ELO Merchant

Юрий Болконский

Elo Merchant Category Recommendation

Бизнес: Elo Brazil предоставляет клиентам карты банков, по которым можно покупать товары на выгодных условиях. Для продавцов это платформа для продажи своих товаров и услуг, а для клиентов - скидки/бонусы/специальные предложения

Основная задача соревнования:

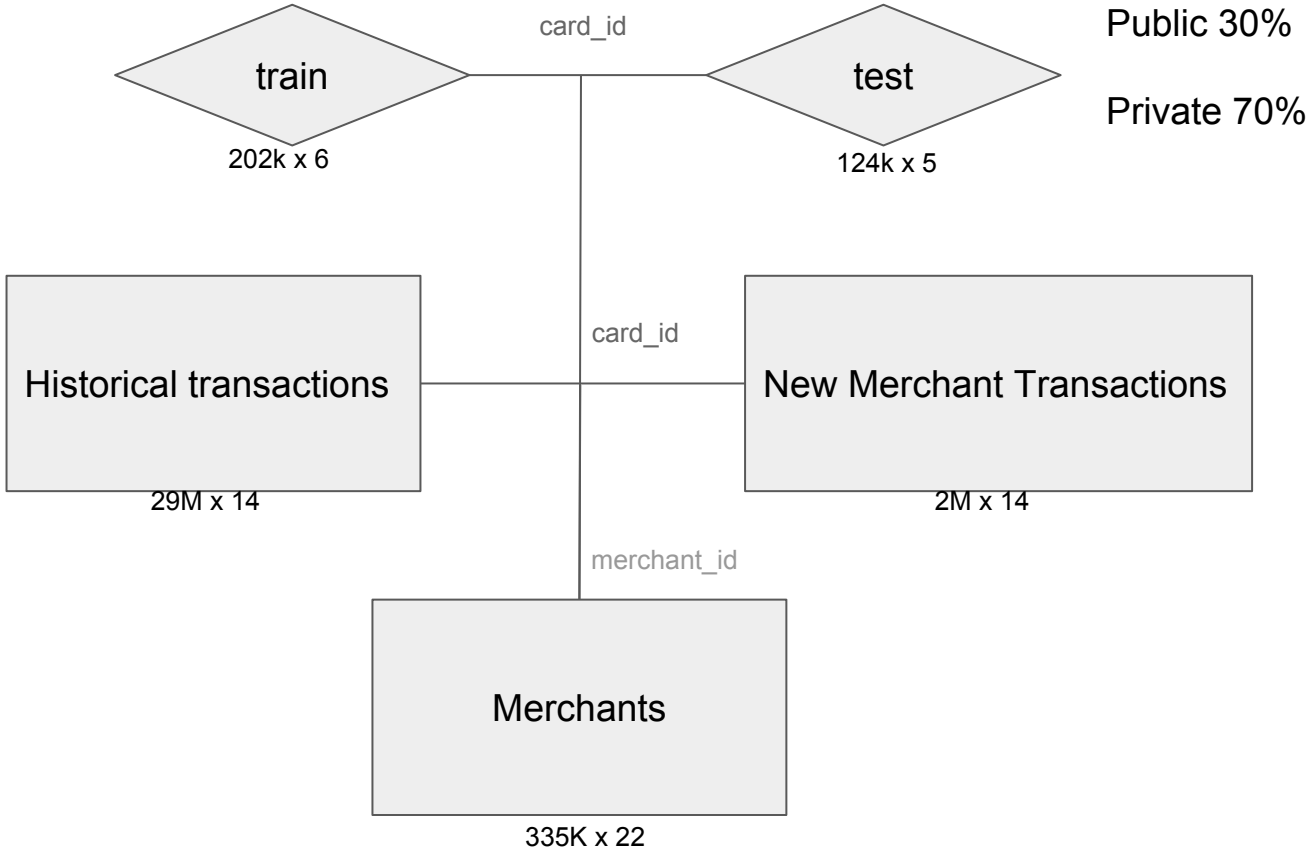
“In this competition, Kagglers will develop algorithms to identify and serve the most relevant opportunities to individuals, by uncovering signal in customer loyalty. Your input will improve customers’ lives and help Elo reduce unwanted campaigns, to create the right experience for customers”.

