Prosper Marketplace Investment Recommendations

Problem Identification

Prosper is a peer-to-peer lending marketplace that was founded in 2005 and has facilitated more than $18 billion in loans to more than 1 million people in the United States. Borrowers apply online for personal loans ranging between $2,000 and $40,000 and people can invest in these loans.[1]

Personal loans are unsecured notes since they are not collateralized by property such as an automobile or real estate. These loans typically have higher interest rates to account for the higher risk of borrower default. Prosper investments have historically yielded return of **5.3%** so they can offer a way for individuals to diversify their portfolio beyond traditional investments.[2]

In this project, I will be creating an investment recommendation system modeled based on historical loan default. The objective is to achieve a total portfolio return of at least **6.5%** within the next 3 years by identifying high quality loan listings that maximizes interest yield but minimizes principal charge off loss exposure.

Data Source

There were 3 primary sources of data for this project:

1. Listings Historical Files

The Prosper Investor portal publishes historical listings segmented in annual periods back to 2005. All files were downloaded covering a range from January 2005 to December 2020. Below is a link to the Prosper Marketplace download page:

Marketplace Download

2. Core Monthly Loan Level File

The historical listing files only contains data elements about the loan at the time of the listing. I was required to sign a Prosper Data License Agreement to gain access to the core historical loan performance data. This agreement protects the data from being shared publicly and can only be used for personal investment decision purposes. **This is consistent with my problem statement but to remain in compliance with this agreement, all data used for this project has been excluded from the public project repository.**

3. Listings API

Prosper institutes a one-month curtailment on when new listings are available in the current listings download file. For example, a listing created in January 2021 would not be available in the listings download until after March 1, 2021. Therefore, to make investment decisions on new listings, I will have to utilize Prosper's investor listings API for model predictions.

---

[1] Prosper About

[2] Prosper Invest Differently

Data Wrangling

This stage accounted for 75% of the total project time to consume the core loan data and historical files, analyze the credit bureau data elements and clean the data. Some key decisions were made in this stage so in the next sections I will provide insight into those choices.

Process Core Data File

The core data file was fundamental for this project because it provides the connection between historical listing data and the status of the loan. I plotted the data to get an idea on historical volumes.

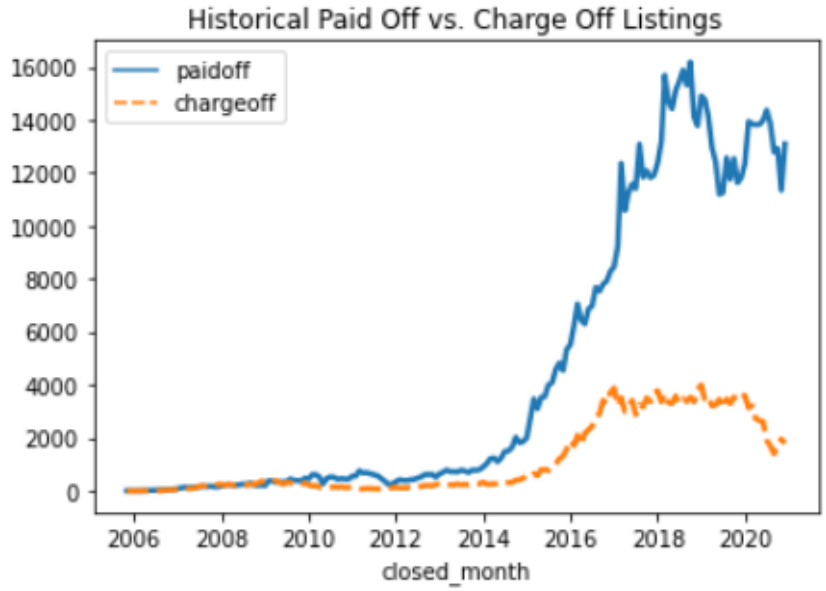

Figure 1: Historical Paid Off vs. Charge Off Listings

According to the Prosper Marketplace Wikipedia, Prosper received a cease-and-desist order from the SEC in November 2008 and they subsequently shut down.[3] Prosper reopened in July 2009 and it took 5 years until 2014 until they started to see divergence of paid off to charge off loans. Based on this information I decided to include only the last 3 years of closed loans 2018 to 2020 since this period represented some degree of normality for credit risk.

Listings Data Dictionary

Next, I created a data dictionary since this was not provided by Prosper for the listings data. I compared the columns available in the historical listings file and the listings API documentation to identify the features that would be available for new listing predictions.

Here is a breakdown of the **865** columns in the listings file:

Listing API Matches	Not Available in Listing API	Experian Columns	TransUnion Columns
56	15	475	319

Figure 2: Listing Columns Breakdown

[3] Prosper Marketplace Wikipedia

Of the 56 columns that had a listing API match, I was able to remove 17 due to 1) Deprecation but still referenced in the API documentation or 2) Not relevant to historical performance such as unique identifiers. 92% of the columns are from each of the 2 credit bureaus: Experian and Transunion. The next logical step was to take a deep dive into the credit bureau data.

