Identifying Loans with High Risk of Default

By Neil Oza

Proposal

Lending Club is an online platform for peer to peer lending. Through Lending Club, individuals can fill out loan applications and other individuals can review the application and choose to lend that applicant money. Lending Club published an anonymized version of all loans issued on its platform to Kaggle. In this analysis, I will use that data to identify current loans that are at high risk of default. This could help lenders work out payment plans with clients working through precarious financial situations and estimate/ budget for shortfalls in revenue.

Goals

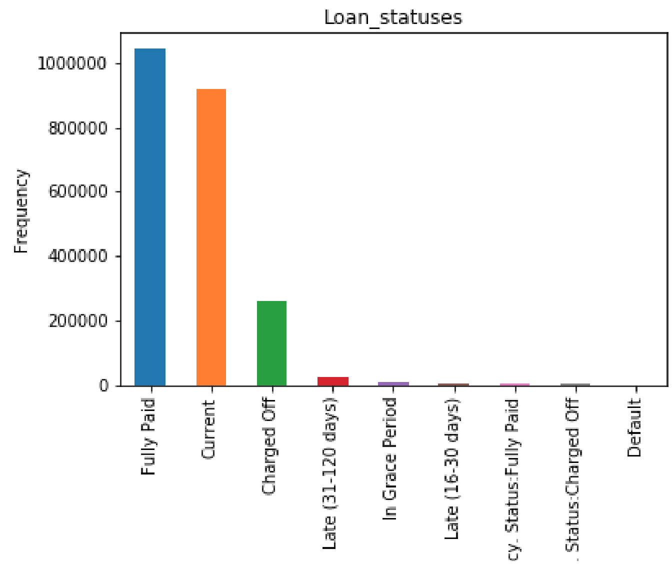
- Understand Lending Club's Loan Dataset, visualize its features, and prepare the data for machine learning
- Train machine learning algorithms to predict which loans are in default and which ones are current. Evaluate and compare model performances and select the best model to deploy on the entire dataset
- Interpret the results from the best model to determine the effectiveness of machine learning on this problem. Identify useful insights that can be derived from the algorithms predictions

Data Collection and Elementary Data Analysis

To begin my analysis, I downloaded Lending Club's Loan Data from Kaggle as a csv. I opened the data in a Jupyter Notebook using pandas and began inspection. The original dataset contains approximately 2.5 million rows by 150 columns . At minimum, machine learning algorithms need 10 columns and 10000 rows to train effectively, so there should be plenty of data to work with in this dataset. Lending Club provided a data dictionary for this dataset, linked in the appendix. I read through it to get a general idea of the column contents and then began my EDA.

EDA stands for Elementary Data Analysis.I had two goals for this EDA. The first wasto identify a target variable column, and the second to create the featureset to identify the target variable. I started with the former. By looking through the data dictionary, I identified loan_status as a suitable target variable; the goal of this analysis is to predict which loans will fail to pay as agreed, and the loan_status column contains data on which loans failed to pay as agreed. The column contains categorical data, so I used matplotlib.pyplot to create the bar plot to the right, summarizing the distribution of loan_status values. The column takes on a total of nine values. These categories can be clustered into three broader groups: Fully Paid, Current, and Default. Fully Paid encompasses the categories ‘Fully Paid’ and ‘does not meet the credit policy. Status:Fully Paid', and represents columns that have been completely paid off. The Current category contains the labels ‘Current’ and ‘In Grace Period’ represent loans that are currently paid as agreed. The Default category consists of all remaining labels and represent loans that are not being paid as agreed. This category is called ‘Default’ instead of ‘Charged Off’ to follow industry standards. Since the goal of this analysis is to predict which loans are most likely to default in the future, the Fully Paid category is not needed to train the machine learning algorithms; fully paid loans will never default since it is impossible for them to miss a payment, so we know they default with a probability of 0. Given the rows in the Fully Paid category were unneeded, I created a copy of the original data in the form of a variable called loan_data. I filtered out all rows with a Fully Paid loan status, reducing the data to approximately 1.2 million rows. This still leaves more than enough data for machine learning, and all of the rows can be neatly categorized as ‘Default’ or ‘Current’. This will be relevant later. At this point, the target variable loan_status is cleaned and primed for machine learning.

