LendingClub

Group 6: Haoyuan Huang, Hang He, Ken Qin

Agenda

- Lending Club & Team Overview
- Data Visualization
- Data Wrangling and Model Selection
- Prediction & Recommendation
- Deployee on AWS

Company Overview

LendingClub is a US peer-to-peer lending company, headquartered in San Francisco, California. It was the first peer-to-peer lender to register its offerings as securities with the Securities and Exchange Commission (SEC), and to offer loan trading on a secondary market. LendingClub is the world's largest peer-to-peer lending platform.

Solving this case study will give us an idea about how real business problems are solved using EDA and Machine Learning. In this case study, we will also develop a basic understanding of risk analytics in banking and financial services and understand how data is used to minimise the risk of losing money while lending to customers.

Team: Our Mission

We work for the 'Lending Club' company which specialises in lending various types of loans to urban customers. When the company receives a loan application, the company has to make a decision for loan approval based on the applicant’s profile. Two types of risks are associated with the bank’s decision: If the applicant is likely to repay the loan, then not approving the loan results in a loss of business to the company. If the applicant is not likely to repay the loan, i.e. he/she is likely to default, then approving the loan may lead to a financial loss for the company. The data given contains the information about past loan applicants and whether they ‘defaulted’ or not.

Business question

How to lower the default rate?

The aim is to identify patterns which indicate if a person is likely to default, which may be used for taking actions such as denying the loan, reducing the amount of loan, lending (to risky applicants) at a higher interest rate, etc.

How to growth client’s business?

Data description

1. credit_policy: 1 if the customer meets the credit underwriting criteria of LendingClub.com, and 0 otherwise.

2. purpose: The purpose of the loan such as: credit_card, debt_consolidation, etc.

3. int_rate: The interest rate of the loan (proportion).

4. installment: The monthly installments ($) owed by the borrower if the loan is funded.

5. log_annual_inc: The natural log of the annual income of the borrower.

6. dti: The debt-to-income ratio of the borrower.

7. fico: The FICO credit score of the borrower.

8. days_with_cr_line: The number of days the borrower has had a credit line.

9. revol_bal: The borrower's revolving balance.

10. revol_util: The borrower's revolving line utilization rate.

11. inq_last_6mths: The borrower's number of inquiries by creditors in the last 6 months.

12. delinq_2yrs: The number of times the borrower had been 30+ days past due on a payment in the past 2 years.

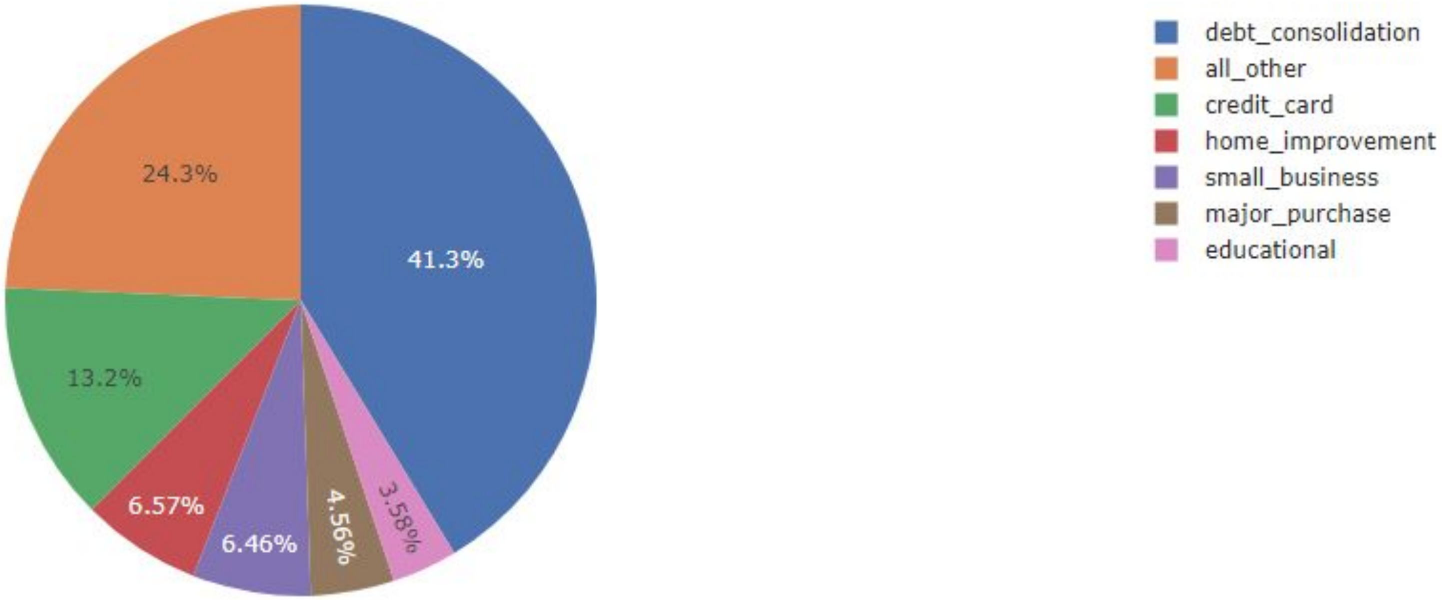
13. pub_rec: The borrower's number of derogatory public records.

