INTEREST RATE PREDICTOR

CAPSTONE PROJECT – DATA SCIENCE CERTIFICATE

Table of Contents

Interest Rate Predictor

Authors: Sara Alaoui, Erika Ruiz, Olimjon Alimov, Andrew Hall

GitHub: https://github.com/georgetown-analytics/Interest-Rate-Predictor

Abstract

This project creates a model that predicts the interest rate for peer-to-peer loans based on consumer input and other variables. The product is an interactive personal loan calculator the user can embed in websites. The calculator helps customers simulate a real-life application process without having to share their sensitive personal information.

Problem Framing

Peer-to-peer lending, a growing method of debt financing, allows people to borrow and lend money without traditional financial institutions. Lenders and borrowers connect through peer-to-peer platforms faster and cheaper than through conventional financial institutions. LendingClub (https://www.lendingclub.com/) is among the most prominent peer-to-peer platforms. LendingClub had 45 percent market share in 2017. LendingClub offers loans from $1,000 to $35,000 for individuals and loans from $15,000 to $300,000 for businesses.

Traditionally, lenders relied on the FICO-based credit model to calculate the ideal interest rate for their customers. Companies like LendingClub increasingly use machine learning to assess and price credit risk. Starting in 2017, the company created a more powerful and complex multivariable model that takes advantage of more data points for each borrower.

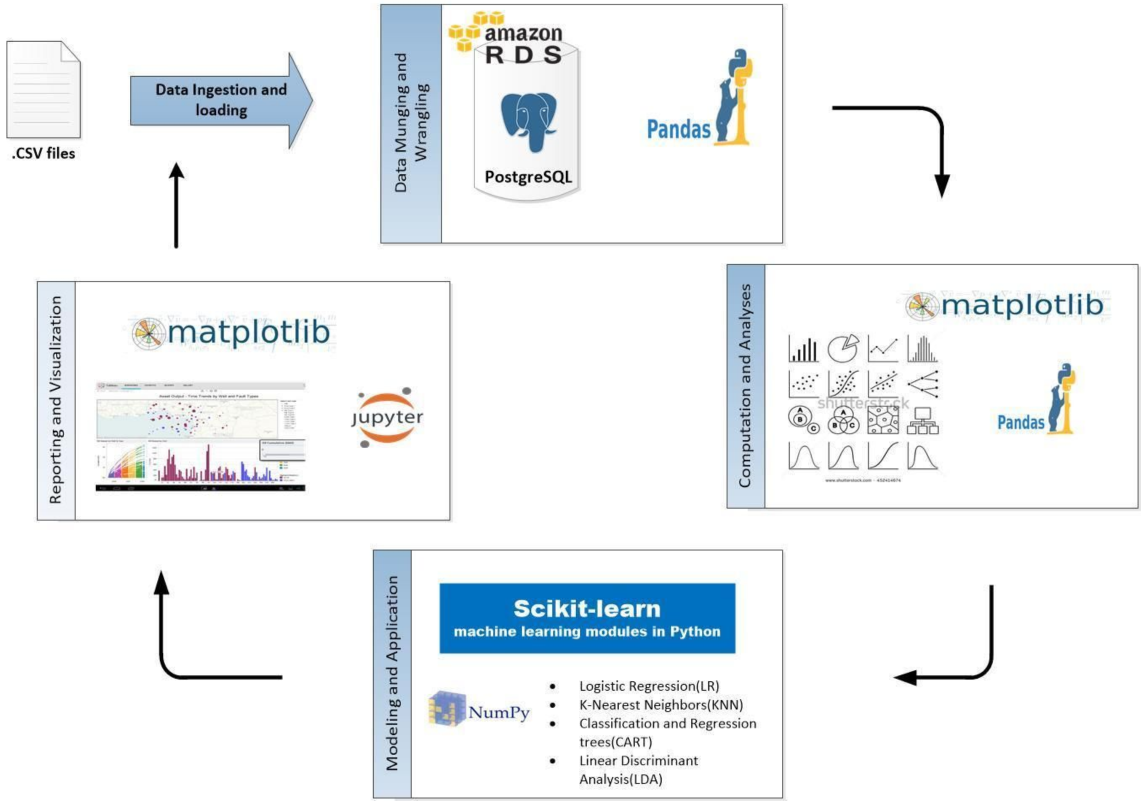
For this project, we will use LendingClub's data to create a similar model that predicts the interest rate and powers our interactive interest rate calculator. The user can embed the interactive calculator in websites, making it easy for customers to simulate a real-life application process without sharing sensitive personal information.

