Machine Learning Engineer Nanodegree

Capstone Project

Predicting charge-off in LendingClub loan data set

Antal Berenyi
May 3, 2018

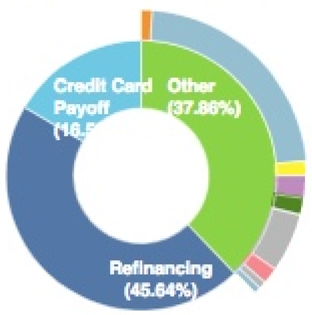
I. Definition

Project Overview

Domain Background

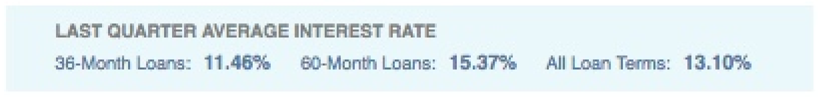
Lending Club (LC) operates an online, peer-to-peer lending platform that enables borrowers to obtain a loan, and investors to purchase notes backed by payments made on loans. The company claims that $15.98 billion in loans had been originated through its platform up to December 31, 2015. wikipedia (https://en.wikipedia.org/wiki/Lending_Club) Loans can be issued for a variety of purposes, such as loan consolidation, car purchase, medical, etc. Loans are issued as $25 notes so that lenders may diversify their investment over many loans. Loan terms are either 36 or 60 months.

REPORTED LOAN PURPOSE



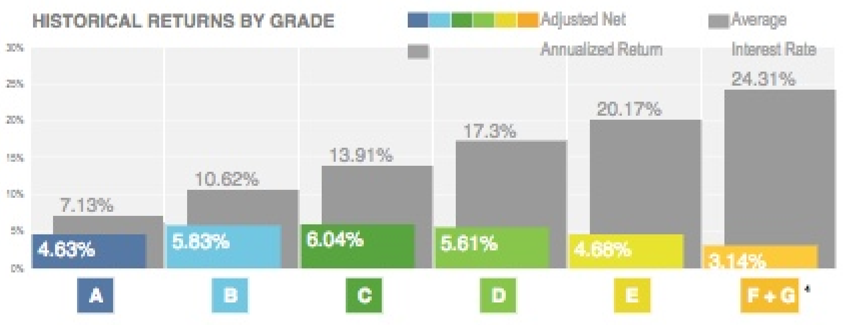
62.14% of Lending Club borrowers report using their loans to refinance existing loans or pay off their credit cards as of 12/31/17.[1]

LC divides loans into categores A,B,C,D,E,FG, based on variables that measure the quality of the loan. A is highest and FG is lowest. Category A loans offers the lowest rate of return backed by borrowes with highest credit rating therefore it is the safest investment. FG are the riskiest loans with high interest rate but least likely to be repaid.



LAST QUARTER AVERAGE INTEREST RATE

36-Month Loans: **11.46%** 60-Month Loans: **15.37%** All Loan Terms: **13.10%**



[source (https://www.lendingclub.com/info/demand-and-credit-profile.action)](https://www.lendingclub.com/info/demand-and-credit-profile.action)

If a borrower does not pay a loan installment on time, its status changes from `current` to `grace period` for 14 days. After that the loan status changes to `delinquent` , then to `late` , then `charged off` after 3 months if not paid.

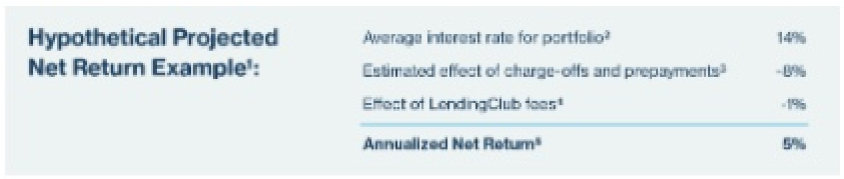
I have been investing in LC for about 3 years. Over this time period I have invested in a range of loan classes. About 50% of the interest earned was erased by load charge-offs. Identifying loans likely to be charged-off means that investor profit could be increased by not investing in those notes. [invesing in LC (https://blog.lendingclub.com/5-key-things-know-investing-lendingclub-notes/)](https://blog.lendingclub.com/5-key-things-know-investing-lendingclub-notes/)

In this section, look to provide a high-level overview of the project in layman's terms. Questions to ask yourself when writing this section:

- *Has an overview of the project been provided, such as the problem domain, project origin, and related datasets or input data?*
- *Has enough background information been given so that an uninformed reader would understand the problem domain and following problem statement?*

Problem Statement

The goal is to identify which loans are likely to be charge-off. Charge-offs impact investor returns because investors lose both investment capital plus the potential to earn interest.



Hypothetical Projected Net Return Example[1]:		
	Average interest rate for portfolio[2]	14%
	Estimated effect of charge-offs and prepayments[3]	-8%
	Effect of LendingClub fees[4]	-1%
	Annualized Net Return[5]	**5%**

[source (https://blog.lendingclub.com/5-key-things-know-investing-lendingclub-notes/)](https://blog.lendingclub.com/5-key-things-know-investing-lendingclub-notes/)

To solve this problem we need to identify a correlation between loan features and "charge-off" probability. In order to classify loans as "charge-off" vs. "non-charge-off" we need to do the following tasks:

- Obtain loan data containing details about the loan terms and borrower information
- Establish benchmark metrics against which to measure classifier
- Clean and pre-process data for feeding the classifier
- Tune the classifier

- Predict loan charge-off

To quantify the problem: the charge-off rate can be calculated as the percentage of loans with status "charge-off" from the data set.

To measure performance: The trained classifier should perform better than the bechmark classifier; it should identify charge-off loans better than a naive classifier. The naive classifier picks notes randomly as `charge-off` loans.

Metrics

In this section, you will need to clearly define the metrics or calculations you will use to measure performance of a model or result in your project. These calculations and metrics should be justified based on the characteristics of the problem and problem domain. Questions to ask yourself when writing this section:

- *Are the metrics you've chosen to measure the performance of your models clearly discussed and defined?*
- *Have you provided reasonable justification for the metrics chosen based on the problem and solution?*

I am planning to measure performance by Recall. The recall is the ratio tp / (tp + fn) where tp is the number of true positives and fn the number of false negatives. The recall is intuitively the ability of the classifier to find all the positive samples (charge-offs). This problem is similar to a fraud detector where we aim to catch all positive cases. We don't want a very high rate of False positives either because that would eliminate valid investment opportunities, so reporting a Precision score is also meaningful.

Evaluating performance

To measure performance: The trained classifier should perform better than a naive classifier; it should identify charge-off loans with a better Recall than a naive classifier. The naive classifier picks notes randomly with the same probability as the percentage of actual `charge-off` loans.

The best evaluation metric would be Recall. We want to label all charged off loans to minimize our loss. It is OK to mislabel a few loans as false positive, so the situation is similar to a fraud detector or spam filter, where we are aiming for high Recall. Metric must be better than for the Naive predictor: Recall > 0.5 and F2 > 0.132.

For the Naive predictor, given about 3.35% charge-off rate, the metrics would be:

Accuracy	Precision	Recall	F2
0.5	0.0335	0.5	0.132

Datasets and Inputs

LC collects extensive statistics on borrowers that they make their rating based on. The borrower data used for this project is publicly available on the LC web site https://www.lendingclub.com/info/download-data.action-data (https://www.lendingclub.com/info/download-data.action). It contains all the LendingClub loan information available for investors to make a decision about whenter to fund the loan or not. The https://resources.lendingclub.com/LCDataDictionary.xlsx (https://resources.lendingclub.com/LCDataDictionary.xlsx) file lists the feature with explanation about each feature. The data is in zipped up .csv format that can be imported in Excel, Pandas or other tools.

This data set should contain a mix of 3yr and 5yr loans that originated up to 5 years ago and new newer loans, with all possible status. For example the Q1 2017 data set contains anonymized information for about 96,781 loans with 151 features.

The zipped file size is about 22MB while unzipped it is 110 MB. Each row represents a loan. Each column is a feature. Column "loan_status" is the target variable to be predicted. The finacial data set is donwloadable in quarterly and yearly chunks. These features relate information about the loand and the borrower: `loan amount, interest rate, grade, purpose; borrower income, geography, employment status, FICO score, etc.` This information will be used to find a good indicator (predictor) of why a borrower may default on a loan.

- https://resources.lendingclub.com/LoanStats_2017Q1.csv.zip (https://resources.lendingclub.com/LoanStats_2017Q1.csv.zip)
- https://resources.lendingclub.com/LoanStats_2017Q2.csv.zip (https://resources.lendingclub.com/LoanStats_2017Q2.csv.zip)
- https://resources.lendingclub.com/LoanStats_2016Q4.csv.zip (https://resources.lendingclub.com/LoanStats_2016Q4.csv.zip)

II. Analysis

Data Exploration

Sample Data

Table 1. This table is an example of the composition of the Traing data set used.

	term	int_rate	installment	grade	emp_title	emp_length
0	36 months	5.32%	451.7300109863281	A	Executive Account Manager	10+ years
1	36 months	7.49%	528.72998046875	A	Air Traffic Controller	10+ years
2	36 months	5.32%	602.2999877929688	A	Associate Dentist	3 years
3	60 months	12.74%	361.92999267578125	C	claims analyst	10+ years
4	36 months	16.99%	71.30000305175781	D	Supervisor	6 years
5	36 months	11.44%	391.260009765625	B	Shop foreman	10+ years
6	36 months	14.99%	173.30999755859375	C	Lead Supervisor	10+ years
7	36 months	12.74%	483.3999938964844	C	nan	nan
8	36 months	13.49%	230.72999572753906	C	Mechanic	< 1 year
9	36 months	6.99%	401.3500061035156	A	Mold Maker	10+ years
10	36 months	7.99%	313.32000732421875	A	Pipelayer	1 year
11	60 months	14.99%	356.7799987792969	C	Senior Superintendent	10+ years
12	36 months	7.49%	311.0199890136719	A	Product Manager	4 years
13	36 months	8.24%	157.24000549316406	B	POLICE COMMUNICATIONS OPERATOR	4 years
14	36 months	11.39%	233.75999450683594	B	nan	nan
15	36 months	12.74%	335.69000244140625	C	Product Specialist	10+ years
16	36 months	11.39%	276.55999755859375	B	IT Tech	8 years
17	36 months	11.39%	151.4499969482422	B	cable technician	4 years
18	36 months	15.99%	165.22000122070312	C	RN Manager	10+ years
19	36 months	6.99%	93.38999938964844	A	Recruiter	10+ years

	home_ownership	annual_inc	verification_status	issue_d	loan_status	pymnt_plan	purpose	zip_code
0	MORTGAGE	182000.0	Not Verified	Mar-2017	Current	n	debt_consolidation	751xx
1	MORTGAGE	120000.0	Not Verified	Mar-2017	Fully Paid	n	debt_consolidation	840xx
2	RENT	120000.0	Not Verified	Mar-2017	Current	n	credit_card	926xx
3	MORTGAGE	130000.0	Not Verified	Mar-2017	Current	n	debt_consolidation	577xx
4	MORTGAGE	62000.0	Not Verified	Mar-2017	Current	n	credit_card	983xx
5	MORTGAGE	55000.0	Not Verified	Mar-2017	Current	n	home_improvement	480xx
6	RENT	68000.0	Not Verified	Mar-2017	Fully Paid	n	debt_consolidation	945xx
7	MORTGAGE	61000.0	Verified	Mar-2017	Current	n	credit_card	864xx
8	RENT	55000.0	Not Verified	Mar-2017	Current	n	credit_card	908xx
9	MORTGAGE	120000.0	Source Verified	Mar-2017	Current	n	debt_consolidation	600xx
10	RENT	69000.0	Not Verified	Mar-2017	Current	n	debt_consolidation	433xx
11	MORTGAGE	125000.0	Not Verified	Mar-2017	Current	n	other	064xx
12	RENT	85000.0	Not Verified	Mar-2017	Current	n	debt_consolidation	021xx
13	RENT	49000.0	Not Verified	Mar-2017	Current	n	credit_card	187xx
14	RENT	32160.0	Not Verified	Mar-2017	Current	n	debt_consolidation	329xx
15	OWN	40000.0	Not Verified	Mar-2017	Fully Paid	n	debt_consolidation	324xx
16	MORTGAGE	50000.0	Source Verified	Mar-2017	Late (31-120 days)	n	other	454xx
17	RENT	44470.0	Not Verified	Mar-2017	Current	n	credit_card	066xx
18	MORTGAGE	100046.0	Source Verified	Mar-2017	Current	n	debt_consolidation	847xx
19	MORTGAGE	83000.0	Not Verified	Mar-2017	Current	n	vacation	301xx

Features

The 2017Q1 LC data set has 145 features and 96781 rows available in the data set for analysis for loans issued in that time period. I am working with this particular subset because it has a mixture there hase been sufficient time since issuance for loans to go to default.

