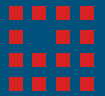


Lending Club

Classifying Loan Defaults

Michael Cho and Sathya Muthukkumar

Github:
https://github.com/michaelcho1
https://github.com/picnicpanic

Lending Club Loans Analysis

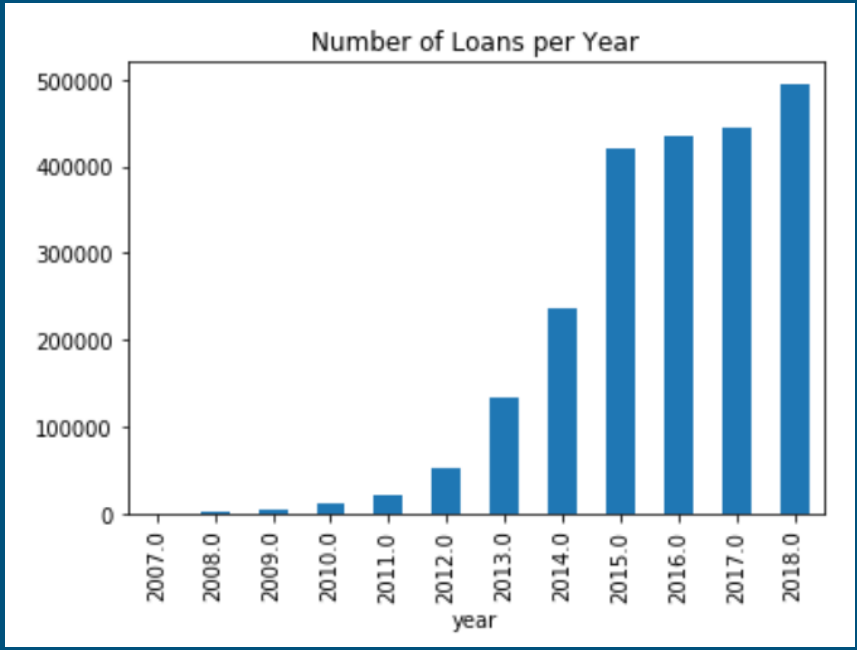
Time Period: June 2007 - December 2018.

Loan Terms Offered: 36 months & 60 months.

Number of Loans Offered: 400K-500K loans in the past four years.

Default Ratio: ~80% pay back their loans vs. ~20% default.

Prepayment Risk: Out of the loans that are fully paid, 85-90% prepay.



Our Position

We analyze and model the data from the perspective of investment advisors.

We run available lending club loans through our model, and tell our client whether or not we believe it would be a safe investment.

Instead of working directly for Lending Club, we work for a client looking to invest in loans on Lending Club.

We use only features available to these clients to develop the models. (See appendix for details)

Lending Club Loans Analysis

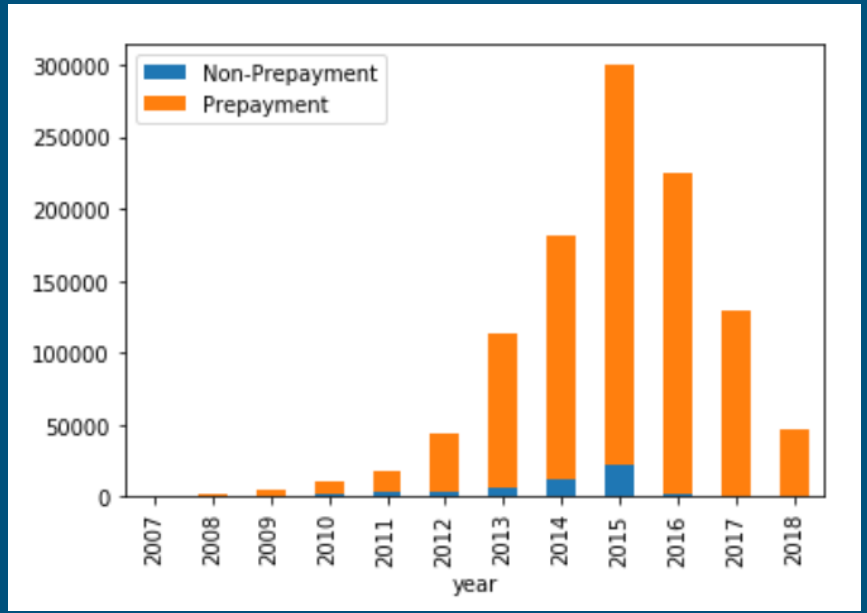
Debtors pay for an average duration of 21 months.

The average term duration is 42 months.