Необходимо создать алгоритм, который благодаря изучению лояльности пользователей сможет уменьшить количество нерелевантных акций, и как итог улучшить опыт пользователей.

Первое золото и Kaggle Master

#	△pub	Team Name	Kernel	Team Members	Score	Entries	Last
1	▲ 21	Look alive			3.57875	24	10d
2	▲ 14	陪elo一起过年			3.59019	275	10d
3	▲ 4	GideonTeo			3.59319	404	10d
4	▲ 8	TH & 袋鼠 & Q			3.59375	318	9d
5	▲ 76	[ods.ai] Evgeny Patekha			3.59422	46	9d
6	▲ 39	Lucky stars			3.59707	183	10d
7	▼ 4	You'll Never Overfitting Alone			3.59779	399	9d
8	▲ 205	Tom124			3.59791	38	10d
9	▲ 79	Karachun Michael			3.59801	127	10d
10	▲ 58	[ods.ai] YuryBolkonskiy			3.59904	111	9d
11	▲ 50	Stack It All			3.59940	386	9d
12	▼ 1	horizon			3.60001	414	10d
13	▼ 7	Loyalty overrated			3.60046	163	10d
14	▼ 5	Skynet			3.60050	386	10d
15	▲ 41	HRed			3.60062	225	10d

Данные


card_id
train
test
202k x 6
124k x 5
Public 30%
Private 70%
Historical transactions
card_id
New Merchant Transactions
29M x 14
2M x 14
merchant_id
Merchants
335K x 22

Задача

Бизнес: определить LTV Score для каждой карты

Математическая: минимизировать $\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$

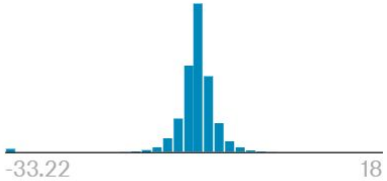
Данные

transactions

	authorized_flag	card_id	city_id	category_1	installments	category_3	merchant_category_id	merchant_id	month_lag	purchase_amount	purchase_date	category_2	state_id	subsector_id
	Y 100%	290001 unique values	-1 347	N 97% Y 3%	-1 999	A 47% B 43% Other (2) 10%	-1 891	[null] 1% M_ID_00a6ca8a8a 1% Other (226128) 97%	1 2	-0.75 263	1Mar17 1May18	1 5	-1 24	-1 41
1	Y	C_ID_415bb3a509	107	N	1	B	307	M_ID_b0c793002c	1	-0.55757375	2018-03-11 14:57:36	1.00000000	9	19
2	Y	C_ID_415bb3a509	140	N	1	B	307	M_ID_88920c89e8	1	-0.56957993	2018-03-19 18:53:37	1.00000000	9	19
3	Y	C_ID_415bb3a509	330	N	1	B	507	M_ID_ad5237ef6b	2	-0.55103721	2018-04-26 14:08:44	1.00000000	9	14
4	Y	C_ID_415bb3a509	-1	Y	1	B	661	M_ID_9e84cda3b1	1	-0.67192550	2018-03-07 09:43:21		-1	8
5	Y	C_ID_ef55cf8d4b	-1	Y	1	B	166	M_ID_3c86fa3831	1	-0.65990429	2018-03-22 21:07:53		-1	29
6	Y	C_ID_ef55cf8d4b	231	N	1	B	367	M_ID_8874615e00	2	-0.63300684	2018-04-02 12:53:28	1.00000000	9	16
7	Y	C_ID_ef55cf8d4b	69	N	1	B	333	M_ID_6d061b5ddc	1	5.26369692	2018-03-28 19:50:19	1.00000000	9	10
8	Y	C_ID_ef55cf8d4b	231	N	1	B	307	M_ID_df1e022f41	2	-0.55378707	2018-04-05 08:06:52	1.00000000	9	19
9	Y	C_ID_ef55cf8d4b	69	N	1	B	278	M_ID_d15eae0468	2	-0.59664268	2018-04-07 18:37:40	1.00000000	9	37
10	Y	C_ID_ef55cf8d4b	69	N	1	B	437	M_ID_5f9bffd028	1	-0.60719129	2018-03-17 18:10:41	1.00000000	9	15
11	Y	C_ID_ef55cf8d4b	69	N	-1		45	M_ID_3ffd43b4cd	1	4.45226529	2018-03-31 09:55:40	1.00000000	9	18
12	Y	C_ID_ef55cf8d4b	69	N	1	B	108	M_ID_e6f5213fbf	1	-0.60595911	2018-03-11 12:53:41	1.00000000	9	34
13	Y	C_ID_ef55cf8d4b	69	N	1	B	278	M_ID_aa97bc87f6	1	-0.63420896	2018-03-14 14:07:43	1.00000000	9	37
14	Y	C_ID_ef55cf8d4b	69	N	1	B	157	M_ID_28fbc8c74d	2	-0.49205816	2018-04-14 09:27:45	1.00000000	9	29
15	Y	C_ID_ef55cf8d4b	-1	Y	1	B	302	M_ID_b9f9332438	1	-0.66553923	2018-03-23 21:35:53		-1	22
16	Y	C_ID_ef55cf8d4b	231	N	1	B	80	M_ID_ad15049b64	2	-0.64624519	2018-04-18 12:41:48	1.00000000	9	37

