Service Analytics - Prosper Replication

Shaan Kohli, Ramy Hammam, Shaher yar Jahangir, Noah Mukhtar

Table of Contents

Introduction

The main goal of this report is to present a replication of a Data-Driven Investment Strategy for Prosper Marketplace that takes a holistic overview from processing the raw data to inferring business outcomes. The field revolves around P2P lending and aims to create a predictive model that acts as a prescriptive tool that directly portrays the impact on the investor & business.

Our report will be split into six parts: Introduction & Objectives, Ingestion & Cleaning, Data Exploration, Predictive Models for Default, Investment Strategies and Optimization. We will be following the perception of investors seeking to diversify their investment portfolio away from traditional norms in search of higher returns.

Background on P2P Lending

Small businesses make up 97.9% of the local economy, and they are now enjoying a very competitive marketplace with an unlimited array of financing firms competing for their business. Whilst banks have acted as the main provider of lending, small businesses needs are evolving and now come with more concerns regarding the rise in operating costs, and fluctuations in demand for their products and services. These are some extreme challenges they face when it comes to raising their capital – a crucial element of their growth, but banks have been overlooking these requirements opening up the pathway for P2P Lending to grow.

The alternative finance market grew up 159% from $334.5m in 2016 to $867.5 in 2017, with this growth trajectory highly likely to continue according to analysts, which is only being exacerbated by tightening credit standards by banks, and the growing market demand for quick & easy funding, along with the broader awareness of these alternative financing products.

Prosper vs. LendingClub

Prosper is a San Francisco based company in the P2P lending industry and offers loans ranging from $2,000 to $40,000 as opposed to LendingClub's $1,000 to $40,000. Additionally, it has a slightly slower turnaround to funding as opposed to LendingClub. Prosper also offers higher interest rates than LendingClub but along with that also charges higher fees, and overall seems to be harder to get a loan from as they have higher credit score requirements.

Therefore, the ideal lender may depend on what the investor's objectives, and we will explore this in detail with our three approaches later in the report.

Study Objective

The aim of this study is to replicate the analysis using LendingClub loan data performed in the "Data-Driven Investment Strategies for Peer-to-Peer Lending: A case study for Teaching Data Science" by Maxime C. Cohen, & C. Daniel Guetta, Kevin Jiao and Foster Provost (referred as the "LendingClub Case") using an alternative peer-to-peer platform known as Prosper. **Our goal is to predict which loans are more likely to default and predict the amount value of the returns.**

The dataset utilized for this analysis was extracted by the Prosper Loan Data API (https://www.prosper.com/investor/marketplace#/download) for the years ranging from 2013 – 2018. A total of 6 excel files were imported, each representing a year of gathered loan data.

Data sets and descriptive statistics

Columns and predictors

As mentioned earlier, the datasets to be utilized in this analysis are publicly available online to download using a registered Prosper account in a similar fashion to LendingClub. The datasets contain comprehensive information on all loans posted annually and a set of 22 features or properties that cover different attributes of an individual loan. These include but are not limited to the loan amount, the amount of interest & fees paid, loan duration, a Prosper rating score, and loan status with reason if completed or defaulted.

When compared to the LendingClub case, the Prosper dataset contains data in a yearly format rather than quarterly and has a much lower number of features of 22 against the 100 and above. In addition, if we group the quarterly data of the LendingClub case into a yearly format, the datapoints highly outweigh that of the Prosper dataset by approximately 4 folds.

The definition of each loan status as seen in the loan_status (LendingClub) and loan_status_description (Prosper) column in both datasets are quite similar. However, the LendingClub loan_status variable has 7 possible outcomes when compared to 4 in the prosper dataset. The 4 loan outcomes in Prosper are given as: Completed, Charged Off, Current, Defaulted.

To simplify our Prosper data analysis we decided to focus on the rows with only **completed** loans by assigning a value of **0** to completed loans and 1 to the remaining 3 types of loan statuses (which refer to loans that have not been completed).

Case Analysis

Part I: Introduction and objectives

Investor Decisions on Prosper

Two fundamental steps are required before a loan investment can be made. Like that in the LendingClub case, an investor on Prosper will also have to decide on how much money to invest (considering potential return) and allocate to other options for investment. In addition, the investor will need to decide on picking the "good" loans (loans which will end up getting completed) in which to invest their money and this is what our loan classification will focus on.

The overall aim is for a potential investor to make as much money or "markup" as possible on their investment. To get a better understanding of which loan to invest in, the investor should be able to split their decision into two parts. The first highlights the importance to decide (which loans to invest in) and the second part is to evaluate them (how much money could they possibly earn).

However, it is not as simple as it seems to estimate or justify how much money investing in a specific loan would make based on the certain criteria given such length of loan, time of default, and the possibility of an early repayment. Therefore, it is important to note that there is no one ideal investment, and it is highly contingent on the risk appetite of the investor.

Use of past data to Predict the future

Likewise, like the LendingClub case, the potential investor should leverage past data to figure out what predictors are significant in indicating what makes a **good loan** to invest in. Moreover, it is important to realize the correlation & interaction amongst these variables in order to also acquire an intuition of identifying good loans.

However, the definition of a good loan is quite ambiguous so it is vital that the prediction should be based on one of these goals: whether a loan will default, paid back early, time of default or time taken to pay back if paid back early.

Prosper Data description

As seen in the prosper excel files (2013,2014,2015,2016,2017, and 2018) each of the 22 predictors can be characterized as one of the following data types: Integer, Float/Decimal, Categorical/String, Binary (True, False) and Date. Figure 1 shows a list of all the predictors in each of the 6 Prosper excel files with a data description and the data type. There exists a similar scenario just like the LendingClub case where some predictors are updated annually, and some are held constant throughout. These predictors should be addressed and could be detrimental when building a model in the future.

It is also worth noting that there is no variable in the Prosper dataset that indicates the total payment or return on each of the loan. Therefore, a lot of our data prep will revolve around replicating these missing variables. The base return can be easily calculated by summing up the principal paid amount with the interest paid. In the following sections, we will be calculating the return on all the loans at a more granular level bucketed into 1 of the 3 categories Pessimistic, Optimistic, Fixed-time Horizon at various % interest.

Predictor Name	Predictor Type	Description
loan_number	Integer	The identifier of the loan
amount_borrowed	Float/Decimal	The amount that was borrowed for the entire loan
borrower_rate	Float/Decimal	The borrower rate for this loan
prosper_rating	Categorical/String	Prosper rating of the listing for the loan at the time the listing was created.Possible values (Highest to Lowest): **AA,A,B,C,D,E,HR**
term	Integer	The term of the loan in months.Possible values: **36 , 60**
age_in_months	Integer	Age of the loan in months
origination_date	Date	The date the loan originated (was funded)
days_past_due	Integer	The number of days this loan is past due
principal_balance	Float/Decimal	The principal amount outstanding on this loan
service_fees_paid	Float/Decimal	The current amount of service fees paid on the loan
principal_paid	Float/Decimal	The current principal amount that has been repaid toward this loan
interest_paid	Float/Decimal	Total current interest paid on this loan
prosper_fees_paid	Float/Decimal	The current amount of Prosper fees paid, including failed payment and alternate payment convenience fees
late_fees_paid	Float/Decimal	The current amount of late fees paid on this loan
debt_sale_proceeds_received	Float/Decimal	Money received as the result of selling post-chargeoff balances to a collection agency
loan_status	Integer	Ranges from 1:6 0 = ORIGINATIONDELAYED, 1 = CURRENT, 2 = CHARGEOFF, 3 = DEFAULTED, 4 = COMPLETED, 5 = FINALPAYMENTINPROGRESS Deprecated, 6 = CANCELLED
loan_status_description	Categorical/String	Text description for the loan status listed above.
loan_default_reason	Integer	Reason for Default 1 = Delinquency, 2 = Bankruptcy, 3 = Deceased, 4 = Repurchased, 5 = PaidInFull, 6 = SettledInFull, 7 = Sold, null = not in default
loan_default_reason_description	Categorical/String	Text description for the loan default reason listed above
next_payment_due_date	Date	The next scheduled payment date
next_payment_due_amount	Float/Decimal	The amount due on the next payment due date. This can vary to include items such as late fees or a different last payment
co_borrower_application	Binary	FALSE, TRUE

***Figure 1**: Prosper Data Description*

Prosper Data Leakage

Just like the LendingClub dataset, the concept of data leakage should also be addressed. Data Leakage occurs when a model is built using predictors that will not be available at the time of future predictions. It is extremely vital to tackle leakage in order to refrain from producing a model that shows misleading accuracy scores which would catastrophically fail to replicate its initial scores when deployed in real time, as most of its feature selection would be on data that wouldn't be available.

There are two main cases on how leakage can be identified:

1. A predicator that is highly correlated with the target variable. This can be checked using a heatmap or correlation matrix of all 22 predictors.
2. The predictor information is not available during the prediction of total return. An example of this is the late_fees_paid variable.

Based on this definition of leakage our team decided to identify and limit the use of highly correlated predictors such as:

1. loan_number
2. late_fees
3. age_in_months
4. days_past_due
5. origination_date
6. principal_balance
7. principal_paid
8. interest_paid
9. late_fees_paid
10. debt_sale_proceeds_received
11. loan_default_reason
12. loan_default_reason_description
13. next_payment_due_date
14. next_payment_due_amount
15. co_borrower_application

All 15 predictors tend to be highly correlated with our target variable and not be available when predicting whether a loan will default or not.

Prosper Rating Leakage

As stated earlier, another type of leakage that can occur is when the value of the Prosper rating can be updated overtime leading to a high correlation with our outcome variable. This is equivalent to the fico_range_low variable in the LendingClub case. To illustrate, if a loan defaults, the borrowers Prosper score may decrease in the future, setting a pattern that could lead to a high correlation between the target variable (loan status) and the Prosper rating. This leakage can be quite hard to detect and will be further explored in a later section of this report.

Part II: Data ingestion and cleaning

Data Import (Refer to the Service_Analytics_Data_2013_2018_complete.ipynb notebook for detailed code)

First, data preprocessing and visualization packages such as NumPy, Pandas, Matplotlib and Seaborn were imported into our notebook.

Next, all six excel files that contain prosper loan data for years 2013-2018 were imported using the pandas read function and assigned to data frames respectively. The six data frames were easily concatenated into a single data frame called **"df".**

Like the LendingClub case data importing format, each excel file follows the same structure, contains a set of primary keys (loan_number) that are unique and sequential.

Binary Transformation of "Completed" Loans

In a similar fashion to that of the LendingClub case, our df was filtered out to include only loan statuses belonging to "Completed", "Defaulted", "ChargedOff" and "Cancelled" state. The loan_status_description column that contains the statuses was then transformed into a binary variable where **0** represents "Completed" loans and **1** represents all uncompleted (defaulted) loans (the remaining 3 statuses: "Defaulted", "ChargedOff" and "Cancelled"). This target variable will be used at a later stage to classify the probability of a loan being defaulted.

Term Selection

The term selection chosen when subsetting the Prosper dataset is greater than or equal to 36 months. This was done because there some loans in the Prosper dataset that only ran for 12 months. To make sure that the replication study is consistent, the term duration selected for Prosper analysis was made equivalent to that of the LendingClub case. This filter was added early on while completing the binary transformation of the completed loans.

Data Leakage Variables Removal

After creation of the binary transformed column "loan_status_description", leakage associated variables such as loan_status, loan_default_reason and loan_defeault_reason_description were dropped from our dataset.

Creation of New Variables

As mentioned earlier in Part 1, the creation of new variables is needed to maintain analysis consistency and utilize in the regression model building stage to predict the total return payment. To do so, our team created a "total_payment" column that is simply the sum of the principal paid with the interest paid. An assumption was made to simplify our understanding and avoid insights confusion by renaming and assigning the "principal_paid" column to "recoveries".

Next our team worked on calculating the actual length of a loan to build a new predictor. To achieve this, the loan start and end date were converted into the datetime format and renamed to "issue_d" and "last_pymnt_d". To prevent anomalies from entering the dataset, the df was filtered out to ensure that the closed date length is always higher than the issue date.

A simple calculation was performed to build the "loan_length" variable by subtracting the last payment date by the issue date. This loan length variable was converted to integer and any loan with length equal to 0 was dropped.

To check if a loan is uncompleted or not, the next payment due date should be positive. This also verified that all loans would have been "expired" like the LendingClub case.

The "loan_status_description" and "prosper_rating" variable was also renamed to "loan_status" and "grade" for easier insight understanding and comparison to the LendingClub case.