14. not_fully_paid: indicates whether the loan was not paid back in full (the borrower either defaulted or the borrower was deemed unlikely to pay it back).

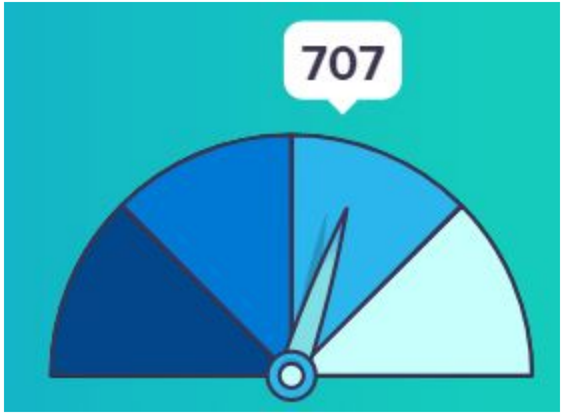
Data Frame Overview-labeled Data

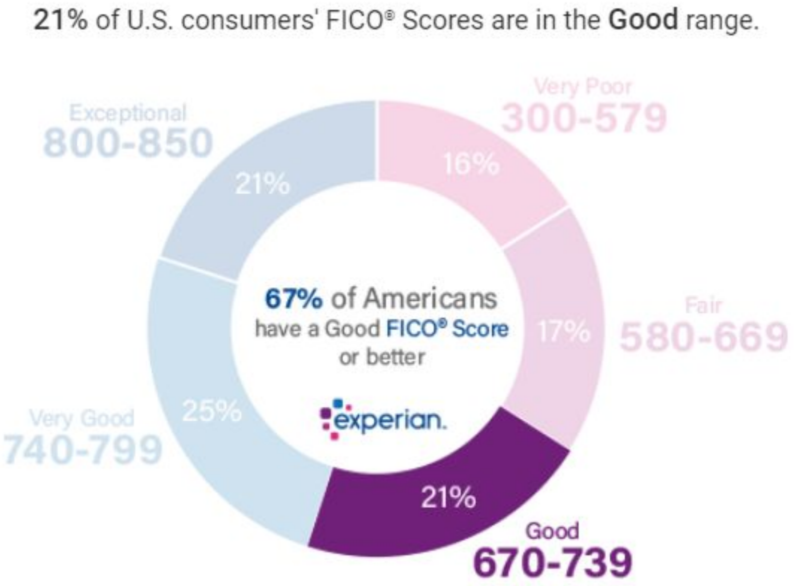
credit_policy	purpose	int_rate	installment	log_annual_inc	dti	fico	days_with_cr_line	revol_bal	revol_util	inq_last_6mths	delinq_2yrs	pub_rec	not_fully_paid
1	debt_consolidation	0.1189	829.1	11.35040654	19.48	737	5639.958333	28854	52.1	0	0	0	0
1	credit_card	0.1071	228.22	11.08214255	14.29	707	2760	33623	76.7	0	0	0	0
1	debt_consolidation	0.1357	366.86	10.37349118	11.63	682	4710	3511	25.6	1	0	0	0
1	debt_consolidation	0.1008	162.34	11.35040654	8.1	712	2699.958333	33667	73.2	1	0	0	0
1	credit_card	0.1426	102.92	11.29973224	14.97	667	4066	4740	39.5	0	1	0	0
1	credit_card	0.0788	125.13	11.90496755	16.98	727	6120.041667	50807	51	0	0	0	0
1	debt_consolidation	0.1496	194.02	10.71441777	4	667	3180.041667	3839	76.8	0	0	1	1
1	all_other	0.1114	131.22	11.00209984	11.08	722	5116	24220	68.6	0	0	0	1
1	home_improvement	0.1134	87.19	11.40756495	17.25	682	3989	69909	51.1	1	0	0	0
1	debt_consolidation	0.1221	84.12	10.20359214	10	707	2730.041667	5630	23	1	0	0	0
1	debt_consolidation	0.1347	360.43	10.4341158	22.09	677	6713.041667	13846	71	2	0	1	0
1	debt_consolidation	0.1324	253.58	11.83500896	9.16	662	4298	5122	18.2	2	1	0	0
1	debt_consolidation	0.0859	316.11	10.93310697	15.49	767	6519.958333	6068	16.7	0	0	0	0
1	small_business	0.0714	92.82	11.51292546	6.5	747	4384	3021	4.8	0	1	0	0
1	debt_consolidation	0.0863	209.54	9.487972109	9.73	727	1559.958333	6282	44.6	0	0	0	0

Purpose of Borrowing



Median Fico Score





Data Wrangling

After the exploratory data analysis, there are no missing values and duplicate values in the dataset.