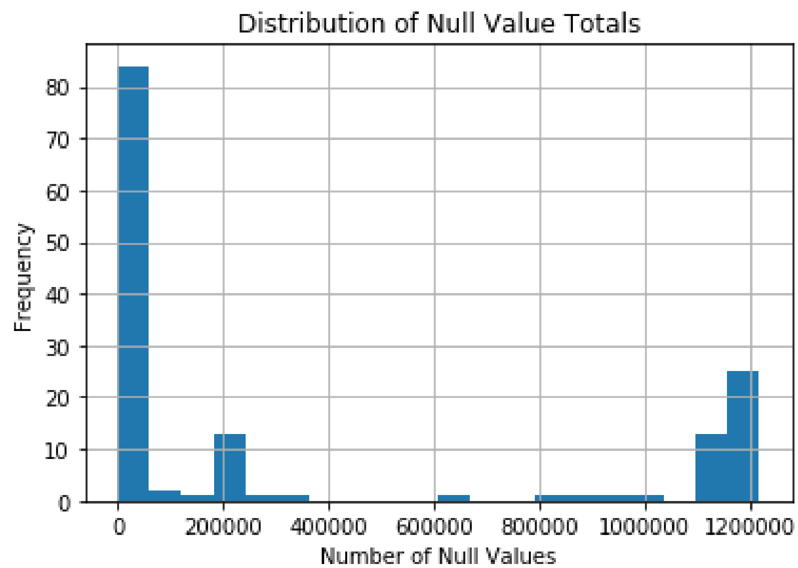
Plot of values in the loan_status column



To create the featureset, unusable columns must be filtered out of the dataframe. In particular, columns with large numbers of null values or strong correlations with the target variable must be removed. Columns with large numbers of null values must be removed because machine learning algorithms do not handle large numbers of null values well. Columns highly corrrelated with the target variable must be removed because the models can become overly reliant on those variables and those variables are often sources of data leakage. In addition, Data that cannot easily be converted into numpy arrays will be excluded because scikit-learn requires numpy arrays to run and train it’s algorithms.

To start, let’s address columns with high numbers of null values. I used pandas to create a histogram representing the total number of null values in each column of the dataset. The histogram is reproduced on the top right of the next page. The histogram appears bimodal, with peaks at both no null values and all null values. Very few columns contain intermediate numbers of null values. Since machine learning algorithms struggle estimating large numbers of null values, I removed all columns with over 600,000 null values. This still leaves circa 100 columns available. Applying this filter, the following columns were removed.

Plot of total null values in all columns

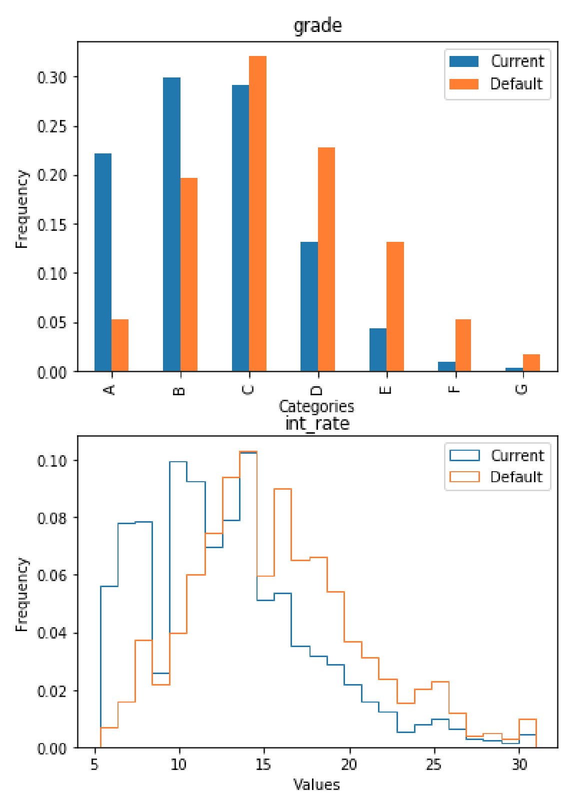


Id member_id	url	desc
mths_since_last_delinq	mths_since_last_record	Mths_since_last_major_derog
annual_inc_joint	dti_joint	verification_status_joint
Mths_since_recent_bc_dlq	mths_since_recent_revol_delin q	revol_bal_joint
Sec_app_earliest_cr_line	sec_app_inq_last_6mths	sec_app_mort_acc
sec_app_open_acc	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_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	debt_settlement_flag_date
settlement_status	settlement_date	settlement_amount
settlement_percentage	settlement_term	