Column type Buckets were also built to easily identify which columns will be kept and which will be dropped from the Prosper dataset and the data type of each variable was outputted as seen in Figure 2.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 735256 entries, 1 to 198753
Data columns (total 22 columns):
 #   Column                       Non-Null Count   Dtype
---  ------                       --------------   -----
 0   loan_number                  735256 non-null  int64
 1   amount_borrowed              735256 non-null  float64
 2   borrower_rate                735256 non-null  float64
 3   term                         735256 non-null  float64
 4   age_in_months                735256 non-null  int64
 5   days_past_due                735256 non-null  int64
 6   principal_balance            735256 non-null  float64
 7   service_fees_paid            735256 non-null  float64
 8   principal_paid               735256 non-null  float64
 9   interest_paid                735256 non-null  float64
 10  prosper_fees_paid            735256 non-null  float64
 11  late_fees_paid               735256 non-null  float64
 12  debt_sale_proceeds_received  735256 non-null  float64
 13  next_payment_due_amount      735256 non-null  float64
 14  co_borrower_application      735256 non-null  float64
 15  total_pymnt                  735256 non-null  float64
 16  recoveries                   735256 non-null  float64
 17  issue_d                      735256 non-null  datetime64[ns]
 18  last_pymnt_d                 735256 non-null  datetime64[ns]
 19  loan_length                  735256 non-null  int64
 20  loan_status                  735256 non-null  float64
 21  grade                        735256 non-null  object
dtypes: datetime64[ns](2), float64(15), int64(4), object(1)
memory usage: 149.0+ MB
```

***Figure 2**: Column Names and Dtypes*

Missing Value Check

Two columns in the Prosper dataset called "loan_default_reason" and "loan_default_reason_description" contained missing values and were dropped early on to ensure the removal of NA values. By dropping the high-volume null value columns, we prevent a substantial loss of information effect on the dataset by removing row-wise null values. This is a common approach in data science practices.

Correlation Check

Next, we check the level of correlation that may exist between all the predictors and the loan status (target variable). None of the predictors seemed to have a very highly positive or negative magnitude of correlation.

To compliment this analysis and provide a better visual, a heatmap was also produced (Appendix Exhibit 2). Variables that are associated with data leakage such as the interest paid, and principal paid stood out to have high correlation with loan status.

Distribution of Interest Rates for Completed and Default Loan Status

A density plot (Appendix Exhibit 3) with 2 lines representing each loan status outcome was built for the different interest rates associated with loans. The figure shows that interests rates for borrowers who **did not** default (completely paid back the loan) were denser on the lower interest rates side than those that did default (uncompleted loan payment). This finding is consistent with that of the LendingClub case.

Loan amounts in Completed vs Default loan status per Term

A violin plot was constructed (Appendix Exhibit 4) that highlights the difference in the value of the loan amount for each completed and uncompleted loan status for the two terms (36 months and 60 months).

The plot indicates some interesting insights for the 60-month term (the longer term) such as the highest loan amount being approximately 15,000 dollars for both completed loans as well as uncompleted loans. Borrowers tend to take higher amount loans to pay back over the longer-term.

Whereas loans in the 36-month terms (the shorter term) tend to be of much smaller amounts and is consistently distributed for both completed and uncompleted loans.

However, it is important to note that the average loan amounts for Prosper is higher than that of LendingClub's and may explain why Prosper has higher credit score requirements of 640 compared to LendingClub's 580 in their loan application requirements.

Interest Rate distribution per loan status for each type of grade

Another violin plot was built (Appendix Exhibit 5) that shows the different levels of interest rates for each of the loan status per grade type (ranging from AA – HR) (highest grade to lowest).

High grade loans/borrowers for both completed (0) and uncompleted (1) have lower interest rates. The trend is consistent for both loan statuses and is the same for the LendingClub case.

Additionally, Prosper has higher interest rates than LendingClub for most years, and while LendingClub has come close, it has yet to surpass Prosper's rates.

Part III: Data exploration

Calculation of the types of possible returns

The most significant data required in determining the potential return from a given loan is the calculation of the total payments that were received on each of the loans.

To build an effective investment strategy, a strong indicator variable on the return amount of each loan must be built. Just like the LendingClub case, it is vital that the return should consider partially paid off defaulted loans and loans that have been paid off earlier than the due date.

In general, there are three effective return measures that can be created using the following variables:

1. Total payment account denoted by variable **p**
2. Total amount invested in the loan (**amount borrowed by borrower**) denoted by variable **f**
3. Nominal length of the loan in months (**term of loan**) denoted by the variable **t**
4. Actual length of the loan in months (**loan length**) denoted by the variable **m**

Like the LendingClub case, the Prosper dataset has been preprocessed to include all the 4 variables (p,f,t,m) needed to create the 3 effective return measures which are: Pessimistic (M1), Optimistic (M2),
Fixed-time Horizon based on different interest rate percentages: 1%,2.5%, 6% (M3)

Pessimistic Approach (M1)

The pessimistic approach states that when the loan is paid back, the investor cannot re-invest it until the term of the loan.

$$\frac{p-f}{f} \times \frac{12}{t}.$$

***Equation (1)**: Pessimistic Approach Formula*

Using equation 1, our team was able to create a new variable called: “ret_PESS”.
It was worth noting that this approach favors long-term loans for loans that default early due to the loss spreading over a greater span.

Optimistic Approach (M2)

The Optimistic approach states that once the loan is paid back, the investor’s money is returned, and the investor can immediately invest in another loan with the same return.

$$\frac{p-f}{f} \times \frac{12}{m}.$$

***Equation (2)**: Optimistic Approach Formula*

Using equation 2, our team was able to create a new variable called: “ret_OPT”.

This method can be rephrased as the annual monthly return of the loan over the time it was active. However, it is plagued by the assumption that money can be reinvested with the same rate and that if the loan is defaulted early,

the loss can be huge overestimate of the negative returns. On the plus side, short and long loans are treated equally using this approach.

Fixed-time Horizon return based on different interest rate percentages: 1%, 3%, 6% (M3)

The Fixed-time Horizon approach led to 3 new variables being built "ret_INTa", "ret_INTb", "ret_INTc" by calculating the fixed-time horizon return for 3 interest rates: 1%, 2.5%, 5% respectively.

$$\frac{12}{T}\cdot\frac{1}{f}\left\{\left[\frac{p}{m}\left(\frac{1-(1+i)^m}{1-(1+i)}\right)\right](1+i)^{T-m}-f\right\}.$$

***Equation (3)**: M3 Approach Formula*

A function was built, called ret_method_3(), to replicate equation 3 and calculate the 3 new variables for each loan in the dataset.

This approach can be the most accurate by equalizing the differences between loan length and defaulted loans. However, it tends to disregard the depreciation value of the money over time.

Variables Visualization & Outlier Removal

To identify if there are any outliers present in our dataset, like the LendingClub case, a variety of plots were built for the different predictors that exist in the Prosper dataset.

- Box-and-whisker plots for continuous variables
- Lists of distinct values for categorical columns
- A timeline density for dates

Using the visualize_columns() function, the respective graphs were plotted for each variable. All the visualization plots are attached in (Appendix Exhibit 6). Present outliers were identified by being out of the upper and lower limit of the box-and-whisker plots, and were removed.

Prosper Rating (Grade) Breakdown

As seen in the LendingClub case, a deeper dive into the prosper rating ("grade") variable was performed to explore any significant trends with each of the type of return (M1-M3). Figure 3 shows the summary statistics.

				Mean return			
Grade	% of loans	% Default	Av. interest	M1	M2	M3 (1.2%)	M3 (3%)
A	16.68	6.33	7.22	1.66	3.89	2.05	3.71
B	28.86	13.48	10.85	1.58	5.01	2.02	3.68
C	27.99	22.41	14.07	0.62	5.39	1.39	3.02
D	15.44	30.37	17.54	0.05	5.71	0.92	2.51
E	7.59	38.83	20.73	−0.91	5.95	0.11	1.64
F	2.72	44.99	24.47	−1.43	6.43	−0.44	1.05
G	0.73	48.16	27.12	−2.58	6.66	−1.40	0.05

***Figure 3**: LendingClub Grade Summary Statistics*

The LendingClub case states that indicates that the lower the grade, the higher the probability that the loan will default and the higher the interest rate LendingClub charges.

Calculating the grade summary statistics for the Prosper Dataset can be seen in Figure 4.

	grade	default	int_rate	return_PESS	return_OPT	return_INTa	return_INTb	return_INTc
A	19.973938	8.612483	9.474686	2.274137	4.541028	2.254222	3.486214	5.706550
AA	10.377824	4.014627	6.775790	2.019641	3.706281	2.023254	3.275717	5.534159
B	22.979313	12.956079	12.394934	2.395357	5.703969	2.466247	3.724494	5.997924
C	24.597005	18.515979	16.501650	2.495612	7.242789	2.583270	3.886307	6.246494
D	12.239538	23.292177	22.047630	2.688308	9.355363	2.692042	4.072759	6.581256
E	7.167875	25.417007	27.214131	3.449610	11.314350	3.154841	4.604898	7.242146
HR	2.664507	25.881872	30.950736	4.027710	12.390232	3.372662	4.887931	7.647172

***Figure 4**: Prosper Grade Summary Statistics*

The prosper dataset also shows a similar trend to that of the LendingClub case, where the higher the grade, the smaller percentage of defaulting and the lower the interest rates.

One key difference between the two datasets is that the Prosper “grade” summary statistics show that the mean return % are very identical with minor increases across the different grades when compared to that of the LendingClub case. In addition, there are no negative mean return values across any of the grades.

A thorough analysis of the prosper grade breakdown reveals that numbers are like that of LendingClub across the different return measures but scaled relatively higher in. This is consistent with Prospers business model of providing the highest returns to investors, but also charging high amounts of service charges and fees. What we also notice is that the % of defaulting loans by grade is much higher for LendingClub in comparison to Prosper.

Tackling the Balanced Dataset Option

A quick count of the number of loan status for each type shows a significantly higher number of completed loans when compared to uncompleted loans as seen in Figure 5. This indicates an imbalance in the dataset, so the most probable option is to try to rebalance the dataset to consist of 50% of completed loans and 50% of uncompleted loans. However, rebalancing the dataset can prove to be detrimental and cause the average return to be negative in some cases (Appendix Exhibit 10).

This phenomenon is occurring since there are equal amounts of borrowers that have defaulted which means that the expected return for the investor may become negative (more losses on investments than gaining money) over time. This goes completely against the objective of a potential investor signing up on Prosper or LendingClub to make money. To capture actual real-life trends and true proportion, physically balancing the dataset can hinder an effective understanding of the analysis.