We choose two models from the scikit-learn package, which are logistic regression and random forest.

```
# How large is the dataset?
loan.shape
# Analogous to dim() in R.
```

```
(7182, 14)
```

```
# Any missing data?
loan.isnull().sum()
```

```
credit_policy        0
purpose              0
int_rate             0
installment          0
log_annual_inc       0
dti                  0
fico                 0
days_with_cr_line    0
revol_bal            0
revol_util           0
inq_last_6mths       0
delinq_2yrs          0
pub_rec              0
not_fully_paid       0
dtype: int64
```

Normalize/Standardize the data

Reason:

Regularization is an increasingly popular method for controlling overfitting. For example, by default the Logistic Regression in scikit-learn package uses L2 regularization

Normalization

In [6]:

```python
# recall the summary statistics
loan.agg(['mean','std','skew'])
```

Out[6]:

	credit_policy	int_rate	installment	log_annual_inc	dti	fico	days_with_cr_line	revol_bal	revol_util	inq_last_6mths	delinq_2yrs	pu
mean	0.804970	0.122640	319.089413	10.932117	12.606679	710.846314	4560.767197	16913.963876	46.799236	1.577469	0.163708	0.0
std	0.396245	0.026847	207.071301	0.614813	6.883970	37.970537	2496.930377	33756.189557	29.014417	2.200245	0.546215	0.2
skew	-1.539621	0.164420	0.912522	0.028668	0.023941	0.471260	1.155748	11.161058	0.059985	3.584151	6.061793	5.1

In [7]:

```python
# below we normalize/standardize some input columns
# in practice, remember to update your data description file afterwards!

loan['installment1000'] = loan.installment / 1000
loan.drop('installment', axis=1, inplace=True)

loan['fico_ratio'] = loan.fico / 850
loan.drop('fico', axis=1, inplace=True)

loan['decades_with_cr_line'] = loan.days_with_cr_line / 3650
loan.drop('days_with_cr_line', axis=1, inplace=True)

loan['log_revol_bal'] = np.log(loan.revol_bal + 1)
loan.drop('revol_bal', axis=1, inplace=True)

loan.revol_util = loan.revol_util / 100
```

In [8]:

```python
# double check the resulting data
loan.agg(['mean','std','skew'])
```

Out[8]:

	credit_policy	int_rate	log_annual_inc	dti	revol_util	inq_last_6mths	delinq_2yrs	pub_rec	not_fully_paid	installment1000	fico_ratio	decades
mean	0.804970	0.122640	10.932117	12.606679	0.467992	1.577469	0.163708	0.062122	0.160054	0.319089	0.836290	
std	0.396245	0.026847	0.614813	6.883970	0.290144	2.200245	0.546215	0.262126	0.366676	0.207071	0.044671	
skew	-1.539621	0.164420	0.028668	0.023941	0.059985	3.584151	6.061793	5.126434	1.854592	0.912522	0.471260	

Convert column “Purpose” to dummies

There are 7 possible values in column “purpose”.

1. Debt_consolidation,
2. All_other, (dropped)
3. Credit_card,
4. Home_improvement,
5. Small_business,
6. Major_purchase,
7. Educational

```
# create the dummies
loan = pd.get_dummies(loan, columns=['purpose'])
loan.head()
```

_all_other	purpose_credit_card	purpose_debt_consolidation	purpose_educational	purpose_home_improvement	purpose_major_purchase	purpose_small_business
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	0	1	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0

```
# drop one of the dummies (why?)
loan.drop('purpose_debt_consolidation', axis=1, inplace=True)
loan.head()
```

