GeekBrains

Курсовой проект на факультете AI GB от Megafon

Подтема

Предсказание вероятности подключения услуг абонентами

Дмитрий Яковлев

Закончил факультет искусственного интеллекта GeekBrains..
Закончил курсы повышения квалификации в МГТУ им. Баумана.

Немного о себе.

- Проживаю в г. Новосибирск, ищу работу в г. Москва.
- Инженер по работе с клиентами и проектировщиками на предприятии, производящем высоковольтное оборудование

План проекта

1. Пройденный этап: DataPreprocessing
2. Пройденный этап: EDA
3. Пройденный этап: FeatureGeneration
4. Пройденный этап: настройка классификаторов
5. Пройденный этап: создание общего пайплайна
6. Пройденный этап: создание product ready скриптов
7. Будущий этап: доработка проекта
8. Будущий этап: создание рекомендательной системы
9. Будущий этап
10. Будущий этап
11. Будущий этап
12. Будущий этап
13. Будущий этап
14. Будущий этап
15. Будущий этап
16. Будущий этап

Решение задачи / План работы

- Разработка метода генерации рабочего датасета из исходных features.csv и data..train.csv (ноутбук
- Исследование рабочего датасета: поиск аномалий, анализ распределений, корреляций, исследование методов работы с количественными и категориальными признаками, исследование выбросов, поиск оптимального алгоритма построения классификатора, определение полезных признаков,. (ноутбуки EDA_before_merge, EDA_after_merge)
- Генерация новых признаков, исследование их корреляции, попытка нахождения полезных кластеров (ноутбук Featureengeneering)
- Разработка отдельнънго классификатора для самой проблемной из предлагаемых услуг (ноутбук vas_id_6)
- Разработка классификатора для всего датасета (ноутбук vas_id_all)
- Разработка product ready скрипта на основе фреймворка Luigi (скрипт Luigi _pipeline)
- Рекомендации по разработке принципов составления индивидуальных предложений для выбранных абонентов.

Трудности

Очень трудно на этом датасете поднять целевую метрику. А на услуге № 6 эта метрика близка к 0,57. И ничем ее поднять не получается. Предлагаю для этой услуги использовать UpLift-метод.

Очень большой дисбаланс классов почти на всех предлагаемых услугах. И даже выровнять его для большинства услуг навряд ли получится.

Выброс на временном ряду с 317 по 350 день. Из-за дисбаланса классов правильно обработать его не возможно.

Исходные признаки для клиентов имеют малую значимость для классификатора и они плывут при разбиении на трейн и тест.

Кластеризация данных не помогает улучшить предсказание.

Нет возможности разбиения на 3 выборки: тренировочную, валидационную и тестовую.

Нет возможности разбиения датасета на трейн и тест по времени.

Baseline

[18]:
```
df.tail()
```

Out[...

	Unnamed: 0	id	vas_id	buy_time_train	target	buy_time	0
408719	831648	555080	5.0	1546203600	0.0	1.536527e+09	-96.799971
408720	831649	1729471	5.0	1546203600	0.0	1.545599e+09	-86.209971
408721	831650	3676177	2.0	1546203600	0.0	1.535922e+09	-96.799971
408722	831651	2255038	2.0	1546203600	0.0	1.539551e+09	-96.799971
408723	831652	3022610	2.0	1546203600	0.0	1.540760e+09	-49.339971

5 rows × 255 columns

[16]:
```
y_pred = np.where(df['vas_id'].isin([4.0, 6.0]), 1, 0)
```

[17]:
```
f1_score(df['target'].astype(int), y_pred, average='macro')
```

Out[17]:
```
0.6772798625826826
```

Создал наивный baseline. Если vas_id равен 4 или 6 – предсказываю единицу, иначе ноль.