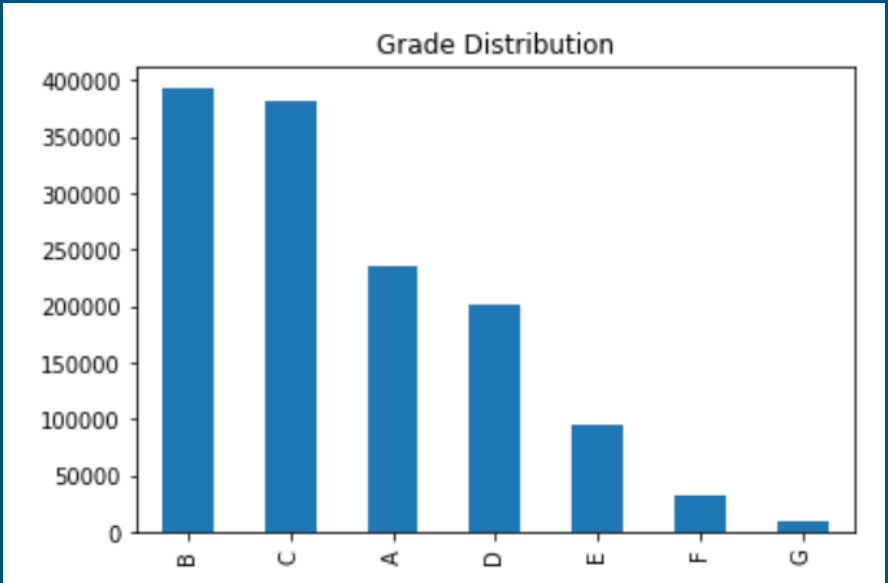
Out of the loans that are fully paid, 85-90% prepay.

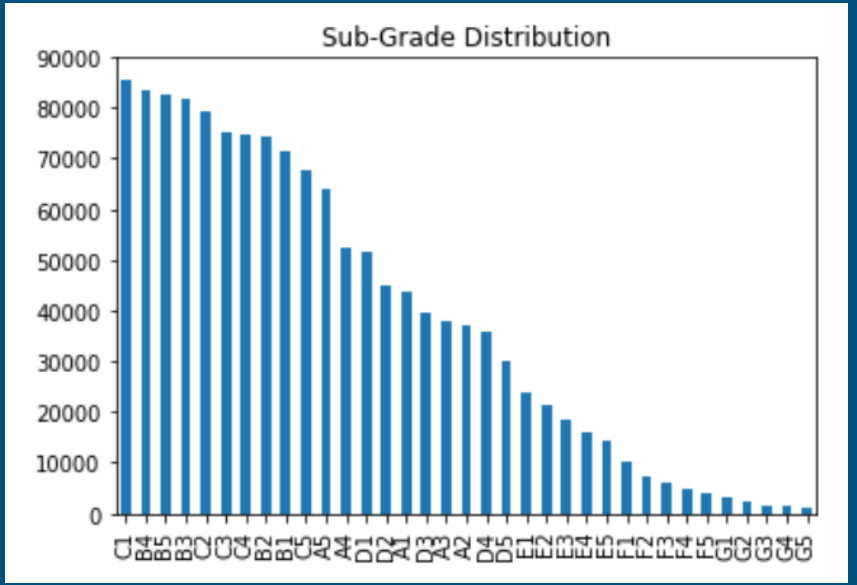
This means that the interest is less than if they paid over the full term.

This also means that defaults are partial, and in some cases, the debtor defaults at a point where their payment totals are higher than the principal.