```
0.0    624679
1.0    110504
Name: loan_status, dtype: int64
```

***Figure 5**: Label counts for Loan Status*

The LendingClub case discusses this balancing option and for the same above reason decides to continue with the analysis using an unbalanced dataset. A calibration curve measure will be used in the model building phase to check if a constructed model is biased by the data imbalance or not.

Part IV: Predictive Model Analysis

The predictive modeling will cover a two-stage approach. First, we will be constructing a binary classification model to predict the loan default probability using a variety of industry standard algorithms and evaluating the performance for the optimal and most accurate model. The second stage involves constructing a regression model using a variety of industry standard regressors to predict the amount of return a loan may generate to an investor for each of the return approaches (M1-M3). The models in both phases will be optimized for higher accuracy by hyper parameter tuning through cross-validation.

STAGE 1: SEARCHING FOR THE OPTIMAL CLASSIFICATION MODEL

First a seed value = 1 was set to allow the results to be reproduceable every time the notebook is re-run. The exported CSV dataset was imported into a new notebook called **"Service_Analytics_Modeling_2013_2018_vf (1)".** We will refer to this as the modeling notebook **from hereafter**. An output file called "output_sample" was also created to save all our results.

Train Test Split

To train a classification model, the dataset needs to be split into a training, test, and cross-validation set. Our aim is to fit a variety of models on various number of time periods and would like every period to use the **same** training/test split.

There are various ways to achieve this; however, the random assignment of each loan into the training and testing set ended up being the better option.

This is for two reasons as outlined in the LendingClub case. First, it allows the replication study to be performed using a consistent train-test split method to better evaluate the two studies. It is also much easier to implement using standard libraries and allows thousands of train and test sets to be created with the use of different random seeds. The random method train test split percentages utilized were 60-40% respectively.

Dummifying variables and Model function definition

To be able to correctly feed a categorical variable into a classification model, all categorical predictors needed to be dummified.

Several functions such as prepare_data(), fit_classification() & fit_regression(), test_investments() were created.

The prepare_data() function will prepare the data using a variety of parameters such as the training data and testing data from the random split to allow the dataset to be fit into a classification and regression model with ease.

The fit_classification() and fit_regression() functions in the modeling notebook enables the data to be fit and build an optimized classification and regression model respectively. The function also outputs their various evaluation metrics by defining a few input parameters such as model_name and cv_paramaters and searches for the optimal hyperparameters.

The test_investments() function tests a variety of investment methodologies and their returns using the test dataset. This will be discussed at a later stage in the report.

Model Evaluation Metrics

The main metrics used in the study to evaluate the performance of a certain classification and regression model are as follows: accuracy, precision & recall, f1-score, and the AUC/ROC curve which are like that in the LendingClub case.

Accuracy

Accuracy is an industry standard metric for evaluating classification models. Generally, accuracy is the fraction of predictions the model got right. Formally, accuracy has the following definition:

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ Number\ of\ fPredictions}$$

For binary classification, accuracy can also be calculated in terms of the positives and negatives as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Where TP = True Positives, TN = True Negatives, FP = False Positives, and FN = False Negatives.

The higher the accuracy of the model, the more predictions the model got right with respect to the true value of the datapoint. However, accuracy alone is not a strong determinant for the predictive power of a binary classification model especially in the case when working with an imbalanced dataset which is what occurs in the Propser and LendingClub datasets. Precision and recall are considered to compliment the accuracy metric when modeling with class-imbalanced datasets.

Precision and recall

Precision

The Precision metric of a model aims to determine the proportion of positive predictions that were truly correct.

It can be calculated using the following equation:

$$Precision = \frac{TP}{TP + FP}$$

Recall

The recall metric of a model aims to determine the proportion of truly positive that were predicted to be correct.

It can be calculated using the following equation:

$$Recall = \frac{TP}{TP + FN}$$

In an ideal scenario, the precision and recall should be as close as possible to 1 to indicate a robust prediction model. However, the two metrics come with a trade-off situation, whereas when the precision increases, the recall decreases and vice versa. There is always a notion of finding the right balance for the two metrics depending on the aim of the use case.

The precision and recall can be dubbed as the specificity and sensitivity of a model respectively and plotted to find the optimal balance threshold.

F1-Score

The F1 score is another way to measure the test accuracy in a binary classification model. When compared to the accuracy metric, the F1 score takes a weighted average into account by using the precision **p** and the recall **r** to calculate a unique metric.

In general, the higher the F1-score the better the predictive power of the binary classification model. This is metric even more useful in the case of an imbalanced dataset which is what occurs in the Propser and LendingClub datasets.

AUC/ROC Curve

An AUC/ROC curve will also be plotted to understand the overall model performance. A ROC curve (receiver operating characteristic curve) is a graph showing the performance of a classification model at all thresholds. This curve plots the true positive rate and false positive rate. The ROC Curve that bends and is closer to the top left of the total positive rate axis is indicative of a strong model performance.

The AUC (area under curve) curve on the other hand, shows an aggregate measure of performance across all possible thresholds. One way of understanding the AUC is the probability that the classification model ranks a random positive prediction more highly than a random negative prediction. The AUC metric ranges from 0 to 1. A model whose predictions are **100% incorrect** has an AUC value of **0** and one whose predictions are **100% true** has an AUC value of **1**.

Calibration curves

The calibration of the model refers to the extent to which the probabilities predicted by the model correspond to the frequency of the event happening for some natural grouping of events. Typically, this is measured by sampling instances grouped into bins of similar estimated probabilities.

Our goal is to use the probabilities output by the classification model to predict and calculate the status and the expected return of the loans. Thus, the checking of the calibration of the estimated probabilities is important. To

check whether the model can account for the issues such as data discrimination that can occur with an imbalanced dataset a calibration curve measure will be plotted alongside. A calibration curve that has all points present or as close to the 45-degree straight line shows a well calibrated model.

Grade and Interest Classification Model

Since the grade and interest rate predictors are also present in the Prosper dataset and are strong determinants of the potential status of a loan and the amount or return, it was important to check the predicative power of the model using these two predictors solely and seperately.

To do so, likewise in the LendingClub case, two logistical classification and regression models using grade and interest predictor only were fit to determine the performance of the models.

As seen in the (Appendix Exhibit 11-12) the AUC for both the grade only and interest only models were identical at around 0.65 which is quite good, and the accuracy was at around 85% which is the industry standard for strong classification models.

It turns out that when all predictors are utilized and the grade and interest predictor are dropped, the model AUC tends to stay the same. In other words, only using these single variables provides as much predictive power as when all predictors are utilized.

Our objective is to utilize the underlying predictors and not attributes derived by Prosper and LendingClub such as grade and interest. All predictors without the grade and interest will be utilized as our features for the following classification and regression models.

This insight is consistent with that in the LendingClub case where the accuracy remains very high when predicting whether the loan will be completed or defaulted even when the grade and interest feature are dropped out of the list of features.

Next, we decided to fit a variety of different classification models ranging from: Ridge classifier, Naïve Bayes, l2 penalized logistic regression, decision tree, random forest, bagged tree, multi-label perceptron, & Light GBM.

The features that will be used are: "co_borrower_application", "term", "prosper_fees_paid", "service_fees_paid" and "amount_borrowed". Once again, these features do not possess data leakage and our considered to be well rounded predictors for our classification models.

The target variable for classification: "loan_status"

Classification Model Analysis

Classification Model	Accuracy	Precision	Recall	F1-Score	AUC	Calibration Level
Ridge Classifier	0.8489	0.8489	1	0.9183		
Naive Bayes	0.8489	0.8489	1	0.9183	0.65	Medium
l2 penalized logistic regression	0.8491	0.8493	0.9996	0.9184	0.64	Perfectly Calibrated
Decision tree	0.8489	0.8489	1	0.9183	0.71	Perfectly Calibrated
Random forest	0.8498	0.8629	0.9785	0.9171	0.74	High
Bagged trees	0.85195	0.8573	0.9905	0.9191	0.74	High
Multi-layer perceptron	0.85075	0.8566	0.99	0.9184	0.73	High
Light GBM	0.85185	0.8566	0.9914	0.9191	0.74	High

***Table 1**: Classification Model Comparisons*

Table 1 depicts the summary of the classification model metrics for all the tested types of models (Appendix Exhibit 13-22).

As mentioned in the prior section, the accuracies, precision, recall and f1-scores are very high without the inclusion of interest and grade variable for all the models above. Since all the models gave a very high similar accuracy, we picked random forest classifier as our optimal model to incorporate into the upcoming investment strategy section for comparability reasons with the LendingClub case.

In addition, what is also interesting is that the calibration curves and AUC scores for all the above models are high indicating strong performance. The calibration code shows a measure to see if the model is affected by the imbalance. If the calibration is perfect, it means the model is not biased by the imbalanced dataset. In other words, it will not always try to predict the outcome to be the grand majority which is the completed loans **(0)** case.

Calibration curves generally aim to answer the following question: How well does your algorithm perform with an imbalanced dataset?

Since Random forest was the best selected model, we decided to explore the best in class for tree algorithms also known as Light GBM classifier. It is considered the latest industry trend and winner of multiple Kaggle completions, developed by Microsoft, so we were curious to see how well it would perform in the Prosper case. As presumed, all the model metrics were very high for the Light GBM classifier.

Feature importance

The feature importance plots for most of the models were also generated in the classification stage. In the random forest model, the feature importance can be seen in (Appendix Exhibit 19). Service fees paid, Propser fees paid and amount borrowed were termed by the model to be the most useful at predicting our target variable: "loan_status" which matches well with investing intuition.

As a side note, Prosper offers a lower minimum fee rate of 1% compared to Prosper's 2.41%, but LendingClub offers a higher maximum fee rate of 6% as opposed to Prosper's 5%.

STAGE 2: REGRESSION MODEL ANALYSIS

A variety of regression models were built in a similar fashion to the Stage 1 approach (Appendix Exhibit 23-27). The objective of building the regression models was to predict or calculate the expected return for the different return approaches (M1-M3). The main metric to be used when comparing multiple regression models was the R^2 value.