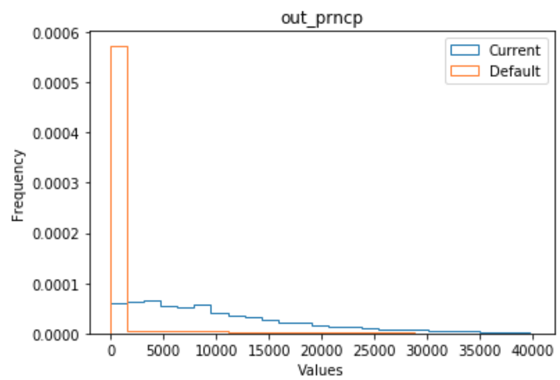
After removing the columns with high null values, I began inspecting the distribution of data in the remaining features. Specifically, I wanted to know if any of these features were too highly correlated to loan_status. Columns that are too highly correlated to the target variable cause machine learning models to become overly dependent on those features and those features are often the source of data leakage, and so should be removed. To determine which columns are highly correlated with the target variable, I compared the probability distributions of current vs default loans using barplots and histograms. I created histograms to represent the distribution of numeric variables and bar plots to visualize categorical variables with fewer than 12 levels in their categories. I created a function to iterate through the columns and construct the plots for all of the features,

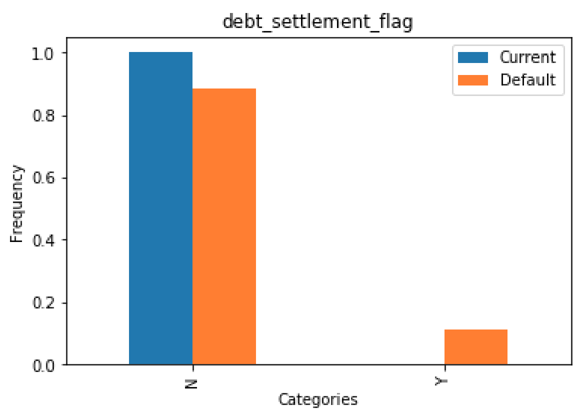
Example distributions of feature values



and I eyeballed the distributions for significant differences in distribution. Some of plots created are reproduced to above to the left.

Example Problematic Feature Distributions





In general the distributions of features were similar between default and current loans. However, in select cases there are noticeable differences. For example, out_prncp often takes on the value of 0 for default loans and larger values for current loans and the feature debt_settlement_flag only takes on the value ‘Y’ when the loan is in default. Variables such as these interfere with proper machine learning and so will be removed.

In addition, certain categorical variables could not be plotted due to having too many possible values. One of these was a datetime value for the issue date. As it turns out, loans issued early in the year are more likely to default. This variable was properly encoded and added to the dataset. The other variables with too many possible categories, such as zip code and occupation, could not be properly encoded and so will be dropped from the data. The columns with either too many possible values or highly correlated categorical variables are as follows:

Emp_title	pymnt_plan	zip_code
out_prncp_inv	addr_state	out_prncp
title	total_pymnt	total_pymnt_inv
total_rec_prncp	total_rec_int	total_rec_late_fee
recoveries,, last_pymnt_d	collection_recovery_fee	last_pymnt_amnt
next_pymnt_d	last_credit_pull_d	issue_d

earliest_cr_line	debt_settlement_flag	

As a corollary, here are the columns that will be used for machine learning.

funded_amnt	tot_cur_bal	mths_since_recent_bc
funded_amnt_inv	open_acc_6m	mths_since_recent_inq
term	open_act_il	num_accts_ever_120_pd
int_rate	open_il_12m	num_actv_bc_tl
installment	open_il_24m	num_actv_rev_tl
grade	mths_since_rcnt_il	num_bc_sats
sub_grade	total_bal_il	num_bc_tl
emp_length	il_util	num_il_tl
home_ownership	open_rv_12m	num_op_rev_tl
annual_inc	open_rv_24m	num_rev_accts
verification_status	max_bal_bc	num_rev_tl_bal_gt_0
purpose	all_util	num_sats
dti	total_rev_hi_lim	num_tl_120dpd_2m
delinq_2yrs	inq_fi	num_tl_30dpd
inq_last_6mths	total_cu_tl	num_tl_90g_dpd_24m
mths_since_last_delinq	inq_last_12m	num_tl_op_past_12m
open_acc	acc_open_past_24mths	pct_tl_nvr_dlq
pub_rec	avg_cur_bal	percent_bc_gt_75
revol_bal	bc_open_to_buy	pub_rec_bankruptcies
revol_util	bc_util	tax_liens
total_acc	chargeoff_within_12_mths	tot_hi_cred_lim
initial_list_status	delinq_amnt	total_bal_ex_mort
collections_12_mths_ex_med	mo_sin_old_il_acct	total_bc_limit
policy_code	mo_sin_old_rev_tl_op	total_il_high_credit_limit
application_type	mo_sin_rcnt_rev_tl_op	hardship_flag
acc_now_delinq	mo_sin_rcnt_tl	disbursement_method
tot_coll_amt	mort_acc	month_issued