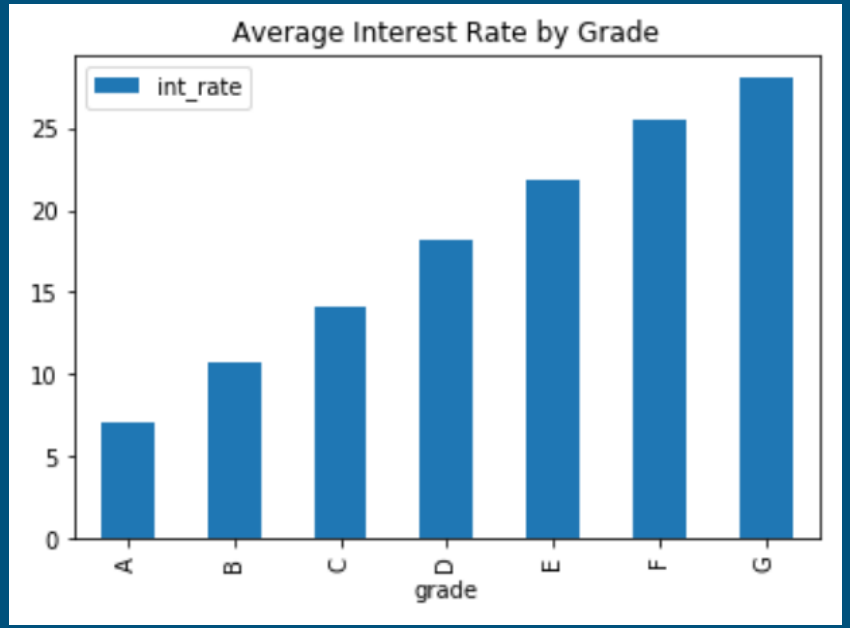


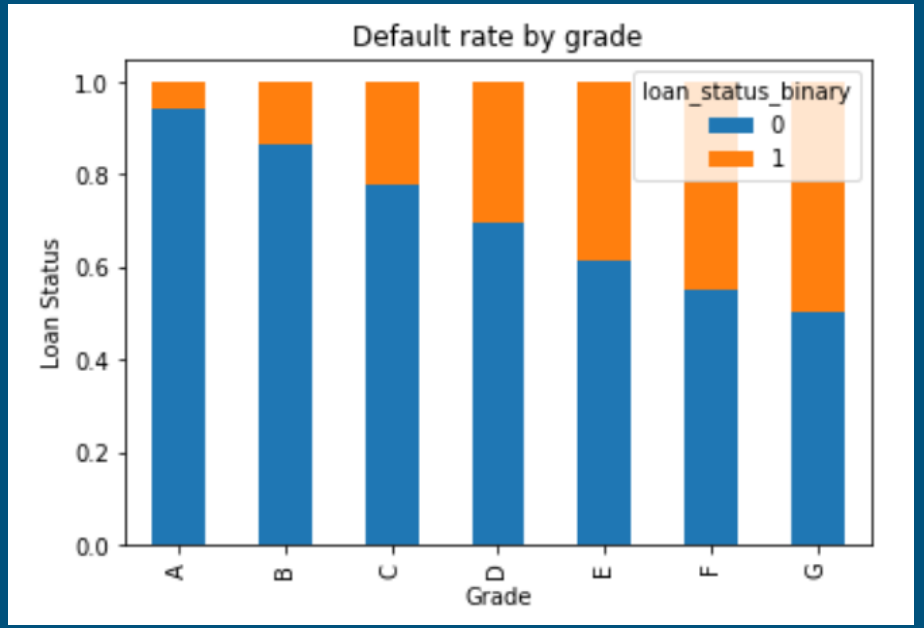
Lending Club Loans Analysis





Lending Club Loans Analysis





Positioning and Scoring

How do we score our models? Which model do we use for each client? Given a certain amount of money, which model returns the best?

We took the total payment feature and subtracted the loan amount feature. This resulting value is negative for most defaults, and positive for all fully paid loans.

This new calculated column, the interest collected, is summed for each model where it predicts a non-default. Similarly the loan amount, or principal, is summed.

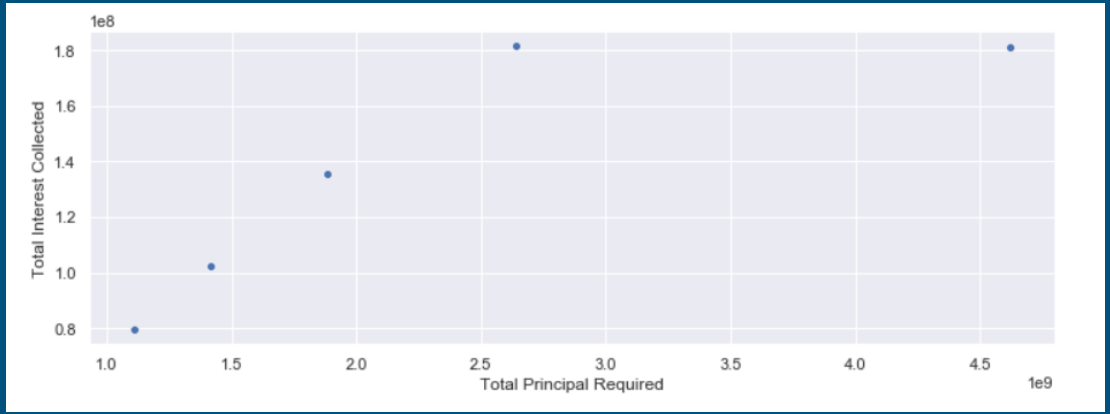
The ratio of these is our interest rate that the model scored, and is calculated on our test data.