I ended up using only 100 features; some of the original ones, and engineered features such as polynomial interactions to boost prediction performance. A description of the features can be read in the data dictionary https://resources.lendingclub.com/LCDataDictionary.xlsx (https://resources.lendingclub.com/LCDataDictionary.xlsx).

- member_id - This will be removed because the data set is anonymized anyways.
- loan_status - Current status of the loan. This is a categorical feature but I encode 1 = "charge-off", 0 = "good"
- loan_amnt - The listed amount of the loan applied for by the borrower. The max loan amount is $ 40,000.
- term - The number of payments on the loan. Values are in months and can be either 36 or 60.
- int_rate - Interest Rate on the loan
- installment - The monthly payment owed by the borrower if the loan originates.
- grade - LC assigned loan grade. A,B,C,D,E,F - that need to be one-hot-encoded
- emp_length - Employment length in years. Possible values are between 0 and 10 where 0 means less than one year and 10 means ten or more years.
- home_ownership - The home ownership status provided by the borrower during registration or obtained from the credit report. Our values are: RENT, OWN, MORTGAGE, OTHER
- annual_inc - The self-reported annual income provided by the borrower during registration. It is numerical and has several outliers, which were removed.
- verification_status - Indicates if income was verified by LC, not verified, or if the income source was verified
- issue_d - The month which the loan was funded
- purpose - A category provided by the borrower for the loan request.

- zip_code - The first 2 numbers of the zip code provided by the borrower in the loan application.
- dti - A ratio calculated using the borrower’s total monthly debt payments on the total debt obligations, excluding mortgage and the requested LC loan, divided by the borrower’s self-reported monthly income.
- delinq_2yrs - The number of 30+ days past-due incidences of delinquency in the borrower's credit file for the past 2 years
- earliest_cr_line - The month the borrower's earliest reported credit line was opened
- fico_range_low - The lower boundary range the borrower’s FICO at loan origination belongs to. Drop.
- fico_range_high - The upper boundary range the borrower’s FICO at loan origination belongs to. Drop.
- fico_range_mean - The mean values of fico_range_low and fico_range_high, which will beused instead of high and low values.
- inq_last_6mths - The number of inquiries in past 6 months (excluding auto and mortgage inquiries)
- mths_since_last_delinq - The number of months since the borrower's last delinquency.
- mths_since_last_record - The number of months since the last public record.
- open_acc - The number of open credit lines in the borrower's credit file.
- pub_rec - Number of derogatory public records
- emp_title - The job title supplied by the Borrower when applying for the loan.* The title to loan ration is 1/3, therefore most values are unique. I have decided to encode title as missing-not missing.
- revol_bal - Total credit revolving balance
- revol_util - Revolving line utilization rate, or the amount of credit the borrower is using relative to all available revolving credit.
- total_acc - The total number of credit lines currently in the borrower's credit file
- initial_list_status - The initial listing status of the loan. Possible values are – W, F
- last_fico_range_high - The upper boundary range the borrower’s last FICO pulled belongs to.
- last_fico_range_low - The lower boundary range the borrower’s last FICO pulled belongs to.
- collections_12_mths_ex_med - Number of collections in 12 months excluding medical collections
- mths_since_last_major_derog - Months since most recent 90-day or worse rating
- application_type - Indicates whether the loan is an individual application or a joint application with two co-borrowers
- annual_inc_joint - The combined self-reported annual income provided by the co-borrowers during registration
- dti_joint - A ratio calculated using the co-borrowers' total monthly payments on the total debt obligations, excluding mortgages and the requested LC loan, divided by the co-borrowers' combined self-reported monthly income
- acc_now_delinq - The number of accounts on which the borrower is now delinquent.
- tot_coll_amt - Total collection amounts ever owed
- tot_cur_bal - Total current balance of all accounts
- open_acc_6m - Number of open trades in last 6 months
- open_act_il - Number of currently active installment trades
- open_il_12m - Number of installment accounts opened in past 12 months
- open_il_24m - Number of installment accounts opened in past 24 months
- mths_since_rcnt_il - Months since most recent installment accounts opened
- total_bal_il - Total current balance of all installment accounts
- il_util - Ratio of total current balance to high credit/credit limit on all install acct

- open_rv_12m - Number of revolving trades opened in past 12 months
- open_rv_24m - Number of revolving trades opened in past 24 months
- max_bal_bc - Maximum current balance owed on all revolving accounts
- all_util - Balance to credit limit on all trades
- inq_fi - Number of personal finance inquiries
- total_cu_tl - Number of finance trades
- inq_last_12m - Number of credit inquiries in past 12 months
- acc_open_past_24mths - Number of trades opened in past 24 months.
- avg_cur_bal - Average current balance of all accounts
- bc_open_to_buy - Total open to buy on revolving bankcards.
- bc_util - Ratio of total current balance to high credit/credit limit for all bankcard accounts.
- chargeoff_within_12_mths - Number of charge-offs within 12 months
- delinq_amnt - The past-due amount owed for the accounts on which the borrower is now delinquent.
- mo_sin_old_il_acct - Months since oldest bank installment account opened
- mo_sin_old_rev_tl_op - Months since oldest revolving account opened
- mo_sin_rcnt_rev_tl_op - Months since most recent revolving account opened
- mo_sin_rcnt_tl - Months since most recent account opened
- mort_acc - Number of mortgage accounts.
- mths_since_recent_bc - Months since most recent bankcard account opened.
- mths_since_recent_inq - Months since most recent inquiry.
- num_accts_ever_120_pd - Number of accounts ever 120 or more days past due
- num_actv_bc_tl - Number of currently active bankcard accounts
- num_actv_rev_tl - Number of currently active revolving trades
- num_bc_sats - Number of satisfactory bankcard accounts
- num_bc_tl - Number of bankcard accounts
- num_il_tl - Number of installment accounts
- num_op_rev_tl - Number of open revolving accounts
- num_rev_accts - Number of revolving accounts
- num_rev_tl_bal_gt_0 - Number of revolving trades with balance >0
- num_sats - Number of satisfactory accounts
- num_tl_30dpd - Number of accounts currently 30 days past due (updated in past 2 months)
- num_tl_90g_dpd_24m - Number of accounts 90 or more days past due in last 24 months
- num_tl_op_past_12m - Number of accounts opened in past 12 months
- pct_tl_nvr_dlq - Percent of trades never delinquent
- percent_bc_gt_75 - Percentage of all bankcard accounts > 75% of limit.
- pub_rec_bankruptcies - Number of public record bankruptcies
- tax_liens - Number of tax liens
- tot_hi_cred_lim - Total high credit/credit limit
- total_bal_ex_mort - Total credit balance excluding mortgage
- total_bc_limit - Total bankcard high credit/credit limit
- total_il_high_credit_limit - Total installment high credit/credit limit
- Join borrowers columns: I have filled the missing values with zeros for Individual borrowers.

** I have decided to not consider hardship_XXX features since it is very sparse, all values except one are N. I have removed all the settlement columns since they are indiactive what happens AFTER the charge-off.

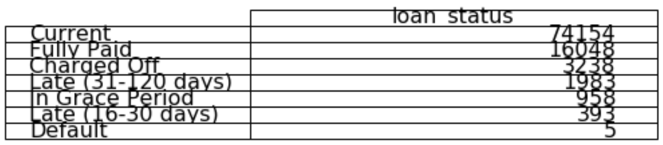
Exploratory Visualization

I have developed a visualization methods that compares the distribution of the null class agains the positive class (charge-off). My aim was to visualize the difference between positive and negative class means; the most relevant feature would appear as having difference between the means of the positive and negative distribtion.

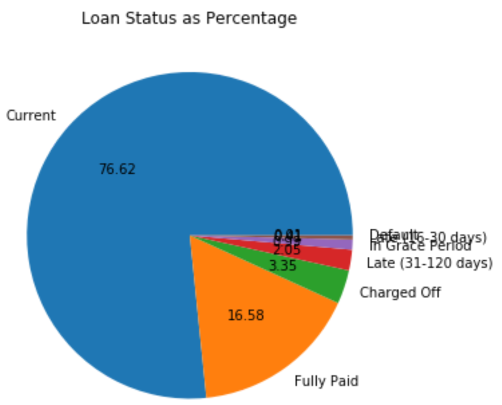
loan_status

First we look at the target distribution, loan_status. We can see that the target class is a minority class so we'll need to compensate for it during training to achieve a balanced training set. We'll also encode it as Charged-off=1, all other = 0.

This is the target to be predicted. We need to encode it: Charged-off and default is 1, all the others are 0.