At this point I have identified columns with many null values, columns highly correlated with the target variable, and categorical columns with too many possible values to encode. These columns were all dropped from the data. The data is now ready for machine learning.

Model Training and Selection

The goal of training a machine learning model is to use past data to analyze and make predictions for new data. There are several different machine learning models available that can learn from the data and make these predictions. The goal is to identify the model that makes the best predictions. In this analysis, I will train logistic regression, SVC, naive bayes classifier, random forest, and xgboost models on the data. I will describe the models briefly below.

Logistic Regression- Logistic Regression is a classification algorithm that uses a linear equation of features to predict the target variable. It requires regularization through techniques such as ridge, lasso, and elastic net. It performs best when the features are linearly independent and when relationships are simple enough to model linearly. Logistic Regression algorithms struggle with complicated relationships between the target and featureset and struggles with partially correlated features
SVC- Support Vector Machines classify data by creating a hyperplane that serves as a border between categories. Support Vector Machines have two major hyperparameters:C and gamma. C is a regularization parameter and gamma influences the importance of points far away from the decision boundary when creating the decision boundary
Naive Bayes Classifier- Naive Bayes Classifier uses Bayes Theorem and assumes independence among all features in order to predict the target variable. The algorithm requires a smoothing parameter to help handle previously unseen values
Random Forest- Random Forest Classifier uses multiple decision trees to classify a new data point. Each tree in the random forest is trained independently on a subset of the data. This causes each tree to split the data slightly differently, so when presented with new data the individual trees reach slightly different conclusions. Random forests return the average of all of the trees as the classification.
XGBoost- XGBoost uses a decision tree to create an algorithm modeling the data. Then it identifies which data points the algorithm misclassified,gives them added weight, and then creates a new decision tree which handles the previously missed values better. XGBoost has two hyperparameters: learning rate and max depth.

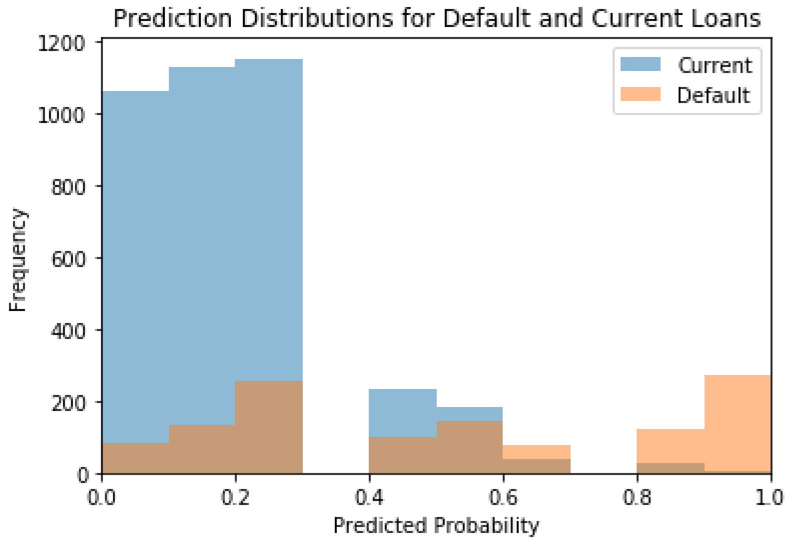
The goal is to train all five of the models, tune their hyperparameters, and identify which model is the best performing. In order to train the models, I will first sample the data, encode the data, split it into a train and test set, and then fit each of the models with the training set. I will use scikit-learn and pandas to train and analyze models in this analysis.