Hypotheses

- What will be the loan interest rate for LendingClub customers given their profile and request?
- What variables help predict the ideal loan interest rate?
- Can we build a simple model with no more than 15 features and still reliably calculate an interest rate prediction?
- Would a classification model be more reliable than a regression model?

Methodology

Our interest rate predictor project will go through the data science life cycle. In addition, we will use different Python libraries and tools in each step as we describe below.


.CSV files
Data Ingestion and loading
Data Munging and Wrangling
amazon
RDS
PostgreSQL
Pandas
Computation and Analyses
matplotlib
Pandas
Modeling and Application
Scikit-learn
machine learning modules in Python
NumPy
Logistic Regression(LR)
K-Nearest Neighbors(KNN)
Classification and Regression trees(CART)
Linear Discriminant Analysis(LDA)
Reporting and Visualization
matplotlib
jupyter

1. Data selection

We downloaded the data from https://www.lendingclub.com/info/download-data.action in CSV format. The data set consists of strings, integers, and float data types. The data set includes information about applications, loans, credit history, and payment history. For the purpose of this project, we will exclude payment history data because this measures post-funding loan performance. In addition, we will exclude declined applications data because this data is inconsistent with the approved applications data. The selected data set contains 1,765,428 unique loan applications from 2007 to 2017. Consumers provide the following information when they apply for a loan with LendingClub:

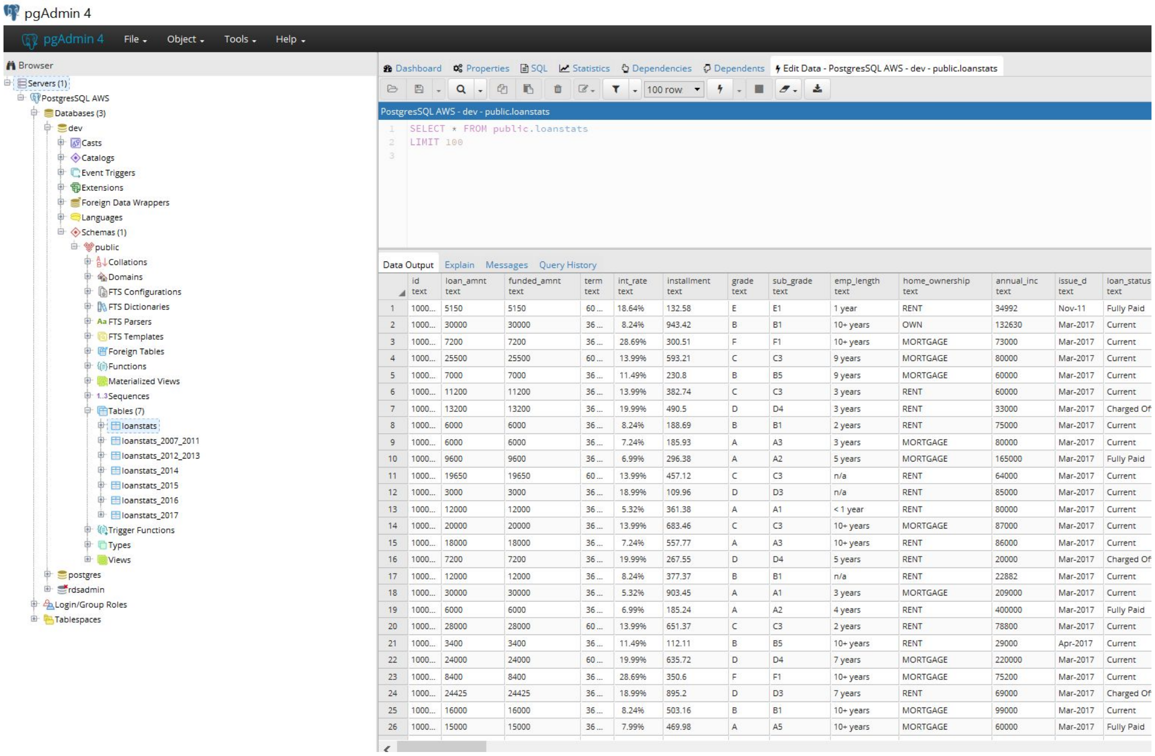
1. Desired loan amount
2. Loan purpose
3. Number of guarantors
4. Date of birth
5. Individual income (verifiable)
6. Additional income
7. Full name
8. Address

2. Data ingestion and loading

For the ingestion process, we loaded data from CSV files into tables in a PostgreSQL database on Amazon RDS. Initially, we had six CSV files, each containing a year's worth of data. We created six tables in the PostgreSQL database using SQL, one for each CSV file. We used pgAdmin to populate the tables with the data from the CSV files. We added the columns "year" and "quarter" to give the data chronological context.