train

	first_active_month	card_id	feature_1	feature_2	feature_3	target
	1Nov11 1Feb18	201917 unique values	1 5	1 3	0 1	-33.22 18
1	2017-06	C_ID_92a2005557	5	2	1	-0.82028260
2	2017-01	C_ID_3d0044924f	4	1	0	0.39291325
3	2016-08	C_ID_d639edf6cd	2	2	0	0.68805599
4	2017-09	C_ID_186d6a6901	4	3	0	0.14249520
5	2017-11	C_ID_cdbd2c0db2	1	3	0	-0.15974919
6	2016-09	C_ID_0894217f2f	4	2	0	0.87158529
7	2016-12	C_ID_7e63323c00	3	2	1	0.23012899



Valid	202k	100%
Mismatched	0	0%
Missing	0	0%
Mean	-0.39	
Std. Deviation	3.85	
Quantiles	-33.22	Min
	-0.88	25%
	-0.02	50%
	0.77	75%
	18	Max

Наличие outliers в target -33.22

month_lags с различной продолжительностью

	card_id	first_active_month	first_month_lag	available_lags_cnt
0	C_ID_92a2005557	2017-06	-8	10
3	C_ID_3d0044924f	2017-01	-12	14
25	C_ID_186d6a6901	2017-09	-5	7
36	C_ID_cdbd2c0db2	2017-11	-3	5
59	C_ID_dfa21fc124	2017-09	-3	5
64	C_ID_fe0fdac8ea	2017-08	-6	8

Отрицательный purchase_amount

Valid	1.96m
Mismatched	0
Missing	0
Mean	-0.55
Std. Deviation	0.69
Quantiles	-0.75
	263

Purchase Amount

```
data.purchase_amount.mean()
```

```
-0.0007032266623490891
```

```
s['delta'] = s.new_amount.diff(1)
s.head(10)
```

:

	new_amount	delta
0	0.000000	NaN
1	0.000015	0.000015
2	0.000030	0.000015
3	0.000045	0.000015
4	0.000060	0.000015
5	0.000075	0.000015
6	0.000090	0.000015
7	0.000105	0.000015
8	0.000120	0.000015
9	0.000135	0.000015

```
data['new_amount'] = data.new_amount / (100 * s.delta.mean())
```

Let's look at most frequent values.

```
data.new_amount.value_counts().head(10)
```

```
50.000000     735619
20.000004     640964
30.000003     547680
10.000005     444249
100.000001    418773
15.000001     379041
40.000002     271846
12.000004     233231
25.000000     232732
5.000003      208044
Name: new_amount, dtype: int64
```