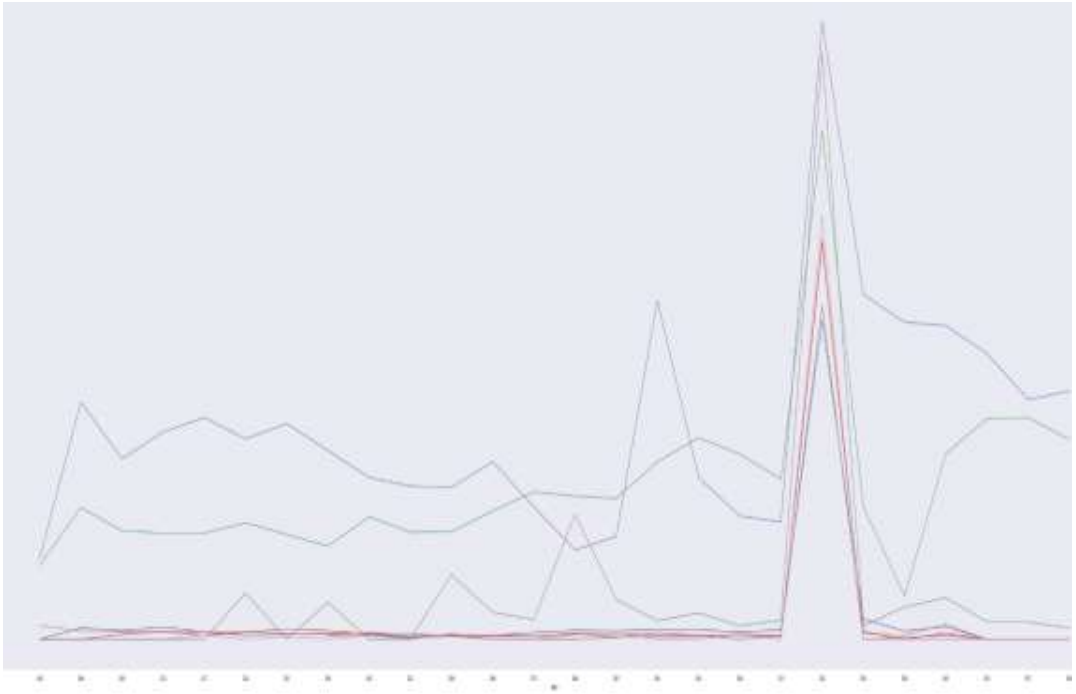
Задача – создать модель лучше чем baseline.

Создание рабочего датасета

```
if 0:
    features_train = features_train.sort_values(by='buy_time')
    data_train = data_train.rename(columns={'buy_time': 'buy_time_train'})
    data_train = data_train.sort_values(by='buy_time_train')
    data_train.shape

if 0:
    data_merged_train = pd.merge_asof(data_train, features_train, by='id',\
            left_on='buy_time_train', right_on='buy_time', direction='backward')
    data_merged_train.head()
```

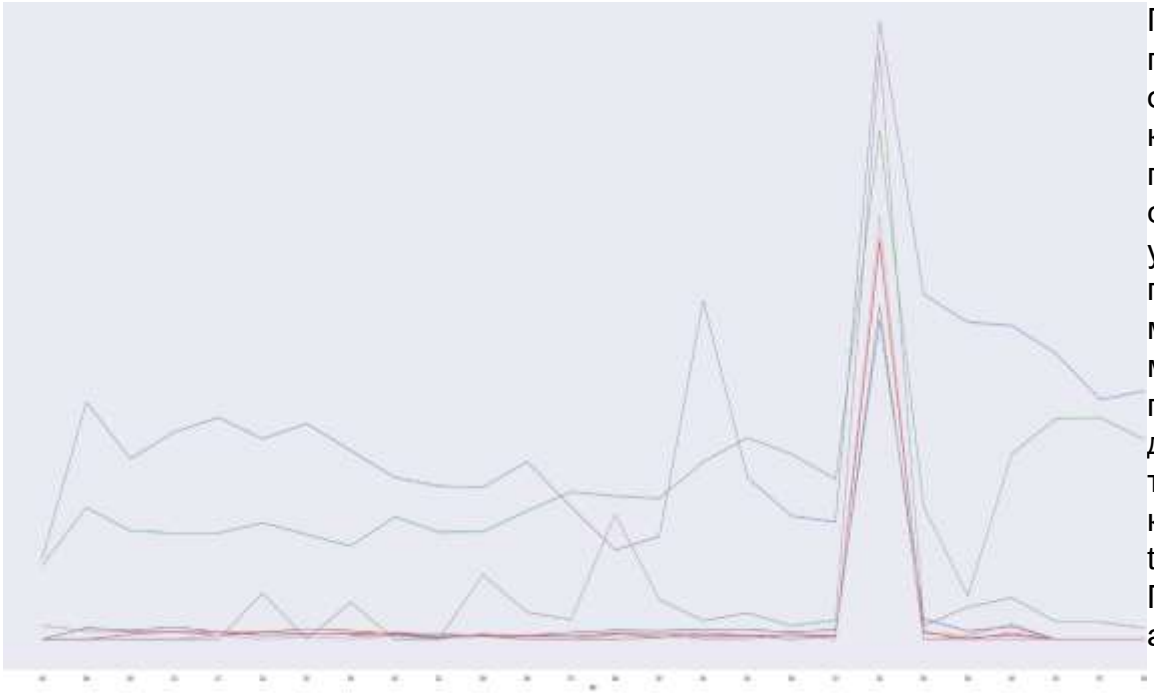
На первом шаге проекта предлагалось собрать общий датасет из двух файлов: features.csv содержащим порядка 300 признаков по каждому клиенту и data_train.csv, содержащим дату, id абонента, номер услуги и результат. Из-за того, что эти файлы не вмещаются в оперативную память, я использовал библиотеку dask. При формировании общего датасета использовал метод ‘backward’, чтобы дата пердложения была равна или позже даты регистрации клиента. Сохранил в полученном датасете обе даты, чтобы сформировать признаки: time_delta, how_old, novelty. Признаки оказались полезными.