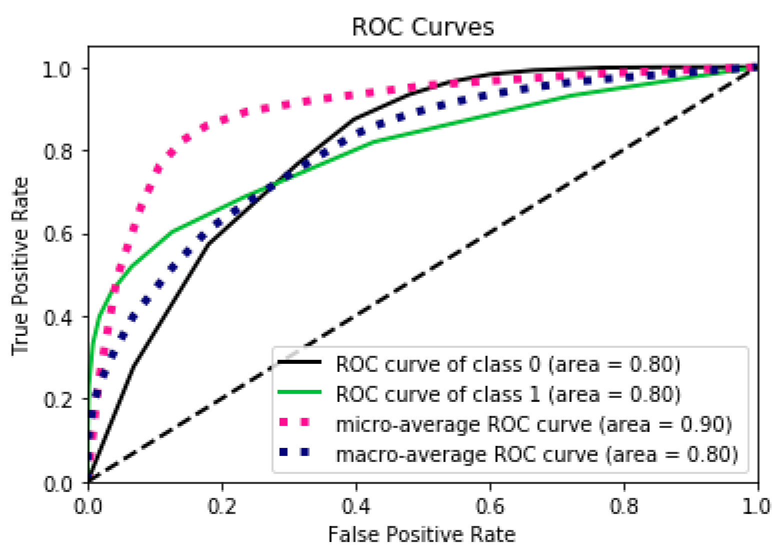
To begin the algorithm training process, I took a 20,000 record sample from loan_data; machine learning algorithms take longer to train given them more data. To save time I sampled the data for model training purposes. Since machine learning models only require circa 10,000 records to train effectively, 20,000 records is sufficient for training

