Описание проблемы

Оценка риска дефолта

Задача – оценка риска неуплаты клиента по кредиту (дефолт).

Дефолт — неуплата процентов по (обязательствам) кредиту или облигациям, непогашение займа в течение определённого времени. Обычно дефолт считают свершившимся, если клиент не совершил выплату по кредиту в течение 90 дней.

*Нужная **модель позволяет** банку или другой кредитной организации **оценить текущий риск** по любым выданным займам и кредитным продуктам и с большей долей вероятности **предотвратить неисполнение** кредитных **обязательств** клиентом.*

*Таким образом, банк **меньше рискует понести убытки**.*

Feature Preparation

Feature Engineering

OneHotEncoder (ohe) – инструмент для этапа Feature Engineering. Генерация новых характеристик с помощью ohe. Использовались все переменные, кроме ‘id’.

Data Engineering .sum(), .count()

Преобразование данных с помощью агрегирующих функций sum() и count().

[5]:

	rn_1	rn_2	rn_3	rn_4	rn_5	rn_6	rn_7	rn_8	rn_9	rn_10	...	pre_loans3060_6	pre_loans6090_0	pre_loans5_10	pre_loans530_5	pre_loans530_8	rn_56	rn_57	rn_5
0	1	1	1	1	1	1	1	1	1	1	...	0	0	0	0	0	0	0	
1	1	1	1	1	1	1	1	1	1	1	...	0	0	0	0	0	0	0	

2 rows × 479 columns

[17]:

	pre_since_opened_0	pre_since_opened_1	pre_since_opened_2	pre_since_opened_3	pre_since_opened_4	pre_since_opened_5	pre_since_opened_6	pre_since_opened_7	p
id									
0	10	10	10	10	10	10	10	10	
1	14	14	14	14	14	14	14	14	

2 rows × 419 columns

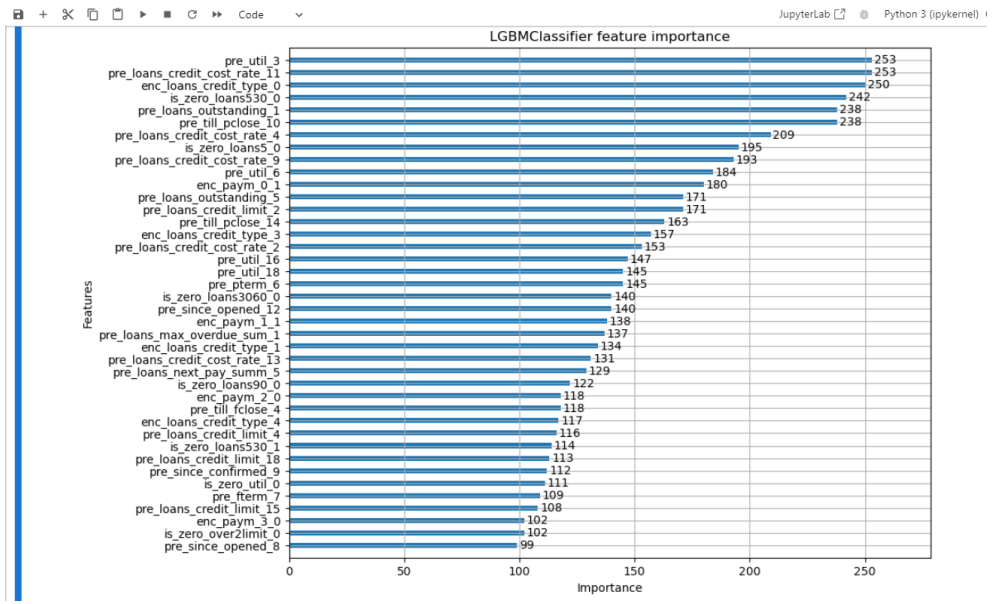
Feature Preparation

Поиск лучших характеристик

model.feature_importances_, cv = KFold(n_splits=5)
– инструменты для поиска лучших фич в финальной модели.