CPMP: https://www.kaggle.com/cpmpml/raddar-magic-explained-a-bit/

Target

Организаторы прологарифмировали target по основанию 2

```
train['target_raw'] = 2**train['target']
```

```
'{:.10f}'.format(2**(-33.22))
```

```
'0.0000000001'
```

Возведем outlier 2 в степень -33.22 и получим *0.0000000001*

```
prices = [19.90, 22.90, 27.90, 29.90, 37.90]
sorted({ i/j for j in prices for i in prices})
```

```
[0.525065963060686,
 0.604221635883905,
 0.6655518394648829,
 0.7132616487455197,
 0.7361477572559366,
 0.765886287625418,
 0.7889182058047494,
 0.8207885304659498,
 0.868995633187773,
 0.9331103678929766,
 1.0,
 1.07168458781362,
 1.150753768844221,
 1.2183406113537119,
 1.2675585284280937,
 1.3056768558951966,
 1.3584229390681004,
 1.4020100502512562,
 1.5025125628140703,
 1.6550218340611353,
 1.9045226130653268]
```

Таким образом, target - отношение суммы покупок в прошлом к настоящему периоду

raddar: https://www.kaggle.com/raddar/target-true-meaning-revealed

Решение

	authorized_flag	card_id	city_id	category_1	installments	category_3	merchant_category_id	merchant_id	month_lag
0	Y	C_ID_4e6213e9bc	88	N	0	A	80	M_ID_e020e9b302	-8
1	Y	C_ID_4e6213e9bc	88	N	0	A	367	M_ID_86ec983688	-7
2	Y	C_ID_4e6213e9bc	88	N	0	A	80	M_ID_979ed661fc	-6
3	Y	C_ID_4e6213e9bc	88	N	0	A	560	M_ID_e6d5ae8ea6	-5
4	Y	C_ID_4e6213e9bc	88	N	0	A	80	M_ID_e020e9b302	-11

1. Объединение historical transactions и new merchant transaction с authorized_flag=1
2. Count Vectorizer merchant id, subsector id, merchant category id + PCA, SVD
3. Группировка month_lag data (auth month-13, auth month-12 and etc.) for count purchases, and sum adjusted purchase_amount, std, min, max

auth_month-13	auth_month-12	auth_month-11	auth_month-10	auth_month-9	auth_month-8	auth_month-7	auth_month-6	auth_month-5	auth_month-4	auth_month-3	auth_month-2	auth_month-1	auth_month0	auth_month+1	auth_month+2
0.00	0.00	0.00	0.00	0.00	897.85	132.0	493.10	51.66	415.55	208.92	345.6	80.00	586.59	0.00	308.5
741.77	712.48	1285.97	2266.72	906.86	0.00	0.0	0.00	0.00	0.00	0.00	0.0	0.00	147.00	228.50	288.9
0.00	0.00	0.00	0.00	0.00	0.00	0.0	149.76	0.00	349.50	1089.67	1529.5	0.00	759.80	1114.00	0.0
0.00	0.00	0.00	0.00	0.00	0.00	0.0	0.00	0.00	0.00	0.00	2569.1	652.43	590.99	830.74	317.5
8428.79	1951.13	1143.33	1331.00	1370.39	483.00	1240.9	4271.54	11560.78	10502.19	6300.00	16001.5	417.80	5328.27	10186.50	825.0

4. Отношение auth_month_purchase_amount-13,…auth_month_purchase amount+2 к предыдущему с периодом 2,4,6