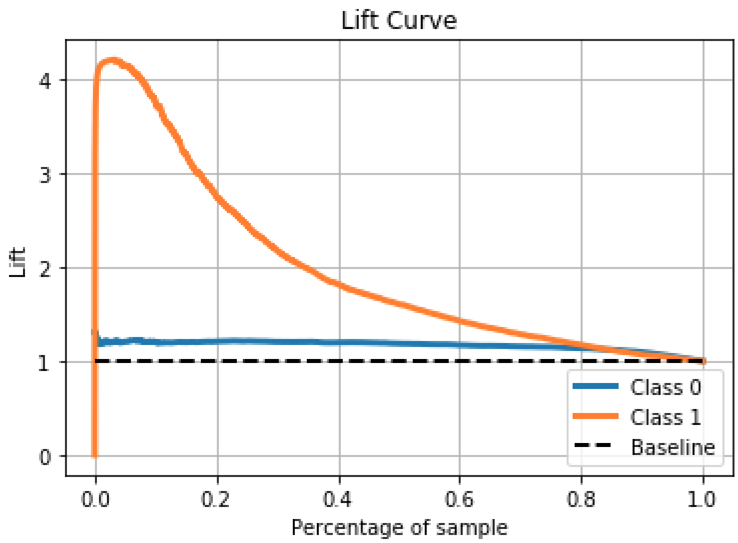
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
import xgboost
rf=RandomForestClassifier()
lr=LogisticRegression()
gnb=GaussianNB()
svc=SVC(probability=True)
xgb=xgboost.XGBRegressor()
```

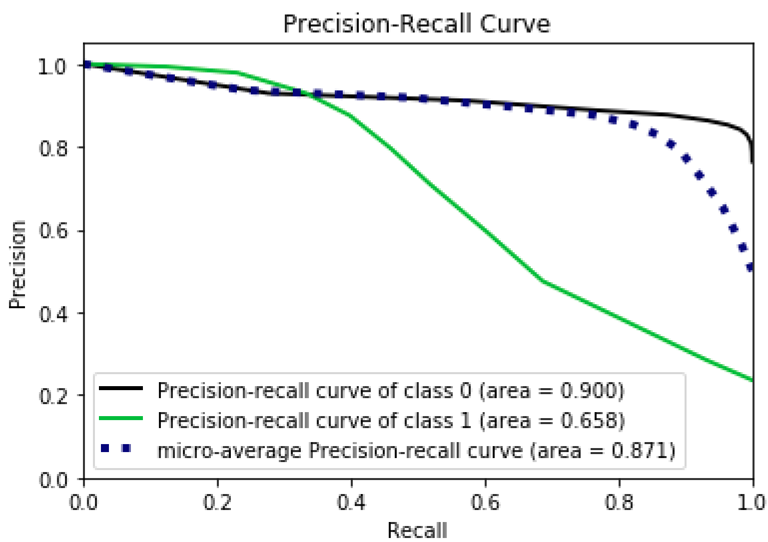
Once I had the 20,000 record sample, I split the target variable from the featureset as the variable y and I encoded the categorical features in the featureset as numbers using one-hot encoding. I branched the featureset into two versions using mean and median fillna methods respectively; algorithms such as logistic regression and svc prefer filling null values with the mean, whereas algorithms like random forest prefer filling with medians. I imported some relevant packages from scikit-learn and initialized the algorithms as shown above to the left. After I initialized the algorithms, I created a function that analyzed the performance of the various algorithms on the sample data. To do this, I created a train test split, trained the model on the training data, evaluated the model using the test data, and plotted/printed performance metrics such as the precision, recall, accuracy,ROC-Curve, lift curve, etc. I used this model on all five algorithms and compared the results. For reference, I reproduced all of my evaluation metrics for the tuned Random Forest and XGBoost algorithms on the following pages.

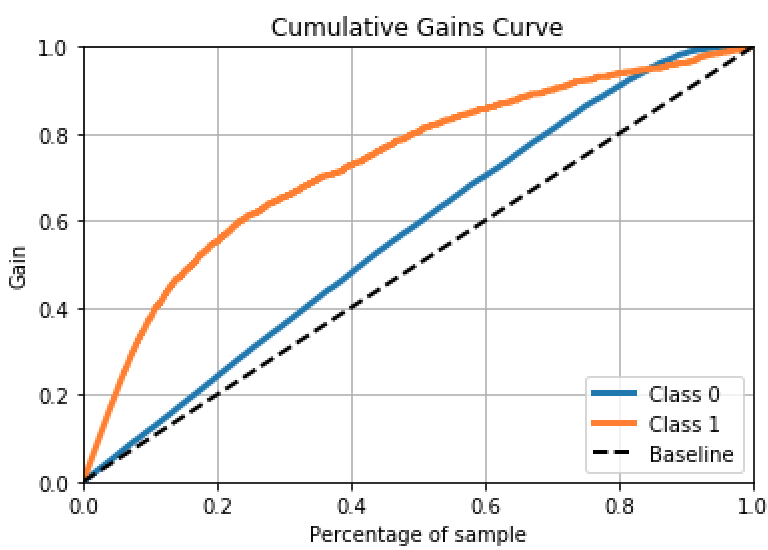
Performance Metrics and Visualizations for Random Forest





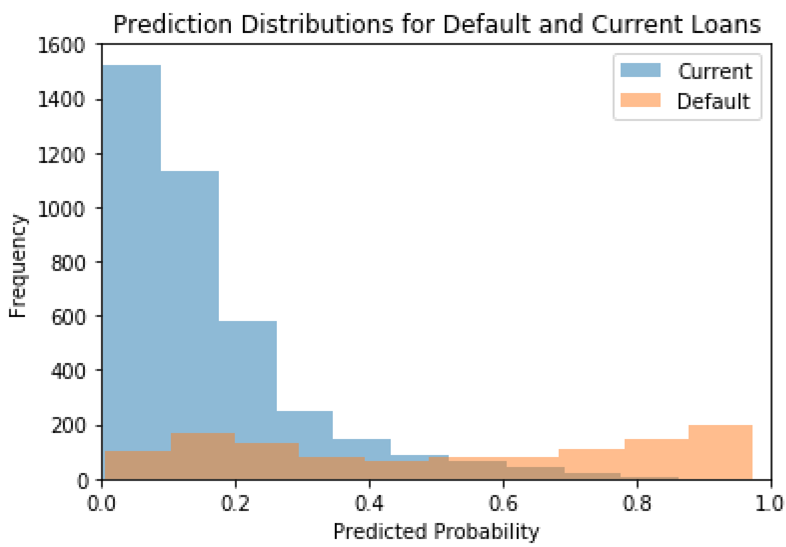

Lift Curve
Lift
Percentage of sample
Class 0
Class 1
Baseline

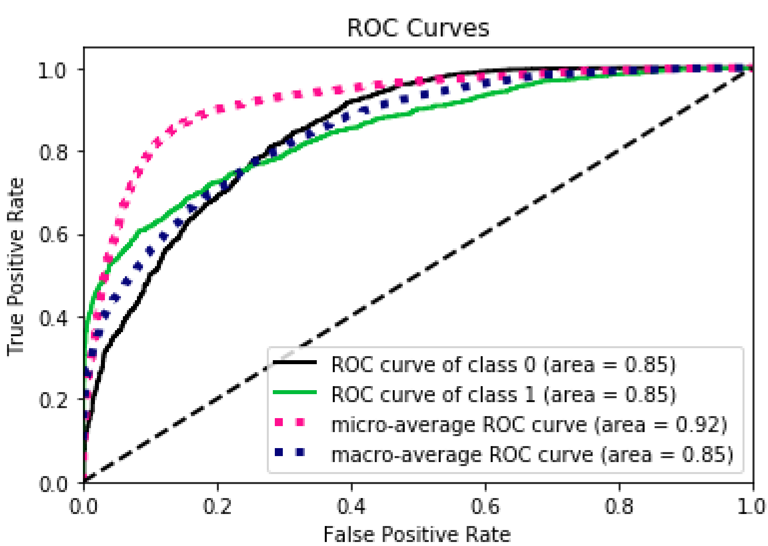

Precision-Recall Curve
Precision
Recall
Precision-recall curve of class 0 (area = 0.900)
Precision-recall curve of class 1 (area = 0.658)
micro-average Precision-recall curve (area = 0.871)

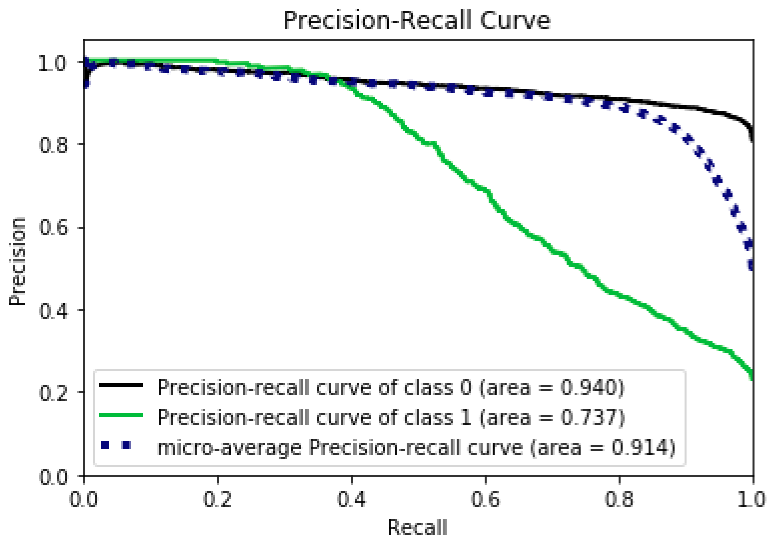

Cumulative Gains Curve
Gain
Percentage of sample