Результаты эксперимента:

len_features=238 результаты: train_roc_auc_score=0.77343, CV_roc_auc_score=0.73257
len_features=248 результаты: train_roc_auc_score=0.78253, CV_roc_auc_score=0.74741 …
len_features=435 результаты: train_roc_auc_score=0.79577, CV_roc_auc_score=0.76173 …
len_features=448 результаты: train_roc_auc_score=0.79504, CV_roc_auc_score=0.76212 …
len_features=468 результаты: train_roc_auc_score=0.79498, CV_roc_auc_score=0.76204
len_features=477 результаты: train_roc_auc_score=0.79498, CV_roc_auc_score=0.76204



Поиск лучших характеристик

Не дало прироста качества модели:

- добавление к фичам (ohe+.sum()) фичи от (ohe+.count()), несмотря на явную потерю информации при преобразовании данных в схеме Data -> ohe Feature Engineering -> data.groupby(by=['id']).sum(),
- использование части наиболее весомых фич (контрольная проверка на 300 и 440 шт.),

В модели можно было использовать меньшее количество сгенерированных фич для снижения вычислительных ресурсов, при сохранении обозначенного порогового качества модели.

Итоговое решение использовать все сгенерированные фичи.

Feature Preparation

StandardScaler()

StandardScaler() – инструмент для стандартизации данных.
С использованием инструмента качество модели незначительно улучшается.

Финальный датасет

[7]:

	rn_1	rn_2	rn_3	rn_4	...	pre_loans3060_6	pre_loans6090_0	pre_loans5_10	pre_loans530_5	pre_loans530_8	rn_56	rn_57	rn_58	id	flag
0	0.0	0.281688	0.419639	0.541014	...	-0.001414	-0.000577	-0.000577	-0.001291	-0.001	-0.000577	-0.000577	-0.000577	0	0
1	0.0	0.281688	0.419639	0.541014	...	-0.001414	-0.000577	-0.000577	-0.001291	-0.001	-0.000577	-0.000577	-0.000577	1	0
2	0.0	0.281688	0.419639	-1.848381	...	-0.001414	-0.000577	-0.000577	-0.001291	-0.001	-0.000577	-0.000577	-0.000577	2	0

3 rows × 479 columns

В результате ohe преобразования для дальнейшего моделирования используются все полученные 477 фич, т.о. все значения категориальных переменных становятся новыми фичами.

Результаты моделирования

Поиск лучшей модели и гиперпараметров

Разделение данных на тренировочную, валидационную и тестовую части.

[19]:
```
df_train, df_test = train_test_split(df, train_size=0.8, random_state=842)
print('Размерность файла df_train', df_train.shape)
print('Размерность файла df_test', df_test.shape)
```

```
Размерность файла df_train (2400000, 479)
Размерность файла df_test (600000, 479)
```

[25]:
```
df_train_cut, df_train_cv = train_test_split(df_train, train_size=0.9, random_state=842)
df_train_cv[:2]
```

[21]:
```
df_train_cut.shape[0]+df_train_cv.shape[0]+df_test.shape[0]
```

[21]:
```
3000000
```

GridSearchCV, (train:0,9, val:0.1)=0.8, test=0.2			
Модель	Диапазоп исследуемых параметров	лучшие параметры	Test ROC-AUC
LogisticRegression	{class_weight='balanced', 'solver': ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'], 'C':np.arange(5.0, 15, 1)}	C=4.1, class_weight='balanced', solver='liblinear'	0.734559
LGBMClassifier	{class_weight='balanced', early_stopping_rounds=250, verbose=-1, n_jobs=8, objective='binary', metric='auc', 'learning_rate': [0.05], 'max_depth': np.arange(5,31,5), 'num_leaves':np.arange(50,150,10), 'n_estimators': np.arange(800,3001,100)}	class_weight='balanced', early_stopping_rounds=500, verbose=0, n_jobs=8, learning_rate=0.05, max_depth=20, metric='auc', n_estimators=500, num_leaves=100, objective='binary'	0.763045
RandomForestClassifier	{class_weight='balanced', 'n_estimators' : np.arange(70,91,10), 'max_depth' : np.arange(3,7,1), 'min_samples_leaf' : np.arange(6,9,1), 'min_samples_split' : np.arange(6,9,1)}	class_weight='balanced', max_depth=15, min_samples_leaf=8, min_samples_split=8, n_estimators=90	0.74078