After familiarizing ourselves with the data, we decided to merge the six tables into one table for maximum efficiency. Once we merged the tables, we exported the data into a CSV file and stored the CSV file in a secure place, consistent with best practices from the data ingestion curriculum. After creating one table, we deleted all non-relevant fields in the table.

Below is a screenshot showing the six tables on the left and the first 100 rows of the merged table "loanstats."



```
SELECT * FROM public.loanstats
LIMIT 100
```

	id text	loan_amnt text	funded_amnt text	term text	int_rate text	installment text	grade text	sub_grade text	emp_length text	home_ownership text	annual_inc text	issue_d text	loan_status text
1	1000...	5150	5150	60 ...	18.64%	132.58	E	E1	1 year	RENT	34992	Nov-11	Fully Paid
2	1000...	30000	30000	36 ...	8.24%	943.42	B	B1	10+ years	OWN	132630	Mar-2017	Current
3	1000...	7200	7200	36 ...	28.69%	300.51	F	F1	10+ years	MORTGAGE	73000	Mar-2017	Current
4	1000...	25500	25500	60 ...	13.99%	593.21	C	C3	9 years	MORTGAGE	80000	Mar-2017	Current
5	1000...	7000	7000	36 ...	11.49%	230.8	B	B5	9 years	MORTGAGE	60000	Mar-2017	Current
6	1000...	11200	11200	36 ...	13.99%	382.74	C	C3	3 years	RENT	60000	Mar-2017	Current
7	1000...	13200	13200	36 ...	19.99%	490.5	D	D4	3 years	RENT	33000	Mar-2017	Charged Of
8	1000...	6000	6000	36 ...	8.24%	188.69	B	B1	2 years	RENT	75000	Mar-2017	Current
9	1000...	6000	6000	36 ...	7.24%	185.93	A	A3	3 years	MORTGAGE	80000	Mar-2017	Current
10	1000...	9600	9600	36 ...	6.99%	296.38	A	A2	5 years	MORTGAGE	165000	Mar-2017	Fully Paid
11	1000...	19650	19650	60 ...	13.99%	457.12	C	C3	n/a	RENT	64000	Mar-2017	Current
12	1000...	3000	3000	36 ...	18.99%	109.96	D	D3	n/a	RENT	85000	Mar-2017	Current
13	1000...	12000	12000	36 ...	5.32%	361.38	A	A1	< 1 year	RENT	80000	Mar-2017	Current
14	1000...	20000	20000	36 ...	13.99%	683.46	C	C3	10+ years	MORTGAGE	87000	Mar-2017	Current
15	1000...	18000	18000	36 ...	7.24%	557.77	A	A3	10+ years	RENT	86000	Mar-2017	Current
16	1000...	7200	7200	36 ...	19.99%	267.55	D	D4	5 years	RENT	20000	Mar-2017	Charged Of
17	1000...	12000	12000	36 ...	8.24%	377.37	B	B1	n/a	RENT	22882	Mar-2017	Current
18	1000...	30000	30000	36 ...	5.32%	903.45	A	A1	3 years	MORTGAGE	209000	Mar-2017	Current
19	1000...	6000	6000	36 ...	6.99%	185.24	A	A2	4 years	RENT	400000	Mar-2017	Fully Paid
20	1000...	28000	28000	60 ...	13.99%	651.37	C	C3	2 years	RENT	78800	Mar-2017	Current
21	1000...	3400	3400	36 ...	11.49%	112.11	B	B5	10+ years	RENT	29000	Apr-2017	Current
22	1000...	24000	24000	60 ...	19.99%	635.72	D	D4	7 years	MORTGAGE	220000	Mar-2017	Current
23	1000...	8400	8400	36 ...	28.69%	350.6	F	F1	10+ years	MORTGAGE	75200	Mar-2017	Current
24	1000...	24425	24425	36 ...	18.99%	895.2	D	D3	7 years	RENT	69000	Mar-2017	Charged Of
25	1000...	16000	16000	36 ...	8.24%	503.16	B	B1	10+ years	MORTGAGE	99000	Mar-2017	Current
26	1000...	15000	15000	36 ...	7.99%	469.98	A	A5	10+ years	MORTGAGE	60000	Mar-2017	Fully Paid

3. Data munging and wrangling

In our data munging and wrangling process, we pulled data from PostgreSQL into Pandas DataFrames and explored the data. We verified the data, removed irrelevant features, addressed null values, and encoded categorical variables.