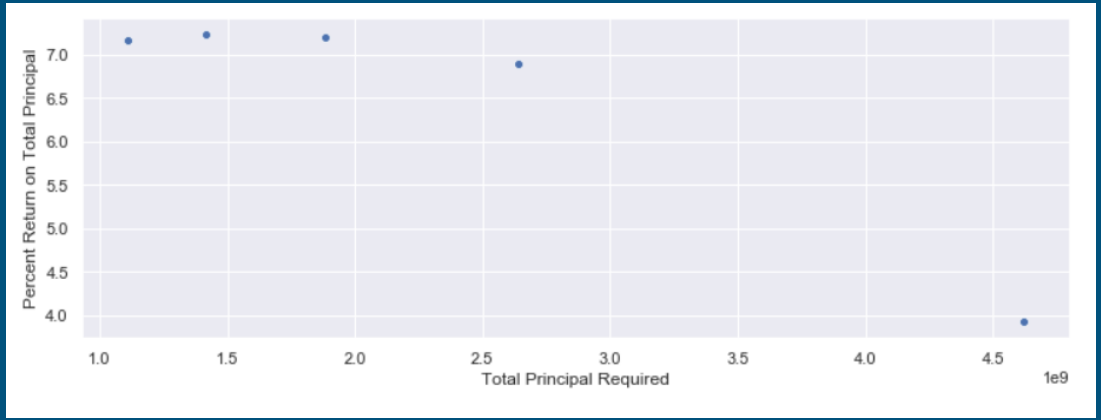
Model

We chose an XGBoost model.

Varying the class weight parameter yielded the most flexibility for us out of any other parameter.

Trade offs between percent interest returned, the raw value of interest returned, and the amount needed to be invested.

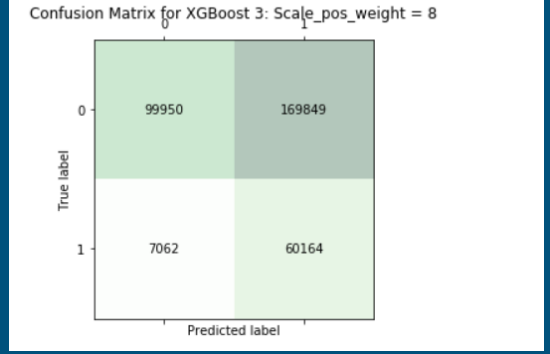


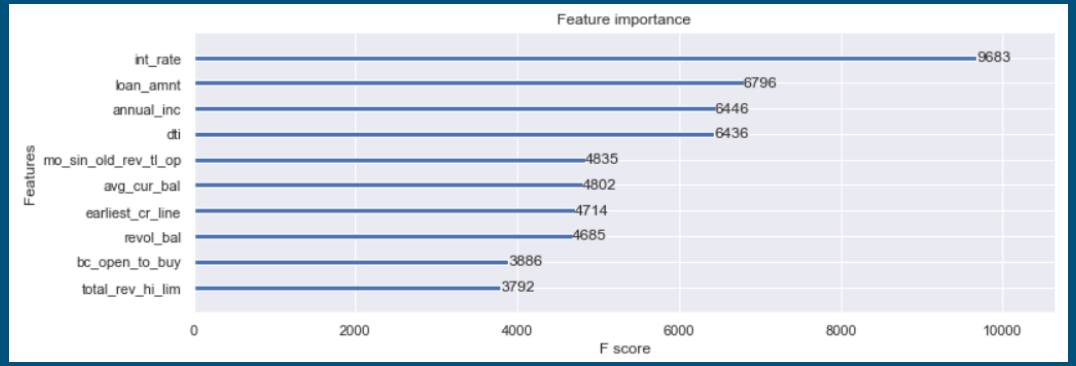


Model Cont.

There is a local maxima for percent interest returned based on different principals (by adjusting class weight). We chose this as our model. It returns 7.24% for a 1.4b investment.

	precision	recall	f1-score	support
0	0.93	0.37	0.53	269799
1	0.26	0.89	0.40	67226
accuracy			0.48	337025
macro avg	0.60	0.63	0.47	337025
weighted avg	0.80	0.48	0.51	337025

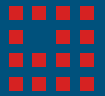




Average Return on Investment

	Annualized Returns
Risk Free Rate *	1.63%
Null Model	1.71%
Our Model	4.28%

*Average of 3yr and 5yr treasury yields.