Результаты моделирования

Результаты тестов KFold

Best parameters, best model, KFold(n_splits=5)	
train_roc_auc_score	CV_roc_auc_score
0.8	0.76299

KFold(n_splits=5, random_state=842, shuffle=True)		
№ fold, mean, std	train_roc_auc_score	CV_roc_auc_score
fold 1	0.80804	0.76321
fold 2	0.80497	0.76308
fold 3	0.80966	0.76365
fold4	0.80948	0.76052
fold 5	0.80334	0.76452
mean	0.8071	0.763
std	0.00282	0.0015

Контрольная проверка для фич

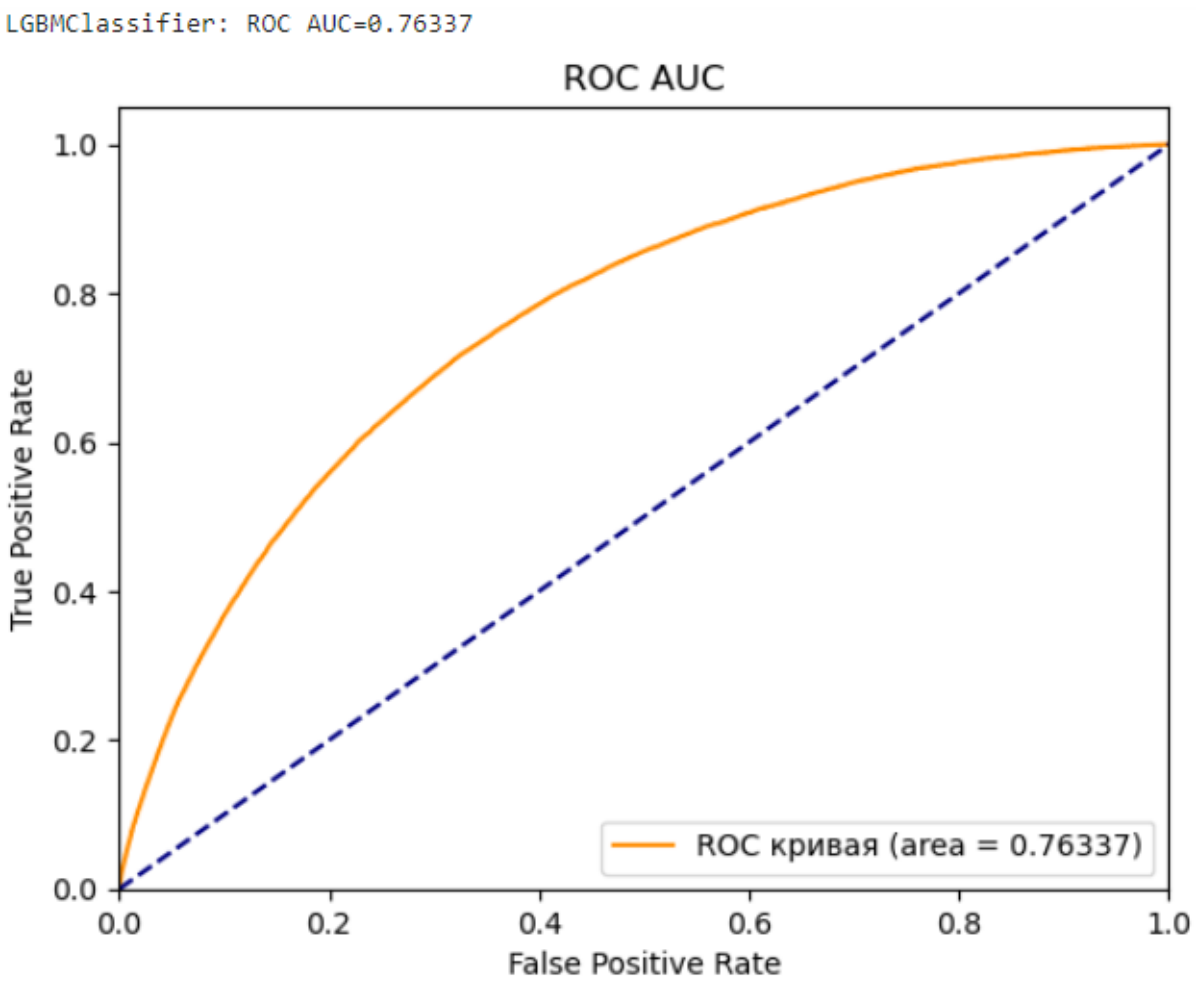
best model, all/only 440 features		
val_size	all_features, roc_auc_score	best_440_columns, roc_auc_score
0.1	0.76264	0.76227
0.05	0.76318	0.763
0.01	0.76337	0.763