3.1 Data verification

In the data verification phase, we defined our target data. Up to this point, our data contained 235,629 instances and 153 features. Among the features was loan type, defined either as a joint or individual loan application. After careful consideration, we dropped the data points for joint applications, leaving instances for individual applications. Pooling joint and individual applications would affect calculation accuracy because joint and individual loan application evaluation processes use different credit models.

3.2 Removing irrelevant features

We determined features containing data about the loan after LendingClub granted it to the customer were irrelevant to our focus on interest rate prediction. We dropped these post-loan features and only retained features containing data on the loan application.

We used the data dictionary to identify other irrelevant features and dropped those features. For example, we used the data dictionary to determine the feature “funded amount,” the sum LendingClub pays the customer, was irrelevant because it had the same value as the feature “loan amount.”

3.3 Addressing null values

Very few of our main variables contained null values. When a main variable contained a null value, we examined its data dictionary description to determine the meaning of a null value for the specific variable. We then decided whether to drop or replace the variable.

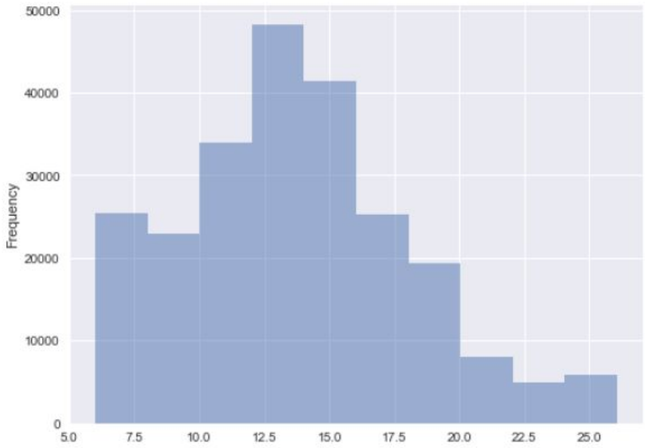
For example, the variable “employment length” had a null value. The data dictionary stated null values represented unemployment so we replaced the null values with zeros.

3.4 Formatting

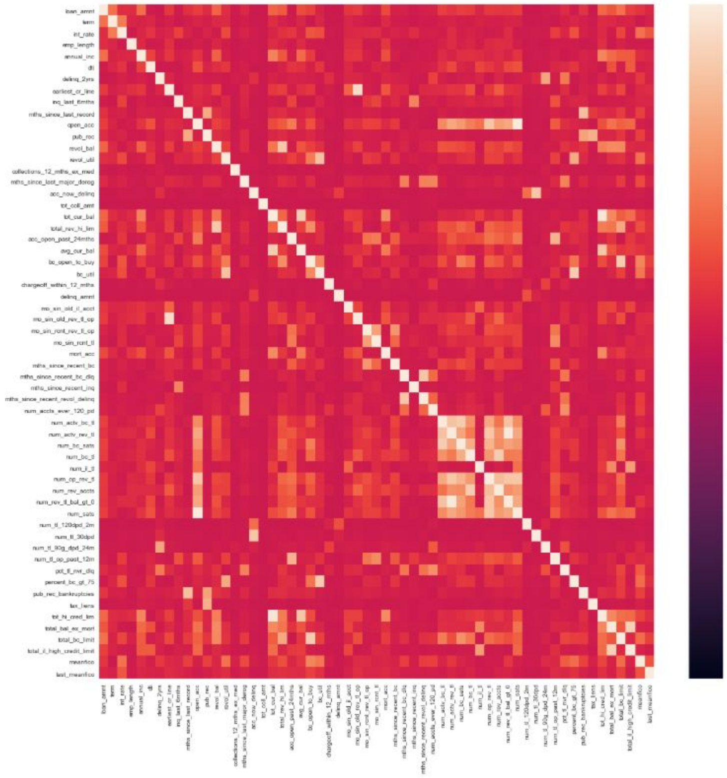
Our data contains both numerical and categorical variables. We converted the categorical variables to numerical values to understand their relationship with the interest rate using scikit-learn’s LabelEncoder. For other variables, we removed the string element from the values. For example, we converted a value like “36 months” to “36.” We also removed leading spaces and trailing spaces in some variables.