The R^2value (coefficient of determination) is a statistical measure of fit that shows how much variation of a target variable is explained by the independent variables in a regression model. In general, the higher the R^2 value when comparing across models, the better the fit of that model compared to the others.

The features that will be used are: "co_borrower_application", "term", "prosper_fees_paid", "service_fees_paid" and "amount_borrowed". Once again, these features do not possess data leakage and our considered to be well rounded predictors for our classification models.

In the regression stage, the target variable is broken into the 4 types of return possible and have the regression model run on each of them separately: Pessimistic Approach (M1), Optimistic Approach (M2), ret_INTa (M3), ret_INTb (M3)

We did not use the ret_INTc variable since our objective was to just capture the occurrence of the increase in the interest rather than the actual increase by value.

The best regressor model with the highest R^2 value for all 4 types of returns in the Prosper dataset was the Random Forest Regressor as seen in Table 2.

	R^2 *scores for Each return definition*			
Model (Regressor)	*M1: Pessimistic*	*M2: Optimistic*	*M3: (1.2%)*	*M3: (3%)*
Ridge	0.176	0.025	0.186	0.153
Ordinary Least Squares	0.176	0.025	0.186	0.153
Multi-Layer Perceptron	0.239	0.029	0.255	0.175
Random Forest	0.313	0.066	0.288	0.245

***Table 2**: Prosper Regression Model Comparisons*

The Prosper regressor model results and trend coincides like the one in the LendingClub case as seen in Table 3. In both platforms, the Random forest regressor gave the highest R^2 value across all the different types of returns when compared to the rest of the models built. It is worth noting that we utilized a few different types of regressor models than those in the LendingClub case.

	R^2 *scores for each return definition*			
Model	*M1*	*M2*	*M3 (1.2%)*	*M3 (3%)*
ℓ_1 regressor	0.026	0.012	0.027	0.027
ℓ_2 regressor	0.026	0.013	0.027	0.027
Multilayer perceptron regressor	0.016	0.004	0.017	0.017
Random forest regressor	0.028	0.018	0.029	0.032

***Table 3**: LendingClub Regressor Results*

Learning Curve

A learning curve was also plotted to assess learning rate of the optimal random forest models when compared to a sample L2 logistic regression model. The logistic model decreases in Prosper and increases in LendingClub. As a result, the learning curve insights do not completely match between LendingClub and Prosper. However, another trend emerges where, when the training size is increased by intervals of 25 datapoints until a very large amount, the random forest model AUC increases meaning that the model will continue to learn and give a much higher percentage of accurate return predictions. For example, at training size of 12000 loan datapoints, the AUC peaks at 0.75.

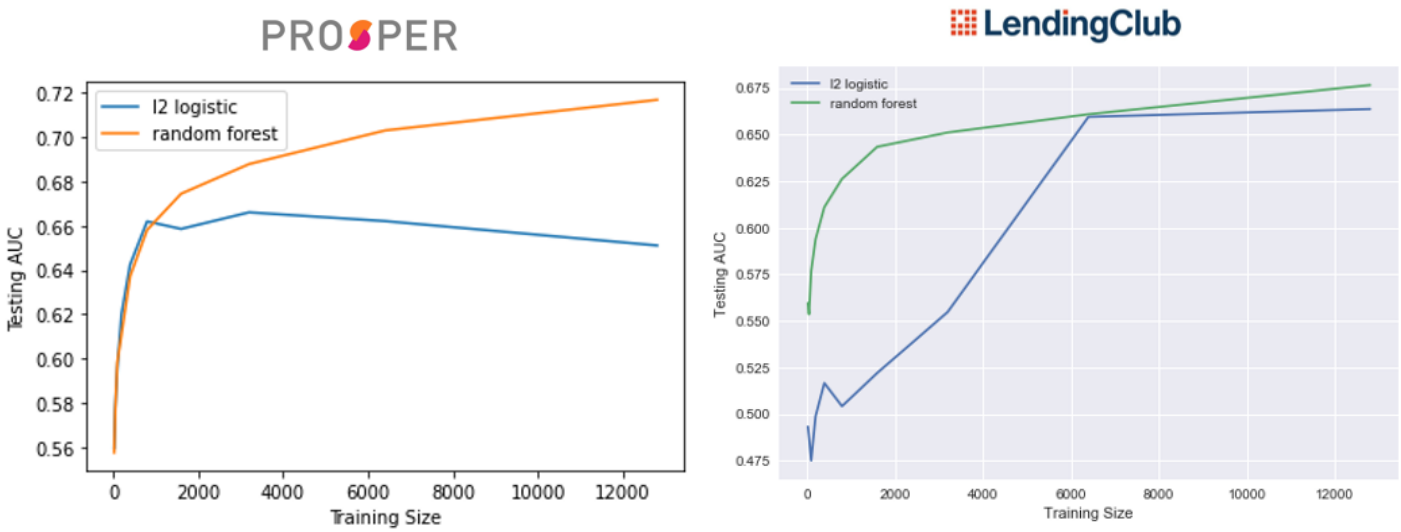


***Figure 6**: Learning Curve Comparisons*

Model Stability Over Time

To verify the stability of the regressor and classification results over different periods of time ranging from 2013 -2018, a variety of different train/test splits were utilized like the LendingClub case.

Timeframe 1: (Appendix Exhibit 28) shows the model being trained and tested using the whole time period and the model performance metrics.

Timeframe 2: (Appendix Exhibit 29) shows the model being trained and tested on the first two years.

Timeframe 3: (Appendix Exhibit 30) shows the model being trained on the first two years and tested on the last year. In the Prosper case, we utilized the loan datapoints of years 2013-2015 (2 years) as our training set and then tested the model using the 2018-2019 loan datapoints (1 year).

The results indicate that there is a strong drop in the performance for the model when the dates are changed to timeframe 3. The AUC worsens to around 50 percent and the model is highly uncalibrated. This is in complete contrast to that in the LendingClub case which mentions that the Random forest model's performance is remarkably stable. Thus, there is strong evidence that Prosper had a very large change in business operation anywhere between 2013 and the end of 2018.

Part V: Investment strategies

Types of Strategies and Analysis

By leveraging both the classification and regression models to predict the probability of whether a loan will be defaulted and the estimated return, an investment strategy can be formulated to maximize an investor's average return.

In a similar fashion to that of the LendingClub case, 4 investment strategies were explored using the Prosper models.

The strategies are outlines as follows:

1. Random strategy (Rand) involves the investor randomly picking loans to invest in.
2. Default-based strategy (Def) involves using the random forest default-predictor models, sorting loans by their predicted default probabilities, and selecting the loans with the lowest probabilities.
3. Simple return-based strategy (Ret) involves training a simple regression model to predict the return on loans directly. The loans can then be sorted by their predicted returns and selecting the highest predicted return loans.
4. Default- and return-based strategy (DefRet) involve training two additional models to predict the return on loans that did not default, and one to predict the return on loans that did default. Then, using the probability of default predicted by the random forest model above to find the expected value of the return from each future loan and invest in those with the highest predicted loans.

The LendingClub investment strategy results for the various types of returns can be seen in Table 4

	Return calculation method			
	M1-PESS, %	*M2-OPT, %*	*M3 (1.2%)*	*M3 (3%)*
Rand	0.6	5.2	1.2	2.7
Def	1.9	5.3	1.4	3.0
Ret	2.6	5.6	1.6	3.2
DefRet	3.0	5.7	1.7	3.3
Best possible	12.0	27.1	10.1	11.7

***Table 4**: Investment Strategy Return Percentages for LendingClub*

As mentioned in the LendingClub case, the two-stage analytical engineering strategy performed the best and the investment returns are positive in all strategy cases. All other strategy methods outperform the random investment strategy given that LendingClub's ability to set interest rate to offset the default risk.

Investment returns continue to improve as the strategy becomes more sophisticated and especially when the return measure is the optimistic approach. The best possible row corresponds to the top 1000 loans that could be invested in and the percentage of return.

In comparison, the Prosper investment strategy return results are shown in Table 5.

Investment Strategy	Pessimistic Return (M1) %	Optmistic Return (M2)%	M3 (1.2%)	M3 (3%)
Rand	1.5	5.7	2.5	3.4
Def	7.3	7.5	2.7	3.2
Ret	1.1	7.3	2.6	3.5
DefRet	1.2	6.3	2.7	3.3

***Table 5**: Investment Strategy Return Percentages for Prosper*

Optimal Strategy for Prosper Analysis

Solely because the % returns under the **Ret** investment strategy in Table 5 for the M2 approach are higher does not necessarily imply that it is a viable option to consider since it does not account for many of the investor's decisions. Rather, we would want to consider the optimistic return and even more likely the M3 approach.

In order to depict real life scenarios, the M3 return approach seems to be a more accurate way to measure the % return when compared to LendingClub's M2 optimal return approach. Thus, the baseline for us to compare the different investment strategies would be under the M3 method. Nonetheless, the returns seem to be very close to one and another regardless of which investment strategy is being used when looking at the M3 return approach.

The magnitude of the % return is also highly dependent on the reinvestment interest rate (M3). This trend is consistent with the LendingClub case results in Table 4 because all the expected % return values are higher with a higher interest rate % regardless of the investment strategy and is very close to one and another in both cases.

A violin plot was also constructed to visualize the R^2 distribution for different investment types as seen in Figure 6. For example, the blue violin shows the variation in R^2 for pessimistic return method across different models. The models perform substantially different when using the pessimistic return method. On the other hand, we see very little variation for the optimistic return method (orange violin).

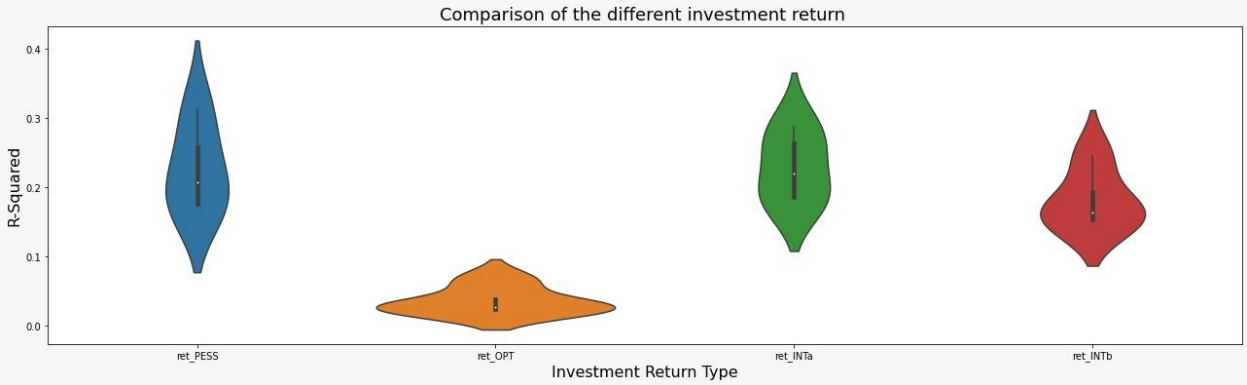


***Figure 7:** R^2 distribution for each Investment Return Type*

Sensitivity test of Portfolio Size

Next, a sensitivity test was run with regards to the portfolio size versus the investment return percentage. Figure 7 shows the sensitivity test results for the Prosper and LendingClub case.

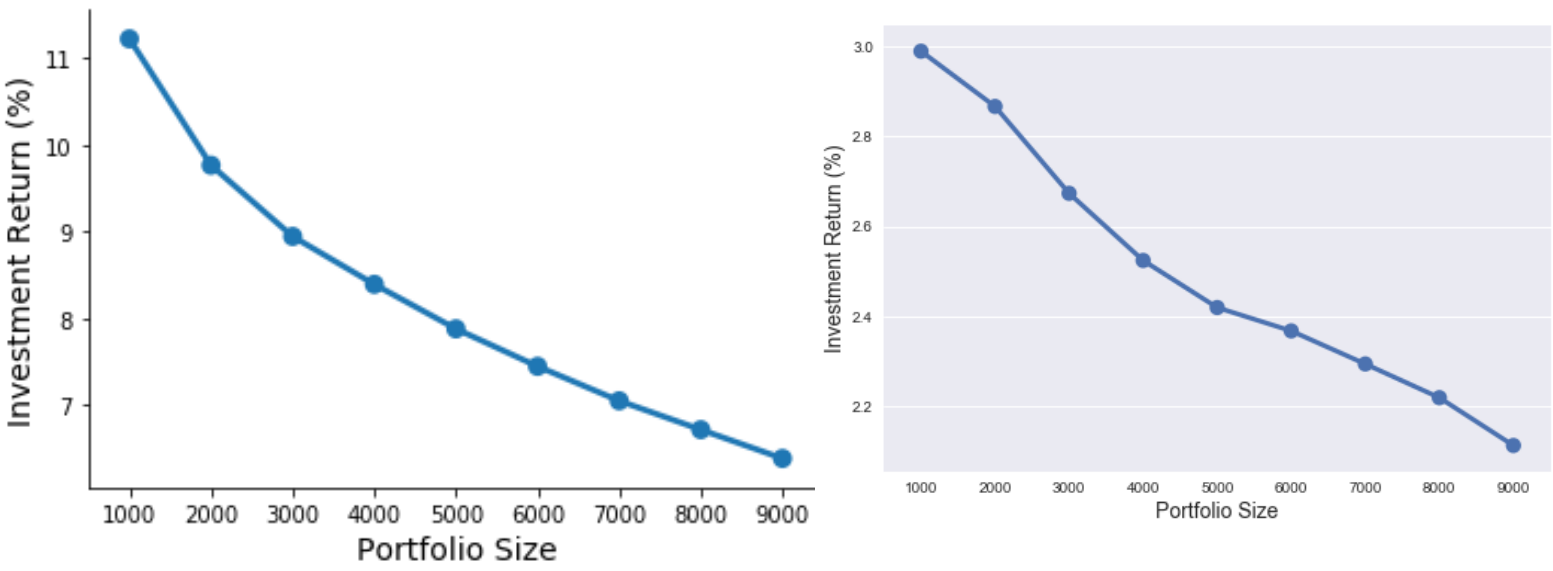


***Figure 8**: Prosper and LendingClub Sensitivity Analysis Plot*

The results match with that of the LendingClub case in the aspect that as the number of loans invested in by a potential investor increases, the percentage of investment return decreases. This is also very intuitive due to the risk factor of a loan getting defaulted increases as the number of loans invested in increases. In other words, an investor may have a similar chance of getting a high return and then losing most of it due to a defaulted loan and this cycle can keep repeating causing the overall percentage of return to decrease as the number of loans invested in increases.

Part VI: Optimization

Optimization Approaches

In this section, we implement three different optimization models to improve an investment strategy using Prosper. To main consistency and comparability with the LendingClub case, we will only use the M1 return approach and the predicted returns from the previously tested two-stage strategy using Gurobi Optimization solver. Gurobi optimization software is a licensed python package that is like excel solver to find the optimal solution of an optimization problem.

The first optimization model building approach involves maximizing the total expected profit. A binary variable is set-up for every loan in our data set. The number of loans constraint is added (as per the maximum number of loans in our dataset) and the objective function is defined (maximize total profit). Using the optimize command output, we can find estimate the optimal percentage of return to be **6.8%** when compared to **2.59**% from LendingClub.

The second optimization model considers the budget constraint of a potential investor. Like the first model, a new budget limiting constraint is added (testing different budgets), and the optimization problem is resolved. Using the optimize command output, we can find estimate the optimal percentage of return to be **10.5%** when compared to **3.18%** from LendingClub.

The third optimization strategy involves incorporating the portfolio risk factor by considering the variance of the returns. To achieve this, first a clustering model is trained with an adjustable k parameter by the investor. Second, the standard deviation is computed for each of the clusters. Third, each loan can be assigned to a specific cluster based on the Euclidean distance from each cluster. Fourth, the standard deviation of the return for each loan can be estimated using the standard deviation of the cluster. The model is then built to maximize profit with the inclusion of the risk-return tradeoff constraint. A sensitivity/penalty factor can be set by the investor to account for the risk tolerance of the investor.

Using the optimize command output, we can find estimate the optimal percentage of return to be **10.3%** when compared to **3.21%** from LendingClub.

	Directly maximize total profit	Maximize profit with budget constraint	Maximize profit with risk-return tradeoff
Prosper	6.8%	10.5%	10.3%
LendingClub	2.59%	3.18%	3.21%

***Table 6**: Optimization for Investment Return under 3 different scenarios*

Conclusion

In conclusion, we saw how information from different attributes can be utilized to create new predictors that may end up boosting the statistical significance of our model as opposed to the original counterparts. It was important to note how a model's performance over a single partition of the data into cross-validation folds can be misleading and running our model several times to develop different iterations of the train / test splits was essential. Another important part was calibration, and how it measured whether the probabilities produced by the model were correct.

Future Extensions

It would be interesting to see how our model would perform if tied to macroeconomic external data that would represent the underlying economy's performance at the time such as oil prices or The World Bank's interest rates, to further extend the pessimistic measure's performance. Additionally, it would be interesting to conduct sentiment analysis over the investor to understand what their risk appetite exactly is as opposed to asking for the numerical input as they may lack the understanding of how the P2P lending market operates.

Appendix