Class 0
Class 1
Baseline

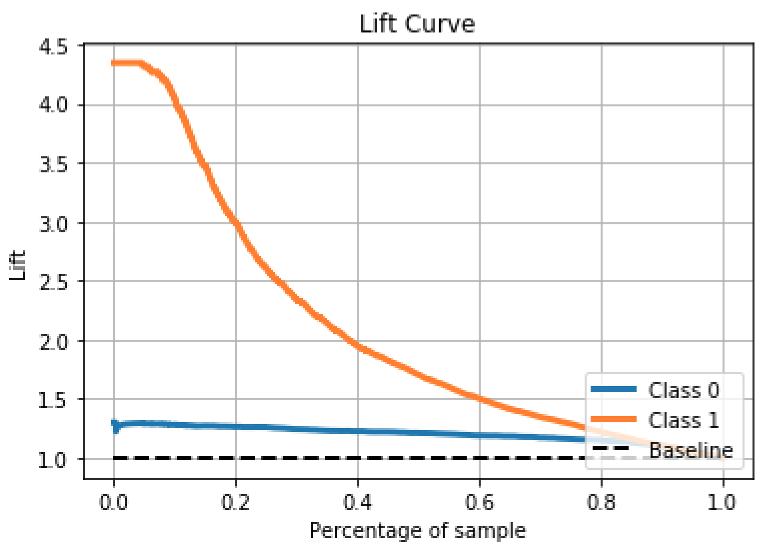
the Precision is 0.7958883994126285
the Recall is 0.45971162001696353
the f1 score is 0.5827956989247312
the roc-auc score is 0.8032537033078436
the accuracy score is 0.8448
The number of loans predicted to default is 681 out of a test size of 5000

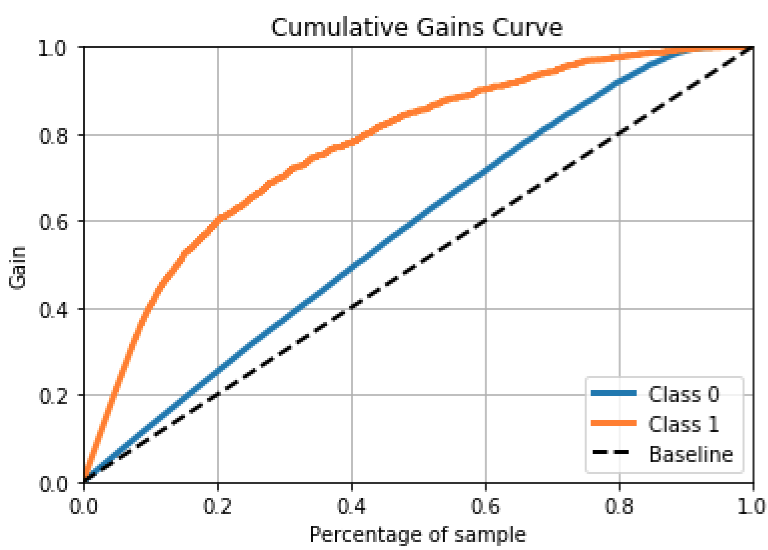
Performance Metrics and Visualizations for XGBoostt






Precision-Recall Curve
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Precision-recall curve of class 0 (area = 0.940)
Precision-recall curve of class 1 (area = 0.737)
micro-average Precision-recall curve (area = 0.914)


Lift Curve
Lift
Percentage of sample
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.0
0.2
0.4
0.6
0.8
1.0
Class 0
Class 1
Baseline


Cumulative Gains Curve
Gain
Percentage of sample
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Class 0
Class 1
Baseline

The Precision is 0.8
the Recall is 0.5108601216333623
the f1 score is 0.6235418875927888
the roc-auc score is 0.848883989184233
the accuracy score is 0.858
The number of loans predicted to default is 735 out of a test size of 5000

Out of the box, the random forest classifier performed best. This isn’t unusual, as random forest is usually very strong out of the box. The naive bayes and SVC classifiers seem to be having trouble, likely because several of the features are actually somewhat correlated with one another. Logistic regression and xgboost perform fine, but not as well as random forest.

In order to improve the algorithm’s performances, we can fine tune the hyperparameters of the algorithms. Each of the algorithms have several hyperparameters available for tuning. The table to the right lists all of the relevant hyperparameters for each of the algorithms.

Logistic Regression	C, penalty
Naive Bayes Classifier	var_smoothing
SVC	C, gamma
Random Forest	N_estimators, max_depth, max_features