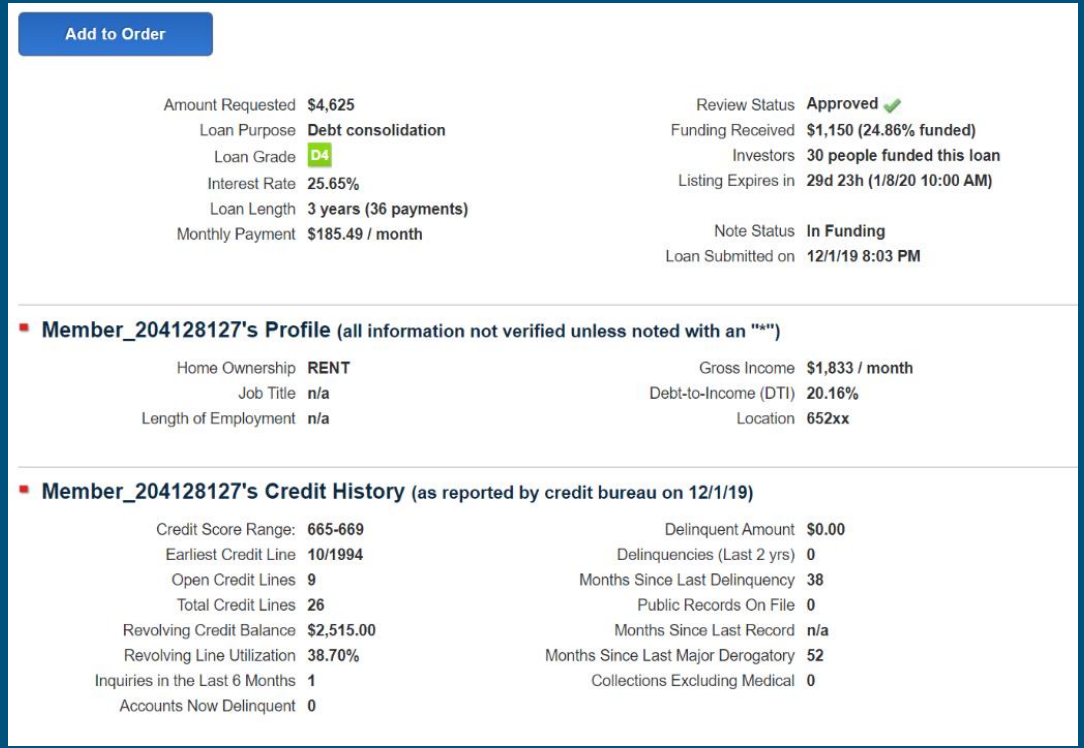
Appendix

Available Data on Prospective Loans

-Additional features available for download.

-Features coincide with dataset provided by class/kaggle. Class/kaggle dataset has more features.

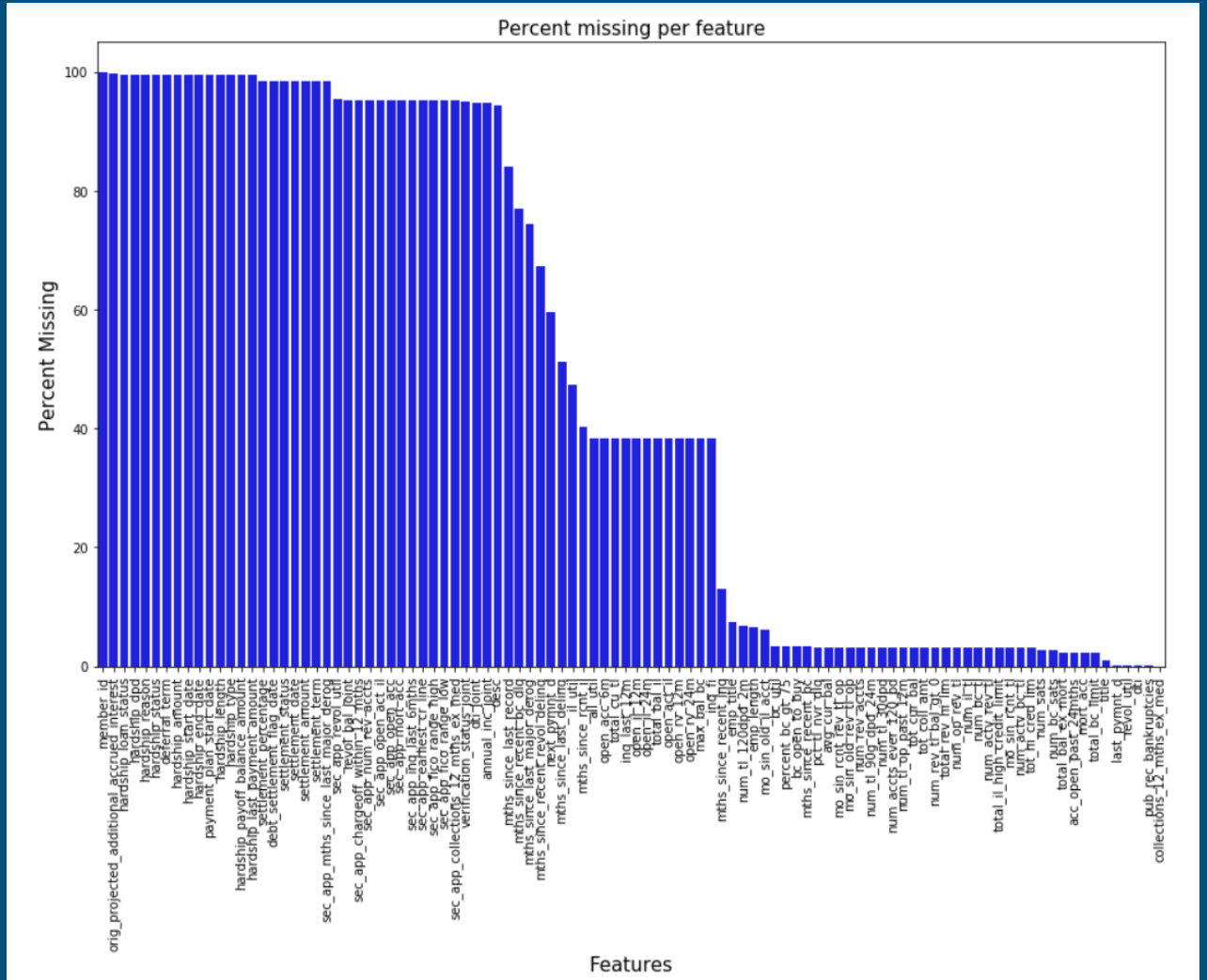
-Only features available for prospective loans were used in the modelling



