Predicting Defaults of Personal Loans

The Problem

LendingClub is a platform that connects investors with borrowers who are seeking a personal loan. Investors can view details on potential loan investments based on information supplied about the borrower, the amount of loan requested, the loan grade, and the loan purpose.

As one would expect, potential loans that are viewed as riskier based on the borrower's or the loan's profile will have a higher interest rate to hedge the risk and increase the likelihood of investors funding the loan.

What if not all risky loan features or borrowers are created equally? Perhaps there is a way to minimize our loan risk by identifying which specific features are most predictive of loan default. This will allow an investor to maximize their return on investment.

The Data

Source: https://www.kaggle.com/wordsforthewise/lending-club

The source of my data is from LendingClub via Kaggle. LendingClub lets members of their site download all loan details from 2007 onward--excluding users' personal information to protect their privacy. LendingClub is a popular dataset in the Data Science community because it is difficult to get access to this sort of data unless you were an employee of such a financial institution.

Though we have access to over a decade worth of data, we will be limiting our analysis to the years 2015 through 2017. These latter years were chosen to minimize the effect that the global recession of 2008 would have on our data. In addition, the scope of three years was chosen to prevent the Federal Funds Rate increases from skewing our interest rates too much. It will also prevent my analysis from taking too long as I do not have a lot of computational firepower at my disposal.

Data Wrangling:

The first priority is to scale down my data from 11 years down to 3 years. I read the data into a dataframe, making sure to parse the dates on the issue date column. From there, I create a new dataframe that consists of every row where the issue date was greater than 12-31-14 and less than 01-01-18. This should give every loan that was issued from the beginning of 2015 to the end of 2017.

After the cut, we now have a DataFrame that is more wieldy at just a meager ~1.3 million rows and 150 columns. Let's hope some of these columns are superfluous or my laptop will melt.

We can't very well have a Data Science project without a Target Feature. Let's create that first. There is a column called 'loan_status' which can have the following values as options:

```
Fully Paid            654747
Current               439211
Charged Off           183193
Late (31-120 days)     13909
In Grace Period         5316
Late (16-30 days)       2673
Default                   32
Name: loan_status, dtype: int64
```

Charged Off refers to a status that lenders move the loan into after they no longer expect to receive any more payments. Once they declare the loan "charged off" it becomes a write-off for the lender. For LendingClub, this occurs 120 days after the last payment. For our purposes, this is our primary category for determining that a loan has defaulted--along with Default obviously. While some of these other categories are delinquent and are in danger of becoming Charged Off, until it actually happens, we will categorize them as non-defaulted loans. We create a new column called 'default' that will either be True or False based on these criteria.

Just from viewing the top of the dataframe, I have a lot of missing values. The vast majority of these missing values are because that column doesn't apply to that particular loan. Specifically, there are a lot of features that only apply to delinquent loans. I will be removing all of these features as using machine learning to predict a loan defaulting with columns that only have values when the loan has defaulted already is not very useful or impressive. Although, it is tempting to have a near 100% success rate with our model, science demands that we challenge our model and ourselves.

There are also many features that are almost all empty because they only apply to joint applications. These joint applications only make up 5% of our total dataset. We're just going to toss them out because they add unneeded complexity.

The drops went like so:

```
drops = ['member_id','funded_amnt','funded_amnt_inv', 'emp_title', 'loan_status','pymnt_plan','url','desc','title',
'initial_list_status','out_prncp','out_prncp_inv','total_pymnt','total_pymnt_inv','total_pymnt_inv','total_rec_prncp',
'total_rec_int','total_rec_late_fee','recoveries','collection_recovery_fee','last_pymnt_d','last_pymnt_amnt','next_pymnt_d',
'policy_code', 'application_type','annual_inc_joint','dti_joint','verification_status_joint','delinq_amnt','revol_bal_joint',
'sec_app_fico_range_low','sec_app_fico_range_high','sec_app_earliest_cr_line','sec_app_inq_last_6mths','sec_app_mort_acc',
'sec_app_open_acc','sec_app_revol_util','sec_app_open_act_il','sec_app_num_rev_accts','sec_app_chargeoff_within_12_mths',
'sec_app_collections_12_mths_ex_med','sec_app_mths_since_last_major_derog','hardship_flag','hardship_type','hardship_reason',
'hardship_status','deferral_term','hardship_amount','hardship_start_date','hardship_end_date','payment_plan_start_date',
'hardship_length','hardship_dpd','hardship_loan_status','orig_projected_additional_accrued_interest',
'hardship_payoff_balance_amount','hardship_last_payment_amount','disbursement_method','debt_settlement_flag',
'debt_settlement_flag_date','settlement_status','settlement_date','settlement_amount','settlement_percentage',
'settlement_term']
```

After removing these columns, along with other columns that are empty, repeat information that is already present in other columns, or are otherwise not useful, I have removed 65 total columns from my dataframe. We're now down to 1.25 million rows by 87 columns. Both my laptop and wallet breathe a sigh of relief.

I check the datatypes for each column and I mostly see strings where I expect them to be and floats/integers as well. Don't worry though, dear reader, we will definitely find things to clean.