Credit Bureau Data

According to the Investor listings API page, Prosper switched from using Experian credit underwriting data to TransUnion to generate loan offers on 3/31/2017. After this date new listings will only contain TransUnion credit bureau data.[4] This presented a challenge since I had to figure out a way to merge as many columns as possible between the 2 credit bureaus. I ran the description column through text similarity algorithms to identify potential matches. The potential matches were manually reviewed, and I was able to match 28 Experian columns to TransUnion. Unfortunately, the rest of the credit bureau columns were discarded since they would not be populated on all historical loans.

Data Cleaning

At this point the data was compiled and I was ready to inspect for cleanliness. My first observation was there may have been some timing issues between the core data file and the listing files because 78 listings did not have a completed status, so these were discarded.

Several columns required missing values to be replaced. The missing values for prosper score and all columns related to the borrower having a prior prosper loan were replaced with zero. The missing values for months employed were replaced with the median and the occupation column was replaced with “Unknown”.

The FICO score column was dropped due to a substantial number of missing values. This was an unfortunate decision, but the missing values might be related to the TransUnion transition. It really was not feasible to replace with another value.

There was some overlap in the DTI (debit-to-income) ratio columns. The combined DTI wprosper loan column was dropped due to substantial number of missing values and the extreme outliers for DTI wprosper loan were removed.

Finally, merging the Experian and TransUnion columns resulted in some abnormal special values so these were replaced with the mode. The credit bureau columns have special values such as -1 for “No trades this type” so it made sense to replace with the most common value of this type.

The final statistics for the dataset was **579,634** rows and **70** columns with no missing values.

[4] Listings API

Exploratory Data Analysis

Data Profile

The average listing amount is $13,379 with an interest rate of 15.35% and a monthly payment of $413.48. Most borrowers opt for the 36-month loan term. In the final clean dataset **18.5%** of closed loans defaulted and were charged off by Prosper.

The lender yield, borrower rate and borrower APR columns have almost identical distributions so only the borrower rate was included. Most listings receive the highest risk score of 11 with an average score of 7.15 and a C is the most common prosper rating.

Several columns were discarded due to heavy skewness to one value including income verifiable, lender indicator and co-borrower application. In the data cleaning step, the prior prosper loan columns were replaced with zero but since so few borrowers had a prior loan these columns were also removed.

Data Relationships

The target variable of interest is charge off so in this section I will summarize some of the key insights that were discovered. As shown in Figure 3, most loans have an interest rate between 7.5% and 12.5% but at higher rates the risk for charge off is much higher.

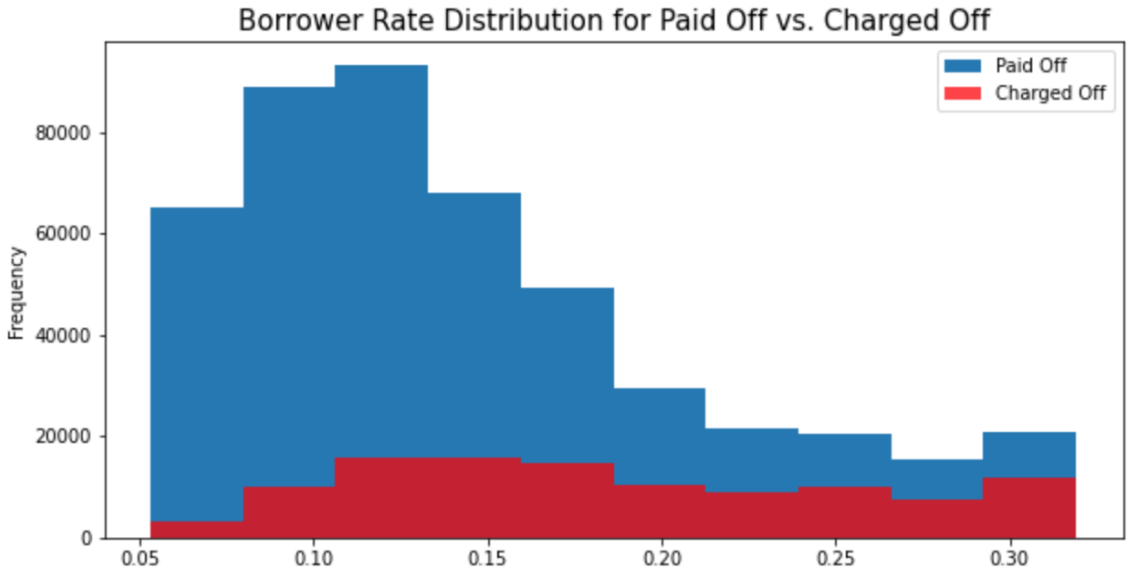

Figure 3: Borrower Rate Distribution for Paid Off vs. Charged Off

Typically, with any type of loan the interest rate reflects the quality of the borrower's credit profile, so I wanted to explore how well the prosper rating aligns to this. Figure 4 clearly aligns with this assumption showing the lower the listing rating the higher the interest rate. This further validates the risk and reward tradeoff between rate and charge off probabilities. In addition, the lowest ratings E and HR had max loan amounts of $12,500.