Missing Data

Dropped columns over 35% missing.

Remaining features imputed using mode for object class, and median for the rest.



Scoring Appendix

		loan_amnt			total_pymnt			int_collected		
		sum	mean	count	sum	mean	count	sum	mean	count
prediction	loan_status_binary									
0	0	1.322666e+09	13233.28	99950	1.464818e+09	14655.51	99950	1.421520e+08	1422.23	99950
	1	9.233728e+07	13075.23	7062	5.259262e+07	7447.27	7062	-3.974465e+07	-5627.96	7062
1	0	2.488089e+09	14648.83	169849	2.972272e+09	17499.50	169849	4.841834e+08	2850.67	169849
	1	9.514011e+08	15813.46	60164	5.021446e+08	8346.26	60164	-4.492565e+08	-7467.20	60164

```
d2
```

	loan_amnt			total_pymnt			int_collected		
	sum	mean	count	sum	mean	count	sum	mean	count
prediction									
0	1.415003e+09	13222.85	107012	1.517411e+09	14179.82	107012	1.024073e+08	956.97	107012
1	3.439490e+09	14953.46	230013	3.474417e+09	15105.31	230013	3.492689e+07	151.85	230013

```
#principal needed (loan amount sum)
1.415003e+09
```

```
1415003000.0
```

```
#interest collected by model (int collected sum)
1.024073e+08
```

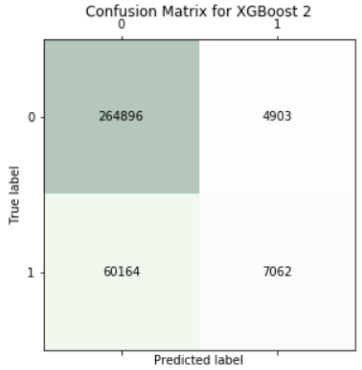
```
102407300.0
```

```
#return, % interest collected divided by principal needed
round((((102407300.0)/(1415003000.0))*100),2)
```

```
7.24
```

Other models: XGBoost no class weight

	precision	recall	f1-score	support
0	0.81	0.98	0.89	269799
1	0.59	0.11	0.18	67226
accuracy			0.81	337025
macro avg	0.70	0.54	0.53	337025
weighted avg	0.77	0.81	0.75	337025



		loan_amnt			total_pymnt			int_collected		
		sum	mean	count	sum	mean	count	sum	mean	count
prediction	loan_status_binary									
0	0	3.715976e+09	14028.06	264896	4.316225e+09	16294.03	264896	6.002481e+08	2265.98	264896
	1	9.068035e+08	15072.19	60164	4.877797e+08	8107.50	60164	-4.190238e+08	-6964.69	60164
1	0	9.477830e+07	19330.68	4903	1.208656e+08	24651.36	4903	2.608732e+07	5320.69	4903
	1	1.369349e+08	19390.39	7062	6.695750e+07	9481.38	7062	-6.997740e+07	-9909.01	7062

```
']).agg({'loan_amnt':['sum','mean','count'],'total_pymnt':['sum','mean','count'],'int_collected':['sum','mea
```

```
b2
```

	loan_amnt			total_pymnt			int_collected		
	sum	mean	count	sum	mean	count	sum	mean	count
prediction									
0	4.622780e+09	14221.31	325060	4.804004e+09	14778.82	325060	1.812243e+08	557.51	325060
1	2.317132e+08	19365.92	11965	1.878231e+08	15697.71	11965	-4.389008e+07	-3668.21	11965

```
#principal needed
4.622780e+09
```

```
4622780000.0
```

```
#interest collected by model
1.812243e+08
```