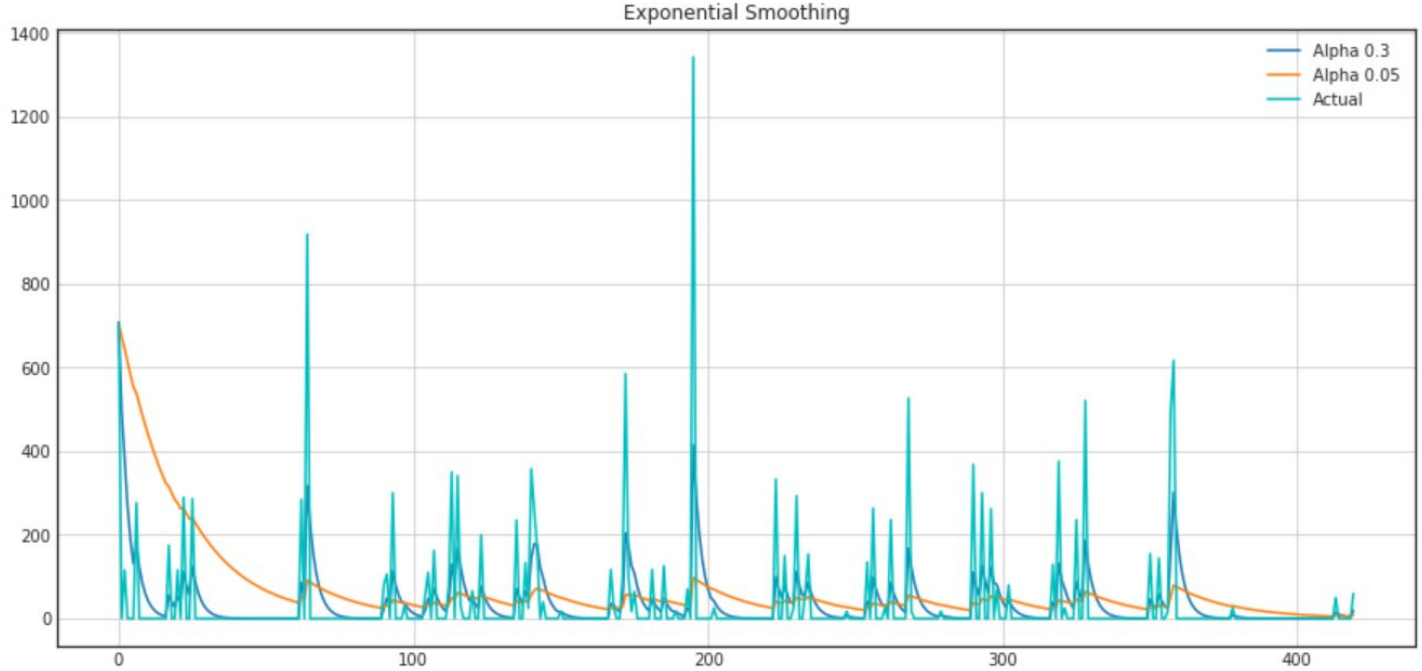
5. Предсказание month_lag+3 и +4 и подсчет diff, ratio, log

auth_month-13	auth_month-12	auth_month-11	auth_month-10	auth_month-9	auth_month-8	auth_month-7	auth_month-6	auth_month-5	auth_month-4	auth_month-3	auth_month-2	auth_month-1	auth_month0	auth_month+1	auth_month+2
0.00	0.00	0.00	0.00	0.00	897.85	132.0	493.10	51.66	415.55	208.92	345.6	80.00	586.59	0.00	308.5
741.77	712.48	1285.97	2266.72	906.86	0.00	0.0	0.00	0.00	0.00	0.00	0.0	0.00	147.00	228.50	288.9
0.00	0.00	0.00	0.00	0.00	0.00	0.0	149.76	0.00	349.50	1089.67	1529.5	0.00	759.80	1114.00	0.0
0.00	0.00	0.00	0.00	0.00	0.00	0.0	0.00	0.00	0.00	0.00	2569.1	652.43	590.99	830.74	317.5
8428.79	1951.13	1143.33	1331.00	1370.39	483.00	1240.9	4271.54	11560.78	10502.19	6300.00	16001.5	417.80	5328.27	10186.50	825.0