The employment length and loan term columns are numerical strings so I will go ahead and convert them both to integers so they can be used in our algorithm. We're slightly distorting our data as one of the values for employment length is 10 years or more but we will be converting it to just 10. Machine Learning Gods forgive me.

Time to check out our remaining missing values:

Out[17]:

zip_code	1
inq_last_6mths	1
mths_since_last_delinq	604416
mths_since_last_record	1022438
revol_util	774
last_credit_pull_d	26
mths_since_last_major_derog	892313
open_acc_6m	399437
open_act_il	399436
open_il_12m	399436
open_il_24m	399436
mths_since_rcnt_il	423244
total_bal_il	399436
il_util	513140
open_rv_12m	399436
open_rv_24m	399436
max_bal_bc	399436
all_util	399503
inq_fi	399436
total_cu_tl	399437
inq_last_12m	399437
avg_cur_bal	15
bc_open_to_buy	13115
bc_util	13757
mo_sin_old_il_acct	35500
mths_since_recent_bc	12438
mths_since_recent_bc_dlq	935778
mths_since_recent_inq	135674
mths_since_recent_revol_delinq	805005
num_rev_accts	1
num_tl_120dpd_2m	62860
percent_bc_gt_75	13436

For those string columns that only have a handful of missing values, I'll just get rid of those rows. Good riddance. For the numerical columns that have < 5% of the total values missing, I'll just fill those missing values with the median of that column.

However, there is a subset of these numerical columns that describe months that have passed since some sort of negative credit event occurred such as a bankruptcy or an inquiry. A lot of these values are missing. We're making a leap of faith here in assuming that these values are missing because they have either not occurred in a very long time--such as opening a new account or have never occurred--in the case of a bankruptcy. For these missing values, we will be filling them in with -999 so that we can still maintain these columns for our model but -999 should adequately represent numerically an event that has not occurred.

After completing this, we now have 13 columns with missing values 12 of which have the exact same amount of missing values (roughly 32%) and 1 column which has around 41% missing values. We will set these columns to the side for now and do some Exploratory Data Analysis.

Exploratory Data Analysis

Our Target Variable:

```
df['default'].value_counts(normalize=True)
```

```
False   0.85710
True    0.14290
Name: default, dtype: float64
```

14.29% is a really good default rate. Lendingclub investors have to be pleased with that. For us though, this means we have a very imbalanced dataset. Something to keep an eye on as many algorithms won't be able to deal with such imbalance.

I'll explore several of these features, starting with the ones that are heavily correlated and then moving on to others I found interesting:

Correlation:

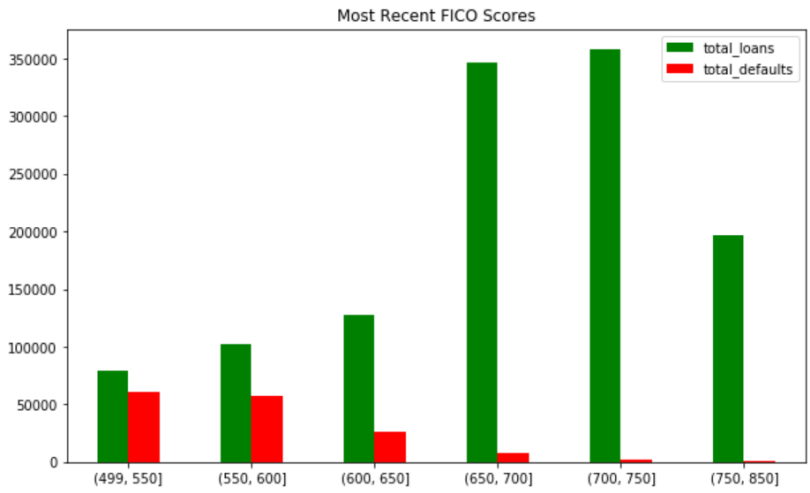
```
df['default'] = df['default']*1
corr = abs(df.corrwith(df['default'])).sort_values(ascending=False).round(2).head(20)
corr
```

```
default                 1.00000
last_fico_range_high    0.64000
last_fico_range_low     0.59000
int_rate                0.22000
fico_range_low          0.12000
fico_range_high         0.12000
acc_open_past_24mths    0.11000
num_tl_op_past_12m      0.10000
open_rv_24m             0.09000
dti                     0.09000
term                    0.09000
```

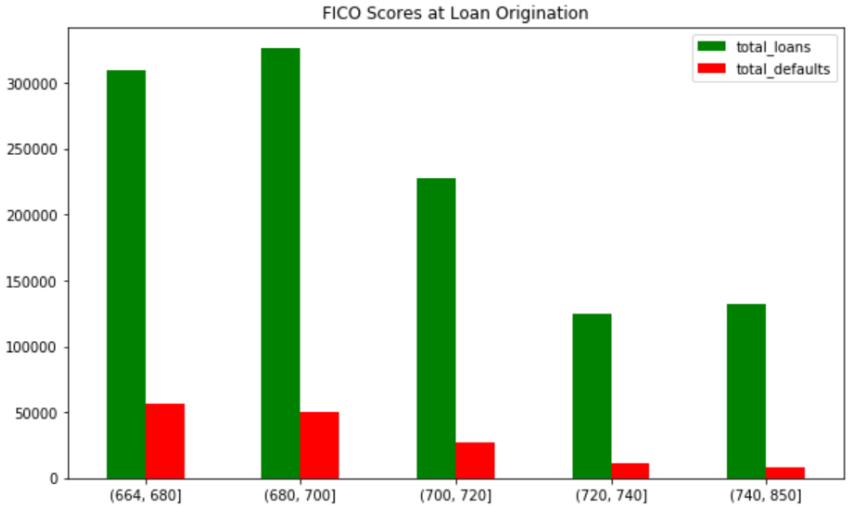
Let's go over some of these correlated features.