4. Computation and analysis

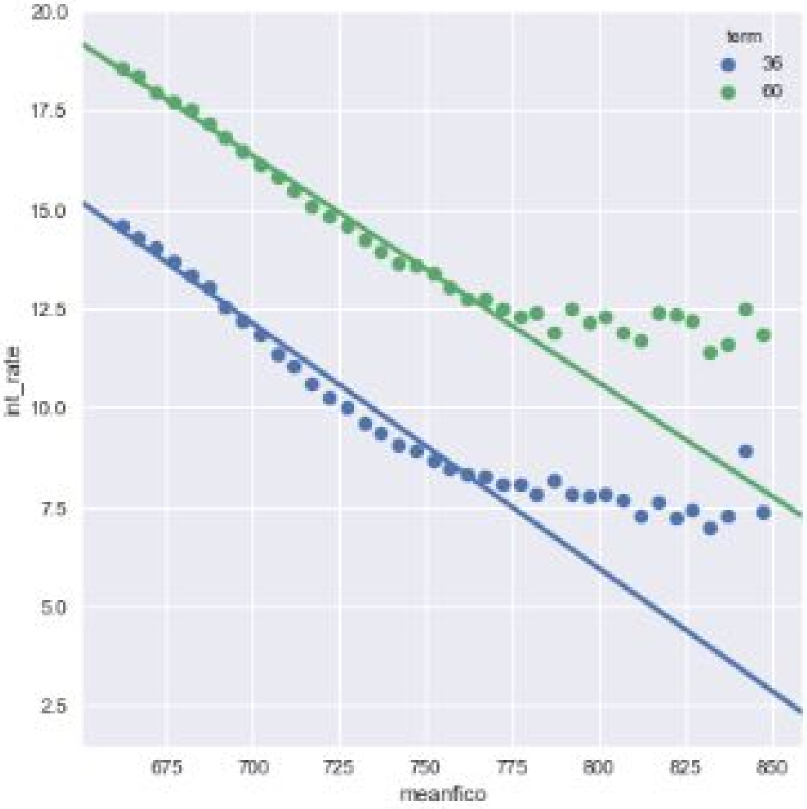
We started our computational analysis with the remaining features with numerical values and then encoded the categorical variables. We used seaborn with Pandas to extract and analyze the information. We utilized a histogram chart to determine the interest rate distribution, which showed a normalized distribution.



We used a correlation heat chart to visualize the features' relationship with the interest rate.



The correlation heat chart showed a relatively high correlation between the average FICO score and the interest rate. We expected this high correlation because most interest rate models use FICO data. We used a regression chart to understand further the relationship between the average FICO score and the interest rate. The regression also parses the data by 36-month and 60-month loan term lengths.



To conclude our computational analysis, we evaluated data with categorical values to determine if it was significant and relevant to our study. We dropped all variables with a high number of null values as well as all features with data irrelevant to our models. This process increased our number of features from 69 to 131.

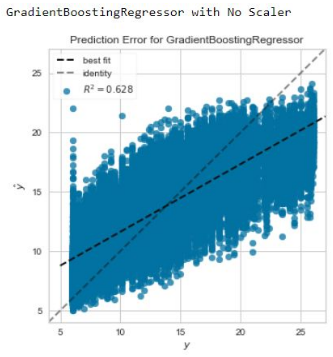
We saved the altered data in a separate SQL database as a secure backup. Although we did not face the challenge of lost or corrupted data, we determined we should retain a copy of both our original and finalized data as a best practice.