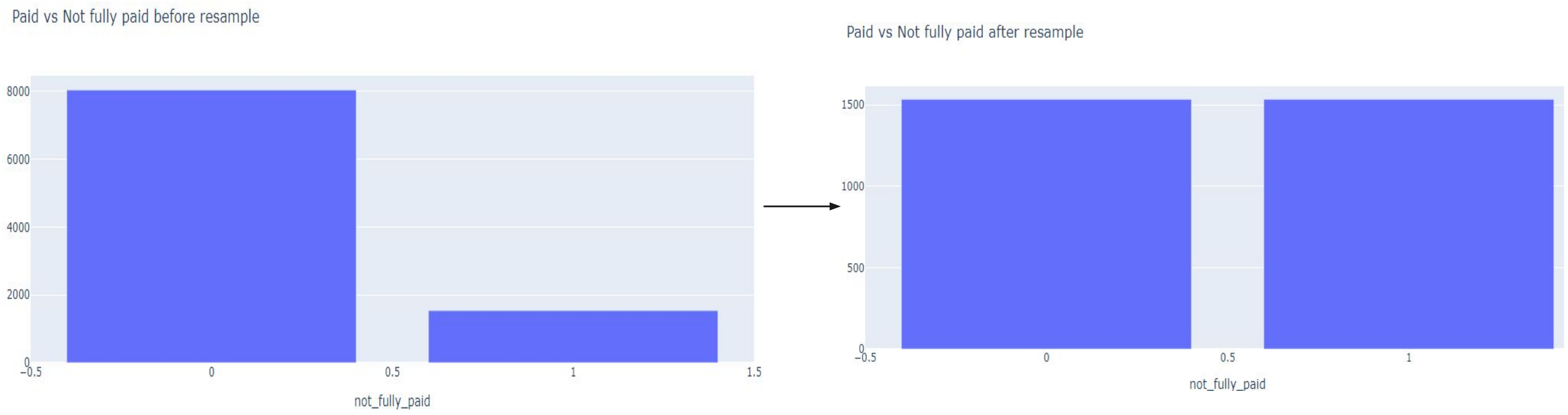
_line	log_revol_bal	purpose_all_other	purpose_credit_card	purpose_educational	purpose_home_improvement	purpose_major_purchase	purpose_small_business
5194	10.270039	0	0	0	0	0	0
5164	10.422995	0	1	0	0	0	0
0411	8.163941	0	0	0	0	0	0
9715	10.424303	0	0	0	0	0	0
3973	8.464003	0	1	0	0	0	0

Balance the Data - Not_Fully_Paid

In the independent variable(not_fully_paid), there are 8045 records equal to 0 (people who paid for their loan) vs 1533 records equal to 1(people who did not pay for their loans).

In order to balance the data, we have two options: undersampling the majority class and oversampling the minority class. We choose undersampling the majority class because of the data leakage issue of oversampling.

After we balanced the data, we have the same number for 0 and 1.


Paid vs Not fully paid before resample
8000
6000
4000
2000
0
−0.5
0
0.5
1
1.5
not_fully_paid
Paid vs Not fully paid after resample
1500
1000
500
0
−0.5
0
0.5
1
not_fully_paid

Modeling on the balanced data

Step 1: 80% as training dataset; 20% as testing dataset

Step 2: determine the hyperparameter values of the chosen learning algorithm

Step 3: feed the train dataset into the learning algorithm to get the trained model (i.e., the algorithm)

Model Selection

Random Forest

- max_depth=4
- accuracy : 64.01%

Logistic Regression

- penalty='L2'
- accuracy: 62.87%

The Random Forest has higher accuracy score so we will use the Random Forest for prediction

Importance of Factors

Top 5 Important Factors:

Int_rate: 0.189

Fico_ratio: 0.138

Credit_policy: 0.095

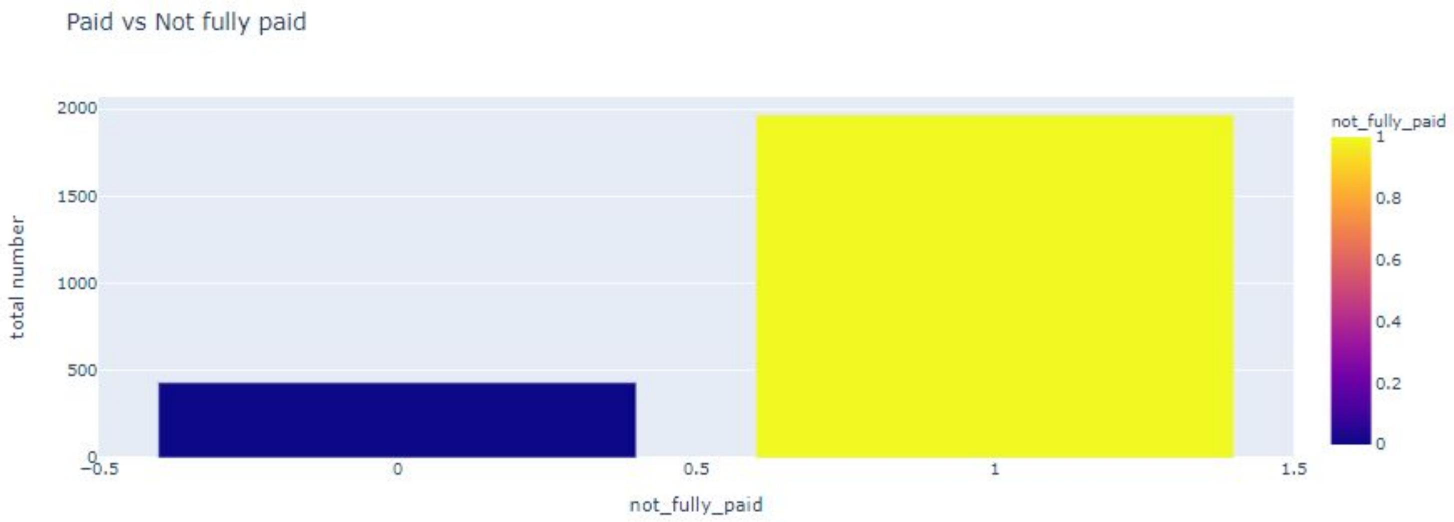
Inq_last_6mths: 0.095

Revol_util: 0.087

Data Frame Overview-Unlabeled Data

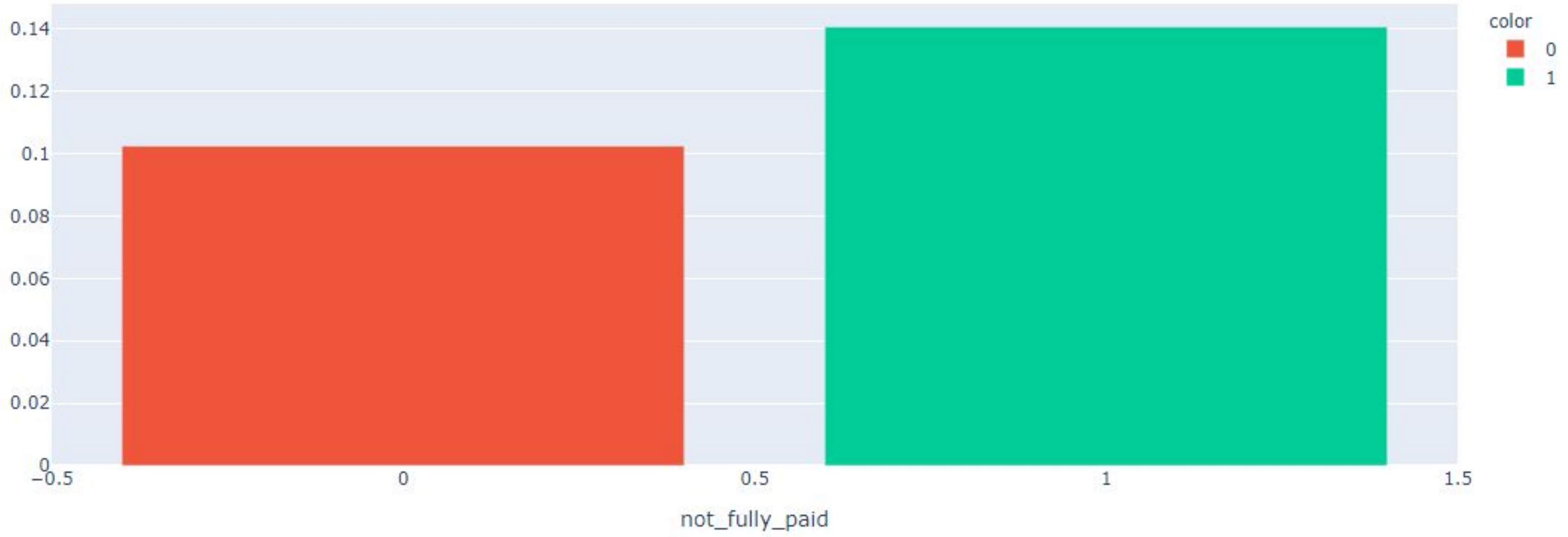
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
credit_policy	int_rate	log_annual_inc	dti	revol_util	inq_last_6mths	delinq_2yrs	pub_rec	installment1000	fico_ratio	decades_with_cr_line	log_revol_bal	purpos	purpos	purpos	purpose_	purpos	purpos
1	0.1273	10.46310334	5.11	0.388	0	0	0	0.26853	0.843529412	0.830148402	8.263332667	1	0	0	0	0	0
1	0.0751	10.08580911	10.8	0.017	3	0	0	0.3111	0.925882353	1.512328767	6.498282149	0	0	0	0	0	0
1	0.1099	10.27505111	23.88	0.588	2	0	0	0.2046	0.825882353	0.526038813	8.627660644	0	0	0	0	0	0
1	0.1348	11.26957921	3.6	0.442	5	0	0	0.59028	0.872941176	2.276997717	5.402677382	0	0	0	1	0	0
1	0.1273	10.18111929	11.91	0.337	2	1	0	0.25175	0.82	1.380833333	8.732788325	0	0	0	0	0	0
1	0.1025	11.69524702	14.2	0.49	1	0	0	0.48578	0.861176471	3.673984019	9.901635839	1	0	0	0	0	0
1	0.1062	11.21559702	23.54	0.796	3	0	0	0.20839	0.831764706	1.05206621	8.653994233	0	0	0	0	0	0
1	0.1385	11.3409502	21.72	0.744	0	0	0	0.83555	0.837647059	1.43836758	10.41373301	0	0	0	0	0	0
1	0.1348	10.77895629	3.13	0.396	1	0	0	0.16963	0.82	1.216438356	5.293304825	0	0	0	0	1	0
1	0.1025	11.23445167	15.46	0.186	0	0	0	0.71247	0.896470588	1.528778539	9.76117486	1	0	0	0	0	0
1	0.0774	11.03838355	22.99	0.421	0	0	0	0.11551	0.872941176	2.000547945	9.837828404	1	0	0	0	0	0
1	0.1062	10.82087768	16.18	0.562	3	0	0	0.4884	0.861176471	0.830148402	9.42270605	0	0	0	0	0	0
1	0.1062	11.40756495	24.16	0.625	0	0	0	0.39072	0.843529412	1.05206621	10.39671931	0	0	0	0	0	0
1	0.0988	10.35774282	20.76	0.398	3	0	0	0.14495	0.849411765	0.608219178	7.491645474	1	0	0	0	0	0
1	0.1025	10.57131693	20.86	0.391	0	0	0	0.32385	0.843529412	1.167134703	9.194718991	0	0	0	0	0	0