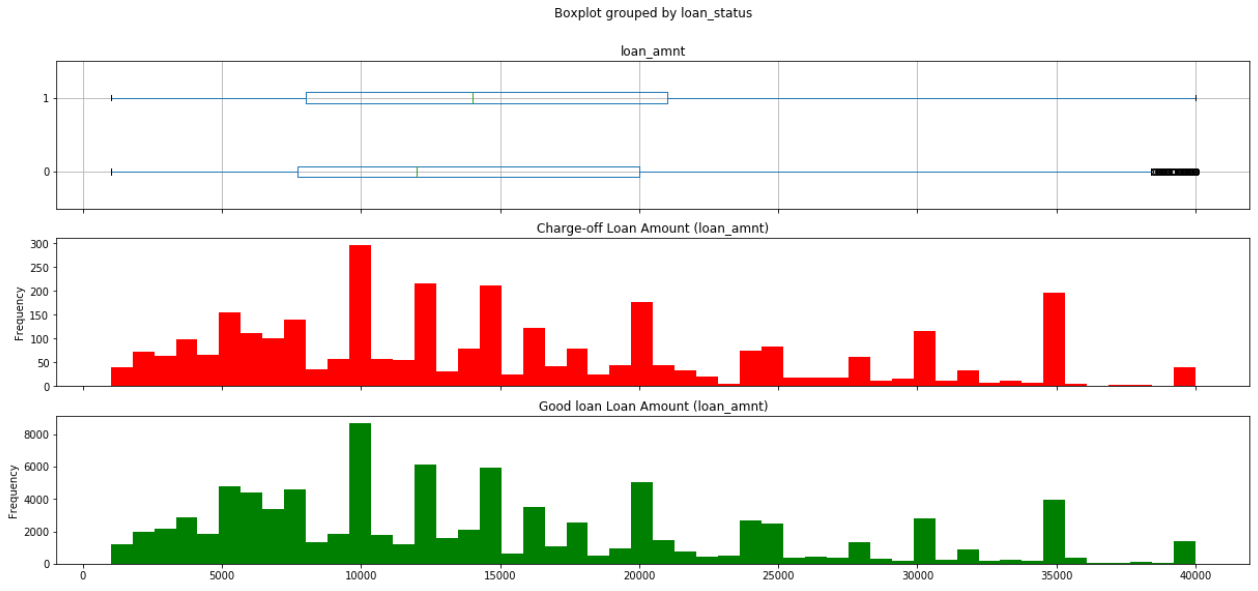


	loan_status
Current	74154
Fully Paid	16048
Charged Off	3238
Late (31-120 days)	1983
In Grace Period	958
Late (16-30 days)	393
Default	5



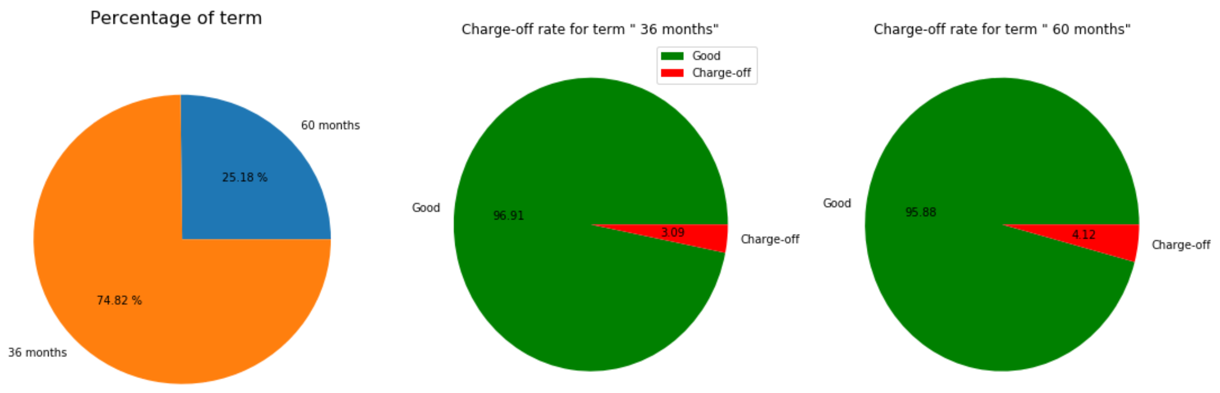
loan_amount

Plot of the Distribution of Loan Amount. It is a numerical feature, mean of 12,500 we can leave use it as is. This plot shows the loan amount grouped by charge-off status and that the mean of charge-off loans is about 2000 higher than of good loans.



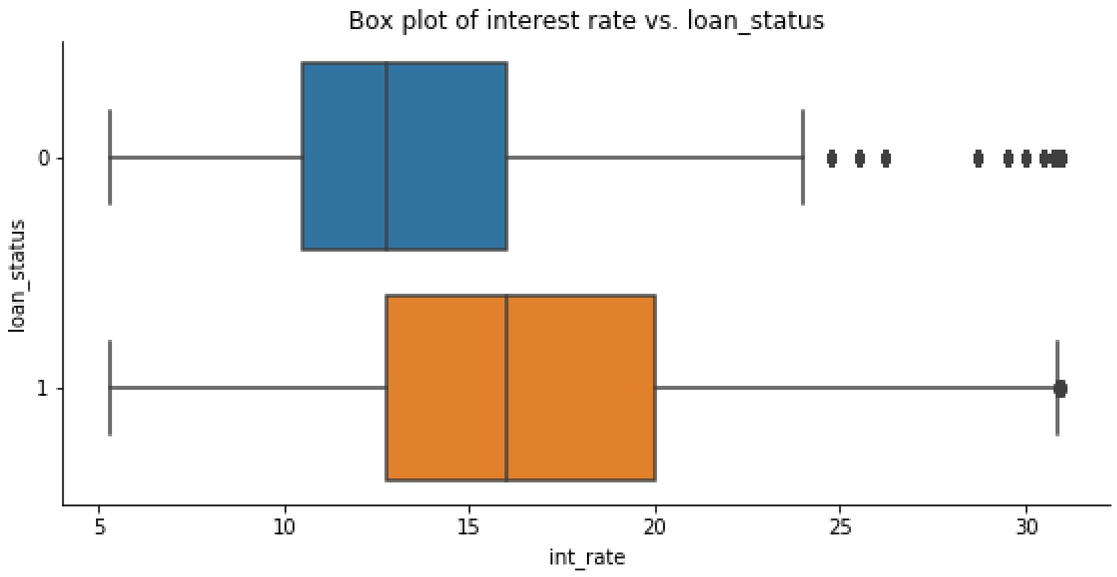
Term

There are only 36 and 60 month terms. Charge-off rate for 60 month loans is 4% vs. 3% for good loans. It is more risky to give out loan for long term.



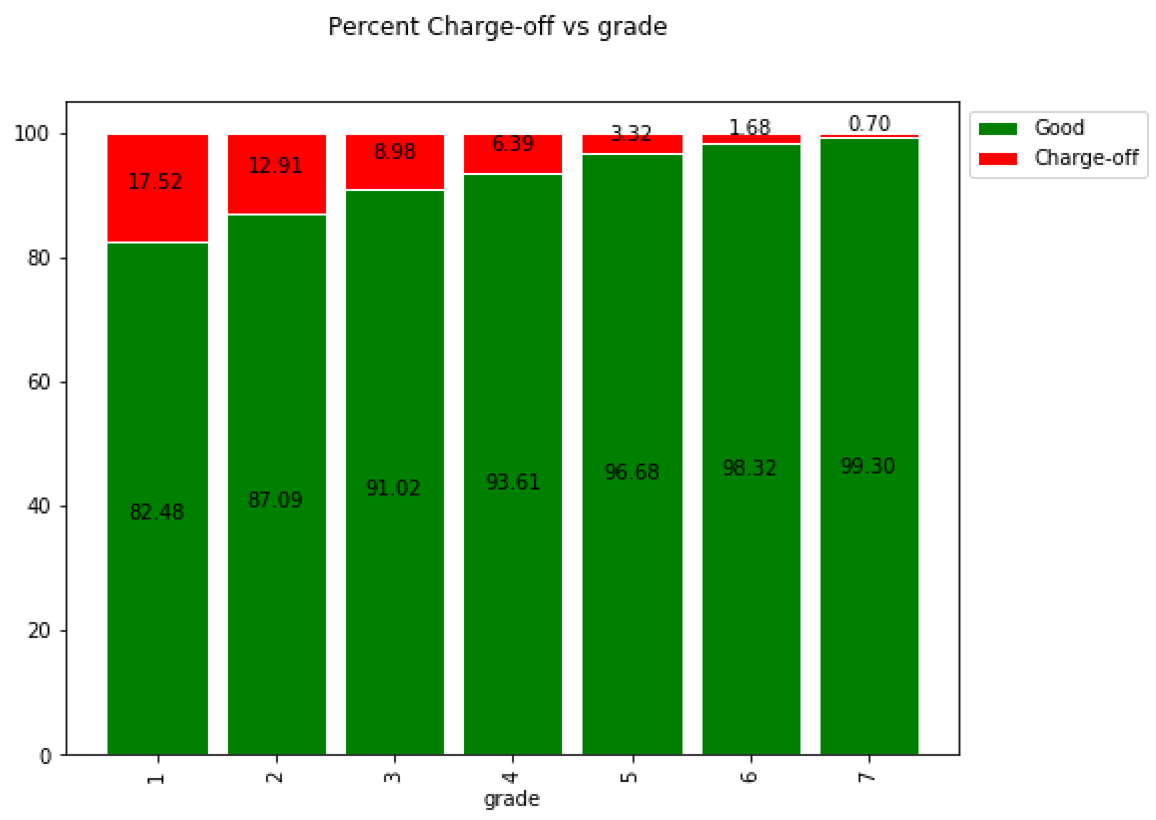
int_rate

Interest rate is a percentage value. We need to keep the float value of the string. The statistics for the interest rate are in the tables below. Overall the charge-off loans have higher mean interest rate of 17.45%, while the good loans have 13.23%.



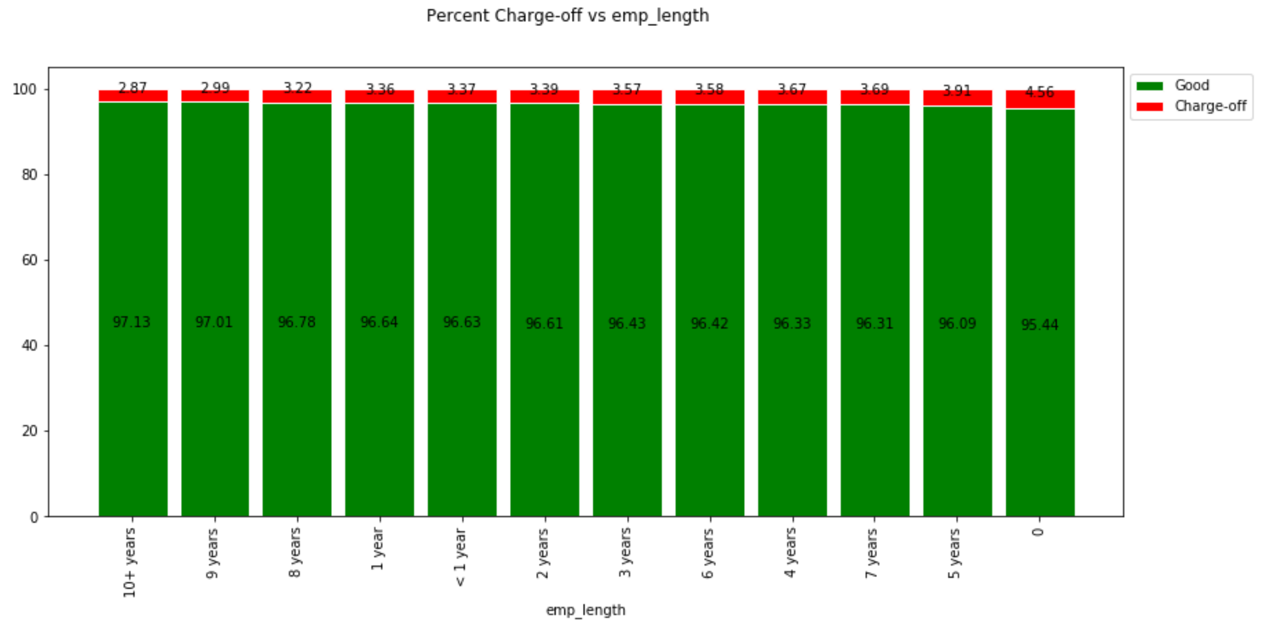
grade

Loans are divided into sub-categores based on borrower rating. A is best and G is worst. I will one-hot encode each grade as a number from 1 to 7. The factor plot also reveals that the worse the loan grade the higher the loan amount, but unfortunately the loan_amount median values for chargeoff and good loans are about the same for each grade.



emp_length

About one third of the loans were given to individuals with 10+ years of employment. The other 2/3 are given in a much lower proportion to shorter term. 10+ is a collective buckes as includes employment lenghts of 11, 12, 15, 20 up to retirement. I encode employment length with the number of years, because there is no clear correlation between emp_length and charge-off. The label is the year. 10+ years is encoded as 12, < 1 year is 0.5, and where not available it is 0.