	auth_month+3	auth_month+4	ratio_auth_month_diff_+3_-13	diff_auth_month_diff_+3_-13	log_ratio_auth_month_diff_+3_-13	ratio_auth_month_diff_+3_-12	diff_auth_month_diff_+3_-12	log_ratio_auth_month_diff_+3_-12	ratio_auth_month_diff_+3_-11
0	644.695420	672.705541	6.446954e+12	644.695420	42.551755	6.446954e+12	644.695420	42.551755	6.446954e+12
1	458.389903	498.111446	4.583899e+12	458.389903	42.059712	8.316293e-02	-5053.560097	-3.587916	1.740327e-01
2	65.851688	82.370152	3.190180e-01	-140.568312	-1.648290	3.638014e-01	-115.158312	-1.458777	5.666124e-01
3	479.628357	554.447899	4.796284e+12	479.628357	42.125054	4.796284e+12	479.628357	42.125054	4.796284e+12
4	1595.252845	1708.145819	1.595253e+13	1595.252845	43.858850	1.595253e+13	1595.252845	43.858850	1.595253e+13

5 rows × 686 columns

6. Simple Rolling mean and SimpleExpSmoothing над auth_purchase_amount-13, …auth purchase amount+4 по month_lags



7. Подсчет avg, min, max временных промежутков между транзакциями

	card_id	tf_purchase_time_period_bins_between_0_and_0.2_sum	tf_purchase_time_period_bins_between_0_and_0.2_mean	tf_purchase_time_period_bins_between_0.2_and_0.5_sum	tf_purchase_time_period_bins_between_0.2_and_0.5_mean	tf_purchase_time_period_bins_between_0.5_and_1_sum
0	C_ID_00007093c1	5	0.043103	8	0.068966	7
1	C_ID_0001238066	8	0.054795	2	0.013699	10
2	C_ID_0001506ef0	6	0.093750	3	0.046875	2
3	C_ID_0001793786	12	0.054545	28	0.127273	21
4	C_ID_000183fdda	13	0.087838	11	0.074324	8

Также были подсчитаны периоды с первой по вторую транзакцию, со второй по третью, с первой по последнюю, с предпоследней по последнюю и т.д.

	card_id	tf_third_purchase_date	tf_all_elapsed_time	tf_hours_from_first_to_second_purchase	tf_hours_from_second_to_third_purchase	tf_hours_from_one_before_purchase_to_last	tf_hours_from_two_before_purchase_to_one_before_last
0	C_ID_00007093c1	2017-02-16 15:37:58	10058.387778	1.783889	47.836944	149.173333	845.977222
1	C_ID_0001238066	2017-10-08 16:19:47	5133.537778	25.135000	208.774167	42.641111	135.988056
2	C_ID_0001506ef0	2017-02-04 10:55:33	10360.974722	74.720556	423.938333	130.875556	657.800556
3	C_ID_0001793786	2017-02-01 17:53:10	8263.343056	3.510556	268.119722	0.169444	0.701111
4	C_ID_000183fdda	2017-09-08 20:46:36	5737.994722	123.177778	4.595556	97.226389	527.911111

Выбор фичей

После генерации дополнительных признаков их количество достигло 6500

1. Boruta (больше 8 часов ушло на выбор 500 лучших признаков) [1]
2. Выбор лучших признаков на основе feature_importance Catboost, LGBM, XGB
3. Удаление признаков на основе Adversarial validation

[1] https://www.jstatsoft.org/article/view/v036i11/v36i11.pdf

Обучение модели с outliers

1. Валидация на 5 Stratified Folds
2. Сохранение .pkl с OOF predictions

CatBoost

LightGBM

XGBoost

CV 3.645 std 0.012 LB 3.671

CV 3.641 std 0.011 LB 3.675

CV 3.649 std 0.013 LB 3.681

DeepFM[1] CV 3.656 std 0.016 LB 3.682

[1]https://arxiv.org/abs/1703.04247

Обучение модели без outliers

1. Валидация на 5 KFolds
2. Сохранение .pkl с OOF predictions

CatBoost

LightGBM

XGBoost

CV 1.547 std 0.011

CV 1.546 std 0.012

CV 1.551 std 0.013

DeepFM[1] CV 1.601 std 0.017

[1]https://arxiv.org/abs/1703.04247

Обучение модели классификатора Outliers

Модель классификации outliers получалась довольно плохой

```
[[165394  34316]
 [   377   1830]]
 Precision 0.05
 Recall 0.83
 F1 Score 0.1
Model: catb_all_adv_ks2_f100 CV AUC:  0.904151  +-  0.008
```