FICO Scores:

'last_fico_range_high' and **'last_fico_range_low'** - correlated 64% & 59% respectively with default rate. These columns refer to the FICO scores on the most recent credit pulls done on the borrower. Recent delinquent credit such as missed or late payments on any account, defaulted loans, or newly opened credit obligations will all bring this number down. We would expect this feature to be a good predictor. Contrast this with **'fico_range_high'** and **'fico_range_low'** which were the credit scores back when the loan was originated. They are still good predictors when compared to other features but 5x less correlated than the 'last_ficos' columns. A lot can happen in a borrower's financial situation to make loan default more likely throughout the term of a loan such as loss of employment or unexpected financial obligations. A borrower who once seemed like a good investment can become a defaulting timebomb.



As you would expect, the defaults are all showing up in the lower tier credit scores. It is a very reliable predictor and I expect the algorithm we choose is going to love this feature the most by far. Now compare this to FICO scores when the loans were started:



We have a much thinner split of scores at loan origination. Notice that our credit score ranges are different too. The previous graph went as low as 499 but at origination our lowest score is 664. This is expected as giving a personal loan to someone with sub 600 credit score is not a good financial investment. You're better off just lighting that money on fire for entertainment purposes.

Interest Rate:

The interest rate on a loan is a delicate balancing act. If the rate is too high, the borrower will have too difficult of a time paying off the loan and it will inevitably lead to default. However, if the rate is too low, the investor has no incentive to take give out a potentially risky loan. Based on their default rate of 14%, LendingClub does a pretty good job of managing this tightrope act. Here's the breakdown:

	total_loans	%_of_loans	total_defaults	%_that_defaulted
(5, 10]	379289	30.714346	23035.0	6.073205
(10, 15]	516322	41.811106	70200.0	13.596167
(15, 20]	252691	20.462599	56283.0	22.273449
(20, 25]	57363	4.645184	17192.0	29.970539
(25, 31]	29227	2.366766	9762.0	33.400623

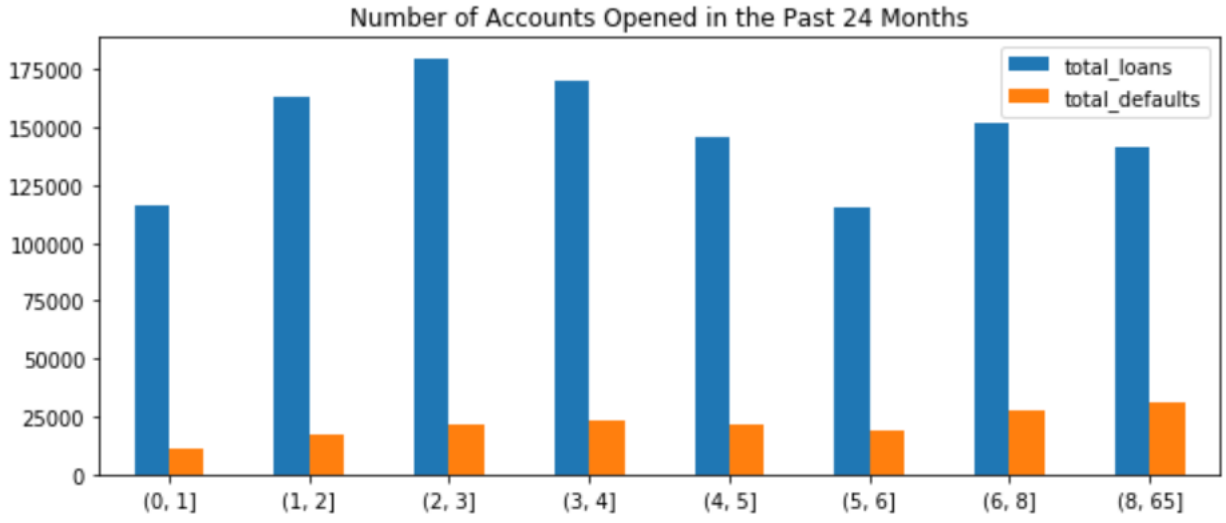
As you can see, there is a clear increase in default rate as we go up in interest rate. 1 in 3 loans in the 25-31% bracket ends in a default! Thankfully, this bracket only makes up a little more than 2% of all the loans that LendingClub gives out.

Accounts Opened:

'acc_open_past_24mths' refers to the number of credit accounts that the borr ower has opened in the past 24 months. **'num_tl_op_past_12m'** is a similar feature but it's only for the past 12 months. Ditto for **'open_rv_24m'** but it only refers to revolving a ccounts such as credit cards.