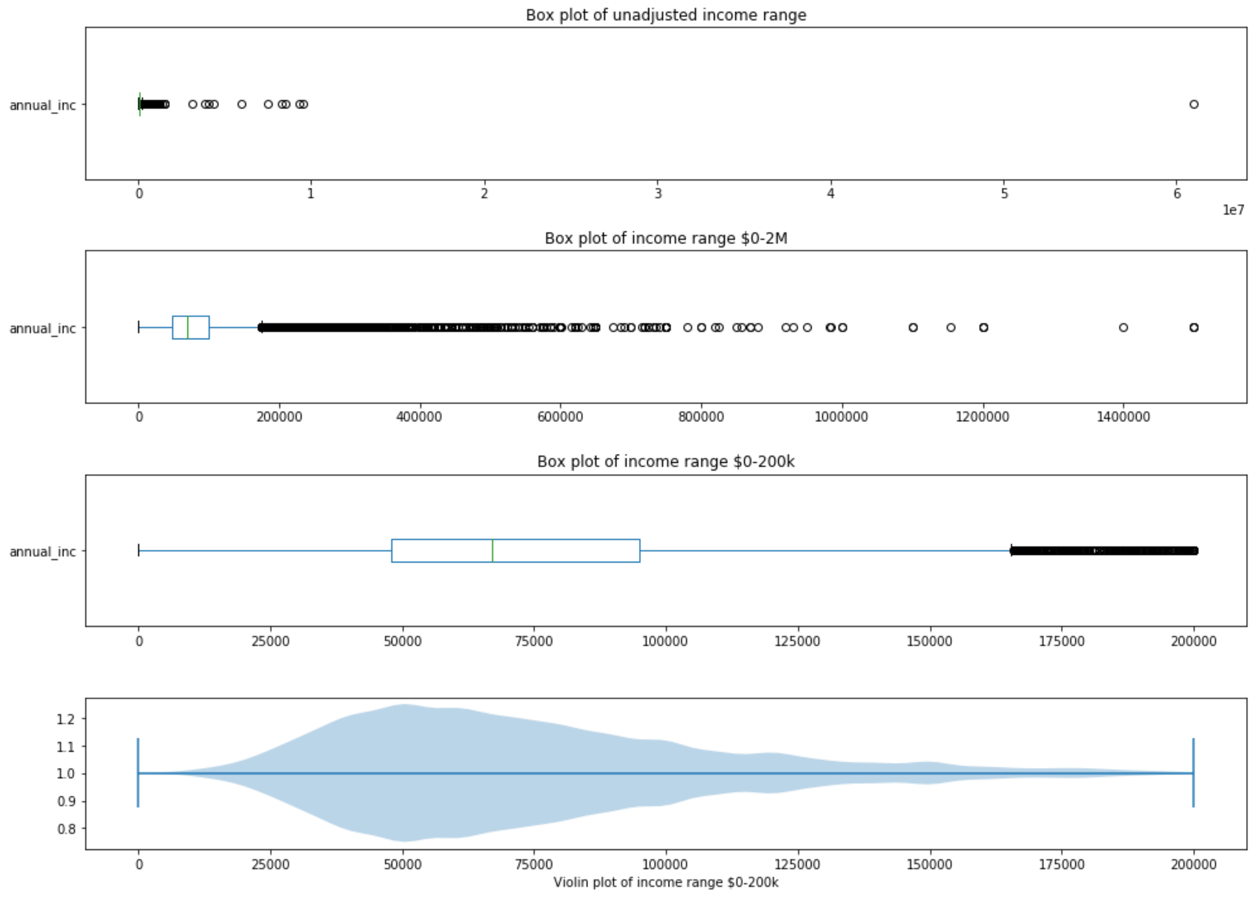


home_ownership

About half of the loan recipients pay mortgage, compared to renting, owning or other. It begs to ask how much money borrowers have left over after paying their mortgages? It seems reasonable to think that people with a long a reliable mortgage history may have higher rating. It will be worth investigating correlation between Home Ownership and other features. Also does NONE mean that a borrower is homeless? Other features such as car ownership would be useful to know: a person may default on other loans before a car loan.

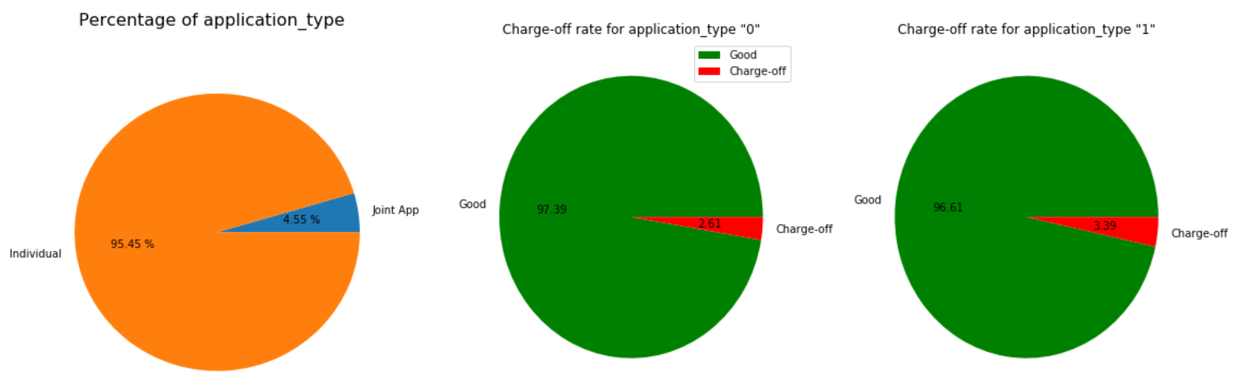
annual_inc

The annual income is perhaps one of the pivotal features. It is worth taking a deeper look. The histogram below shows income distribution in the $0-200k range. There appears to be spurious data, for illustration see the table. There is a borrower with income $61 million and emp_title indicated waitress. There is some data that is suspicious and unlikely. It may be worth asking LC a few questions about their verification process! Also, it appears as if there were three nested distributions, I am not sure if it is real. The log-transformed annual_inc shows a better distribution



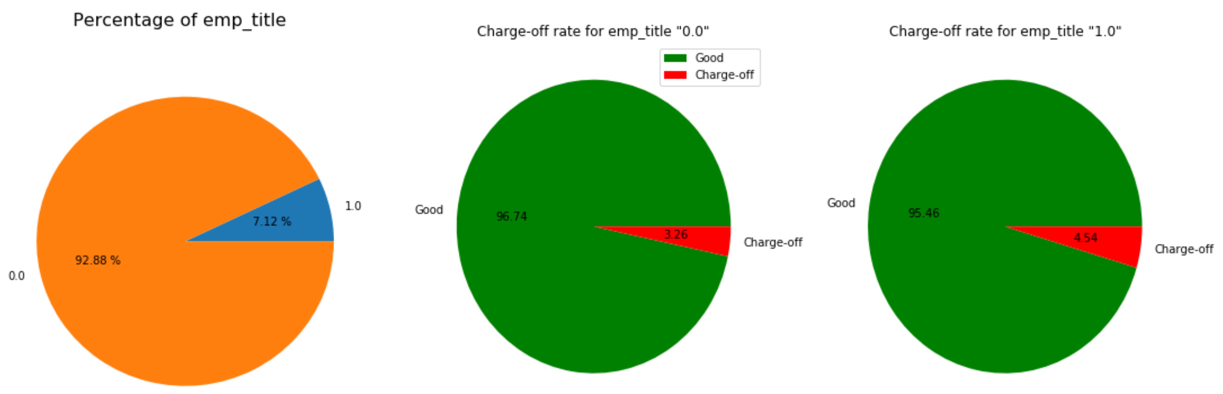
application_type

Some people apply for loans jointly with other individuals. The joint status type of the loan is indicated by this field. Joint application is 0, Individual is 1. Joint loans appear to be safer because there are two people responsible for repaying it.



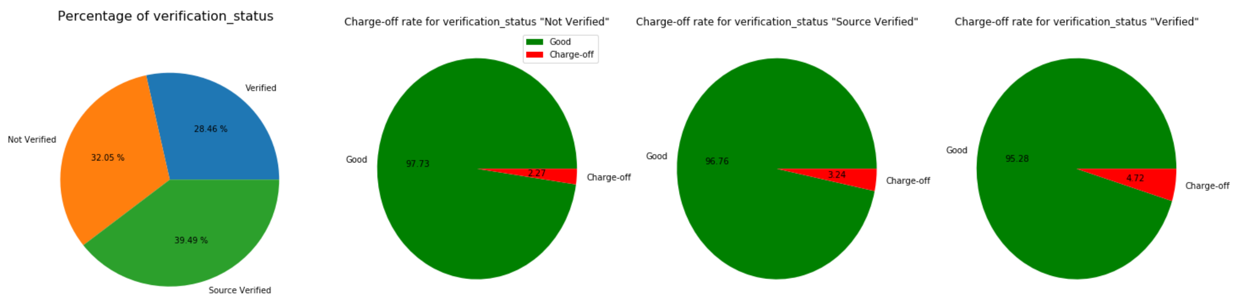
emp_title

The output below shows that about 7% of loans have no employment title. This may be indicative. It may be worth encoding it a title/no title. There is a very large number of different titles, one out of three loans have a unique employment title. Missing employee title has a little higher charge-off rate.



