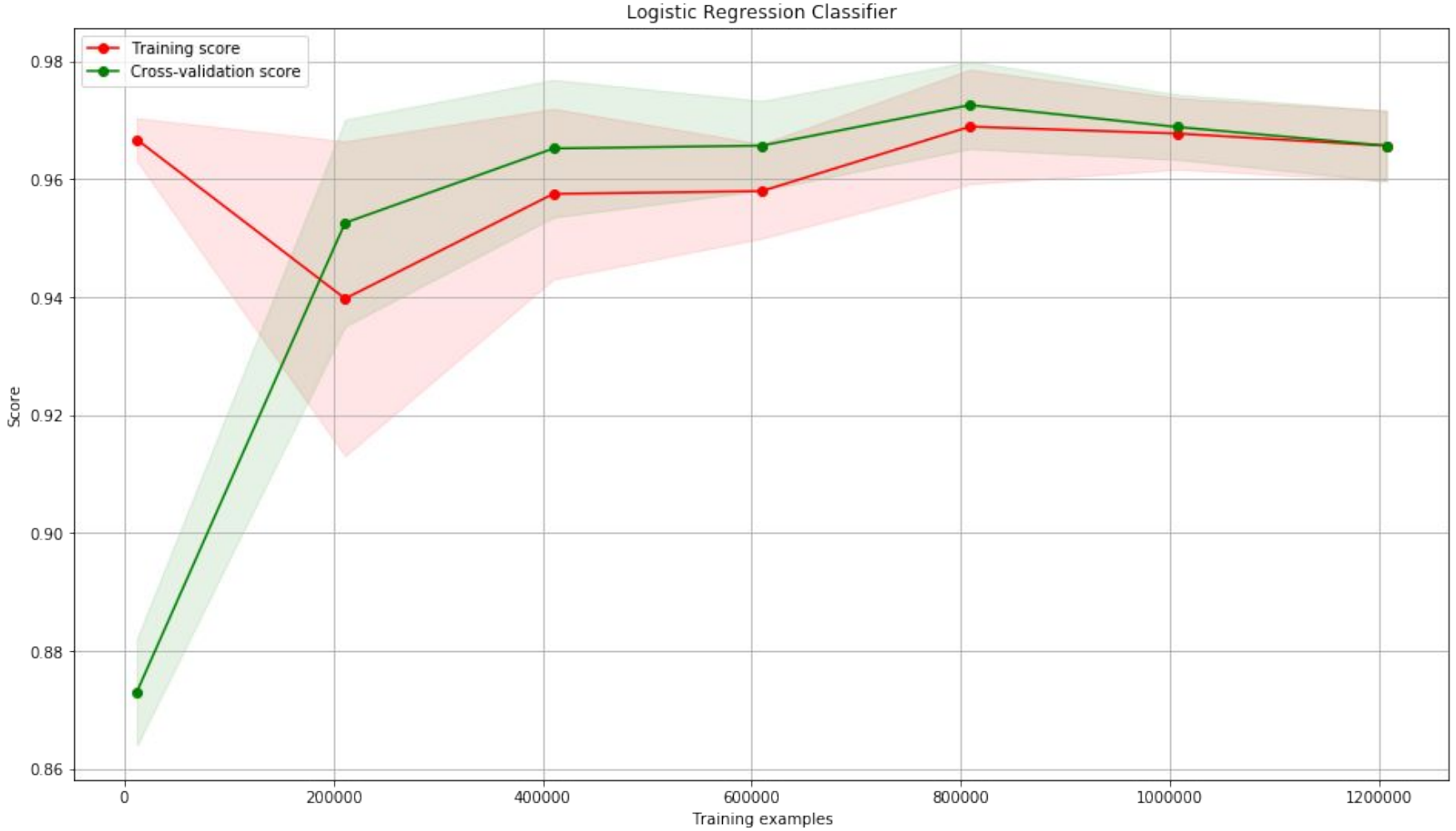
Multilayer Perceptron

Using Neural Network can be useful to find nonlinear relationships, instead of me trying to find manually. After testing different layers, a model with 2 layers with 5 nodes each gave a good performance. Optimizing on recall and precision.

- F1: 0.91
- Accuracy: 0.96
- Precision: 0.99
- Recall: 0.84
- ROC: 0.92

Multilayer Perceptron

Model was run with an optimization algorithm Adam, with default parameters. Loss function used was Binary Cross-entropy. Activation function were ReLu and Sigmoid. There is a big room for improvement. Due to computation constraint, like in Random Forest I couldn’t experiment too much with the parameters.


Logistic Regression Classifier
Training score
Cross-validation score
0.98
0.96
0.94
0.92
0.90
0.88
0.86
Score
0
200000
400000
600000
800000
1000000
1200000
Training examples

Summary

- The performance was mainly improved by using algorithms to find “right” features.
- Best Performing algorithm was a surprisingly Logistic Regression, followed by Random Forest, and Neural Network (Note that little tuning was done)

Thank you!

Regardless of the outcome I would love to hear your feedback in order to improve for the future. I was actually very surprised that the results were much better than from the website.