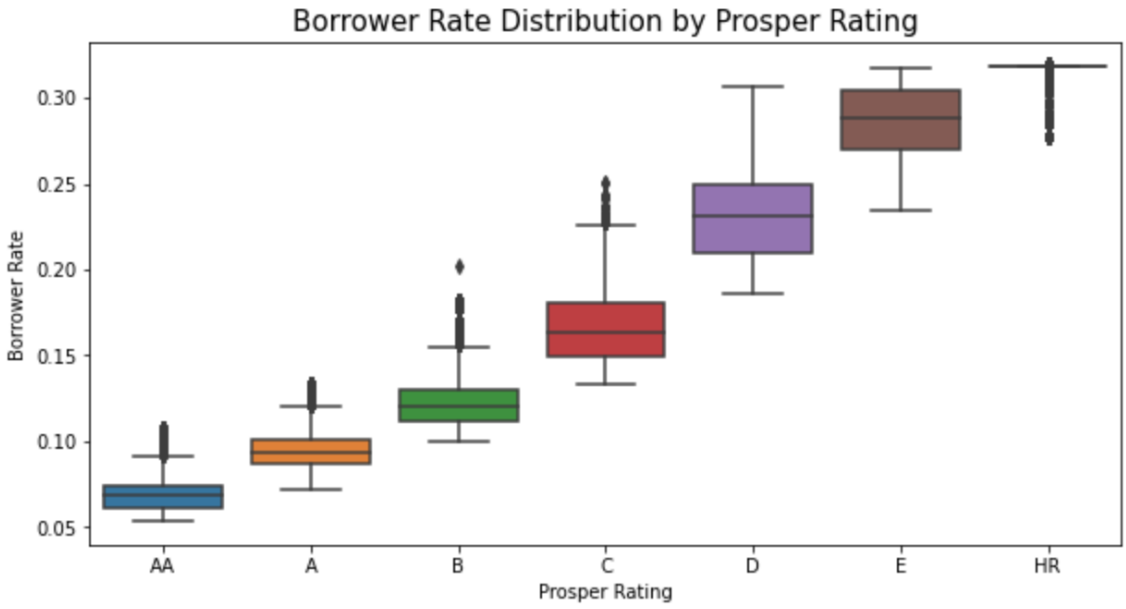

Figure 4: Borrower Rate Distribution by Borrower Rate

Another factor that typically quantifies a borrower's ability to pay their loan obligation back is their current DTI ratio. Borrowers with a DTI ration > 30% typically had ratings of D or lower as well as interest rates > 20%. Charge off %'s are also the highest > 25%.