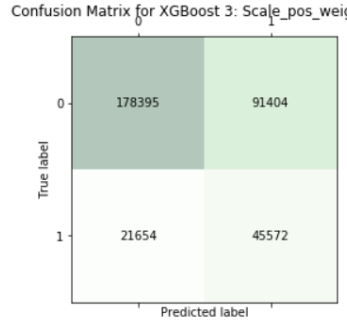
```
181224300.0
```

```
#return, % interest collected divided by principal needed
round((((181224300.0)/(4622780000.0))*100),2)
```

```
3.92
```

XGBoost 4 weight on positive

	precision	recall	f1-score	support
0	0.89	0.66	0.76	269799
1	0.33	0.68	0.45	67226
accuracy			0.66	337025
macro avg	0.61	0.67	0.60	337025
weighted avg	0.78	0.66	0.70	337025



		loan_amnt			total_pymnt			int_collected		
		sum	mean	count	sum	mean	count	sum	mean	count
prediction	loan_status_binary									
0	0	2.354749e+09	13199.64	178395	2.660355e+09	14912.72	178395	3.056055e+08	1713.08	178395
	1	2.835842e+08	13096.16	21654	1.597813e+08	7378.83	21654	-1.238029e+08	-5717.32	21654
1	0	1.456006e+09	15929.34	91404	1.776736e+09	19438.27	91404	3.207299e+08	3508.93	91404
	1	7.601542e+08	16680.29	45572	3.949559e+08	8666.64	45572	-3.651983e+08	-8013.66	45572

```
c2
```

	loan_amnt			total_pymnt			int_collected		
	sum	mean	count	sum	mean	count	sum	mean	count
prediction									
0	2.638333e+09	13188.44	200049	2.820136e+09	14097.23	200049	1.818026e+08	908.79	200049
1	2.216160e+09	16179.18	136976	2.171691e+09	15854.54	136976	-4.446842e+07	-324.64	136976

```
#principal needed (Loan amount sum)
2.638333e+09
```

```
2638333000.0
```

```
#interest collected by model (int collected sum)
1.818026e+08
```

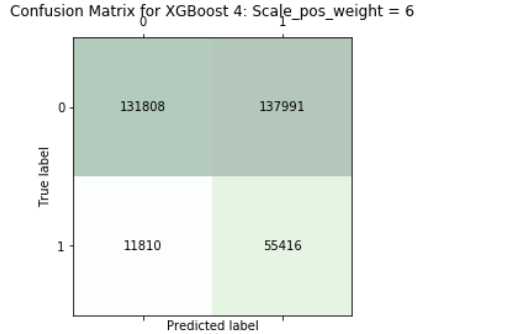
```
181802600.0
```

```
#return, % interest collected divided by principal needed
round((((181802600.0)/(2638333000.0))*100),2)
```

```
6.89
```

XGBoost 6 weight on positive

	precision	recall	f1-score	support
0	0.92	0.49	0.64	269799
1	0.29	0.82	0.43	67226
accuracy			0.56	337025
macro avg	0.60	0.66	0.53	337025
weighted avg	0.79	0.56	0.60	337025



		loan_amnt			total_pymnt			int_collected		
		sum	mean	count	sum	mean	count	sum	mean	count
prediction	loan_status_binary									
0	0	1.730008e+09	13125.21	131808	1.931584e+09	14654.52	131808	2.015755e+08	1529.31	131808
	1	1.526232e+08	12923.21	11810	8.657877e+07	7330.97	11810	-6.604438e+07	-5592.24	11810
1	0	2.080747e+09	15078.86	137991	2.505507e+09	18157.03	137991	4.247599e+08	3078.17	137991
	1	8.911152e+08	16080.47	55416	4.681584e+08	8448.07	55416	-4.229568e+08	-7632.40	55416

e2

	loan_amnt			total_pymnt			int_collected		
	sum	mean	count	sum	mean	count	sum	mean	count
prediction									
0	1.882631e+09	13108.60	143618	2.018162e+09	14052.29	143618	1.355311e+08	943.69	143618
1	2.971862e+09	15365.84	193407	2.973665e+09	15375.17	193407	1.803050e+06	9.32	193407

```
#principal needed (loan amount sum)
1.882631e+09
```

```
1882631000.0
```

```
#interest collected by model (int collected sum)
1.355311e+08
```