```
Top 10 - Positive Correlations:
----------------------------------
term                            0.102811
service_fees_paid               0.104814
late_fees_paid                  0.105903
age_in_months                   0.115527
borrower_rate                   0.170636
principal_balance               0.307483
days_past_due                   0.326288
loan_status                     1.000000
debt_sale_proceeds_received          NaN
co_borrower_application              NaN
Name: loan_status, dtype: float64

Top 10 - Negative Correlations:
------------------------------
principal_paid            -0.388577
recoveries                -0.388577
total_pymnt               -0.339452
loan_number               -0.117796
loan_length               -0.078809
interest_paid             -0.036537
amount_borrowed            0.007077
prosper_fees_paid          0.076323
next_payment_due_amount    0.092813
term                       0.102811
Name: loan_status, dtype: float64
```

***Exhibit 1**: Top 10 positive and negative correlations with loan status variable*

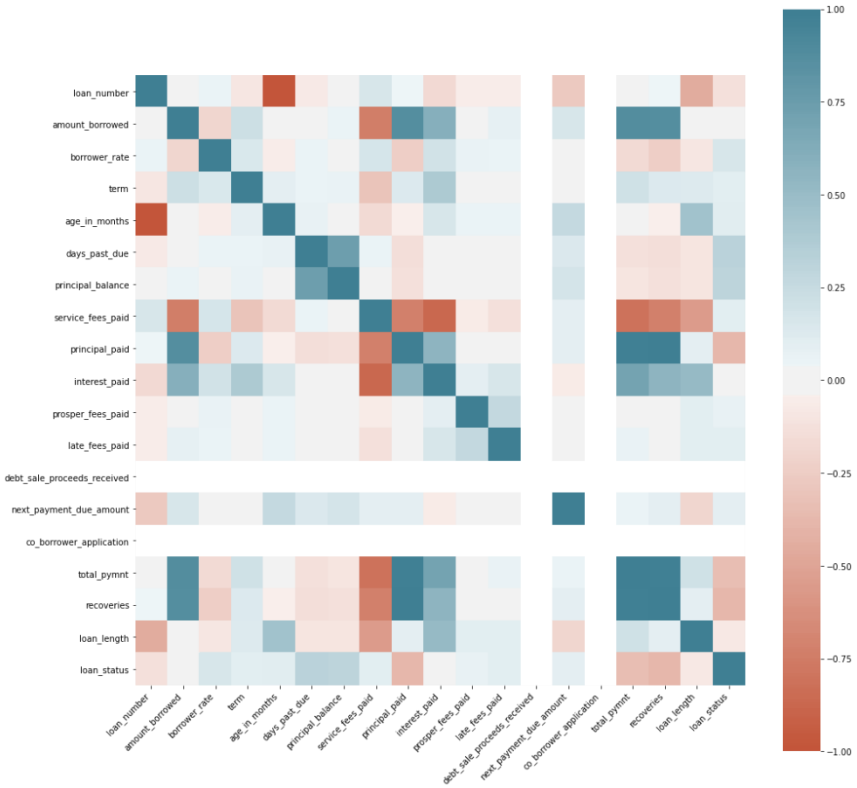


***Exhibit 2**: Correlation Heat Map*

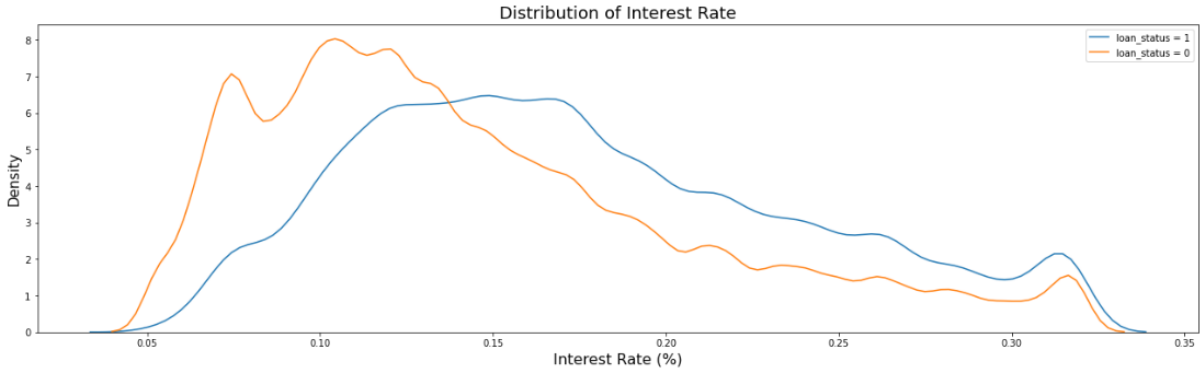


***Exhibit 3**: Loan Status Density Plot with distribution of Interest Rates*

***Exhibit 4**: Violin Plot of Loan amount per Term for each type of loan*

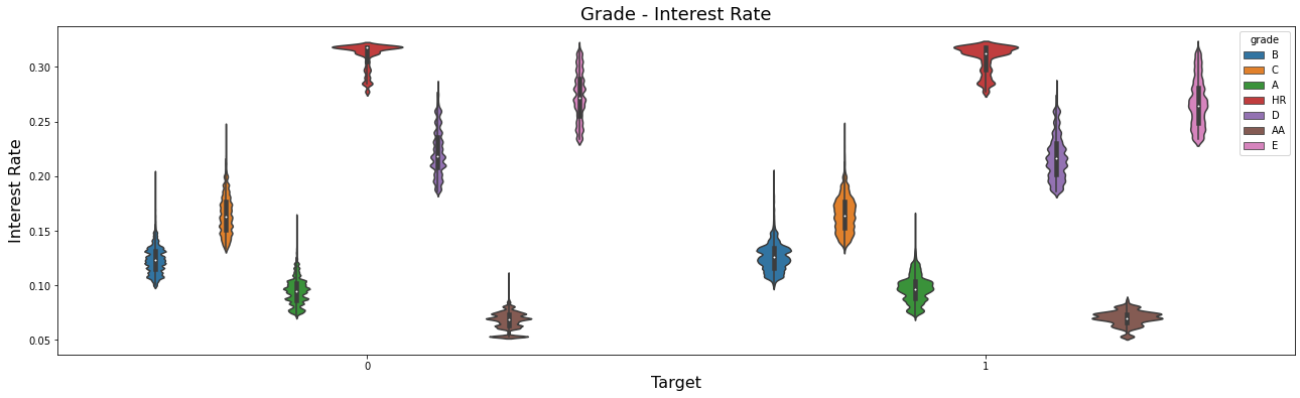
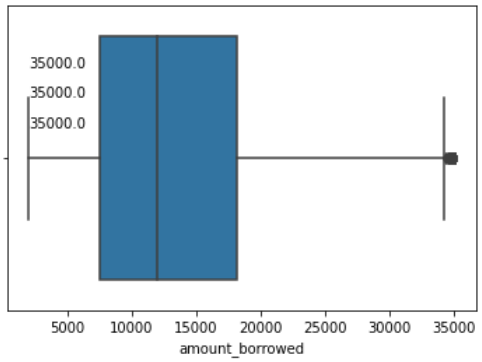
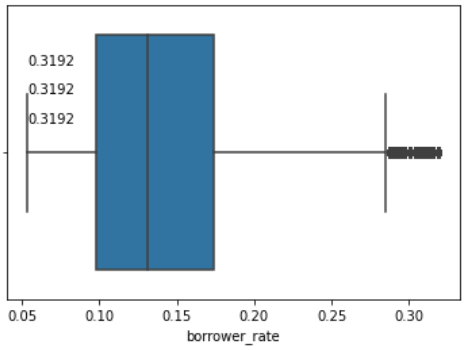
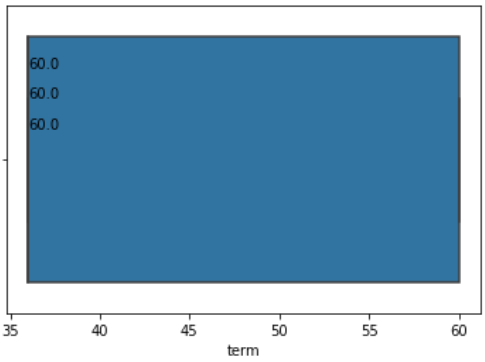
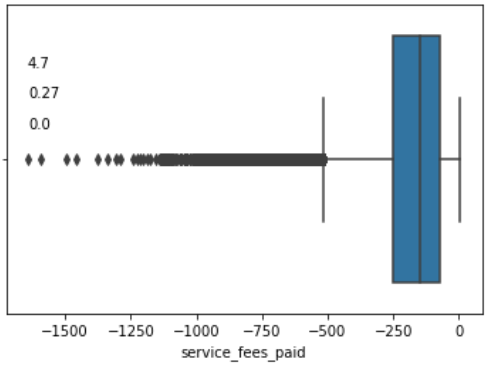


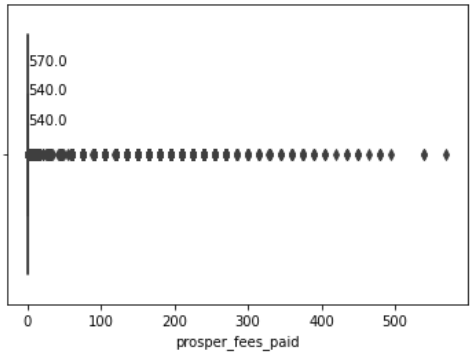
***Exhibit 5**: Violin Plot of Interest Rate for each Loan Status per Grade*

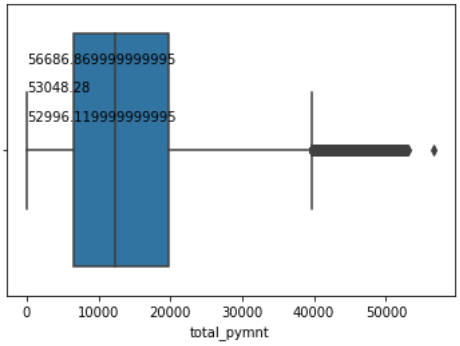

35000.0
35000.0
35000.0
5000
10000
15000
20000
25000
30000
35000
amount_borrowed

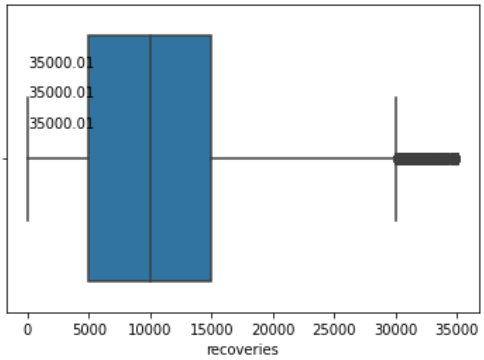

0.3192
0.3192
0.3192
0.05
0.10
0.15
0.20
0.25
0.30
borrower_rate

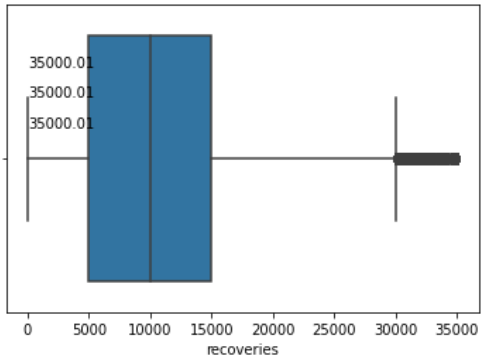

60.0
60.0
60.0
35
40
45
50
55
60
term

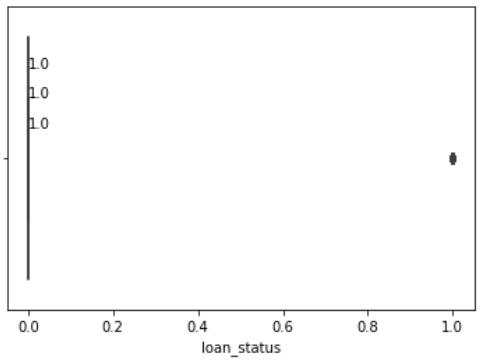

4.7
0.27
0.0
−1500
−1250
−1000
−750
−500
−250
0
service_fees_paid

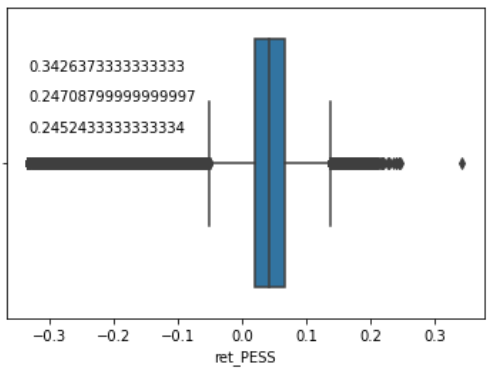

570.0
540.0
540.0
0
100
200
300
400
500
prosper_fees_paid


56686.869999999995
53048.28
52996.119999999995
0
10000
20000
30000
40000
50000
total_pymnt


35000.01
35000.01
35000.01
0
5000
10000
15000
20000
25000
30000
35000
recoveries


35000.01
35000.01
35000.01
0
5000
10000
15000
20000
25000
30000
35000
recoveries


1.0
1.0
1.0
0.0
0.2
0.4
0.6
0.8
1.0
loan_status


0.3426373333333333
0.24708799999999997
0.2452433333333334
−0.3
−0.2
−0.1
0.0
0.1
0.2
0.3
ret_PESS

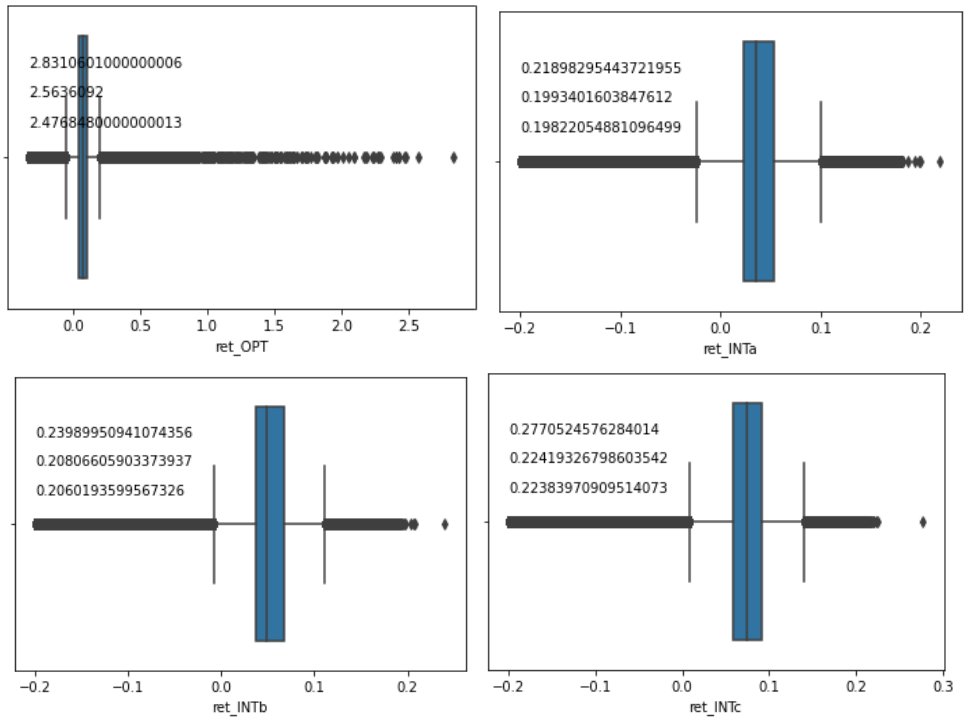


***Exhibit 6**: Various Variable Outlier Plots*

```
term
2 distinct values
36.0    368771
60.0    129950
Name: term, dtype: int64


co_borrower_application
1 distinct values
0.0    498721
Name: co_borrower_application, dtype: int64