Лучшее чего удалось добиться LGBM ROC 0.907 F1 0.1025

Несколько дней до финала

1	Adventurous LB validation		3.61285	394	10d
2	[Aladdin Healthcare Tech]Sna...		3.61383	111	10d
3	You'll Never Overfitting Alone		3.63701	399	10d

63	lennart		3.66698	324	10d
64	l0glikelihood		3.66720	110	10d
65	失业青年		3.66736	43	17d
66	[Datawhale] Next in money		3.66741	280	10d
67	Onlise		3.66744	58	10d
68	[ods.ai] YuryBolkonskiy		3.66772	111	10d

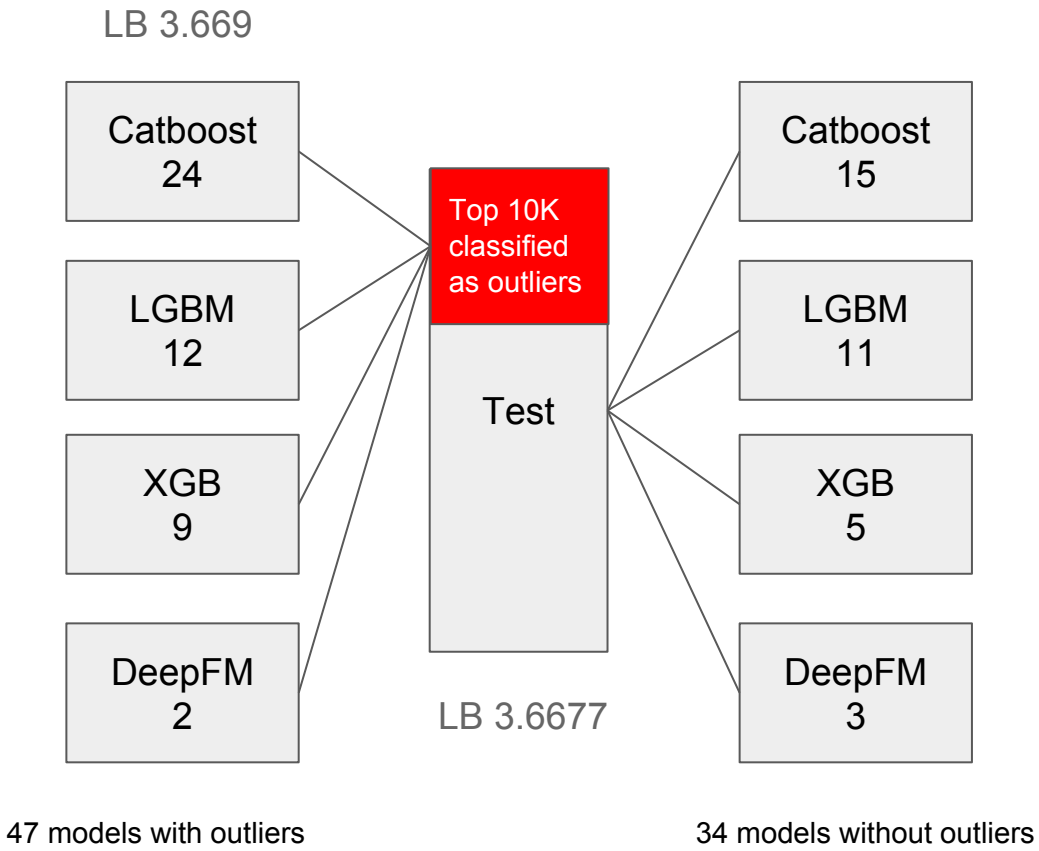
Your Best Entry ↑

Your submission scored 3.66772, which is an improvement of your previous score of 3.66803. Great job! Tweet this!