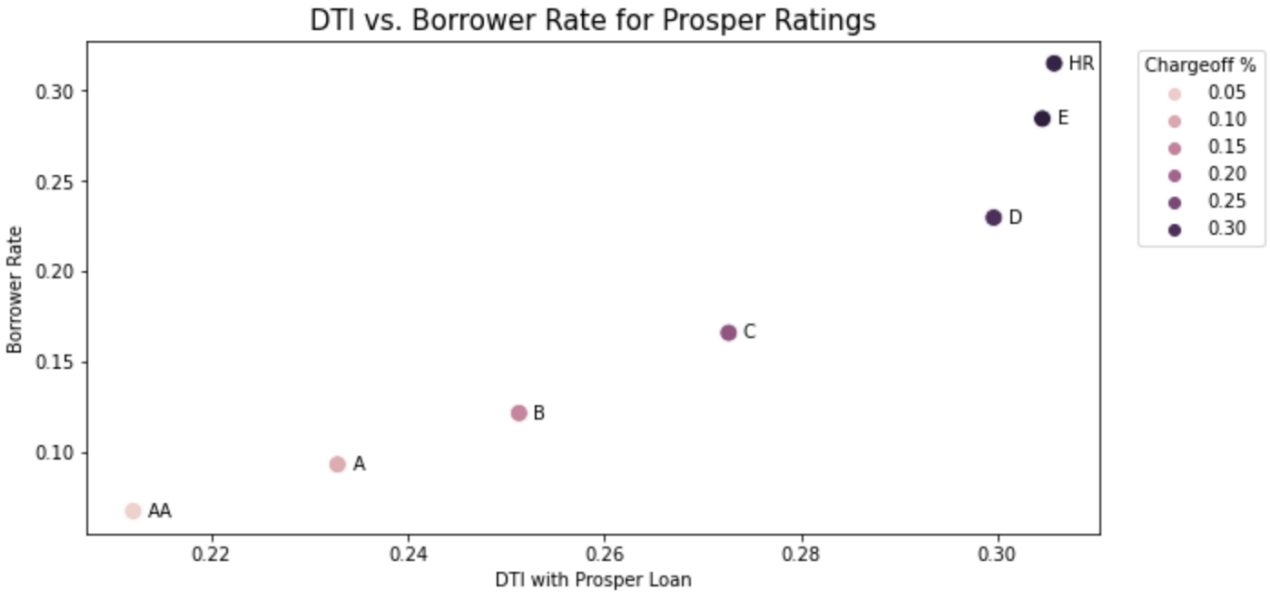

Figure 5: DTI vs. Borrower Rating by Prosper Rating

From a geographic perspective the Mountain region states, Pacific Northwest and West Virginia/New York have the lowest charge off %'s. Iowa is the only US state with no listings. Several variables had no clear relationship with charge offs including income level and how long the borrower has been employed. To a lesser extent the type of employment did not have a strong relationship but some characteristics such as self-employed had higher charge off %'s.

Due to the coronavirus pandemic financial institutions had some limitations on recourse for loans in default. As shown in Figure 6, Prosper had a drastic drop in charge offs to sub 10% levels by September 2020. Since many loans may not have been written off due to the pandemic, I decided to exclude all closed loans from the start of the pandemic in March 2020.

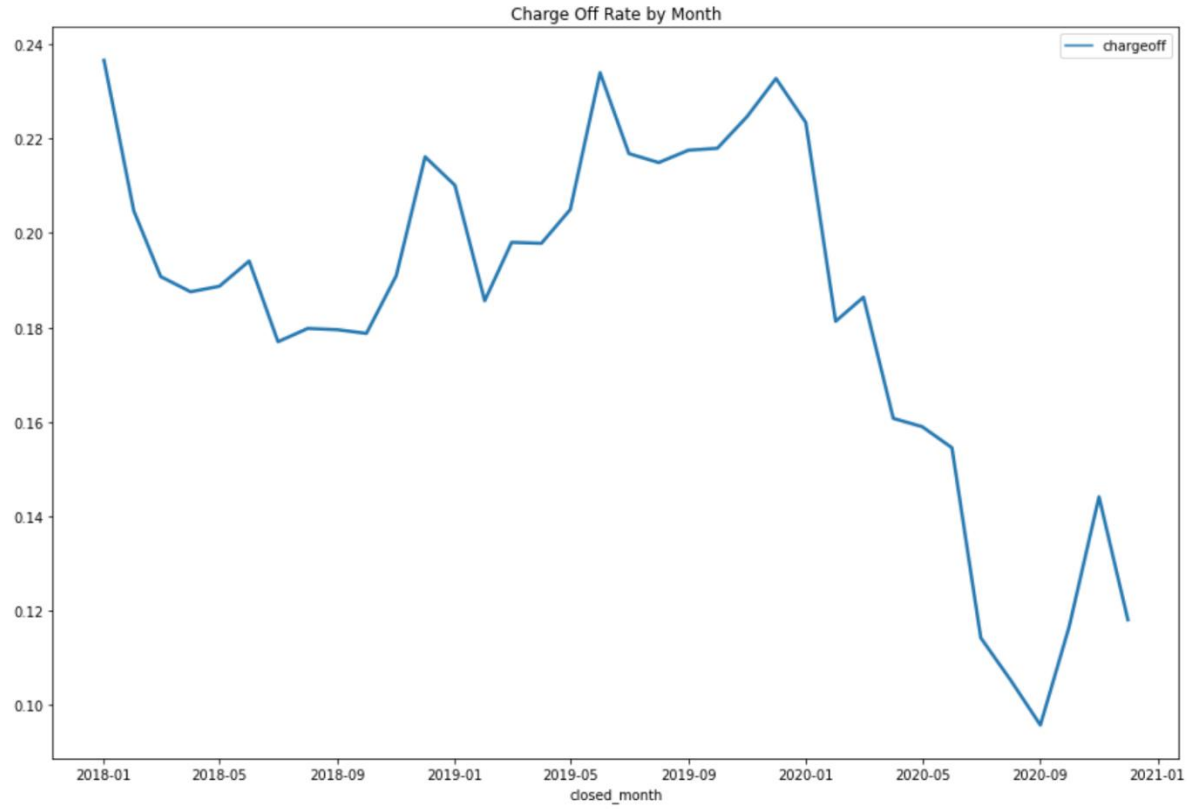

Figure 6: Charge off %'s by Month

The last step in the exploratory data analysis phase was to evaluate all the credit bureau variables. A correlation matrix revealed many variables had higher correlations with each other. I decided to perform Variance Inflation Factor (VIF)[5] analysis to identify multicollinearity issues and removed all columns with a VIF > 5.