A lender doesn't want to see that you have many credit obligations with minimal history. If you must have many lines of credit, it is best that we have a long history of you making these payments on time so that we can be certain it can be managed.

They're all pretty similarly correlated and all accounts in the past 24 months is the strongest feature so I'll look at that one:

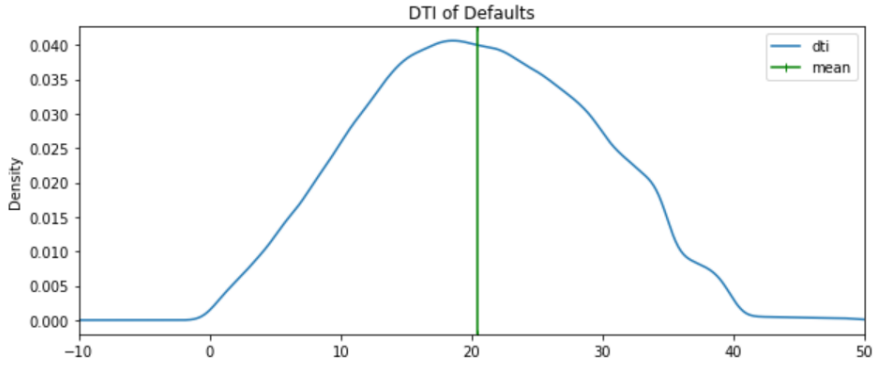


Looks like there's a decent correlation here. I'd love to see how someone opens 64 accounts in the span of 24 months. That's 2.67 per month! Yes, they defaulted but another borrower who opened 61 had not defaulted yet. What a world.

Debt-To-Income

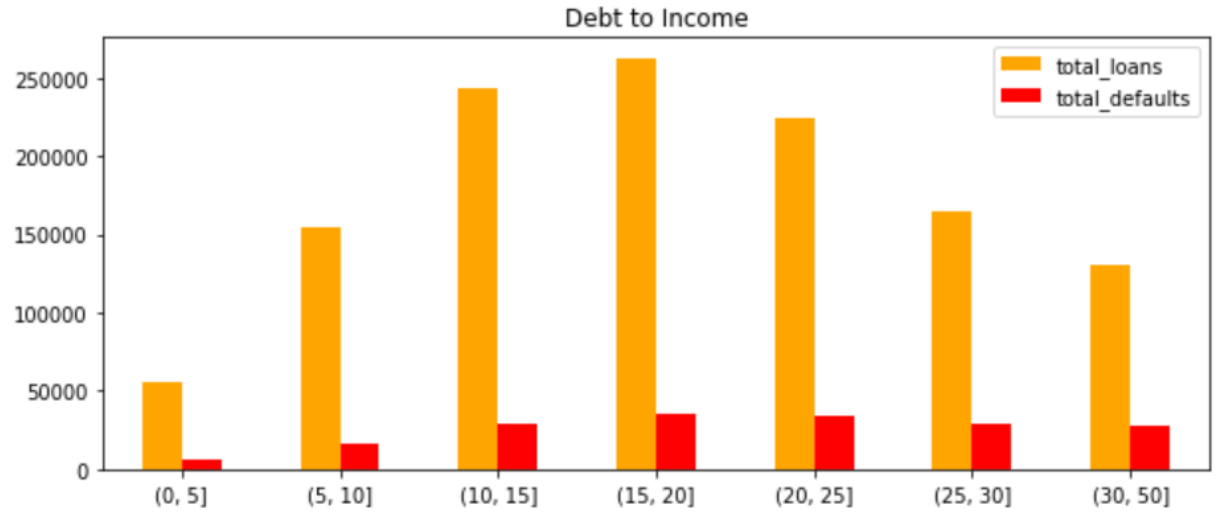
Debt to income (DTI) is a calculation of monthly liabilities to monthly income. Lending club calculates this rather uniquely in that they do not include the cost of housing, nor the loan itself in their calculation. That's important to know for anyone who comes from a lending background who becomes confused by how low the average is.

DTI ranges from 0 to ~50. The DTI ranges for the loans that defaulted appear like so:



I love how gaussian this looks. The mean for a defaulted loan DTI is just above 20.

And the distribution for all loans looks like this:



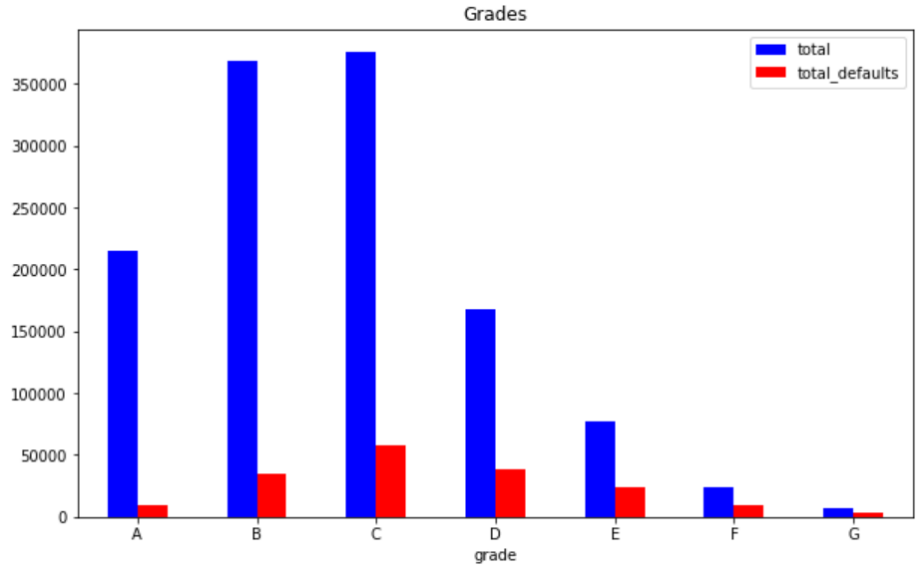
Loan Grades:

Lendingtree grades all of their loans so that potential investors know what to expect in terms of loan quality. The lower the grade, the bigger the risk, the higher the payout (interest rate) for the investor. Since the loan populations are so different across each grade. It's more useful to see this data written out:

grade	total	total_defaults	default_rate
A	214784	9135.00000	4.25311
B	368036	34683.00000	9.42381
C	375543	58203.00000	15.49836
D	168193	38713.00000	23.01701
E	76789	23446.00000	30.53302
F	23991	9248.00000	38.54779
G	7556	3044.00000	40.28587

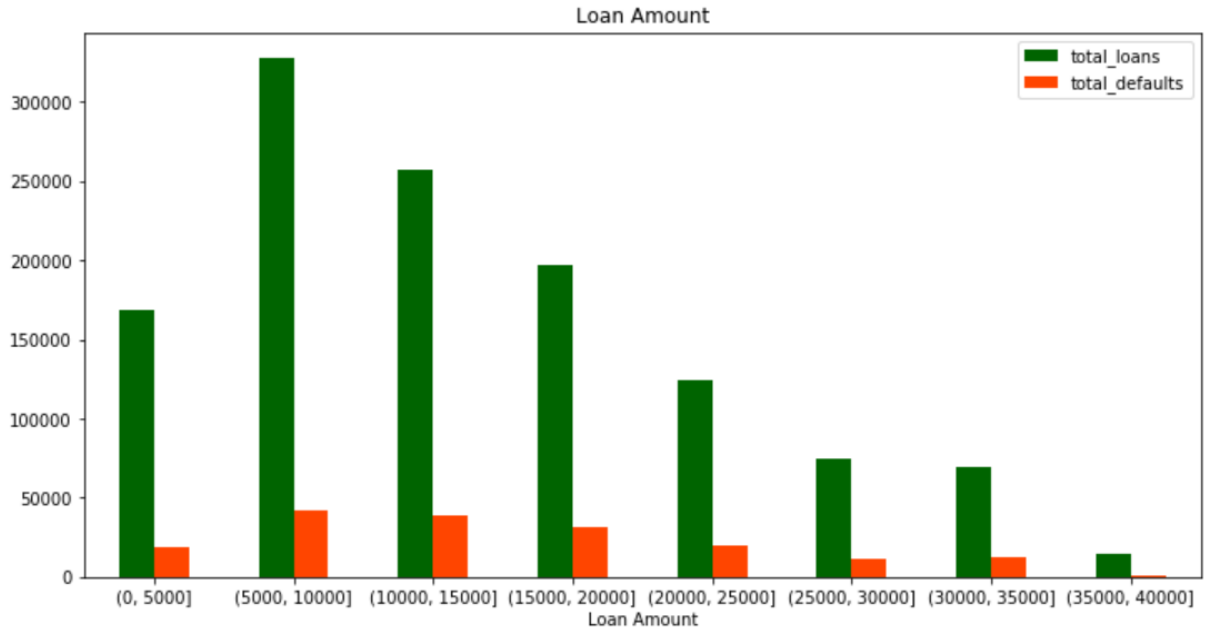
If the loan is graded E or worse, it has a greater than 30% chance of defaulting. It looks like lendingclub is doing a great job grading their loans but that's an incredible amount of risk for anyone to take on.

And here's a visual representation for everyone whose eyes glazed over when they saw the table:



Loan Amount:

Loan amounts have a range from $1000 to $40,000. Hopefully it goes without saying that the larger the loan amount, the larger the payment.

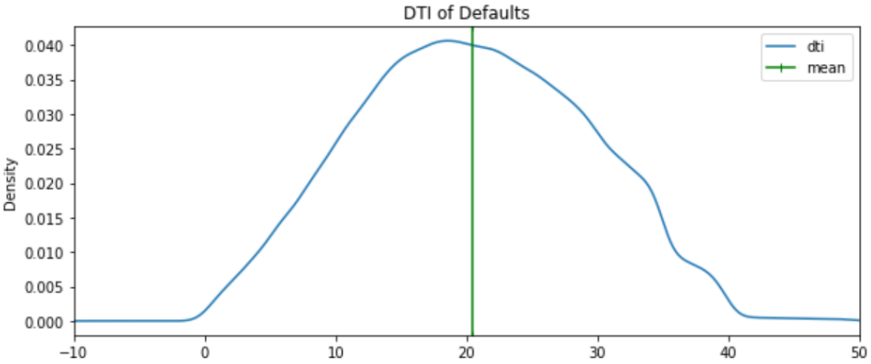


Looks like the higher the loan amount, the less loans we have, but the greater the amount of defaults as a percentage of the total number of loans.