1	0.0714	11.36210258	3.42	0.125	0	0	0	0.18564	0.902352941	2.457545662	8.818926087	1	0	0	0	0	0
1	0.1459	10.858999	13.64	0.803	0	0	0	0.44805	0.796470588	0.780833333	9.261603666	0	0	0	0	0	0
1	0.0788	9.852194258	24.32	0.088	1	0	0	0.10949	0.843529412	3.246586759	7.765569081	0	0	0	0	0	0
1	0.1385	11.00209984	5.36	0.683	0	0	0	0.57295	0.82	0.575079909	9.335297611	1	0	0	0	0	0
1	0.1348	10.93310697	21.26	0.527	0	0	0	0.61064	0.82	1.602739726	10.21438554	0	0	0	0	0	0
1	0.0714	10.83958091	16.99	0.078	1	0	0	0.03094	0.896470588	1.528778539	6.603943825	0	0	0	0	1	0
1	0.1099	11.94470788	19.25	0.738	1	0	0	0.39283	0.837647059	2.07946347	11.87409031	0	0	0	0	0	1
1	0.0788	10.20359214	21.56	0.428	1	0	0	0.06257	0.843529412	0.608219178	9.029417836	1	0	0	0	0	0
1	0.1533	11.2515607	16.04	0.954	3	0	1	0.27863	0.796470588	1.65206621	10.14933149	0	0	0	0	0	0
1	0.1136	11.15665044	0.34	0.145	2	0	0	0.06583	0.82	0.591780822	5.38907173	1	0	0	0	0	0
1	0.1348	11.51292546	20.5	0.623	2	0	0	0.67849	0.825882353	1.027408676	10.19129462	0	0	0	0	0	0
1	0.0788	10.73639581	10.74	0.528	0	0	0	0.22835	0.855294118	0.764394977	8.996651979	0	0	0	0	0	0
1	0.1645	10.16585182	15.83	0.976	0	0	0	0.30071	0.796470588	0.403561644	9.138522084	0	0	0	0	0	0
1	0.1062	10.27505111	12.58	0.334	0	0	0	0.6512	0.896470588	1.126027397	8.701512751	0	0	0	0	0	0
1	0.1099	11.1124479	19.9	0.217	0	0	0	0.81838	0.884705882	1.150696347	9.035153304	0	0	0	0	0	0
1	0.1136	10.30895266	21.68	0.54	1	1	0	0.40317	0.825882353	0.936997717	9.621124642	0	0	0	0	0	0
1	0.1533	10.34174248	11.26	0.707	0	0	0	0.20897	0.784705882	0.427408676	8.602820277	0	0	0	0	0	0
1	0.0788	10.74720759	10.3	0.096	0	0	0	0.18769	0.914117647	1.258093607	7.307202315	1	0	0	0	0	0
1	0.1459	10.93310697	12.19	0.724	0	1	0	0.46011	0.796470588	0.895890411	9.149634497	0	0	0	0	0	0