Preprocessing and Modeling

Data Transformation

The clean dataset required several transformations to prepare for modeling. The employment type had some inconsistencies in categories so that was cleaned. All the ordinal and categorical variables with low cardinality were one-hot encoded to dummy variables. The categorical variables with high cardinality

[5] Variance Inflation Factor

including occupation and state were encoded using weight of evidence. Finally, all the numerical variables were zero mean standardized using a standard scaler.

Modeling

The first step in the modeling stage was to define my main objective. My goal is to identify the highest quality loans to invest in to maximize rate of return. I am most concerned with **False negative** prediction errors, since this is the situation when the true status of the loan is charge off, but the model predicted it was a good loan. This would include the listing in the pool of potential investments and increase my loss probability.

A **False positive** prediction is not as much of a concern since I have a finite amount of capital and cannot invest in every listing. However, a charge off prediction that is a good loan represents an opportunity cost for higher yield investments so will not be completely ignored in model evaluation. I am more concerned with how well the model detects charge offs (**Recall**) than how accurate the predictions are (**Precision**).

Baseline Model

The next step was to set a baseline expectation model performance. For the model evaluation stage, I captured multiple metrics but was primarily focusing on the following metrics.

1. F2 Score – 2x emphasis on Recall
2. Average Precision Score – Summarizes precision-recall curve
3. False Omission Rate – False negatives / Predicted negative

The false omission rate quantifies my loss exposure since I will only invest in negative predictions from the model output. Figure 7 shows the baseline model evaluation metrics for a model that predicts negative for all listings.

F2 Score	Average Precision Score	False Omission Rate
0.0	.2017	20.17%

Figure 7: Baseline Model Metrics

Initial Models

7 different classification models were trained on the entire dataset and Figure 8 below reflects the model evaluation metrics.

Model Description	Training Time	Test Results.Accuracy Score	Test Results.Precision Score	Test Results.Recall Score	Test Results.F1 Score	Test Results.F2 Score	Test Results.False Omission Rate	Test Results.Average Precision Score
LGBM Classifer	2.068896	0.803570	0.565250	0.113836	0.189507	0.135474	0.186338	0.418128
XGB Classifer	15.585451	0.803531	0.551284	0.140232	0.223589	0.164810	0.182825	0.413556
Gradient Boosting Classifier	92.136293	0.803223	0.586448	0.083314	0.145901	0.100571	0.190381	0.411305
Random Forest Classifier	46.294638	0.799162	0.666667	0.008862	0.017492	0.011041	0.200482	0.401020
AdaBoost Classifer	21.565550	0.801928	0.552790	0.095003	0.162141	0.113862	0.189124	0.399459
Logistic Regression	10.287756	0.801227	0.556206	0.072580	0.128404	0.087859	0.192149	0.393958
Gaussian Naive Bayes	0.408361	0.691791	0.332143	0.522194	0.406030	0.468571	0.141159	0.321929

Figure 8: Initial Model Metrics

Most models except for GaussianNB struggled with the imbalanced dataset. As shown in Figure 9, the confusion matrix for the LGBMClassifier shows the model only predicted 5,272 charge offs.

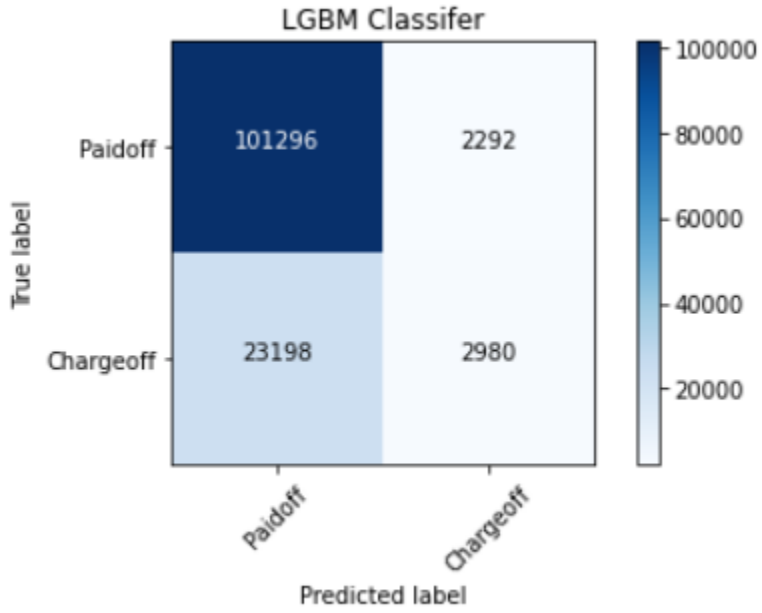

Figure 9: LGBMClassifier Confusion Matrix

Under Sampling

Next, I experimented with an under-sampling technique to force an equal distribution by randomly removing paid off loans from the training data.