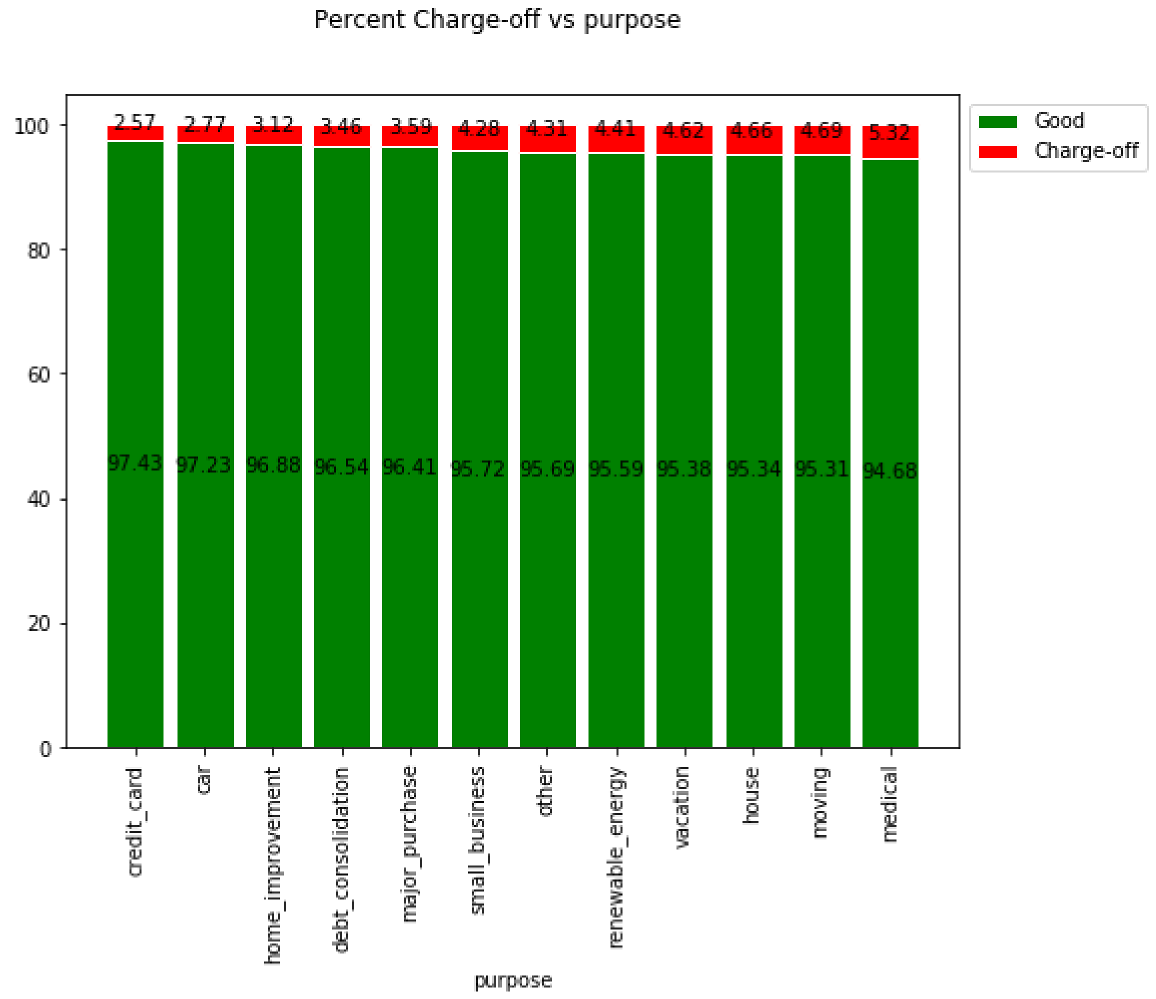
verification_status

Indicates if income was verified by LC, not verified, or if the income source was verified. It can take on values 'Source Verified', 'Not Verified', 'Verified'. Below pie plot shows surpisingly that Verified status is charged off at a higher rate than 'Source Verified', 'Not Verified'. The values need to be one-hot encoded.



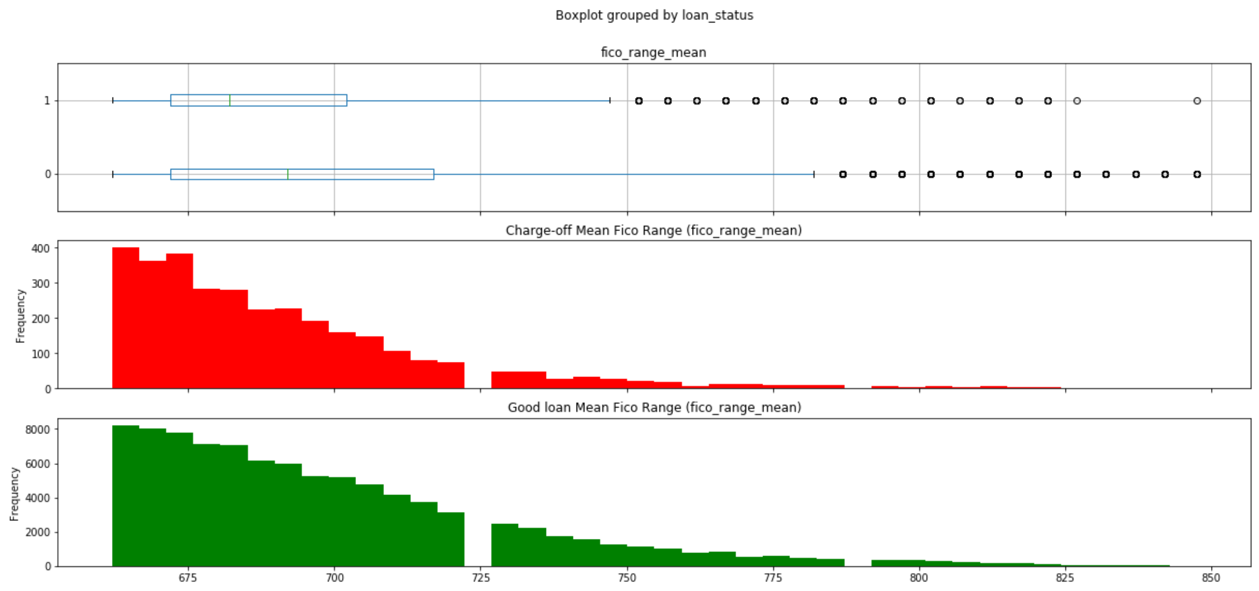
purpose

Correlation can be seen between Purpose and charge-off rate, but since it is a categorical variable, it should be encoded. Low frequency groups could be grouped together.



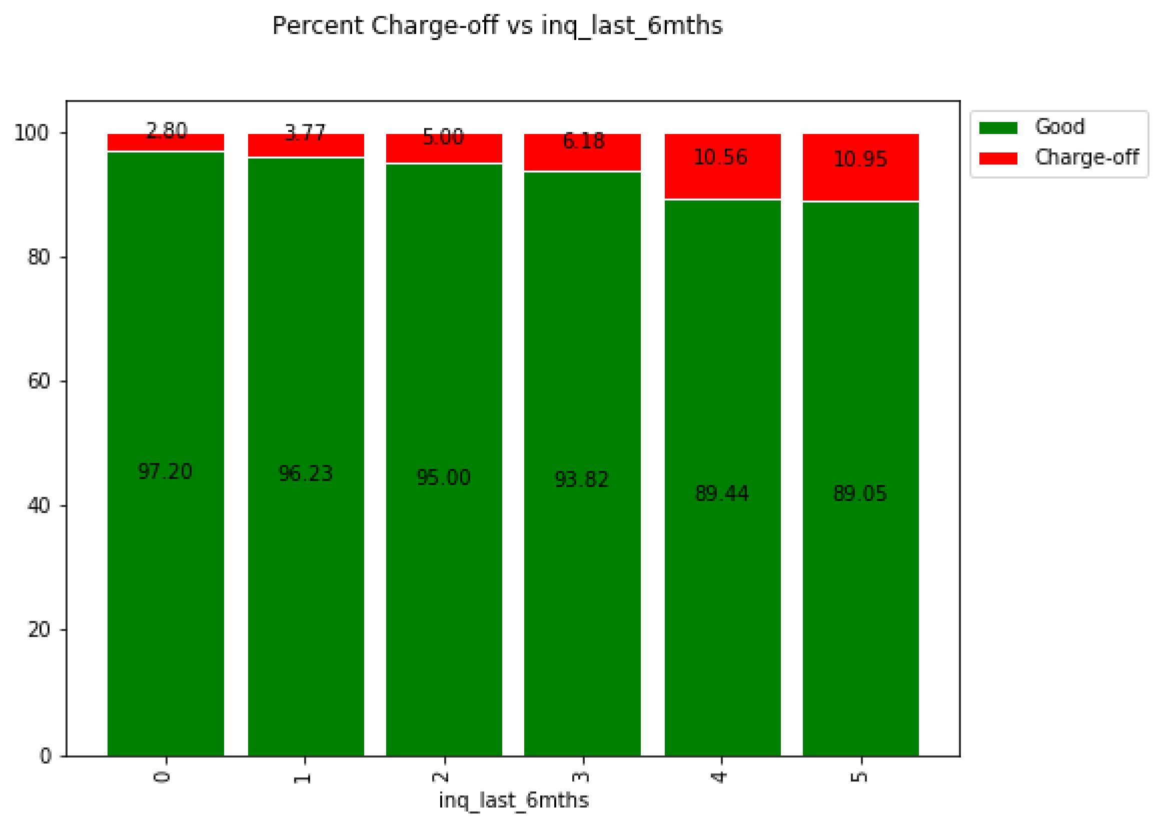
fico_range_mean

Since the high and los FICO values are very close, I took the mean of the two values. There is a cutoff at 660, only loans higher than that value are approved.



inq_last_6mths

This feature strongly correlates with the charge-off status. It indicates the number of times credit inquiries were run on the borrower's credit history.



Algorithms and Techniques

In this section, you will need to discuss the algorithms and techniques you intend to use for solving the problem. You should justify the use of each one based on the characteristics of the problem and the problem domain. Questions to ask yourself when writing this section:

- *Are the algorithms you will use, including any default variables/parameters in the project clearly defined?*
- *Are the techniques to be used thoroughly discussed and justified?*
- *Is it made clear how the input data or datasets will be handled by the algorithms and techniques chosen?*

Algorithms: Random Forest

The Random Forest algorithm is an ensemble method and in theory is well suited for the type of problem at hand. We have a large number of variables of different types: categorical, discreet numerical and continuous numerical. A Random Forest classifier works well with this kind of data with varying and ranges and types. They don't require special preprocessing apart from imputing null values and encoding categorical data.

Random Forest is an ensemble method composed of many Decision Trees and it gives an average of the results reported by the individual trees. It can give better results because it samples a subset of the parameters and a subset of the data with bagging which can help avoid overfitting. I have tried out different parameters values controlling Random Forest to find the best score. The parameters not listed here were left at their default values. Random Forest (http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html) DecisionTreeClassifier (http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTree

The following parameters were tuned during development phase:

- criterion - The function to measure the quality of a split. Supported criteria are “gini” (default) for the Gini impurity and “entropy” for the information gain. "Entropy" function performed better than "gini" when all other parameters were the same.
- max_depth - The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples. The tree was surprisingly shallow; the max depth at which the best results were measured was 3.
- max_features - The number of features to consider when looking for the best split: I have used a range of int values in a search pattern to find the best value of 50.
- class_weight - The “balanced” mode uses the values of y to automatically adjust weights inversely proportional to class frequencies in the input data as n_samples / (n_classes * np.bincount(y))

Techniques:

Outlier detection

Data points outside 1.5 IQR removed after visual inspection

Minority Class balancing

- class_weight - The “balanced” mode uses the values of y to automatically adjust weights inversely proportional to class frequencies in the input data as n_samples / (n_classes * np.bincount(y)) (http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html)

Distribution Analysis

Sqewness of data noted for many variables, but theoretically Random Forest does not need normally distributed data, so no action was takes to normalize distribution.

Imputation of Missing Data

Missing data points were iputed using 'mean' as strategy. sklearn.preprocessing.Imputer (http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.Imputer.html)

Grid Search

The proper way of choosing multiple hyperparameters of an estimator are of course grid search or similar methods that select the hyperparameter with the maximum score on a validation set or multiple validation sets. scikit-learn.org (http://scikit-learn.org/stable/modules/learning_curve.html)

Best parameters were searched for using sklearn.model_selection.GridSearchCV (http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html). 3-fold (Stratified)KFold cross-validation was applied.

Training Curve visualization

It is helpful to plot the influence of a single hyperparameter on the training score and the validation score to find out whether the estimator is overfitting or underfitting for some hyperparameter values. learning_curve (http://scikit-learn.org/stable/modules/learning_curve.html)

Cross Validation

After initial preprocessing as described in the section below, the data was split into training and testing in a 80:20 proportion. The grid search and learning curves used 3-fold cross validation and "recall" scoring method.

The best parameter value was selected and the classifier result reported on the held-out test set.

Benchmark

The benchmark model I intend to use is a naive predictor. It will randomly classify a loan as `charge-off` with the probability of occurence of charge-off loans in the whole data set.

Define the percentage of charged off loans calculated from the data set as P0. That is estimated about 3.335% A naive predictor is built to randomly label a data point as "charge-off" with probability PN = 0.5. Calculate the Recall score of the naive predictor. This will be the benchmark against which the solution will be beasured.