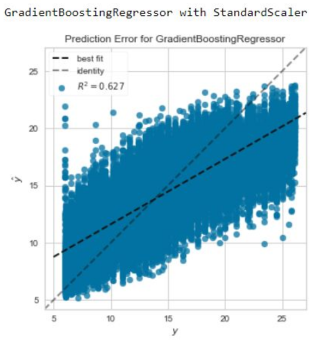
5. Modeling and application

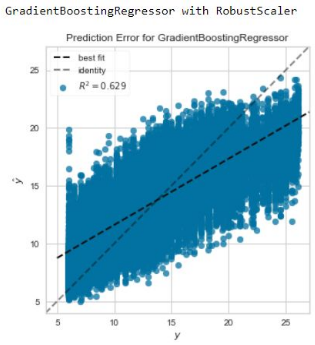
5.1 Regression modelling

After data wrangling, our dataset contained 235,629 instances and 131 features. First, we split the dataset in two pieces. We used the largest piece of the dataset (80 percent) to train our models and held the smaller piece (20 percent) as a validation dataset. We built different models so we could experiment and identify the best model with the highest estimated accuracy score. We used scikit-learn to create a predictive model from our dataset.

During the modeling process, we experimented with different approaches including scaling the data to have consistent ranges, smoothing outliers, and dropping variables with a high number of zeros. The charts below show the data prior to scaling and after scaling with StandardScaler and RobustScaler. We applied the process to all models (the Jupyter notebook contains the charts).





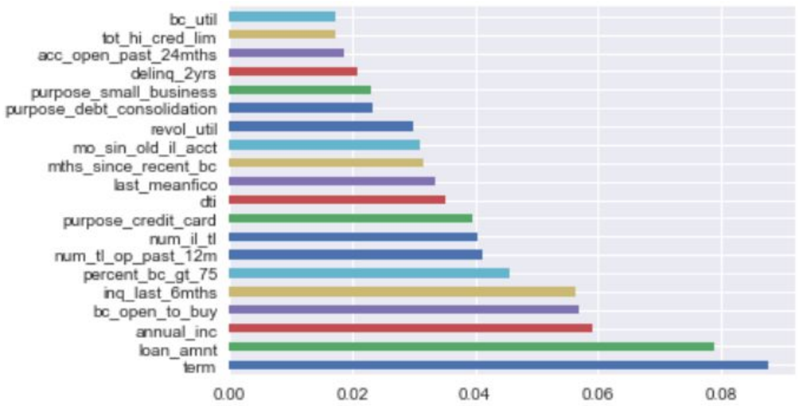


The table below shows the results of each model. The gradient boosting model was the most accurate model the model with the lowest error. We used scikit-learn's GridSearchCV to improve the parameters of the gradient boosting model. As a result, the coefficient of determination increased to 0.649 and the mean squared error decreased to 6.595.

MODEL	MEAN SQUARED ERROR	COEFFICIENT OF DETERMINATION
LinearRegression	7.579	0.595
Ridge	7.553	0.595
RidgeCV	7.741	0.586
LassoCV	9.027	0.519
ElasticNetCV	9.843	0.474
RandomForestRegressor	8.346	0.556

GradientBoostingRegressor	7.482	0.601
Improved GradientBoostingRegressor	6.595	0.649

Using the important feature plot for the gradient boosting model, we obtained the 20 most important features. The chart shows loan amount and term are key elements of the model. We expected these results because of the centrality of loan amount and term to LendingClub's overall interest rate determination and loan offer process.



We continued to experiment with the gradient boosting model. From the list above, we narrowed the selection of variables, choosing only variables other interest rate calculators used and also were easy for customers to remember. We selected loan amount, term, employment length, home ownership, annual income, loan purpose, mean FICO, revolving line utilization rate, inquiries in the last six months, debt to income ratio, and delinquencies in the last two years. Nevertheless, the coefficient of this secondary gradient boosting model decreased to 0.582.

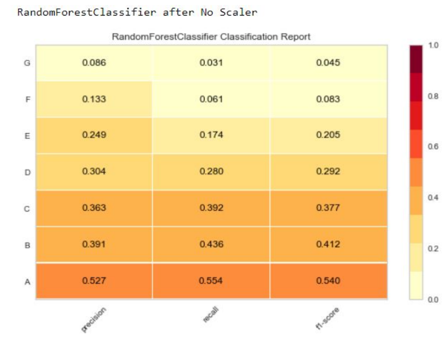
5.2 Classification modelling

Our third attempt to create a simple model with only seven variables involved changing our changing our model-building approach from regression to classification. LendingClub uses their historical data, credit risk, and market conditions to calculate a loan grade. The grades are letters from "A" to "G" and each letter has sub-categories. We used their loan grade as our target. The table below describes the interest rate ranges of each grade.