Model Description	Training Time	Test Results.Accuracy Score	Test Results.Precision Score	Test Results.Recall Score	Test Results.F1 Score	Test Results.F2 Score	Test Results.False Omission Rate	Test Results.Average Precision Score
LGBM Classifer Under Sampled	1.005312	0.670900	0.344480	0.699251	0.461571	0.579823	0.102743	0.415599
Gradient Boosting Classifier Under Sampled	32.008096	0.662970	0.337853	0.698716	0.455470	0.575728	0.104288	0.410577
XGB Classifer Under Sampled	6.745530	0.667548	0.340414	0.691115	0.456149	0.573043	0.105535	0.403010
Random Forest Classifier Under Sampled	13.529953	0.655064	0.331447	0.697952	0.449455	0.571551	0.105934	0.398485
AdaBoost Classifer Under Sampled	8.833120	0.653006	0.329399	0.695164	0.446994	0.568837	0.107085	0.395797
Logistic Regression Under Sampled	2.785545	0.654023	0.330931	0.699786	0.449359	0.572226	0.105618	0.392850
Gaussian Naive Bayes Under Sampled	0.131449	0.621804	0.302237	0.668424	0.416257	0.538046	0.120772	0.322912

Figure 10: Under sampling model metrics

The LGBM Classifier performed the best for the initial default model and using the random under-sampled data. Comparing the default models to the random under-sampled data models demonstrates the tradeoff between Precision and Recall in the Average Precision Score metric. The Average Precision Score was quite similar in each run but with the random under-sampled data Recall improved at the cost of Precision. In Figure 10 I plotted the Precision Recall Curves for both LGBM Classifier models to visualize that tradeoff.

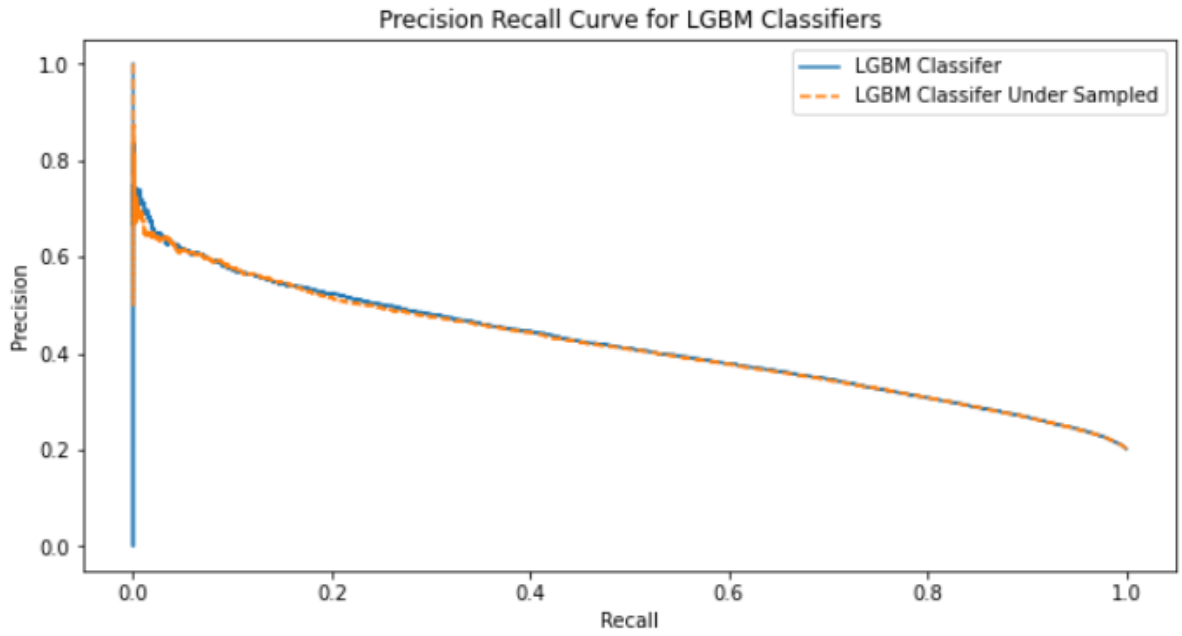

Figure 11: Precision Recall Curve for LGBM Classifiers

The Precision Recall curves demonstrates that the LGBM Classifier using the under-sampled data is the same model, but the balanced class probabilities shift more predictions to charge off and increases Recall. I could accomplish a similar result by adjusting the class weight/positive class (POS) scale weight parameters when training the model using the entire training dataset. Another option would be to lower the threshold for predictions. I explored both options in the model optimization stage.

Model Optimization

As shown in Figure 12, the LGBMClassifier had the highest Average Precision Score on the Test data and the XGBClassifier performed the best on the Train data. I decided to optimize both models for comparison purposes. However, I shifted my focus to the F2 Score evaluation metric since the Average Precision score for the models trained on the imbalanced and random under sampled datasets were the same despite quite very contrasting Recall scores. I realized for my objective this was not the correct evaluation metric.