Результаты моделирования

График ROC-AUC

LGBMClassifier
(class_weight='balanced’,
early_stopping_rounds=500, verbose=-1,
n_jobs=16,
learning_rate=0.05, max_depth=20,
metric='auc', n_estimators=500,
num_leaves=100, objective='binary’)

roc_auc_score=0.76337



Итоговый результат

Jupyter Notebook

подготовлен блокнот со всеми этапами решения задачи

Решение

1.Business Understanding

+ 2 cells hidden

2. Data Understanding

+ 2 cells hidden

3. Data Preparation ¶

+ 75 cells hidden

4. Modeling

+ 24 cells hidden

5. Evaluation

+ 12 cells hidden

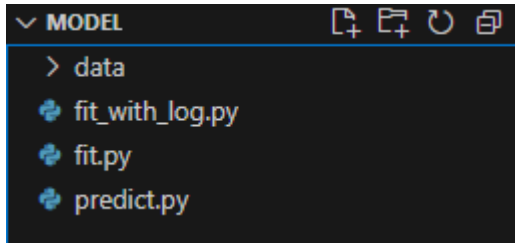
6. Deployment

+ 15 cells hidden

fit.py

с помощью sklearn.pipeline подготовлен автоматизированный пайплан, который по вызову fit готовит данные и обучает модель на наборе данных

Файловая структура



predict.py

по вызову predict обученная модель осуществляет предсказание, на всех данных ROC-AUC=0.819

```
data_all[(data_all.id == 0)]
```

	id	rn	pre_since_opened	pre_since_confirmed	pre_pterm	pre_fterm	pre_till_pclose	pre_till_fclose	pre_loans_credit_limit	pre_loans_next_pay_summ	...	enc_paym_22
0	0	1	18	9	2	3	16	10	11	3	...	3
1	0	2	18	9	14	14	12	12	0	3	...	0
2	0	3	18	9	4	8	1	11	11	0	...	0
3	0	4	4	1	9	12	16	7	12	2	...	3
4	0	5	5	12	15	2	11	12	10	2	...	3
5	0	6	5	0	11	8	12	11	4	2	...	3
6	0	7	3	9	1	2	12	14	15	5	...	3
7	0	8	2	9	2	3	12	14	15	5	...	3
8	0	9	1	9	11	13	14	8	2	5	...	3
9	0	10	7	9	2	10	8	8	16	4	...	3

10 rows × 62 columns

```
data_all[(data_all.id == 1)]
```

```
PS                    \Desktop\Git\ml-junior\final_project\model>
df_test.shape=(10, 61)
предсказание для df_test:[0]
```