Prediction Result

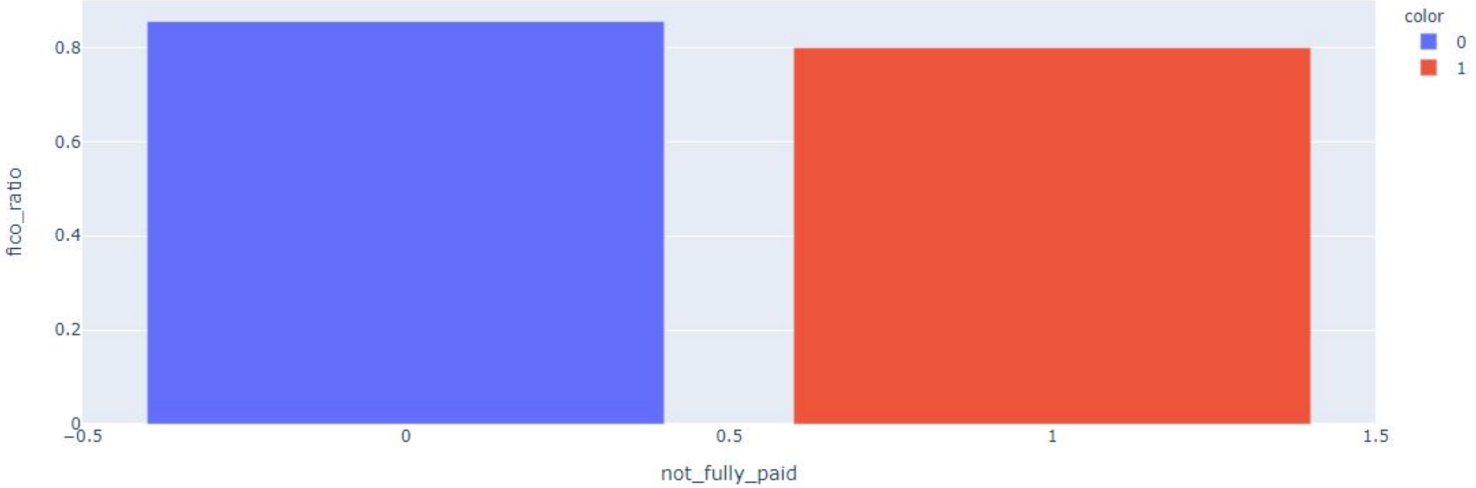


As the figure shown, not_fully_paid casstomers are larger than the fully_paid customers.

The lower the interest rate, the higher the chance that customers would pay back loan.



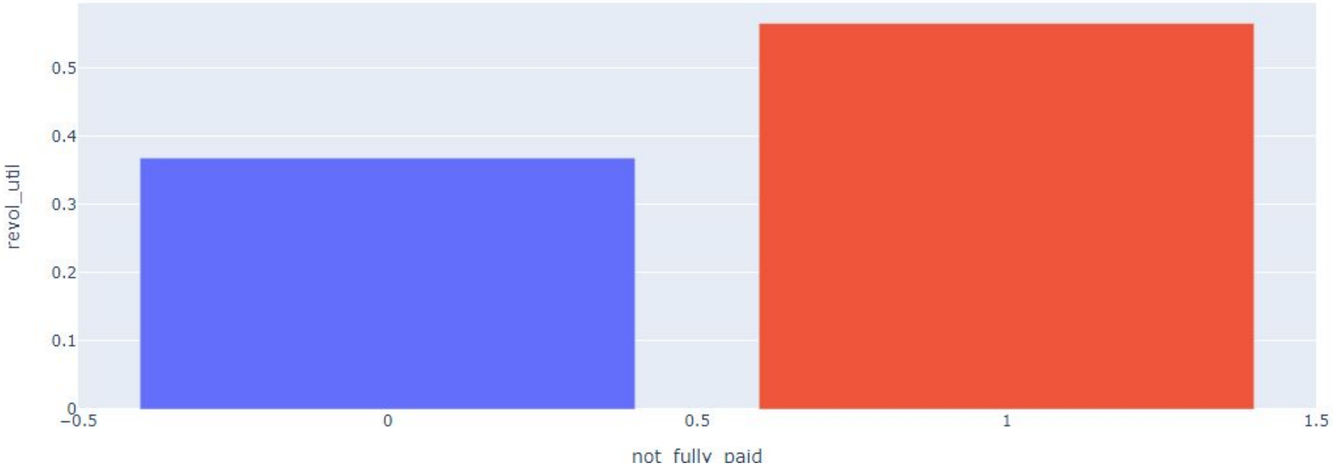
Fico Score



Median:

726 vs. 679

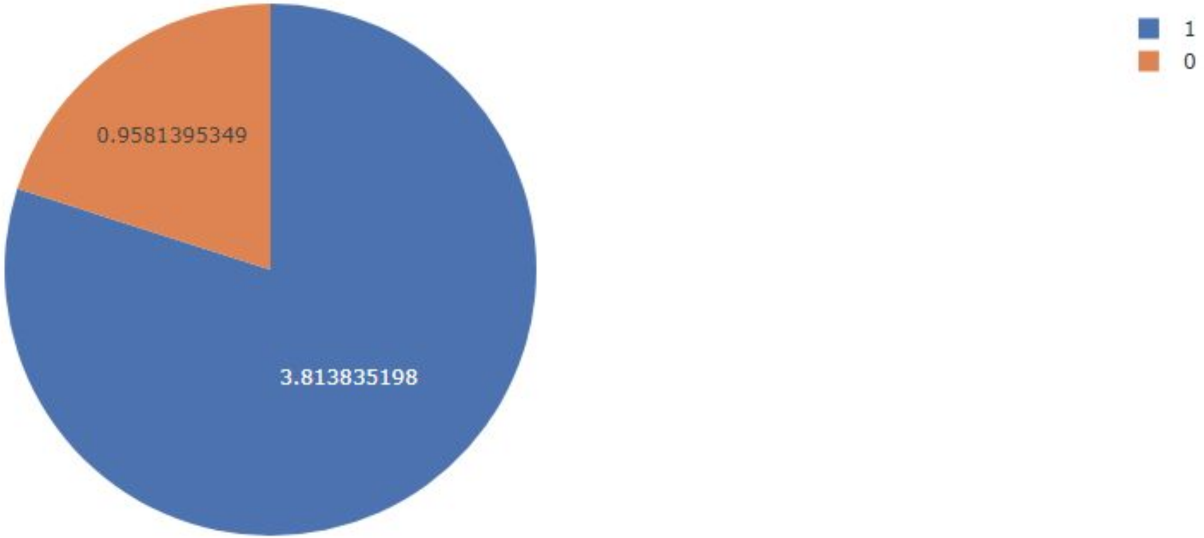
Borrower’s revolving line utilization rate



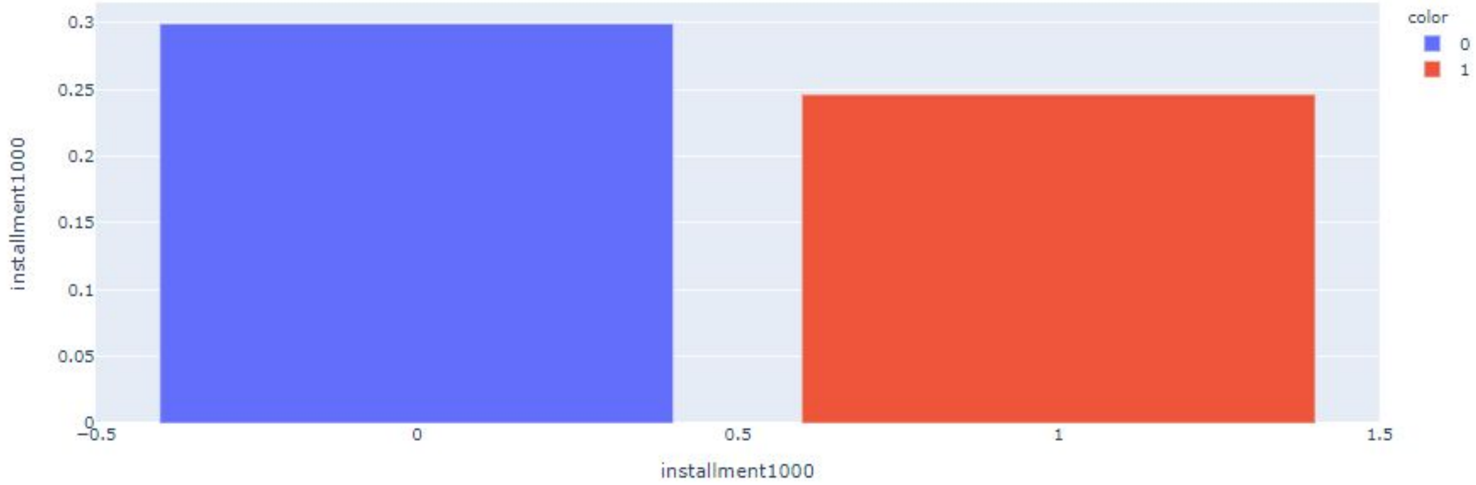
Median:

37% vs. 57%

Number of inquiries by creditors in the last 6 months



The monthly installment of paid customers is lower than the monthly installment of unpaid customers.



From the graph, we find that there is a difference in the median monthly installment between paid customers and not paid customers. The monthly installment of paid customers is lower than the monthly installment of unpaid customers.

Recommendation

- Find consumers who have a good financial position that can afford high monthly installment.
- Slightly lower the interest rate for consumers who are eligible for loan.
- Focus on consumers who borrow for debt consolidation, credit card, and home improvement