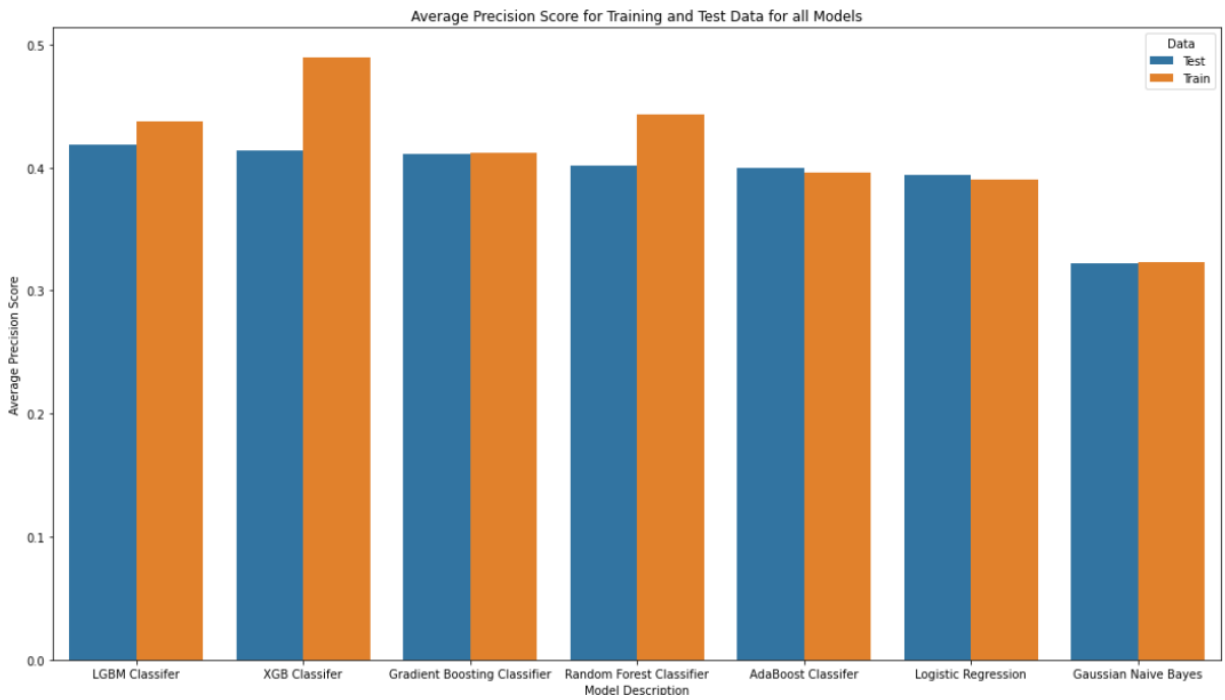

Figure 12: Average Precision Score for Training and Test Data

First, I used a grid search to identify the optimal POS scale weight parameter. The optimal hyper parameter for both models was 10. Next, I used a Bayesian optimization approach using the Hyperopt library to tune several numerical hyper parameters.

Model	Recall	F2 Score	False Omission Rate
XGBClassifier	.918	.606	5.95%
LGBMClassifier	.927	.607	5.53%

Figure 13: Optimized Model Metrics

Finally, I evaluated adjusting the prediction threshold with the same model but with a POS scale weight of 1. The best F2 score was .6030 with a prediction threshold of .10. This is not better than the model with the tuned scale POS weight and standard prediction threshold of .5.