Debt To Income

Debt to income (DTI) is a calculation of monthly liabilities to monthly income. Lending club calculates this rather uniquely in that they do not include the cost of housing, nor the loan itself in their calculation. That's important to know for anyone who comes from a lending background who becomes confused by how low the average is.

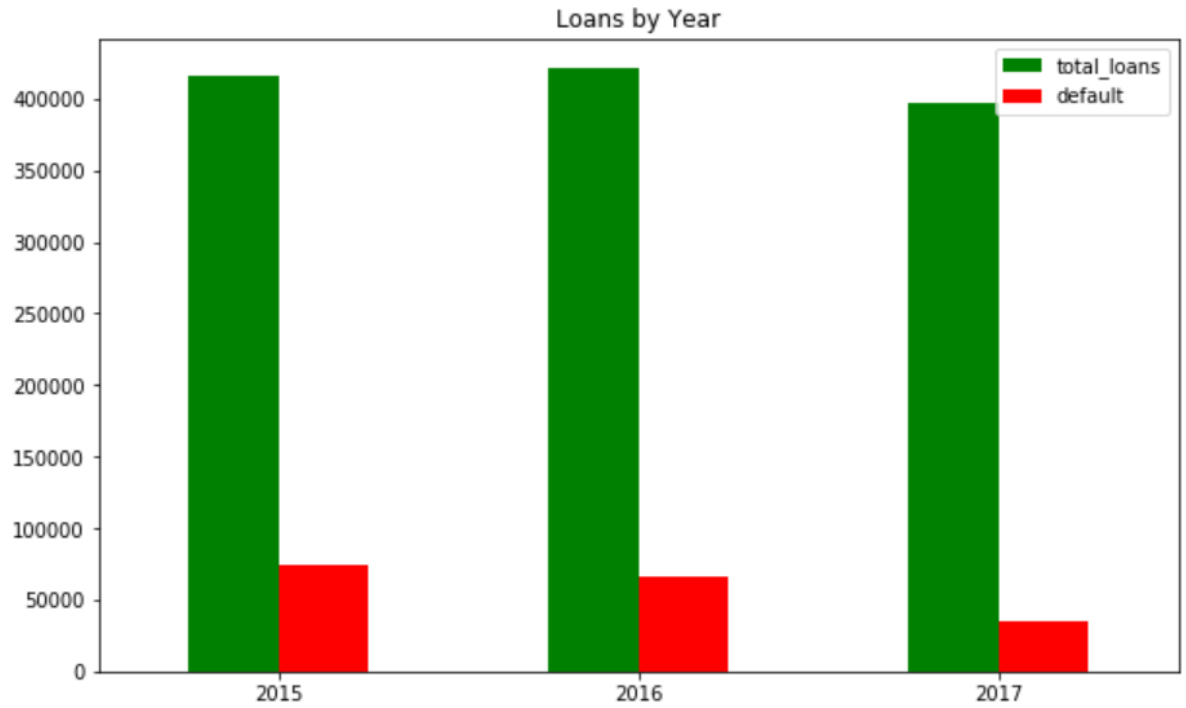
DTI ranges from 0 to ~50. The DTI ranges for the loans that defaulted appear like so:



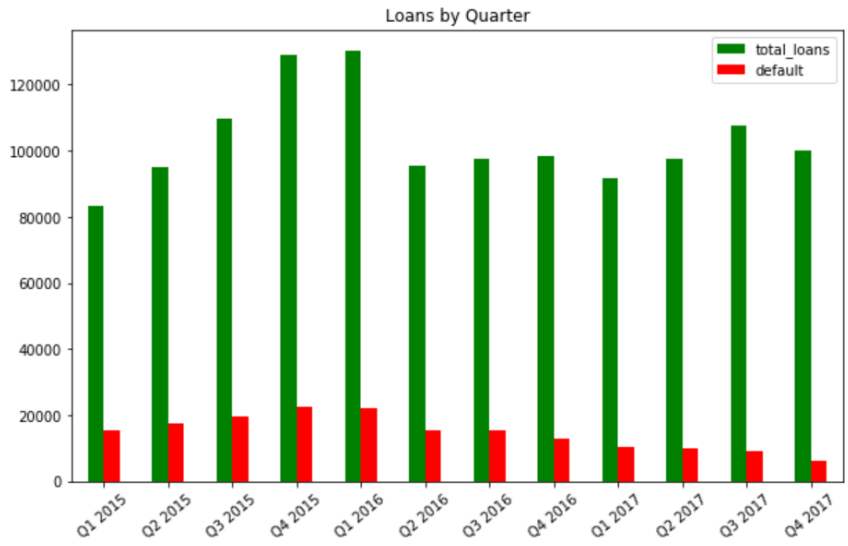
I love how gaussian this looks. The mean for a defaulted loan DTI is just above 20.