grade
7 distinct values
C     119034
B     118014
A     112124
D      55127
AA     54772
E      29394
HR     10256
Name: grade, dtype: int64
```

***Exhibit 7**: Frequency of Loans over Different Grades*

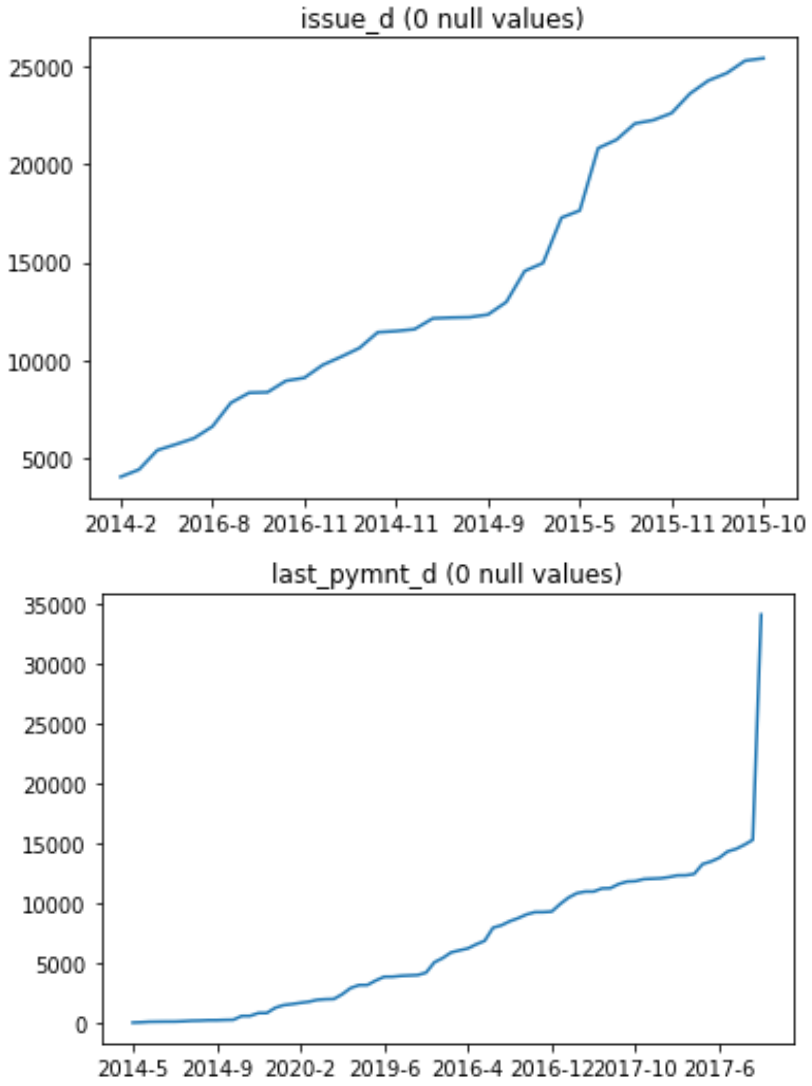


***Exhibit 8**: Issue and Payment date Graphs*

	grade	default	int_rate	return_PESS	return_OPT	return_INTa	return_INTb	return_INTc
A	19.973938	8.612483	9.474686	2.274137	4.541028	2.254222	3.486214	5.706550
AA	10.377824	4.014627	6.775790	2.019641	3.706281	2.023254	3.275717	5.534159
B	22.979313	12.956079	12.394934	2.395357	5.703969	2.466247	3.724494	5.997924
C	24.597005	18.515979	16.501650	2.495612	7.242789	2.583270	3.886307	6.246494
D	12.239538	23.292177	22.047630	2.688308	9.355363	2.692042	4.072759	6.581256
E	7.167875	25.417007	27.214131	3.449610	11.314350	3.154841	4.604898	7.242146
HR	2.664507	25.881872	30.950736	4.027710	12.390232	3.372662	4.887931	7.647172

***Exhibit 9**: Unbalanced Dataset Returns by grade*

	grade	default	int_rate	return_PESS	return_OPT	return_INTa	return_INTb	return_INTc
A	16.375878	34.944187	9.538783	-2.474031	-0.878019	-0.910899	0.147415	2.054459
AA	7.315120	18.946001	6.812487	-0.541292	0.901011	0.408420	1.568575	3.660394
B	21.652157	45.740079	12.460648	-3.750521	-1.679114	-2.000569	-0.990740	0.833388
C	26.888167	56.344973	16.524345	-4.971645	-2.390153	-3.167542	-2.191823	-0.425022
D	15.023438	63.123814	21.864717	-6.024556	-2.782839	-4.024851	-3.049293	-1.277420
E	9.258488	65.457922	26.980508	-6.464774	-2.762397	-3.933130	-2.920515	-1.078863
HR	3.486752	65.792889	30.860243	-7.820892	-3.933467	-4.032661	-2.986616	-1.081351

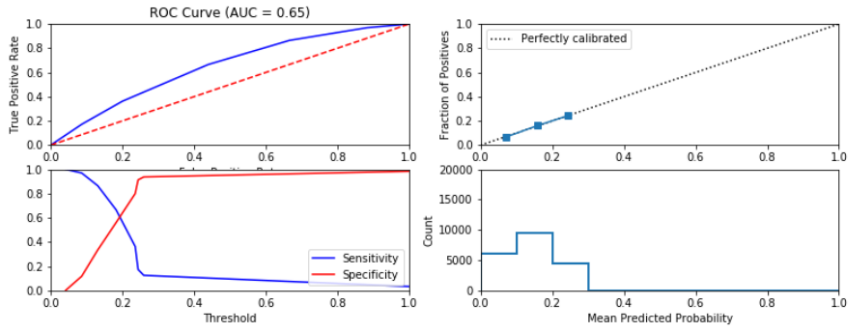
***Exhibit 10**: Balanced Dataset Returns by grade*

```
============================================================
  Model: Grade only logistic l2
============================================================
Fit time: 0.56 seconds
Optimal parameters:
{}

Accuracy-maximizing threshold was: 1
Accuracy:  0.84895
              precision    recall  f1-score   support

  No default     0.8489    1.0000    0.9183     16979
     Default     0.0000    0.0000    0.0000      3021

    accuracy                         0.8489     20000
   macro avg     0.4245    0.5000    0.4592     20000
weighted avg     0.7207    0.8489    0.7796     20000
```




```
Similarity to LC grade ranking:  0.8863989075158132
Brier score: 0.12373502386731795
Were parameters on edge? : False
Score variations around CV search grid : 0.0
[0.8499]
```

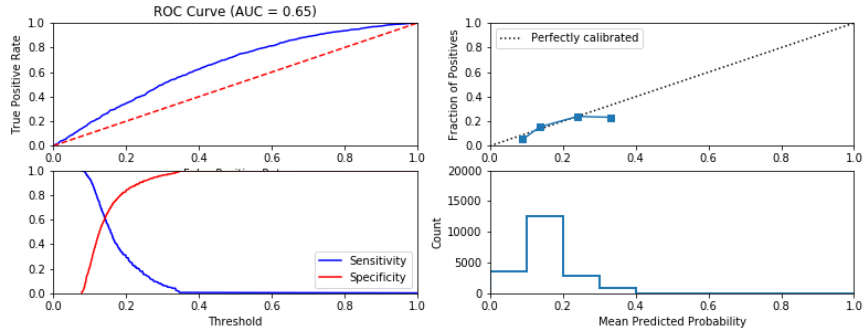
***Exhibit 11**: Grade only Classification Model Results*

```
============================================================
  Model: Interest rate only logistics l2
============================================================
Fit time: 0.34 seconds
Optimal parameters:
{}

Accuracy-maximizing threshold was: 1
Accuracy:  0.84895
              precision    recall  f1-score   support

  No default     0.8489    1.0000    0.9183     16979
     Default     0.0000    0.0000    0.0000      3021

    accuracy                         0.8489     20000
   macro avg     0.4245    0.5000    0.4592     20000
weighted avg     0.7207    0.8489    0.7796     20000
```




```
Similarity to LC grade ranking:  0.7919594781733205
Brier score: 0.1248645919851115
Were parameters on edge? : False
Score variations around CV search grid : 0.0
[0.8499]
```

***Exhibit 12**: Interest only Classification Model Results*

```
============================================================
  Model: Ridge Classifier
============================================================
Fit time: 1.72 seconds
Optimal parameters:
{'alpha': 166.81005372000558}

Accuracy:  0.84895
              precision    recall  f1-score   support

  No default     0.8489    1.0000    0.9183     16979
     Default     0.0000    0.0000    0.0000      3021

    accuracy                         0.8489     20000
   macro avg     0.4245    0.5000    0.4592     20000
weighted avg     0.7207    0.8489    0.7796     20000


Similarity to LC grade ranking:  0.1926358825472961
```

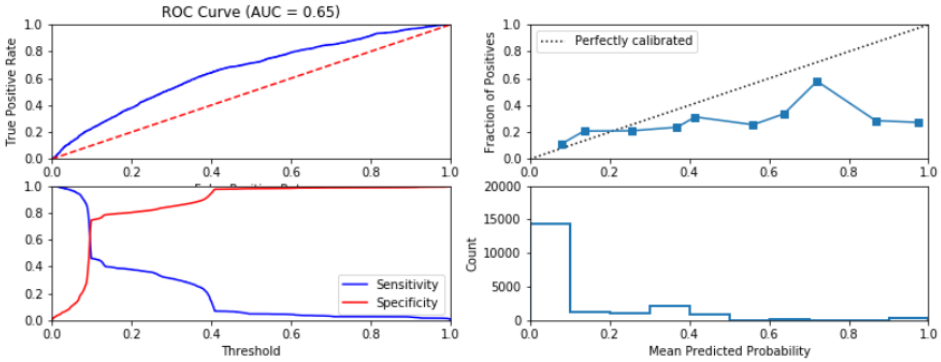
***Exhibit 13**: Ridge Classification Model Results*

```
  Model: Gaussian Naive Bayes
============================================================
Fit time: 0.17 seconds
Optimal parameters:
{}

Accuracy-maximizing threshold was: 1
Accuracy:  0.84895
              precision    recall  f1-score   support

  No default     0.8489    1.0000    0.9183     16979
     Default     0.0000    0.0000    0.0000      3021

    accuracy                         0.8489     20000
   macro avg     0.4245    0.5000    0.4592     20000
weighted avg     0.7207    0.8489    0.7796     20000
```




```
Similarity to LC grade ranking:  0.17502946948272416
Brier score: 0.13498558967104715
Were parameters on edge? : False
Score variations around CV search grid : 0.0
[0.8421]
```

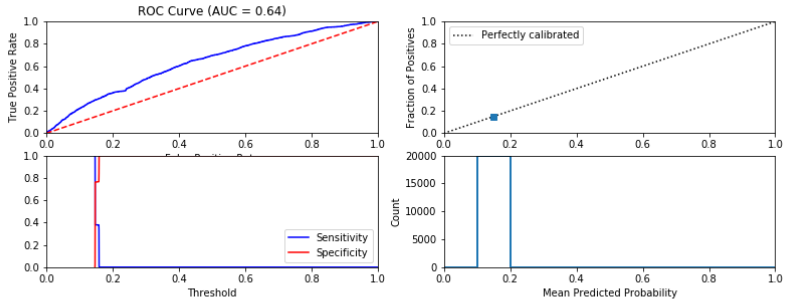
***Exhibit 14**: Naïve Bayes Classification Model Results*

```
------------------------------------------------------------
  Model: l2 Penalized Logistic Regression
============================================================
Fit time: 4.67 seconds
Optimal parameters:
{'C': 0.0001}

Accuracy-maximizing threshold was: 0.1590103006760455
Accuracy:  0.8491
              precision    recall  f1-score   support

  No default     0.8493    0.9996    0.9184     16979
     Default     0.6000    0.0030    0.0059      3021

    accuracy                         0.8491     20000
   macro avg     0.7246    0.5013    0.4621     20000
weighted avg     0.8116    0.8491    0.7805     20000
```




```
Similarity to LC grade ranking:  0.21978648492929584
Brier score: 0.12782571325642217
Were parameters on edge? : True
Score variations around CV search grid : 0.3686708240185236
[0.8499     0.8499     0.8499     0.8499     0.849      0.84743333
 0.84696667 0.84676667 0.84676667 0.84676667]
```

***Exhibit 15**: L2 Penalized Logistic Regression Classifier Model Results*

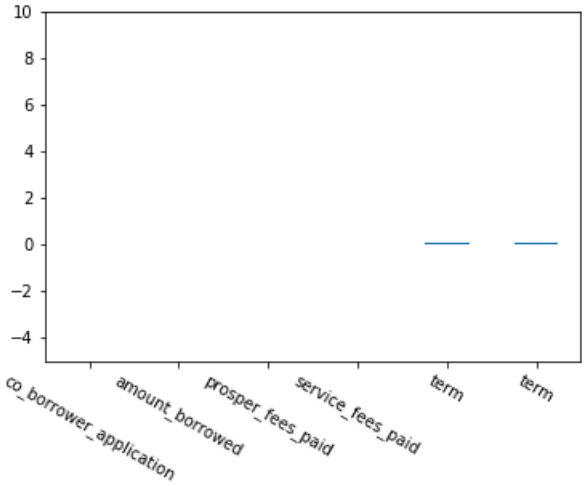


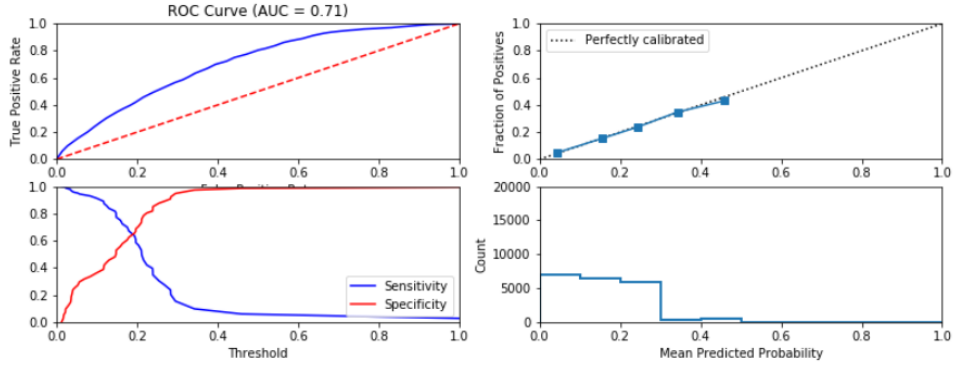
***Exhibit 16**: L2 Penalized Logistic Regression Feature Importance*

```
============================================================
  Model: Decision tree
============================================================
Fit time: 2.96 seconds
Optimal parameters:
{'min_samples_leaf': 500}

Accuracy-maximizing threshold was: 1
Accuracy:  0.84895
              precision    recall  f1-score   support

  No default     0.8489    1.0000    0.9183     16979
     Default     0.0000    0.0000    0.0000      3021

    accuracy                         0.8489     20000
   macro avg     0.4245    0.5000    0.4592     20000
weighted avg     0.7207    0.8489    0.7796     20000
```




```
Similarity to LC grade ranking:  0.16188419085472244
Brier score: 0.1192939107245286
Were parameters on edge? : True
Score variations around CV search grid : 0.0
[0.8499 0.8499 0.8499 0.8499 0.8499 0.8499 0.8499 0.8499 0.8499]
```

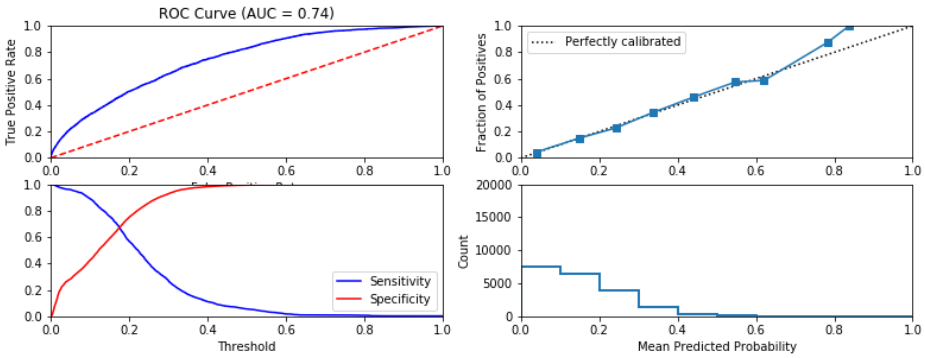
***Exhibit 17**: Decision Tree Classifier Model Results*