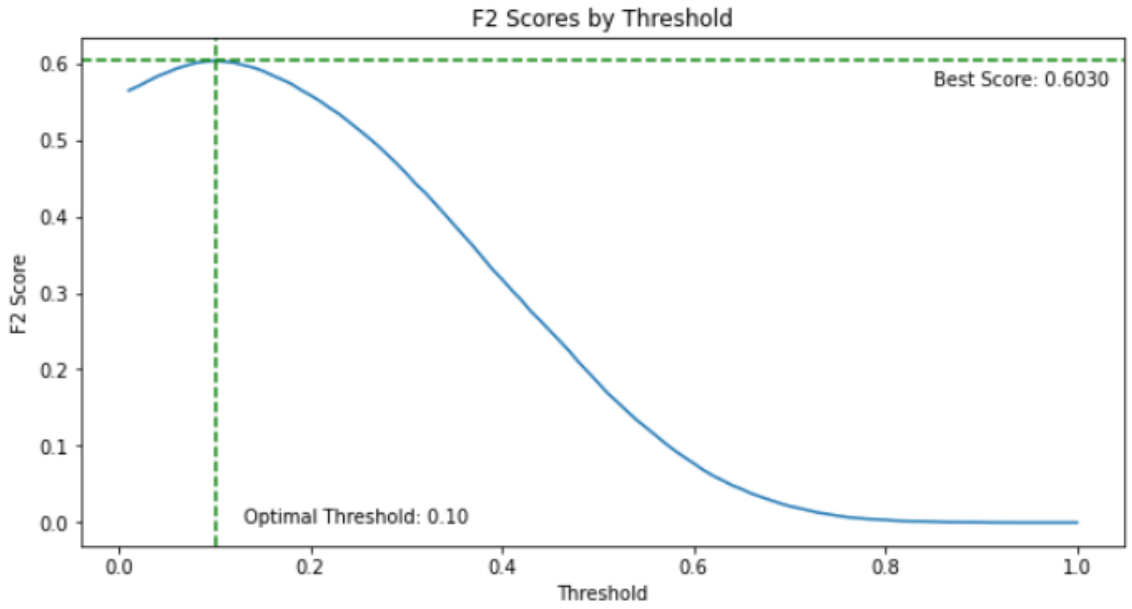

Figure 14: F2 Scores by Threshold for LGBMClassifier

Model Selection

7 different classification models were evaluated and the top 2 were optimized for final model selection. The LGBMClassifier model with POS scale weight of 10 performed the best with an F2 Score .607. The False Omission Rate has improved significantly from **20.16%** in the naive Dummy Classifier to **5.53%**.

As shown in Figure 15, the final model over predicts charge offs to achieve maximum performance. The model is not the strongest but in the final stage I will evaluate 3 different investment scenarios to determine if it is a viable solution for my problem.

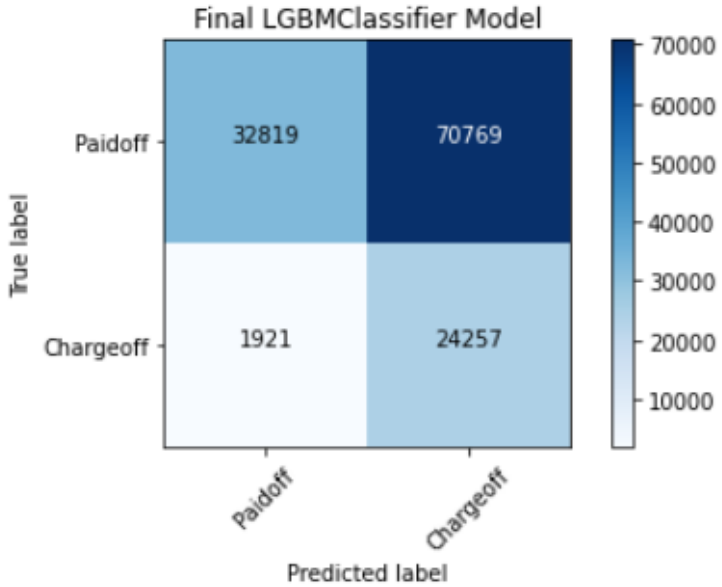

Figure 15: Final LGBMClassifier Confusion Matrix

Investment Scenario Analysis

The core data file has information for the total principal paid, interest paid and debt sale recovery for every loan. As a result, I can simulate investment scenarios and to back-test the rate of return for the selected investments. I will evaluate 3 different strategies:

1. Random

Random selection is a simple strategy where investments are selected randomly.

2. Rule-Based

Rule based could implore different criteria but to keep it simple I will only select investments where prosper rating = AA.

3. Loan Scorecard

The Loan Scorecard will utilize the final model predictions to score each loan. I excluded loans that the model predicted charge off and then created a weighted score based on interest rate and prediction probabilities.

For each scenario I selected 40 loans from the same period in the test data. I selected loans originated in the month of January 2018 with a term of 36 months. The core data file was obtained in January 2021 so all loans if held to maturity would have paid off prior to the data being acquired. Figure 16 summarizes the results for each scenario.

Scenario	Average Interest Rate	Charge Off %	Average Rate of Return
Random	15.77%	20%	-1.69%
Rule-Based	6.18%	5%	1.71%
Loan Scorecard	12.6%	5%	7.25%

Figure 16: Investment Scenario Comparison

The random selection scenario has the highest interest yield but due to high charge off % this strategy ends of losing money with a **–1.69%** rate of return. The rule-based selection only selects higher quality loans, so the interest rate yield decreases significantly and improves the charge off losses and rate of return to **1.71%**. The Loan Scorecard combines model predictions with a weighted scoring system improves on rule-based selection by maximizing interest rate yield but minimizing charge off %. This is the optimal scenario with a **7.25%** rate of return.

My original objective was to achieve a **6.5%** rate of return, so this is certainly promising. I will need to evaluate the results over a longer period, but I am going to tentatively conclude that this project was a success!

Future Improvements

Prosper's switch to TransUnion was the biggest constraint for this project. I needed more historical data for closed loans for model training and as a result, I could not utilize ~91% of the credit bureau features. In addition, key columns such as FICO score due to many missing values. Once I can gather a larger population of closed loans that were evaluated with the TransUnion scoring system, I will be able to retrain the model with the additional features.