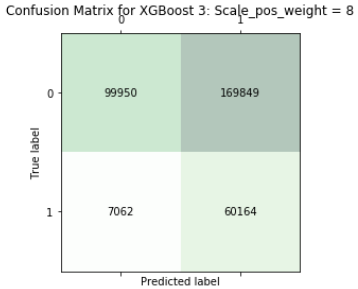
```
135531100.0
```

```
#return, % interest collected divided by principal needed
round((((135531100.0)/(1882631000.0))*100),2)
```

```
7.2
```

XGBoost 8 weight on positive (chosen model)

	precision	recall	f1-score	support
0	0.93	0.37	0.53	269799
1	0.26	0.89	0.40	67226
accuracy			0.48	337025
macro avg	0.60	0.63	0.47	337025
weighted avg	0.80	0.48	0.51	337025



		loan_amnt			total_pymnt			int_collected		
		sum	mean	count	sum	mean	count	sum	mean	count
prediction	loan_status_binary									
0	0	1.322666e+09	13233.28	99950	1.464818e+09	14655.51	99950	1.421520e+08	1422.23	99950
	1	9.233728e+07	13075.23	7062	5.259262e+07	7447.27	7062	-3.974465e+07	-5627.96	7062
1	0	2.488089e+09	14648.83	169849	2.972272e+09	17499.50	169849	4.841834e+08	2850.67	169849
	1	9.514011e+08	15813.46	60164	5.021446e+08	8346.26	60164	-4.492565e+08	-7467.20	60164

```
d2
```

	loan_amnt			total_pymnt			int_collected		
	sum	mean	count	sum	mean	count	sum	mean	count
prediction									
0	1.415003e+09	13222.85	107012	1.517411e+09	14179.82	107012	1.024073e+08	956.97	107012
1	3.439490e+09	14953.46	230013	3.474417e+09	15105.31	230013	3.492689e+07	151.85	230013

```
#principal needed (loan amount sum)
1.415003e+09
```

```
1415003000.0
```

```
#interest collected by model (int collected sum)
1.024073e+08
```

```
102407300.0
```

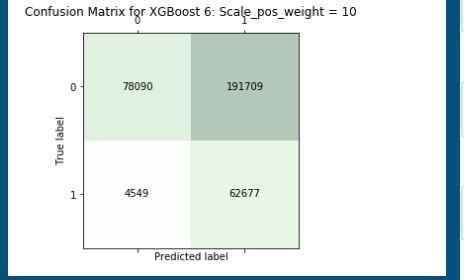
```
#return, % interest collected divided by principal needed
round((((102407300.0)/(1415003000.0))*100),2)
```

```
7.24
```

```
from xgboost import plot_importance
plot_importance(clf4, max_num_features=10) # top 10 most important features
plt.show()
```

XGBoost 10 weight on positive

	precision	recall	f1-score	support
0	0.94	0.29	0.44	269799
1	0.25	0.93	0.39	67226
accuracy			0.42	337025
macro avg	0.60	0.61	0.42	337025
weighted avg	0.81	0.42	0.43	337025



		loan_amnt			total_pymnt			int_collected		
		sum	mean	count	sum	mean	count	sum	mean	count
prediction	loan_status_binary									
0	0	1.046967e+09	13407.19	78090	1.152482e+09	14758.39	78090	1.055153e+08	1351.20	78090
	1	6.091552e+07	13390.97	4549	3.480815e+07	7651.82	4549	-2.610738e+07	-5739.15	4549
1	0	2.763788e+09	14416.58	191709	3.284608e+09	17133.30	191709	5.208201e+08	2716.72	191709
	1	9.828228e+08	15680.76	62677	5.199290e+08	8295.37	62677	-4.628938e+08	-7385.39	62677

f2

	loan_amnt			total_pymnt			int_collected		
	sum	mean	count	sum	mean	count	sum	mean	count
prediction									
0	1.107883e+09	13406.29	82639	1.187291e+09	14367.19	82639	79407887.28	960.90	82639
1	3.746610e+09	14728.05	254386	3.804537e+09	14955.76	254386	57926306.81	227.71	254386

```
#principal needed (loan amount sum)
1.107883e+09
```

```
1107883000.0
```

```
#interest collected by model (int collected sum)
79407887.28
```

```
79407887.28
```

```
#return, % interest collected divided by principal needed
round((((79407887.28)/(1107883000.0))*100),2)
```

```
7.17
```