```
============================================================
  Model: Random forest
============================================================
Fit time: 846.33 seconds
Optimal parameters:
{'min_samples_leaf': 20, 'n_estimators': 100}

Accuracy-maximizing threshold was: 0.3857648186064266
Accuracy:  0.84975
              precision    recall  f1-score   support

  No default     0.8629    0.9785    0.9171     16979
     Default     0.5107    0.1261    0.2023      3021

    accuracy                         0.8498     20000
   macro avg     0.6868    0.5523    0.5597     20000
weighted avg     0.8097    0.8498    0.8091     20000
```




```
Similarity to LC grade ranking:  0.15449947428141564
Brier score: 0.11452158176917314
Were parameters on edge? : False
Score variations around CV search grid : 6.463729556303314
[0.79743333 0.79893333 0.79796667 0.79686667 0.7978     0.8325
 0.835      0.8341     0.83386667 0.8344     0.84353333 0.84403333
 0.84393333 0.84333333 0.8441     0.84846667 0.8489     0.8491
 0.8494     0.8494     0.8507     0.85083333 0.85063333 0.85106667
 0.85076667 0.8515     0.85176667 0.85146667 0.8513     0.85146667
 0.85133333 0.85186667 0.851      0.8518     0.8519     0.85156667
```

***Exhibit 18**: Random Forest Classifier Model Results*

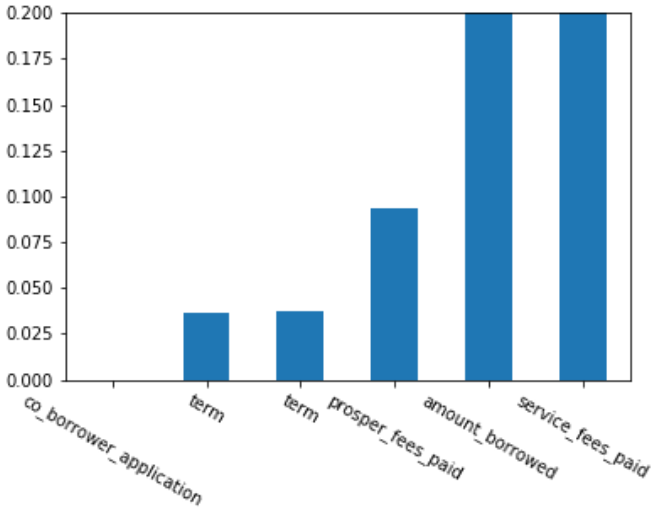


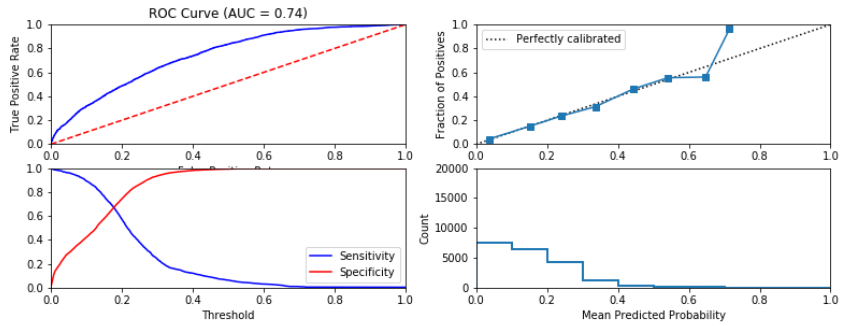
***Exhibit 19**: Random Forest Feature Importance*

```
============================================================
  Model: Bagged trees
============================================================
Fit time: 913.59 seconds
Optimal parameters:
{'min_samples_leaf': 50, 'n_estimators': 200}

Accuracy-maximizing threshold was: 0.4741333189559673
Accuracy:  0.85195
              precision    recall  f1-score   support

  No default     0.8573    0.9905    0.9191     16979
     Default     0.5785    0.0732    0.1299      3021

    accuracy                         0.8519     20000
   macro avg     0.7179    0.5318    0.5245     20000
weighted avg     0.8152    0.8519    0.7999     20000
```




```
Similarity to LC grade ranking:  0.14681397985135544
Brier score: 0.11521947939687466
Were parameters on edge? : False
Score variations around CV search grid : 0.9870353687673765
[0.84336667 0.84263333 0.8439     0.84353333 0.84363333 0.84843333
 0.84823333 0.8478     0.84843333 0.84856667 0.8505     0.8506
 0.85033333 0.8505     0.8504     0.8507     0.85096667 0.85076667
 0.85103333 0.8509     0.8501     0.85036667 0.85       0.85006667
 0.85023333 0.84953333 0.84956667 0.84946667 0.84946667 0.84956667
 0.8498     0.8499     0.8499     0.8498     0.84983333]
```

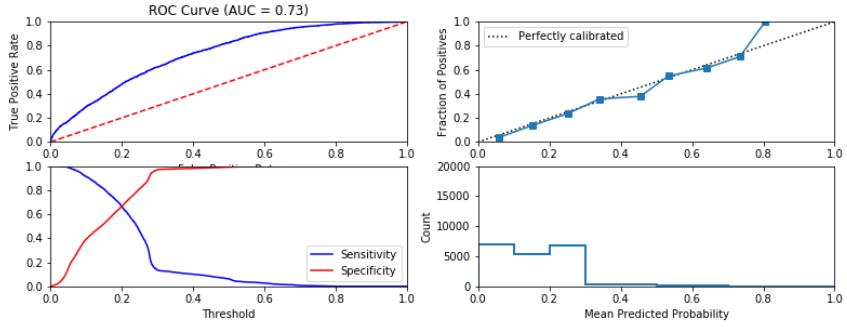
***Exhibit 20**: Bagged Trees Classifier Model Results*

```
============================================================
  Model: Multi-Layer Perceptron
============================================================
Fit time: 607.33 seconds
Optimal parameters:
{'hidden_layer_sizes': 100}

Accuracy-maximizing threshold was: 0.4861110408709919
Accuracy:  0.85075
              precision    recall  f1-score   support

  No default     0.8566    0.9900    0.9184     16979
     Default     0.5479    0.0682    0.1213      3021

    accuracy                         0.8508     20000
   macro avg     0.7022    0.5291    0.5199     20000
weighted avg     0.8099    0.8508    0.7980     20000
```




```
Similarity to LC grade ranking:  0.1481178386306645
Brier score: 0.11662118241920238
Were parameters on edge? : False
Score variations around CV search grid : 0.13317143864320338
[0.8499     0.8501     0.85046667 0.85103333 0.85046667 0.85076667]
```

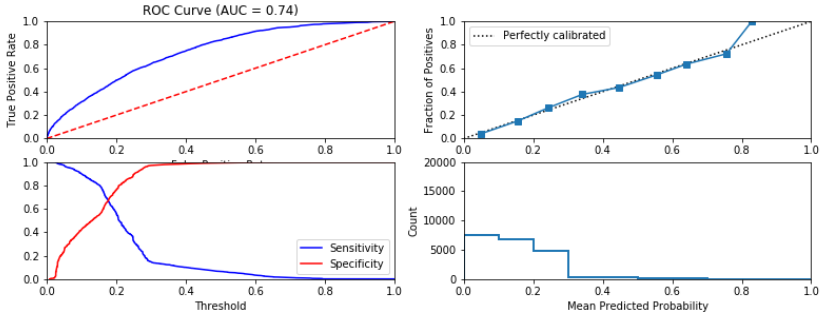
***Exhibit 21**: Multi-Layer Perceptron Classifier Model Results*

```
============================================================
  Model: Light GBM
============================================================
Fit time: 33.39 seconds
Optimal parameters:
{'n_estimators': 60, 'num_leaves': 20}

Accuracy-maximizing threshold was: 0.4881470597056431
Accuracy:  0.85185
              precision    recall  f1-score   support

  No default     0.8566    0.9914    0.9191     16979
     Default     0.5829    0.0675    0.1210      3021

    accuracy                         0.8518     20000
   macro avg     0.7197    0.5295    0.5201     20000
weighted avg     0.8153    0.8518    0.7986     20000
```




```
Similarity to LC grade ranking:  0.1536271700878448
Brier score: 0.11501730375121172
Were parameters on edge? : False
Score variations around CV search grid : nan
[       nan 0.8511     0.85206667 0.85186667 0.8516           nan
 0.85183333 0.8522     0.85133333 0.8517           nan 0.85173333
 0.8519     0.85123333 0.85156667       nan 0.85206667 0.85156667
 0.85206667 0.8517           nan 0.85206667 0.8518     0.85086667
 0.8509    ]
```

***Exhibit 22**: Light GBM Classifier Model Results*

```
============================================================
  Model: Lasso  Return column: ret_PESS
============================================================
Fit time: 0.28 seconds
Optimal parameters:
{'alpha': 1e-05}

Testing r2 score: 0.15777175569802915
Were parameters on edge (model) : False
Score variations around CV search grid (model) : 108.73503712186032
All test scores : [ 0.12391231  0.12391231  0.1246795   0.12906773  0.00250605 -0.01127411
 -0.01127411 -0.01127411]
============================================================
  Model: Lasso  Return column: ret_OPT
============================================================
Fit time: 0.32 seconds
Optimal parameters:
{'alpha': 1e-05}

Testing r2 score: 0.021470588584063588
Were parameters on edge (model) : False
Score variations around CV search grid (model) : -2.632487185023721
All test scores : [-0.08014381 -0.08014381 -0.07980375 -0.07859004 -0.08065891 -0.08065891
 -0.08065891 -0.08065891]
============================================================
  Model: Lasso  Return column: ret_INTa
============================================================
Fit time: 0.3 seconds
Optimal parameters:
{'alpha': 1e-05}

Testing r2 score: 0.17361224970705003
Were parameters on edge (model) : False
Score variations around CV search grid (model) : 111.33402798354275
All test scores : [ 0.1153231   0.1153231   0.11675571  0.12238064 -0.01267885 -0.01387066
 -0.01387066 -0.01387066]
============================================================
  Model: Lasso  Return column: ret_INTb
============================================================
Fit time: 0.34 seconds
Optimal parameters:
{'alpha': 1e-05}

Testing r2 score: 0.1416995310879764
Were parameters on edge (model) : False
Score variations around CV search grid (model) : 123.06939059837619
All test scores : [ 0.07653298  0.07653298  0.07789847  0.08338214 -0.01922039 -0.01923575
 -0.01923575 -0.01923575]
```

***Exhibit 23**: Lasso Regressor Model Results*

```
============================================================
  Model: Ridge  Return column: ret_PESS
============================================================
Fit time: 0.29 seconds
Optimal parameters:
{'alpha': 0.1}

Testing r2 score: 0.17612661976804678
Were parameters on edge (model) : True
Score variations around CV search grid (model) : 0.05325241533362927
All test scores : [0.12492967 0.12492967 0.12492967 0.12492967 0.12492974 0.12493034
 0.12493635 0.12499623]
============================================================
  Model: Ridge  Return column: ret_OPT
============================================================
Fit time: 0.24 seconds
Optimal parameters:
{'alpha': 0.1}

Testing r2 score: 0.02478052210025561
Were parameters on edge (model) : True
Score variations around CV search grid (model) : -0.031369333862783094
All test scores : [-0.08014381 -0.08014381 -0.08014381 -0.08014381 -0.08014379 -0.08014356
 -0.08014129 -0.08011868]
============================================================
  Model: Ridge  Return column: ret_INTa
============================================================
Fit time: 0.27 seconds
Optimal parameters:
{'alpha': 0.1}

Testing r2 score: 0.1856910463889362
Were parameters on edge (model) : True
Score variations around CV search grid (model) : 0.07750198154986125
All test scores : [0.1153231  0.1153231  0.11532311 0.11532311 0.11532319 0.115324
 0.11533208 0.11541255]
============================================================
  Model: Ridge  Return column: ret_INTb
============================================================
Fit time: 0.28 seconds
Optimal parameters:
{'alpha': 0.1}

Testing r2 score: 0.15304869374759578
Were parameters on edge (model) : True
Score variations around CV search grid (model) : 0.10924594695323299
All test scores : [0.07653298 0.07653298 0.07653298 0.07653299 0.07653306 0.07653382
 0.07654138 0.07661668]
```

***Exhibit 24**: Ridge Regressor Model Results*

```
============================================================
  Model: Linear Regression  Return column: ret_PESS
============================================================
Fit time: 0.06 seconds
Optimal parameters:
{}

Testing r2 score: 0.17613361794005966
Were parameters on edge (model) : False
Score variations around CV search grid (model) : 0.0
All test scores : [0.12504423]
============================================================
  Model: Linear Regression  Return column: ret_OPT
============================================================
Fit time: 0.05 seconds
Optimal parameters:
{}

Testing r2 score: 0.024802562043752197
Were parameters on edge (model) : False
Score variations around CV search grid (model) : -0.0
All test scores : [-0.08008129]
============================================================
  Model: Linear Regression  Return column: ret_INTa
============================================================
Fit time: 0.06 seconds
Optimal parameters:
{}

Testing r2 score: 0.1856946108607851
Were parameters on edge (model) : False
Score variations around CV search grid (model) : 0.0
All test scores : [0.11543783]
============================================================
  Model: Linear Regression  Return column: ret_INTb
============================================================
Fit time: 0.05 seconds
Optimal parameters:
{}

Testing r2 score: 0.1530518011812041
Were parameters on edge (model) : False
Score variations around CV search grid (model) : 0.0
All test scores : [0.07665579]
```

***Exhibit 25**: Ordinary Least Squares Regressor Model Results*

```
============================================================
  Model: Multi-Layer Perceptron  Return column: ret_PESS
============================================================
Fit time: 334.59 seconds
Optimal parameters:
{'alpha': 0.001, 'hidden_layer_sizes': 200}