For a model that predicts all notes as charge-off, the metrics would be:

Accuracy	Precision	Recall	F2
0.0335	0.0335	1	0.148

For a model that predicts all notes as OK, the metrics would be:

Accuracy	Precision	Recall	F2
0.9665	0	0	0

For the Naive predictor, given about 3.35% charge-off rate, the metrics would be:

Accuracy	Precision	Recall	F2
0.5	0.0335	0.5	0.132

Clearly, a better model would have a better Recall than 50% and F2 score greater than 0.132

classification-accuracy-is-not-enough (https://machinelearningmastery.com/classification-accuracy-is-not-enough-more-performance-measures-you-can-use/)

III. Methodology

Data Preprocessing

The Random Forest algorithm does not require special preprocessing and that data is normaly distributed or that the values are in similar ranges. However it requires that the data set has no missing values and that they are numerical. Preprocessing consisted of the following steps:

1. Exclude columns with over 95% data missing
2. Convert strings to numerical values
3. Fill in missing data points via sklearn.preprocessing.Imputer using 'mean' strategy
4. One-hot encode categorical data
5. Drop outliers
6. Split data into training and testing sets using sklearn.model_selection.train_test_split

Some of the columns required special consideration:

1. ZIP code: I ended up using the first two digits only as numerical data.
2. employment title: I ended up encoding it as present=0, missing=1, due to the large number of values.
3. dropped all hardship-related columns since they contained very little data
4. dropped columns with very low data frequency or redundant information: pymnt_plan, funded_amt, sub_grade
5. Average FICO high and low values
6. Converted date type into time intervals, to number of months: issue_d - earliest_cr_line

Surprisingly there were many abrormalities I have found in the data. I have expected financial dta to be well-reviewed and clean. Most notably income data had what it appeared to be erroneus. Income levels over 2 million dollars seemed unreasonable, one instance being 61 million reported for a person with occupations "waitress". I had removed outliers after visual inspection that lie 1.5 IQR beyond 25th and 75th percentile range. This step has improved prediction from .70 to .79 in the traing step. A large number of variables dealing with hardship-related exceptions were 99.9% empty and I excluded them from analysis. The distributions of numerical features holding data were often squewed. This is not a concern with decision trees as they don't required normal distributed data. Percent values in the data set were represented as string data types, these had to be converted to floats. The following variables were one-hot encoded: 'verification_status', 'home_ownership', 'verification_status_joint', 'purpose', 'emp_length', 'term' and 'initial_list_status'.

1. The data set was read in as DataFrame and the column types, ranges and missing values inspected. Then I have developed specialized data converters to encode and transform the data as it is being read in.
2. The following columns need converting:

    - loan_status: The target varianble is 1 if 'Charged Off' or 'Default' else 0
    - term: "36 month" is encoded as 0, "60 months" encoded as 1
    - int_rate: the % sign stripped from the string and converted to float
    - revol_util: the % sign stripped from the string and converted to float
    - grade: mapped as follows: {'A': 7, 'B': 6, 'C': 5, 'D': 4, 'E': 3, 'F':2, 'G':
    - emp_title: 1 if missing, 0 if a title is given
    - application_type: 'Individual': 1, 'Joint App': 0

   - initial_list_status: 0 if x == 'w' else 1
3. The following features were one-hot encoded:

   - verification_status (three distinct values)
   - verification_status_joint (three distinct values)
   - home_ownership
   - purpose
   - emp_length (while it is numerical, from visual inspection there did not appear to be a correlation to loan_status)
4. After the pre-processing, there remained a large number of numerical variables with some missing values. I have listed them in a table and used sklear.preprocessing.Imputer to fill in missing values with the impute strategy='mean'

The following notebook was a helpful guidance in preprocessing: https://github.com/aprilypchen/depy2016 (https://github.com/aprilypchen/depy2016)

Implementation

In this section, the process for which metrics, algorithms, and techniques that you implemented for the given data will need to be clearly documented. It should be abundantly clear how the implementation was carried out, and discussion should be made regarding any complications that occurred during this process. Questions to ask yourself when writing this section:

- *Is it made clear how the algorithms and techniques were implemented with the given datasets or input data?*
- *Were there any complications with the original metrics or techniques that required changing prior to acquiring a solution?*
- *Was there any part of the coding process (e.g., writing complicated functions) that should be documented?*

The implementation process took its course in three stages: 1. Exploratory data analysis, 2. Algorithm selection, 3. Performance tuning. I have used Jupyter notebooks, scikit-learn and and Python in each stage.

To to perform the various task required in the project - such as data ingestion, data cleaning, model training, evaluation and visualization - some custom scripts had to be developed around the standard scikit-learn libraries. These scripts, in the form of Python functions are collected in preprocess_visuals.py (preprocess_visuals.py).

I have used standard Python and sklearn libraries wherever I could.

- For data ingestion I have developed converters that can be passed in read_csv() (https://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_csv.html) as a dict of functions for converting values in certain columns. I found it very helpful.
- The data cleaning steps required to fill null valuess, remove some outliers, drop columns and impute mean were implemented in the `preprocess()` function. Other than being tedious I have encountered another difficulty that has to do with the way Pandas DataFrame structure works. It is handles data in a smart way avoiding making many data copy operation. One can get surprise if he tries to assign values to a reference instead of an index. When setting values in a pandas object, care must be taken to avoid what is called chained indexing (https://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy).

Otherwise assignment can fail when using chained indexing. Data imputation was straight forward, I just had to call sklearn.preprocessing.Imputer (http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.Imputer.html). To help identify sparse features, I developed col_stats(df) to display null percent, min, max median, squew and unique count statistics on all features. This function is leveraging numpy statistical methods and unique_counts. I needed research the right functions to use.

- Coding the traing process required the most analytical thinking. If I used a grid search with a very large number of parameter values to test, the search computation would take many hours. Using a few values and "filling in" to reduce computation time proved useful, still it took over half hour to run the grid search on a MacBook Pro with 8 GB memory and 2.9 GHz i7 dual core processor, on a dozen parameters. Using many more parameters could have taken days.
- Developing visualization required some creativity, but matplotlib provides many excellent graph types that can be customized to need. Bar plots, pie charts and histograms were adequate elucidate important characteristics of the data.
- In the end the code proved to be too long to include in one Notebook and I had to move it to an external file. This is a little tricky because the kernel has to be restarted if changes are made to the external file, which takes some time.

Notebooks

- Feature Visualization.ipynb (Feature%20Visualization.ipynb) contains visualization methods for features. Figures presented in this reports are developed in this Notebook.

- Project Development.ipynb (Project%20Development.ipynb) contains processing steps related to discovery and images.
- preprocess_visuals.py (preprocess_visuals.py) The helper functions and converters are in the file.

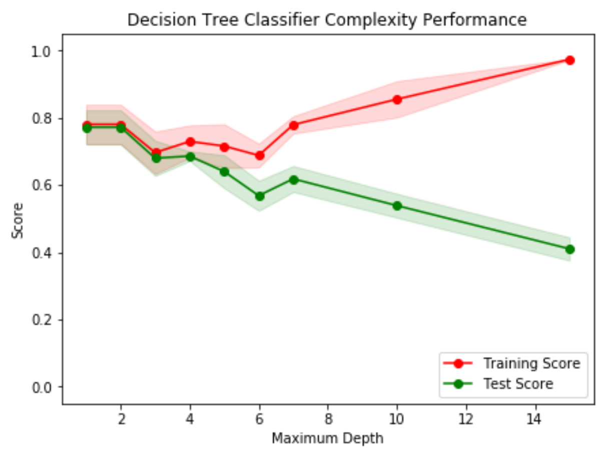
Exploratory data analysis

1. I loaded the traing and testing data into memory, it fit easily into 100MB of physical memory.
2. I have developed converters for pandas Excel reader to transform strings to floats for percent fields.
3. plot_box_hist2() Show a box plot and a histogram of the column values. This function displays side-by-side charge-off and good loan distribution and box plots allow outlier detections and evaluation of the feature variance visually.
4. remove_outliers(df) After detecting outliers this function removes outliers from the data set.
5. I have developed col_stats(df) function that allows looking at all of the 100+ columns and display the percentage of missing values, mean, min, max and squewdness.
6. I have developed percent_col() function to Create a stacked percentage bar chart, indicating good to charge-off loan ratio.
7. I have developed analyze_col() function to display a table, a pie chart showing relative percentages, and piecharts as good-to-chargeoff ratios.
8. The helper functions and converters are in the file preprocess_visuals.py (preprocess_visuals.py)
9. Extensive feature exploration, during which each feature's characters were inspected, required a long time. I had to clean the data, discover which fields were categorical, numerical, decide of imputation strategy.

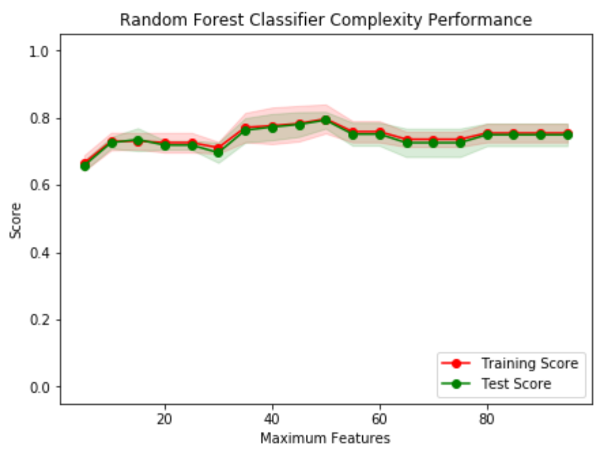
Algorithm selection

My initial choice was Decision Tree but after inital inspection of the validation_curve from sklearn.model_selection curve I realized that it was overfitting. I have switched to Random Forest to reduce overfitting effect. Decision Tree method yielded 0.772 recall on 3-fold cross validation set, but I suspected that it could do better.

Test score for param= 1 recall= 0.772 std= 0.051



Test score for param= 50 max_depth= 1 recall= 0.793 std= 0.025



Performance tuning

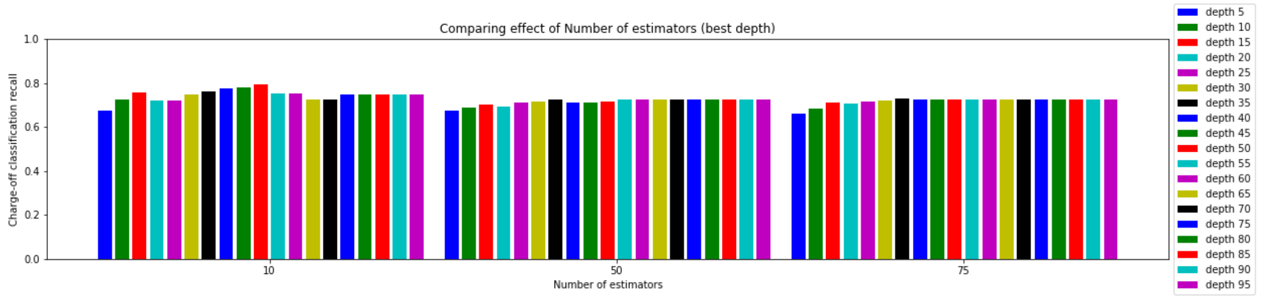
I have a constructed Grid Search to find the optimal parameters for the Random Forest. I varied the model depth as parameter and the both 'gini' and 'entropy' as criterion. Then I implemented the search using RandomForest and trying various paramters to improve accuracty.