Issue Date

I won't lie to you. It was the AI that discovered this during the Machine Learning process, not me. The machines discovered a correlation between the loan's id number and default rate. Since the id numbers are consecutive, the issue date has the same relationship. Look at the defaults for our 3 years:



My theory here is that since this data is from the end of 2018, not enough time has elapsed for the 2017 loan default rate to catch up with its peers in 2016 and 2017. If this is correct, we would expect to see default rate slowly go down by quarter. Let's check it out:



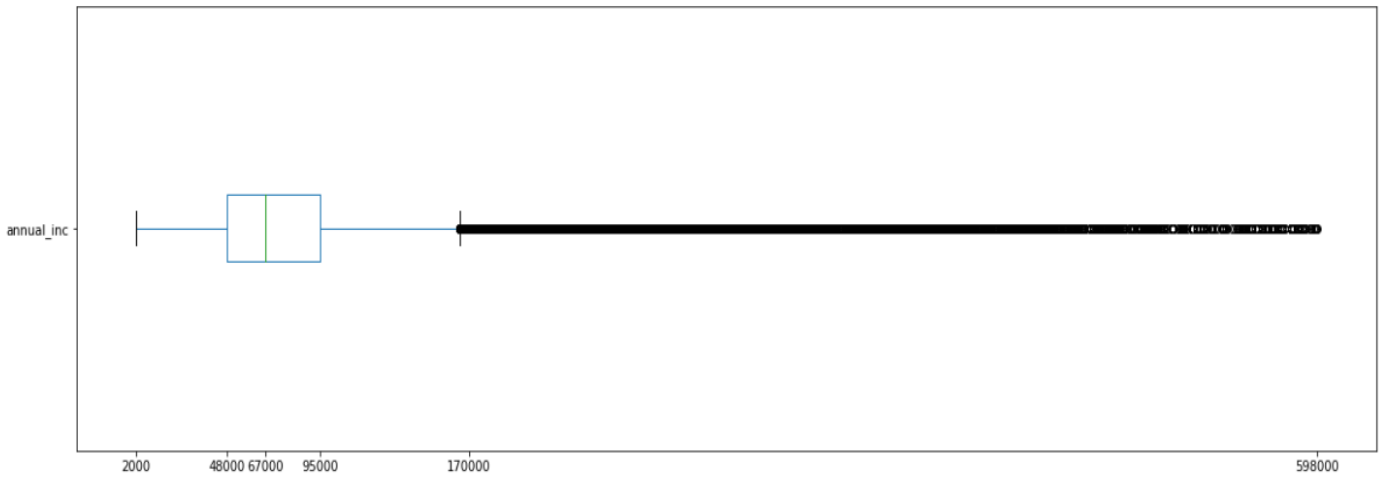
The graph is makes it a little tough to see so let's look at the actual table.

	default	total_loans	default_rate
Q1 2015	15256.0	83474	18.276350
Q2 2015	17521.0	94973	18.448401
Q3 2015	19634.0	109547	17.922901
Q4 2015	22539.0	128794	17.500039
Q1 2016	22159.0	130204	17.018678
Q2 2016	15520.0	95383	16.271243
Q3 2016	15576.0	97554	15.966542
Q4 2016	13036.0	98218	13.272516
Q1 2017	10221.0	91491	11.171591
Q2 2017	9840.0	97699	10.071751
Q3 2017	9181.0	107422	8.546666
Q4 2017	5989.0	100133	5.981045

With the exception of Q1 2015 -> Q2 2015. The default rate goes down every quarter. My theory holds!

Annual Income

Lastly, let's take a look at our annual incomes in this dataset.



That is a very wide range for annual income and an even more impressive number of potential outliers.

Machine Learning

The models I've chosen to test out for this experiment are Logistic Regression, Random Forest, and Gradient Boosting. For each of the 3 algorithms, I'm going to start with a 5 fold cross validation on just our training data before try out our testing set. As this is a very imbalanced dataset, I experimented with upsampling and downsampling to try to minimize the effect that this imbalance would have on our model.

Logistic Regression

Even after experimenting with upsampling and downsampling, this model had a difficult time with predicting default. It would appear that there was a lack of linear relationships between our features and our target.

Random Forest

Random Forest performed much better than Logistic Regression. Upsampled data was included as part of the experiment to see how it would perform when compared to the normal data. We were able to achieve 93% accuracy with the upsampled data used for training improving performance by only the slightest of margins.

Gradient Boosting

Gradient boosting also performed well, and at a much quicker pace than the Random Forest. Once again, 93% accuracy was achieved.

Optimizing Our Model:

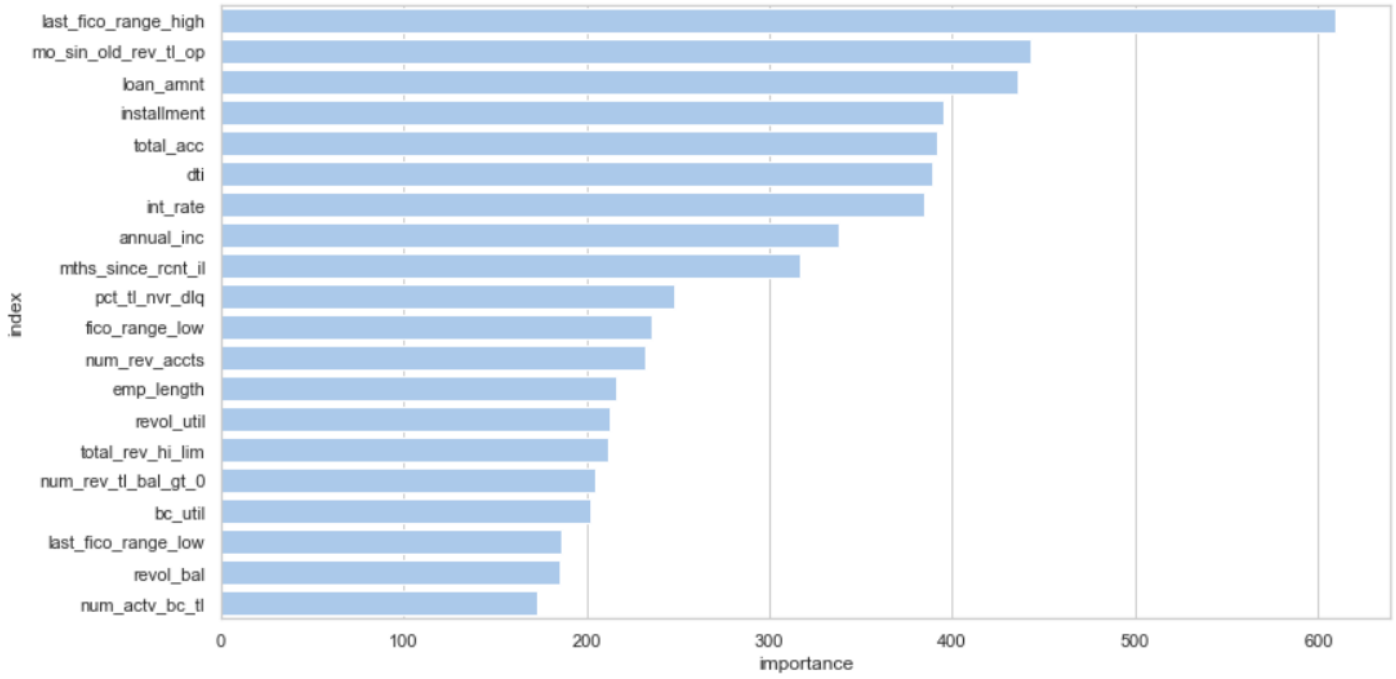
Even though Random Forest technically performed best (by the thinnest of margins), I chose to optimize our Gradient Boosting algorithm as Random Forest is so computationally expensive that I would be optimizing for weeks.

Hyperparameters were optimized by testing out ranges of numbers for each one and measuring performance. Results were optimized for maximum F1 score and maximum ROC AUC score separately and then testing results were compared.

After all that, the optimization barely improved our model at all. Since so much time was spent optimizing, I went with the ROC AUC model even though it almost imperceptibly improved performance.

Feature Importance:

In terms of feature importance, here is what our Gradient Boosting model preferred to use to make predictions:



Let's review the top features:

I'll go over the most important features here. I'll start with the definition of the feature and and explain how our model interprets its importance to predicting default.

last_fico_range_high - Most recent credit pull is the best predictor as we would expect. Recent delinquent credit such as missed or late payments on any account, defaulted loans, or newly opened credit obligations will all bring this number down.

mo_sin_old_rev_tl_op - Months since oldest revolving account opened. Revolving accounts are lines of credit such as credit cards. A low number here indicates a lack of credit history indicating that the borrower doesn't have a history of paying debts off so we would expect them to be more likely to default.

loan_amnt & **installment** - The installment (monthly payment) will vary based on the loan amount and interest rate. The higher the loan amount, and thus the monthly payment, the more likely the default.

total_acc & **dti** - The total number of credit lines currently in the borrower's credit file and debt to income. The more accounts, the more debt obligations, the more likely a customer is to default. DTI is the actual ratio of the monthly debts of these accounts over the borrower's income. It's essentially a measure of the borrower's capacity for more debt.

int_rate - Interest rate is a chicken or egg dilemma. Higher interest rates are assigned to riskier borrowers to get investors to take them on but the interest rate itself makes the loan more difficult to pay off. The lower the interest rate the more likely the borrower is not default.

annual_inc - Annual income. Obviously if a borrower has low annual income, we would expect them to be more likely to default.

mths_since_recent_il - Months since most recent installment accounts opened. Installment accounts are loans. If a borrower hasn't taken on any other installment loans recently, they should be more prepared to pay off their loan.

Takeaways:

Most recent FICO Score was our best feature but that’s a lagging indicator. If our goal is to predict defaults before the loan is even issued, we have better features at our disposal. As an investor reviewing potential borrowers I would focus on the following

features in this order: borrowers with a high number of months since oldest revolving account opened, lower loan amounts & payments, a low number of total accounts, higher annual incomes, and a high number of months since the most recent installment accounts were opened.

None of these features is revolutionary when it comes to financing, but it is helpful to cut down on the noise of 100+ features and find the 5-6 columns that truly matter when it comes to predicting default.

Future Study: If I had the processing power, I'd love to use the whole dataset rather than just the 3 year period I ended up using. In addition, I'd be curious to study the datetime features and see how certain time periods compared in terms of defaults as compared to others.