Testing r2 score: 0.2390840438207419
Were parameters on edge (model) : True
Score variations around CV search grid (model) : 110.26381391022588
All test scores : [-2.92860800e-03  9.65928397e-02  1.77056197e-01  1.98610613e-01
  2.17835393e-01  3.05718326e-02  1.60968950e-01  8.34271175e-03
  7.06226741e-02  1.57758075e-01  2.05444729e-01  1.51339048e-01
  1.06082805e-01  1.45484558e-01 -9.59442317e-03  1.14018881e-01
  1.13556270e-01  1.41529363e-01  1.23240542e-01  1.08668746e-01
  1.45887762e-01 -2.79680597e-03  6.94653973e-03  6.49150809e-05
 -1.39407392e-02 -1.87019282e-02 -1.03177765e-02 -9.75417577e-03
 -8.75084890e-03 -8.66949723e-03 -2.23582194e-02 -1.95075497e-02
 -7.75610984e-03 -1.03389972e-02 -1.27329441e-02 -1.04782245e-02
 -1.44920519e-02 -1.06012913e-02 -1.41621678e-02 -8.88395741e-03
 -9.92451748e-03 -1.13618151e-02]
============================================================
  Model: Multi-Layer Perceptron  Return column: ret_OPT
============================================================
Fit time: 333.28 seconds
Optimal parameters:
{'alpha': 0.01, 'hidden_layer_sizes': 50}

Testing r2 score: 0.028715737188446333
Were parameters on edge (model) : False
Score variations around CV search grid (model) : -136.03953453516993
All test scores : [-0.06865017 -0.09364391 -0.06133931 -0.06822706 -0.084723   -0.07221498
 -0.085772   -0.14225123 -0.08726108 -0.06026585 -0.06207001 -0.07922968
 -0.10929629 -0.1001539  -0.13733734 -0.07133609 -0.08661144 -0.10559166
 -0.06724398 -0.08125016 -0.09773988 -0.08398669 -0.08246701 -0.08941552
 -0.07725701 -0.07657637 -0.06971975 -0.08623431 -0.07630609 -0.08889681
 -0.08160355 -0.07122032 -0.08228184 -0.08482517 -0.08758655 -0.08009864
 -0.07858734 -0.08107488 -0.08820232 -0.08380793 -0.07494689 -0.07665899]
============================================================
  Model: Multi-Layer Perceptron  Return column: ret_INTa
============================================================
Fit time: 345.63 seconds
Optimal parameters:
{'alpha': 0.001, 'hidden_layer_sizes': 100}

   Testing r2 score: 0.2547923332691421
   Were parameters on edge (model) : True
   Score variations around CV search grid (model) : 208.34683950462346
   All test scores : [-0.12706027  0.11969097  0.12998993  0.18060324  0.10425588  0.03375094
     0.14628646 -0.00979292  0.10636011  0.10346162  0.10460552  0.17227736
     0.09715407  0.11908192 -0.19567791  0.03185878  0.10683795  0.13250057
     0.14179843  0.09679741  0.11512122 -0.00662635  0.00272116 -0.01107799
    -0.00777462 -0.01621714 -0.02146239 -0.01503067 -0.01395485 -0.01283318
    -0.02175604 -0.01028472 -0.02330622 -0.01579725 -0.01172696 -0.01379621
    -0.01351292 -0.01500755 -0.01333029 -0.01415291 -0.01425798 -0.0142982 ]
   ============================================================
     Model: Multi-Layer Perceptron  Return column: ret_INTb
   ============================================================
   Fit time: 337.11 seconds
   Optimal parameters:
   {'alpha': 0.01, 'hidden_layer_sizes': 50}

   Testing r2 score: 0.17541059361708122
   Were parameters on edge (model) : False
   Score variations around CV search grid (model) : 683.6101399590427
   All test scores : [-0.00301475 -0.02191092  0.06494859  0.09040759  0.1091098   0.04486798
     0.08523559 -0.71813787  0.05035178  0.12305096  0.0948115   0.09285719
     0.04858547  0.12271202 -0.03964401  0.00148126  0.09316546  0.09275669
     0.08532581  0.03528199  0.07799137  0.04023045  0.00854073 -0.02432491
    -0.02081041 -0.01990862 -0.02584478 -0.01700369 -0.01709033 -0.0162285
    -0.01206832 -0.01052521 -0.01462367 -0.02798708 -0.02492019 -0.01326989
    -0.02004816 -0.02043221 -0.02387319 -0.02593357 -0.01931169 -0.02186974]
```

***Exhibit 26**: Multi-Layer Regressor Model Results*

```
=========================================================
  Model: Random forest regressor  Return column: ret_PESS
=========================================================
Fit time: 196.05 seconds
Optimal parameters:
{'min_samples_leaf': 75, 'n_estimators': 90}

Testing r2 score: 0.3132111667758547
Were parameters on edge (model) : True
Score variations around CV search grid (model) : 12.223286843451328
All test scores : [0.22742234 0.22875706 0.22772106 0.22843033 0.22833516 0.22947257
 0.22870358 0.22232914 0.22225981 0.22376882 0.22321379 0.2233802
 0.22363295 0.22334089 0.21209937 0.2120619  0.21275021 0.21317527
 0.21277307 0.21301128 0.21416096 0.20605004 0.20529838 0.20628124
 0.20663087 0.20672425 0.20589552 0.2069726  0.20182448 0.20336222
 0.20295764 0.20286169 0.20142348 0.20195736 0.2023139 ]
=========================================================
  Model: Random forest regressor  Return column: ret_OPT
=========================================================
Fit time: 207.9 seconds
Optimal parameters:
{'min_samples_leaf': 75, 'n_estimators': 100}

Testing r2 score: 0.06585042004441466
Were parameters on edge (model) : True
Score variations around CV search grid (model) : -24.242367367499856
All test scores : [-0.0696877  -0.06720319 -0.06919765 -0.06820098 -0.06741147 -0.0682948
 -0.06660003 -0.07145661 -0.06983037 -0.07045838 -0.07078508 -0.07015385
 -0.07096163 -0.07042917 -0.07752429 -0.07856048 -0.07861372 -0.07823639
 -0.07761847 -0.07805758 -0.07819228 -0.07930328 -0.07887808 -0.07947359
 -0.08023442 -0.08035987 -0.07955747 -0.07966309 -0.08274545 -0.08167692
 -0.08139095 -0.08090288 -0.08149237 -0.08183322 -0.08074823]
=========================================================
  Model: Random forest regressor  Return column: ret_INTa
=========================================================
Fit time: 182.1 seconds
Optimal parameters:
{'min_samples_leaf': 75, 'n_estimators': 35}

Testing r2 score: 0.28777630041086766
Were parameters on edge (model) : True
Score variations around CV search grid (model) : 9.762663685659895
All test scores : [0.18963995 0.18866826 0.18894098 0.18914241 0.18903751 0.18927176
 0.18909946 0.18566448 0.18644053 0.18777241 0.18799698 0.18797981
 0.18669851 0.18724561 0.17951459 0.17956033 0.17812493 0.17863091
 0.17862599 0.17882211 0.17965261 0.17538075 0.17380629 0.17584273
 0.17632058 0.1756024  0.17392571 0.17452632 0.1721281  0.17112604
 0.17217302 0.17184434 0.17134067 0.17188735 0.17346945]
=========================================================
  Model: Random forest regressor  Return column: ret_INTb
=========================================================
Fit time: 187.04 seconds
Optimal parameters:
{'min_samples_leaf': 75, 'n_estimators': 100}

Testing r2 score: 0.24536926877973642
Were parameters on edge (model) : True
Score variations around CV search grid (model) : 10.710118045478927
All test scores : [0.13770122 0.13665521 0.13652096 0.13623455 0.13727794 0.13704416
 0.13790196 0.13503575 0.13309152 0.13414606 0.1346388  0.13341168
    0.1348943  0.13437239 0.12751545 0.12802447 0.12877887 0.12784231
    0.12704824 0.12811258 0.12869334 0.12487168 0.12517148 0.12427102
    0.12437862 0.12470403 0.12545406 0.12499861 0.12445145 0.12357668
    0.1231325  0.1242939  0.12361091 0.12384777 0.12362587]
```

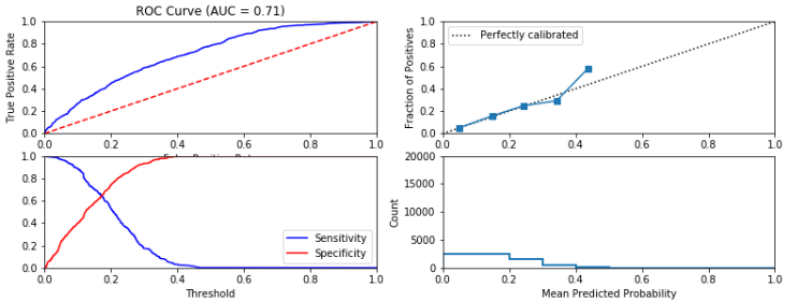
***Exhibit 27**: Random Forest Regressor Model Results - Optimal*

```
------------------------------------------------------------
  Model: Random forest
============================================================
Fit time: 20.14 seconds
Optimal parameters:
{'min_samples_leaf': 40, 'n_estimators': 5}

Accuracy-maximizing threshold was: 0.38883497149343443
Accuracy:  0.8477142857142858
              precision    recall  f1-score   support

  No default     0.8509    0.9946    0.9171      5931
     Default     0.5224    0.0327    0.0616      1069

    accuracy                         0.8477      7000
   macro avg     0.6866    0.5137    0.4894      7000
weighted avg     0.8007    0.8477    0.7865      7000
```




```
Similarity to LC grade ranking:  0.16459869461633106
Brier score: 0.12042820781137027
```

```
Out[55]: {'model': RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=40, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=5,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False),
 'y_pred_labels': array([False, False, False, ..., False, False, False]),
 'y_pred_probs': array([0.00916084, 0.10240727, 0.11321924, ..., 0.19015348, 0.20383766,
        0.20383766])}
```

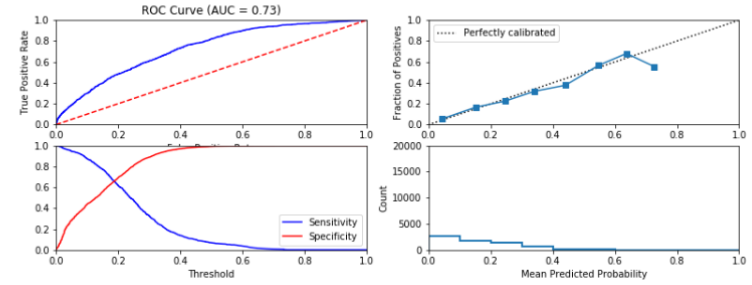
***Exhibit 28**: Train and test on whole time period using RF*

```
  Model: Random forest
============================================================
Fit time: 20.89 seconds
Optimal parameters:
{'min_samples_leaf': 13, 'n_estimators': 20}

Accuracy-maximizing threshold was: 0.3966215604713658
Accuracy:  0.831
              precision    recall  f1-score   support

  No default     0.8508    0.9671    0.9052      5843
     Default     0.4637    0.1435    0.2191      1157

    accuracy                         0.8310      7000
   macro avg     0.6572    0.5553    0.5622      7000
weighted avg     0.7868    0.8310    0.7918      7000
```




```
Similarity to LC grade ranking:  0.15796531860145965
Brier score: 0.1247262811630372
```

```
Out[57]: {'model': RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=13, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=20,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False),
 'y_pred_labels': array([False, False, False, ..., False, False, False]),
 'y_pred_probs': array([0.17903738, 0.10038034, 0.01744186, ..., 0.13552458, 0.12735702,
        0.156478  ])}
```

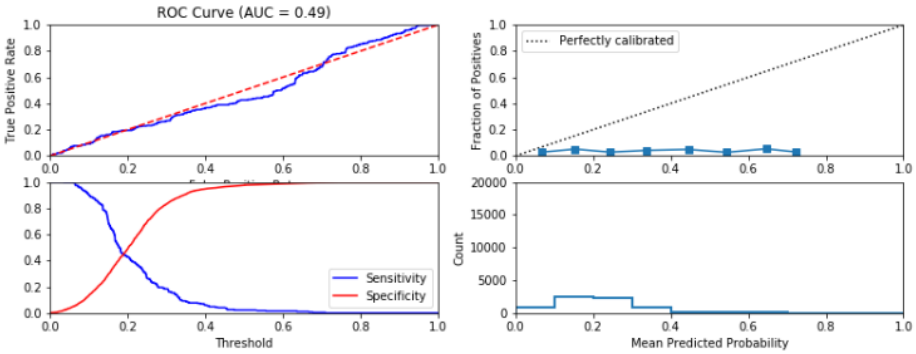
***Exhibit 29**: Train and Test on first 2 years only (2013-2015) using RF*

```
============================================================
  Model: Random forest
============================================================
Fit time: 19.56 seconds
Optimal parameters:
{'min_samples_leaf': 13, 'n_estimators': 20}

Accuracy-maximizing threshold was: 0.3966215604713658
Accuracy:  0.9118571428571428
              precision    recall  f1-score   support

  No default     0.9616    0.9460    0.9538      6728
     Default     0.0472    0.0662    0.0551       272

    accuracy                         0.9119      7000
   macro avg     0.5044    0.5061    0.5045      7000
weighted avg     0.9261    0.9119    0.9189      7000
```




```
Similarity to LC grade ranking:  0.15796531860145965
Brier score: 0.08039909534844232
```

Out[58]:
```
{'model': RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=13, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=20,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False),
 'y_pred_labels': array([False, False, False, ..., False, False, False]),
 'y_pred_probs': array([0.2599607 , 0.21394454, 0.3249159 , ..., 0.31381318, 0.21377097,
        0.20218554])}
```

***Exhibit 30**: Train on the first 2 years and test on the last year using RF*