69	Going deeper		3.66777	211	10d
70	Cedric Damien		3.66779	14	10d
71	[ods.ai] Bugi_bugi		3.66780	79	11d
72	daredevil		3.66782	30	12d
73	Oleksii B.		3.66782	121	24d

Финальный Stacking

1. Stacking models with outliers CV 3.637 std 0.013 LB 3.669
2. Stacking models without outliers CV 1.544 std 0.011
3. Combining models, replacing 10000 values



Как на Private улетать вверх, а не вниз

Public

1	Adventurous LB validation	3.61285	394	10d
2	[Aladdin Healthcare Tech]Sna...	3.61383	111	10d
3	You'll Never Overfitting Alone	3.63701	399	10d

Private

1. Доверять собственной Кросс-валидации
2. Понимать, что Leaderboard по ходу соревнования определяет места на основании части тестового датасета. В этом соревновании лишь 30%.
3. Не уделять слишком много внимания подбору гиперпараметров для модели и других параметров (кол-во фолдов, random_seed и т.д.)

Финальный результат

#	△pub	Team Name	Kernel	Team Members	Score	Entries	Last
1	▲ 21	Look alive			3.57875	24	10d
2	▲ 14	陪elo一起过年			3.59019	275	10d
3	▲ 4	GideonTeo			3.59319	404	10d
4	▲ 8	TH & 袋鼠 & Q			3.59375	318	9d
5	▲ 76	[ods.ai] Evgeny Patekha			3.59422	46	9d
6	▲ 39	Lucky stars			3.59707	183	10d
7	▼ 4	You'll Never Overfitting Alone			3.59779	399	9d
8	▲ 205	Tom124			3.59791	38	10d
9	▲ 79	Karachun Michael			3.59801	127	10d
10	▲ 58	[ods.ai] YuryBolkonskiy			3.59904	111	9d
11	▲ 50	Stack It All			3.59940	386	9d
12	▼ 1	horizon			3.60001	414	10d
13	▼ 7	Loyalty overrated			3.60046	163	10d
14	▼ 5	Skynet			3.60050	386	10d
15	▲ 41	HRed			3.60062	225	10d

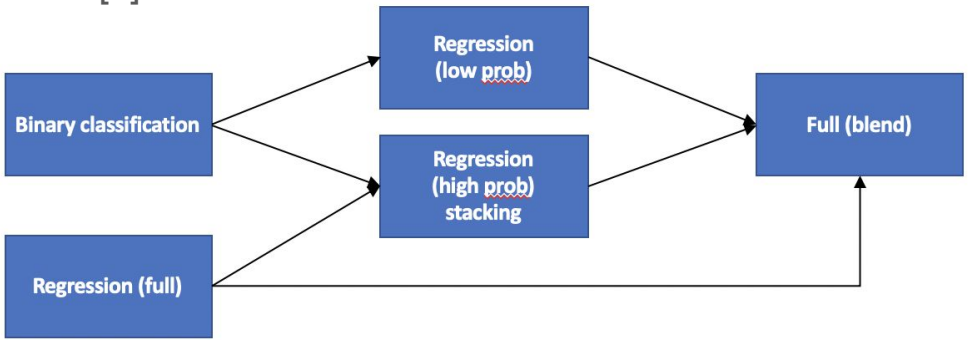
Что могло бы улучшить результат

1. Создать word embeddings на основе merchant_id's , merchantt_categoryt_id's , purchaset_date's … , потом сгруппировать по card_id's min/max/mean/std *senkin 7th place [1]*

2. Вставить train's target в каждую транзакцию, прогнать LGBM и извлечь предсказания, потом сгруппировать по card_id's min/sum, это позволит улучшить результат на 0.005 ~ 0.006 на CV и LB *senkin 7th place [1]*

3. Совершенно иная логика обучения моделей [2]

Evgeny Patekha 5th place



[1] https://www.kaggle.com/c/elo-merchant-category-recommendation/discussion/82055
[2] https://www.kaggle.com/c/elo-merchant-category-recommendation/discussion/82314