Parameters:

- criterion - The function to measure the quality of a split. Supported criteria are “gini” (default) for the Gini impurity and “entropy” for the information gain. "Entropy" function performed better than "gini" when all other parameters were the same.
- max_depth - The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples. The tree was surprisingly shallow; the max depth at which the best results were measured was 3.
- max_features - The number of features to consider when looking for the best split: I have used a range of int values in a search pattern to find the best value of 50.
- class_weight - The “balanced” mode uses the values of y to automatically adjust weights inversely proportional to class frequencies in the input data as n_samples / (n_classes * np.bincount(y))

I have used validation_curve from sklearn.model_selection package to find the best parameters for the classifier. After only 3 levels the model quickly began to overfit which is apparent from the training curve generated by validation_curve.

Figure: Random Forest Grid Search visualization comparing different number of estimators and depths.



The grid search algorithm can be found in [Process%20Input%204.ipynb (Process%20Input%204.ipynb)](Process%20Input%204.ipynb) notebook.

Refinement

Initial solution

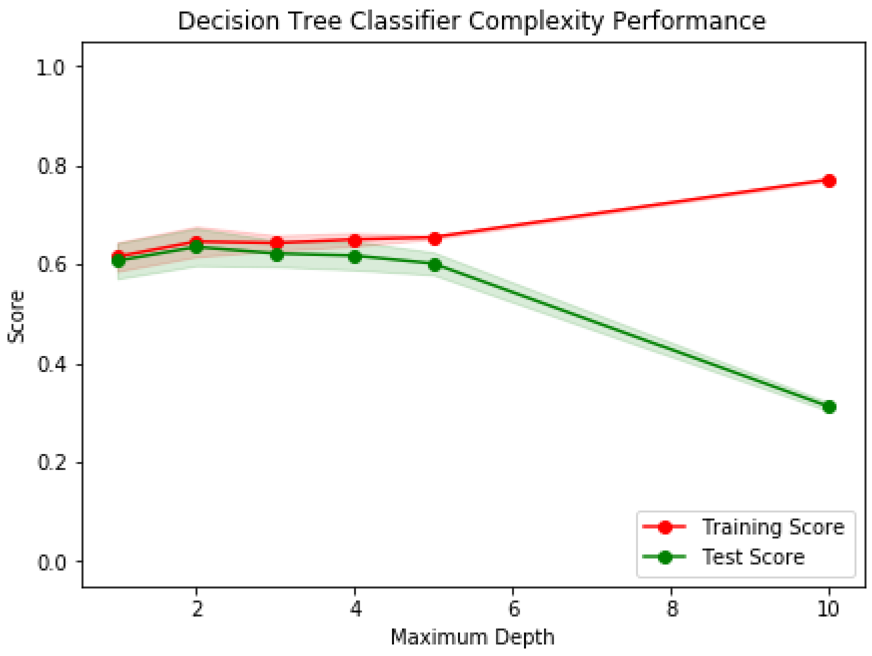
The inital stab at the problem was using DecisionTreealgorithm. The first attempt with max_depth set to 'auto' resulted in recall 0.06 with std=0.003.

Refinement 1: max_depth='auto' to max_depth=2

The initial solution had a poor result, and I decided to try to reduce the max_depth. A training curve has revealed that after about max_depth=3 the traing and testing curves were rapidly diverging, which indicated overfitting. Then I reduced the maxdepth and tried different values:

Refinement 2: criterion:'interactions'

Test score, criterion= entropy for max_depth= 2 recall= 0.635 std= 0.038



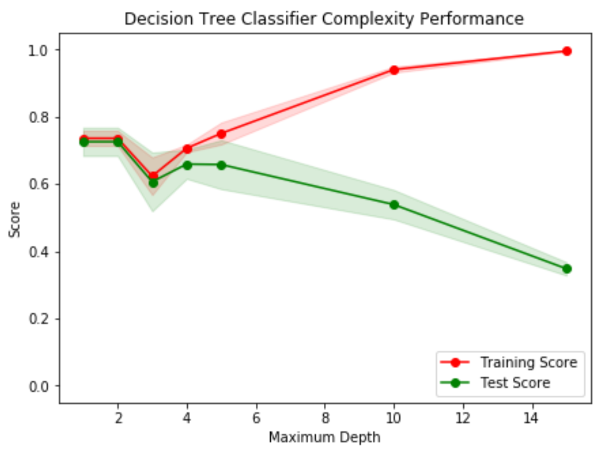
Refinement 3. Outlier deletion

I was not very impressed by the improvements feature interactions yielded so I tried outlier deletion. Deleting outlier for all the 100+ features for outliers removed a large portion of the training data set but still there is plenty of data left, 54,000 data points for just one quarter, so that ws not a concern. Outlier deleletion gave a boost to the recall score of the predictor, it bas improved its performance from 0.635 to 0.726.

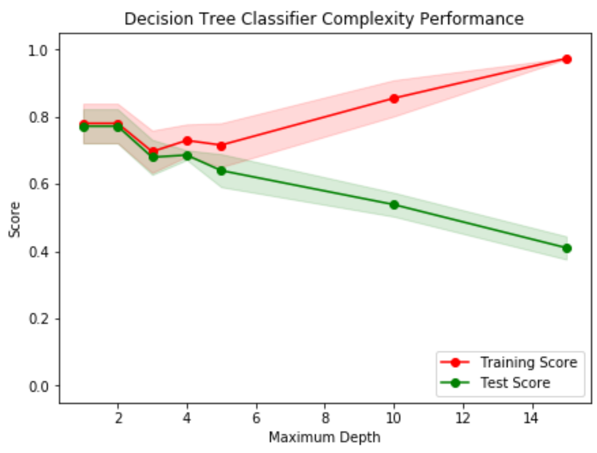
Refinement 3: criterion:'entropy'

I have tried the default 'gini' versus 'entropy' as criterion parameter in the fitting, which detrmines the splitting criteria of a feature. Specifying Entropy gave better results so I decided to use it going forward; the recall was now at 0.772.

Test score for GINI recall= 0.726 std= 0.042



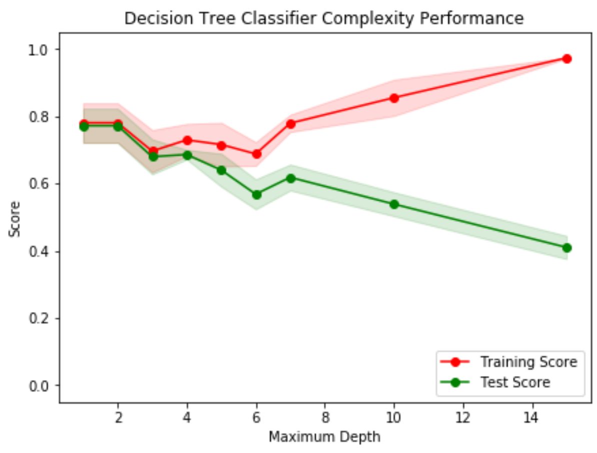
Test score for ENTROPY recall= 0.772 std= 0.051



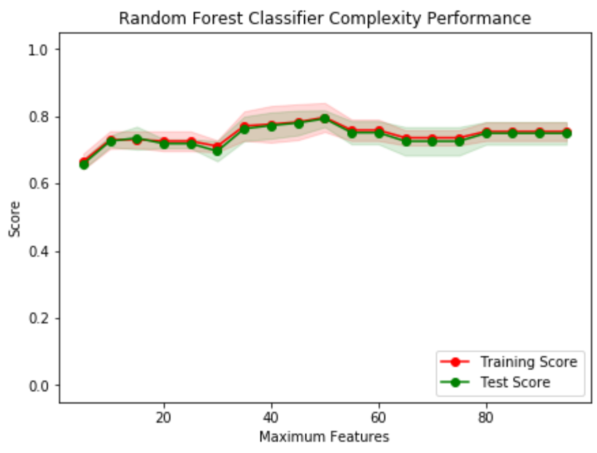
Refinement 5. Random Forest

Then I have swicthed to Random Forest to remove overfitting by applying bagging, estimator number and feature selection tuning. The traing and testing curves are a lot more consistent indicating that the model is not overfitting. Random Forest with entropy criterion, outliers removed is the classifier configuration that gave the highest recall value without overfitting.

Decision Tree, entropy, max depth= 1 recall= 0.772 std= 0.051



Random Forest, entropy, max features = 50 recall= 0.793 std= 0.025



.

IV. Results

Model Evaluation and Validation

The final model was derived by careful model building technique: splitting data into train and test set, grid searching best parameters and removing outliers.

Robustness

To evaluate the robusteness of the final model, I run it with various random states in the train-test split and analyze the mean / variance of the results. For calculations please see the Notebook Robustness Study.ipynb (Robustness%20Study.ipynb). Robustness (https://www.researchgate.net/post/What_is_the_definition_of_the_robustness_of_a_machine_learning

T-test:

- $H0 : \mu = 0.50$ The null hypothesis is that the classifier recall is the same as the naive predictor's recall.
- $H1 : \mu > 0.50$ The alternative hypothesis is that the trained classifier's recall is greater that the naive predictor's.
- The sample mean $\bar{x}$: 0.715
- The population μ: 0.50
- The sample standard deviation s = 0.091
- Number of observations n = 30

Results from T-statistic

- t-statistic = 9.962 pvalue = 7.1916e-11
- KS-statistic D = 0.676 pvalue = 0.0000

The p-value indicates the probablity that the classifier recall is the same as the naive classifier's recall. Since the sample mean is 7.1916e-11, and it is less than alpha=0.05, we can reject the null hypothesis, and accept the alternative that classifier recall mean is higher.

The Kolmogovov-Smirnoff test pvalue < 0.05 indicates that the recall means form a t-distribution with 30 degrees of freedom. t-test-and-ks-test (https://docs.scipy.org/doc/scipy/reference/tutorial/stats.html#t-test-and-ks-test) statisticshowto (http://www.statisticshowto.com/one-sample-t-test/), t-test-using-numpy (https://towardsdatascience.com/inferential-statistics-series-t-test-using-numpy-2718f8f9bf2f)

Trained Model on data set 2017 Q1:

--	Accuracy	Precision	Recall	F2
Mean	0.570	0.072	0.716	0.251
Stdev	0.109	0.010	0.117	0.011

Naive predictor theoretical values:

Accuracy	Precision	Recall	F2
0.5	0.0335	0.5	0.132

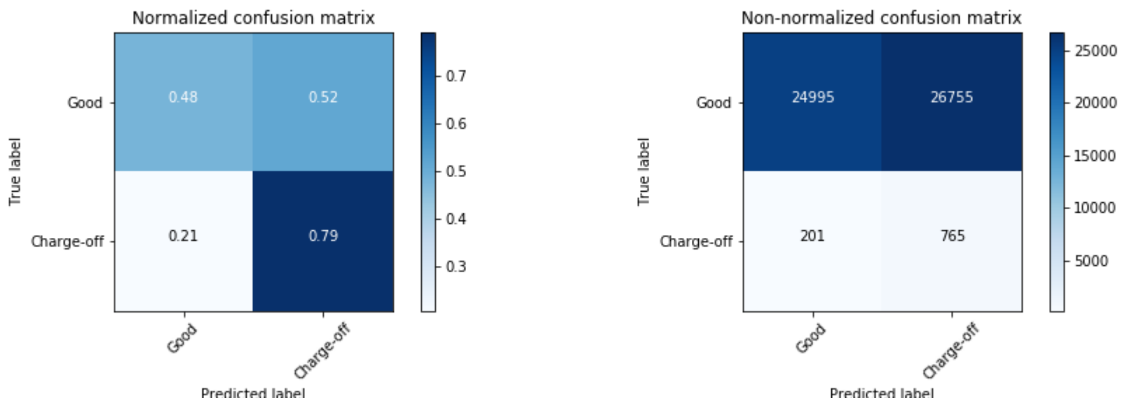
Robustness with respect to data

The model seems robust enough when the data set is changed. In addition to varying random splits, the robustness of the model was also tested on loan data from a different quarter, 2017 Q2 and 2016 Q4. The recall recorded was 0.792 and 0.710, which is in good agreement with the data from Q1 2017.