Loan grade	Interest rate range
A	5.31% - 8.08%
B	9.58% - 12.13%
C	13.06% - 16.46%

D	17.47% - 21.85%
E	22.90% - 26.77%
F	28.72% - 30.75%
G	30.79% - 30.99%

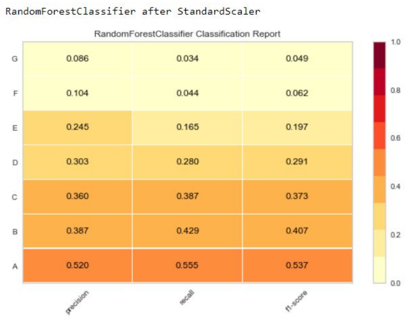
Similar to our regression modelling process, we tried different models in their original state and after applying RobustScaler. We fixed the class imbalance with imbalanced-learn's imblearn.combine.SMOTEENN class.



RandomForestClassifier after No Scaler

RandomForestClassifier Classification Report

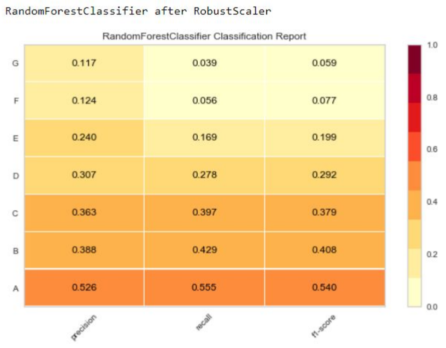
	precision	recall	f1-score
G	0.086	0.031	0.045
F	0.133	0.061	0.083
E	0.249	0.174	0.205
D	0.304	0.280	0.292
C	0.363	0.392	0.377
B	0.391	0.436	0.412
A	0.527	0.554	0.540



RandomForestClassifier after StandardScaler

RandomForestClassifier Classification Report

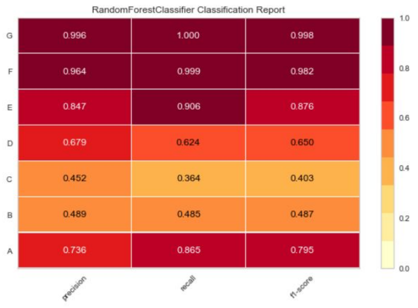
	precision	recall	f1-score
G	0.086	0.034	0.049
F	0.104	0.044	0.062
E	0.245	0.165	0.197
D	0.303	0.280	0.291
C	0.360	0.387	0.373
B	0.387	0.429	0.407
A	0.520	0.555	0.537



RandomForestClassifier after RobustScaler

RandomForestClassifier Classification Report

	precision	recall	f1-score
G	0.117	0.039	0.059
F	0.124	0.056	0.077
E	0.240	0.169	0.199
D	0.307	0.278	0.292
C	0.363	0.397	0.379
B	0.388	0.429	0.408
A	0.526	0.555	0.540



RandomForestClassifier Classification Report

	precision	recall	f1-score
G	0.996	1.000	0.998
F	0.964	0.999	0.982
E	0.847	0.906	0.876
D	0.679	0.624	0.650
C	0.452	0.364	0.403
B	0.489	0.485	0.487
A	0.736	0.865	0.795

The table below shows the classification model results.

MODEL	PRECISION	Accuracy	F1
LinearSVC	0.44	0.53	0.44
LogisticRegressionCV	0.43	0.53	0.44
RandomForestClassifier	0.78	0.79	0.78
GradientBoostingClassifier	0.58	0.60	0.57

The results showed the best classification model is RandomForestClassifier with a 0.78 F1 score.

Results analysis

The gradient boosting model uses all 131 features. If we were to use the gradient boosting model for our interest rate calculator, potential customers will receive an interest rate that is 65 percent accurate. However, it is overly complicated for customers to input so many features especially as few customers have extensive financial history data on themselves.

One of our hypotheses was to construct a simple model with no more than 15 features that reliably calculated an interest rate prediction. However, no combination or option we tried give us a reliable model. Even when we only used variables we found to be the easiest and most widely used in other studies, the model still was not significant. For our third model, we took the same variables and shifted to a classification approach with loan grade as the target. The classification model showed better scores than the regression model. The random forest classifier model was the best model for this approach. With the random forest model allows us to construct a simple interest rate calculator with few features and use LendingClub's loan grading scale.