XGBoost	Learning_rate, max_depth

The hyperparameters can be tuned to improve model performance by several percent. Scikit-learn includes cross validation functionality, which can allow us to determine the best values for the hyperparameters. I imported GridSearchCV from sklearn.model_selection. I used a 5 fold cross validation and created a function that will take in a model and a dictionary of hyperparameters to tune and return the best performing hyperparameters. I applied cross validation to all five models and updated the hyperparameters. I updated the models recalculated performance metrics. There were slight increases of around 1% in precision and recall, but nothing too notable. XGBoost was the best classifier and so I selected it to model the entire dataset.

Analyzing the Results

The XGBoost model had a relatively high precision rate close to 80% and a recall close to 50%. This suggests that if the model predicts a loan to be defaulted, it is very likely to actually be in default. Therefore, if a current loan is classified as defaulted it may likely be on the verge of defaulting and should be addressed accordingly. The fact that the recall is only .5 suggests that there are factors that affect whether or not a loan defaults that are not represented in the data or a significant number of defaults are due to random chance. Additional research and data encoding may produce even better results.

Appendix

The following resources were used to create this analysis. Please look through them for more details.s
Original Data: https://www.kaggle.com/wendykan/lending-club-loan-data
My Jupyter Notebook Analysis:
https://github.com/neiloza/Springboard/blob/master/Lending_Club_Data_Analysis.ipynb
Pandas:https://pandas.pydata.org/pandas-docs/stable/
Numpy: https://docs.scipy.org/doc/
Scikit-Learn: https://scikit-learn.org/stable/documentation.html
Matplotlib: https://matplotlib.org/3.1.1/contents.html