Reasonableness

The final model is reasonable given that it appears to be hard to tell upfront which loan would be default. My expectation was that it would outperform the naive predictor and it does a better job then the naive predictor. The final parameters indicate that a decision is made based on a low number of fetures.

Confusion Matrix of Prediction Results, Normalized and Non-Normalized



Reliability and Trustworthyness

I feel that the model is reliable to classify bad loans with about 70% recall rate on the long run.

Justification

The model is working better than the benchmark model with respect to predicting charge-off. This predictive power comes at the expense of discarding large number of good loans as can be seen from the confusion matrix. If there is a large number of loans to select from it is not a problem, but if the there are not enough applications then we are not left with many to choose from.

I got as final solution a model that predicts charge-off with 74% recall. It also predict 44% of true good label loans as charge-off, which reduces the investment opprtunity by 44%. With a large pool of available loan applications this may be acceptable.

The final solution can be used if it is not a problem to discard many potentially good loan applications if our only only goal is to filter out bad loan applications. The improvement is significant with respect to the recall, 74% of charge-off loans can be caught by this method.

V. Conclusion

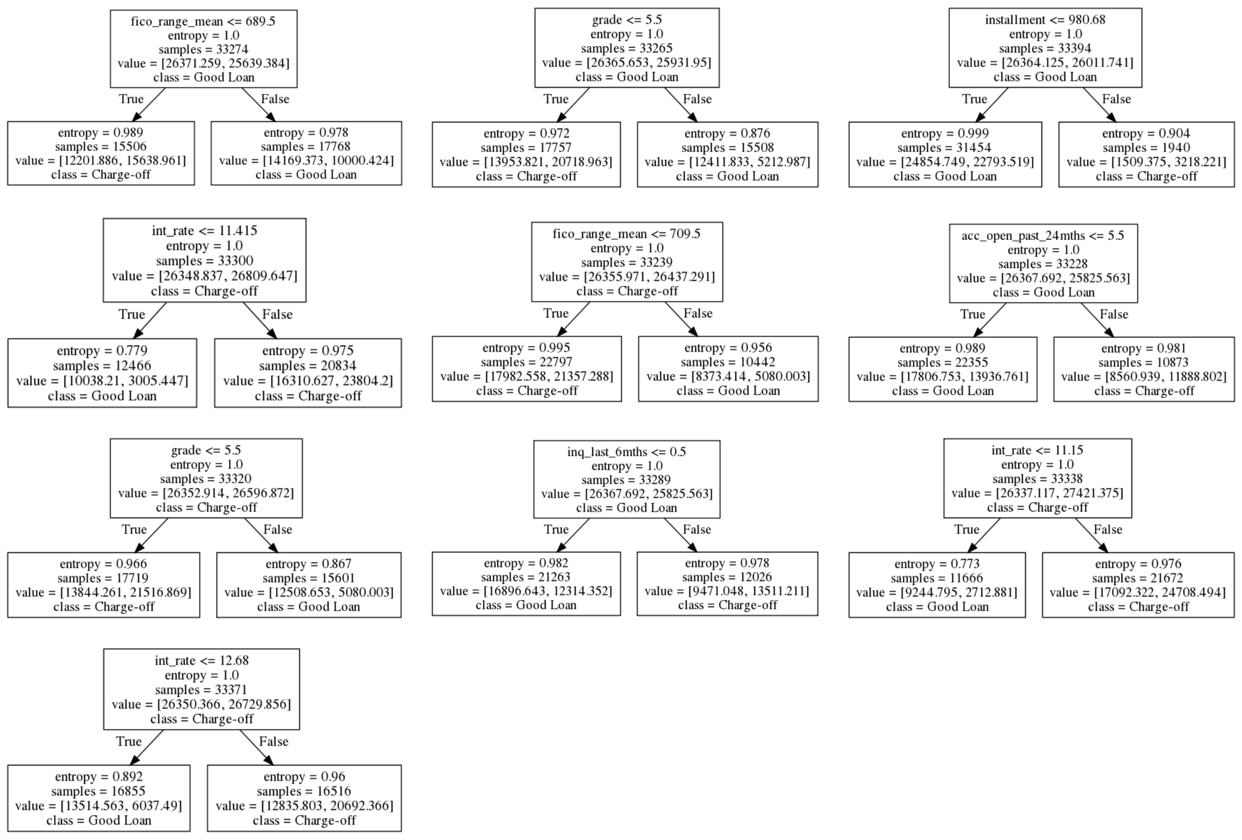
Free-Form Visualization

Visualization of Random Forest.

The images below visualize the trees in the trained Random Forest, on the 2017 Q2 data set. The individual trees in the table below are the trees that make up the Random Forest trained by GridSearch. They are all one level deep, indicating the split value for each feature.

The trained trees seem to confirm some of the findings uncovered during Explotratory Visalization. There are relatively few features that showed clear correlation to loan_status. We could see these correlations by a large difference in their means when grouped by loan_status for the following features: int_rate, inq_last_6mnths, fico_range_mean acc_open_past_24mnths and grade.

```
RandomForestClassifier(bootstrap=True, class_weight='balanced', criterion='entropy', max_depth=1,
    max_features=50, 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=10, n_jobs=1,
    oob_score=False, random_state=7, verbose=0, warm_start=False)
```


fico_range_mean <= 689.5 entropy = 1.0 samples = 33274 value = [26371.259, 25639.384] class = Good Loan
True
False
entropy = 0.989 samples = 15506 value = [12201.886, 15638.961] class = Charge-off
entropy = 0.978 samples = 17768 value = [14169.373, 10000.424] class = Good Loan
grade <= 5.5 entropy = 1.0 samples = 33265 value = [26365.653, 25931.95] class = Good Loan
True
False
entropy = 0.972 samples = 17757 value = [13953.821, 20718.963] class = Charge-off
entropy = 0.876 samples = 15508 value = [12411.833, 5212.987] class = Good Loan
installment <= 980.68 entropy = 1.0 samples = 33394 value = [26364.125, 26011.741] class = Good Loan
True
False
entropy = 0.999 samples = 31454 value = [24854.749, 22793.519] class = Good Loan
entropy = 0.904 samples = 1940 value = [1509.375, 3218.221] class = Charge-off
int_rate <= 11.415 entropy = 1.0 samples = 33300 value = [26348.837, 26809.647] class = Charge-off
True
False
entropy = 0.779 samples = 12466 value = [10038.21, 3005.447] class = Good Loan
entropy = 0.975 samples = 20834 value = [16310.627, 23804.2] class = Charge-off
fico_range_mean <= 709.5 entropy = 1.0 samples = 33239 value = [26355.971, 26437.291] class = Charge-off
True
False
entropy = 0.995 samples = 22797 value = [17982.558, 21357.288] class = Charge-off
entropy = 0.956 samples = 10442 value = [8373.414, 5080.003] class = Good Loan
acc_open_past_24mths <= 5.5 entropy = 1.0 samples = 33228 value = [26367.692, 25825.563] class = Good Loan
True
False
entropy = 0.989 samples = 22355 value = [17806.753, 13936.761] class = Good Loan
entropy = 0.981 samples = 10873 value = [8560.939, 11888.802] class = Charge-off
grade <= 5.5 entropy = 1.0 samples = 33320 value = [26352.914, 26596.872] class = Charge-off
True
False
entropy = 0.966 samples = 17719 value = [13844.261, 21516.869] class = Charge-off
entropy = 0.867 samples = 15601 value = [12508.653, 5080.003] class = Good Loan
inq_last_6mths <= 0.5 entropy = 1.0 samples = 33289 value = [26367.692, 25825.563] class = Good Loan
True
False
entropy = 0.982 samples = 21263 value = [16896.643, 12314.352] class = Good Loan
entropy = 0.978 samples = 12026 value = [9471.048, 13511.211] class = Charge-off
int_rate <= 11.15 entropy = 1.0 samples = 33338 value = [26337.117, 27421.375] class = Charge-off
True
False
entropy = 0.773 samples = 11666 value = [9244.795, 2712.881] class = Good Loan
entropy = 0.976 samples = 21672 value = [17092.322, 24708.494] class = Charge-off
int_rate <= 12.68 entropy = 1.0 samples = 33371 value = [26350.366, 26729.856] class = Charge-off
True
False
entropy = 0.892 samples = 16855 value = [13514.563, 6037.49] class = Good Loan
entropy = 0.96 samples = 16516 value = [12835.803, 20692.366] class = Charge-off

Reflection

The process for this machine learning project can be summarized as follows:

1. I have found a problem that had interested me before, in the finace sector.
2. Formulating the problem was the first step. I decided to use a classifier that predicted if a loan application was likely to default.
3. Then I had to find the data set that would help me train a classifier on. I was fortunate that LendingClub published loan data and target label could easily be defined.
4. The next step was to set up a Notebook, read in the data and understand the features. After data clean-up I had to decided to use a Decision Tree algorithm because of the shape and composition of the data.
5. The next stage after initial tries was in optimization and parameter tuning.

The most interesting aspect of the project was learning about the features that make up the data set. Trying to find correlation between them and the target variable proved to be tricky. Only a small fraction, about 5% of features seem to be relevant to prediction, I expected more features playing a role.

The most difficult aspects of the project were data cleanup and optimization. The large number of features cased most of the difficulty in data cleanup and preprocessing. I was not sure how much attention each feature required and at the outset it was imposible to tell which features had more promise than others, and I have spent a lot of time looking at each one. I have tried feature interaction, feature reduction via selecting K best feaures and outlier removal. The second most difficult aspect was fine tuning the learning algorithm. There is a large number of algorithms each having many parameters to tune. I have set up Grid Search and Learning Curves to run several options as a job and to select the best parameters. The final model has a decent recall rate, 0.793, significantly better than the benchmark classifier that had 0.5. It could be used in a general setting, such as an application, to help investors choose good loans to invest it and to reduce loss due to write-offs.

Improvement

Further ideas for improvements include more extensive experiments with neural net such as Multi Layer Perceptron classifier. I have not had good results with a simple 2-layer perceptron. The percepton had two hidden layers of length 100. The training took very long and the accuracy was very low. Perhaps bottleneck layers would improve the results. An outlier removal function could be developed that would automatically identify data points to remove and suggest removal of those data points to improve performance.

I was considering Natural Language Processing techniques to encode job_description. This feature had very large number of free-text titles that may bear correlation to the charge-off rate. I was worried that the number of features generated would be too large and not have enough processing power. Also I was not sure how to apply word embedding to the problem.

I feel that further feature engineering may improve the final result. Adding interactions (polynomial, sum, difference) may yield improvement. I tried the following but it yielded no significant improvement:

1. Before adding interactions I scaled the features so they have similar values.
2. Since I suspected that there may be interdependce between features, I added 2nd degree polynomial interactions (sklearn.preprocessing.PolynomialFeatures) between the features after the initial classification results were not too great.
3. The polynomial fetures have increased the number of features to over 6000. I then reduced the number of features back to 100 by applying sklearn.feature_selection.SelectKBest.
4. The feature interactions have not improved the prediction values. After some test runs I have decided against using polynomial features.