Conclusion

We determined calculating an accurate interest rate using a regression model required more data than we used for this study. The classification model showed better performance and used only 11 variables. This makes it simple for customers and our data product.

We determined the key variables of the model are loan amount, term, employment length, home ownership, annual income, loan purpose, mean FICO, revolving line utilization rate, inquiries in the last six months, debt-to-income ratio, and delinquencies in the last two years. It is possible to create a simple, reliable model with classification and LendingClub's loan grade scale.

Challenges

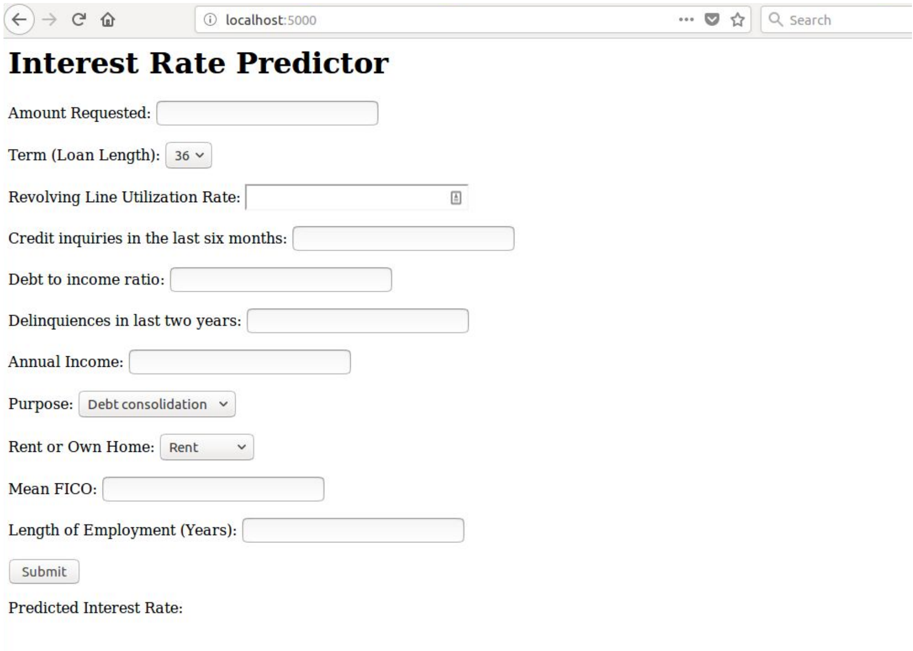
We originally intended to use all LendingClub data from 2007 to 2016 but as we started the pipeline, we realized there was inconsistency, seasonality, and noise because LendingClub frequently updates their interest rate model. We decided to examine only 2017 data because we expected it to provide the most recent and updated scores. Still, the 2017 data had many inconsistencies. In addition, we had a high number of instances and features, which made the machine learning process long and slow. We lacked the computer hardware strength we needed to process our data. In some cases, running models took more than eight hours. Finally, we did not obtain our desired regression model score from which we concluded other external datasets are necessary to improve the model.

Potential future work

For students who desire to improve or continue our study, we recommend considering adding additional data, such as demographic data, to the existing customer and economic data. We also recommend better computer hardware capacity and different models such as neural networks.

Data product prototype

The data product prototype is a web application allowing the user to input financial data and receive a predicted interest rate range. The application utilizes the Flask framework.



Interest Rate Predictor

Amount Requested:

Term (Loan Length): 36

Revolving Line Utilization Rate:

Credit inquiries in the last six months:

Debt to income ratio:

Delinquiences in last two years:

Annual Income:

Purpose: Debt consolidation

Rent or Own Home: Rent

Mean FICO:

Length of Employment (Years):

Submit

Predicted Interest Rate:

Bibliography

https://www.lendingclub.com/info/demand-and-credit-profile.action
https://www.lendingclub.com/foliofn/rateDetail.action
https://tradingeconomics.com/united-states/lending-interest-rate-percent-wb-data.html
https://www.lendingclub.com/info/demand-and-credit-profile.action
https://www.lendingclub.com/info/statistics.action
http://www.lendstats.com
https://data.worldbank.org/indicator/FR.INR.LEND?contextual=default&locations=US&view=chart
https://www.growingfamilybenefits.com/credit-scores-interest-rates-relationship/
https://www.lendingclub.com/public/investing-faq.action