Выбросы на временном ряде



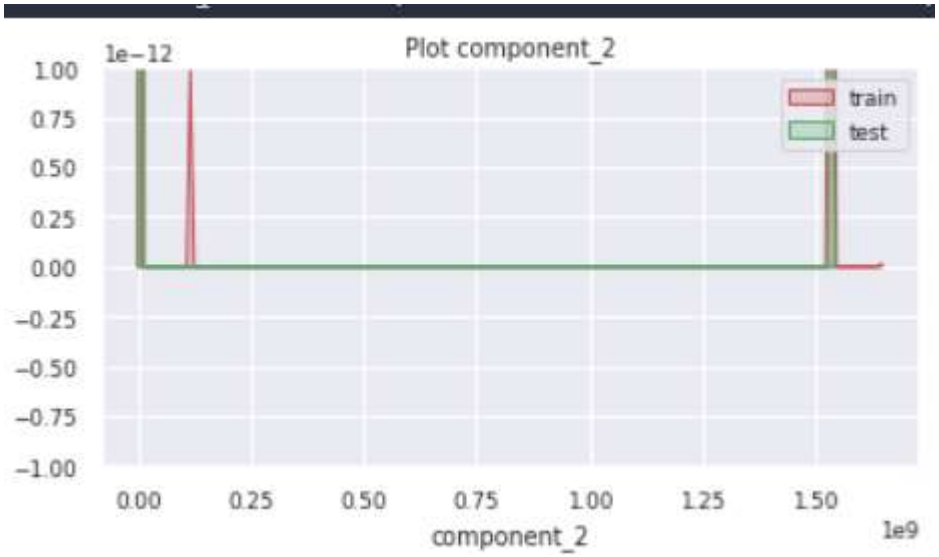
Основной проблемой датасета считаю выброс (распродажу) продаж с 315 по 350 день и близкие к нулю продажи по другим дням у 5 из 8 услуг. Причем при всех методах балансировки таргета f1_macro ухудшалось. Поскольку целевая метрика f1_macro, нам важно предсказать все продажи. При таком раскладе статистический анализ значимости клиентов на распродаже считаю бесполезным. Поэтому сгенерировал модель выявления таких клиентов (признак is_action), f1_score по единицам которой показала всего 0,24. Значит для классификатора эти категории клиентов будут мало различимы, а распродажи могут быть и в предсказываемом периоде (хотя для самой продаваемой услуги №6 после исключения выбросов классификатор избавился от переобучения, но оно было всего 2%). Поэтому принял решение оставить выбросы в датасете.

Разбиение на train и test



Поскольку сгенерированные временные признаки новизны предложения оказались значимыми для классификатора, логично было бы попробовать разбиение по времени, но на самых крупных выборках с номерами услуг 1 и 2 после 350 дня продажи практически отсутствуют, а целевая метрика у нас f1_macro. Поэтому такой метод разбиения не подходит. По той же причине невозможно создать дополнительную валидационную выборку, т.к. после 350 дня для обучения важна каждая продажа. Поэтому разбивал на train и test со стратификацией по таргету. По номеру услуги стратифицировалась автоматически.

EDA



Анализ достаточно обширный чтобы его весь описать. Пытался провести кодировку категориальных признаков различными методами, но почти все они приводили к одинаковой метрике, более низкой, чем без кодировки. Из-за огромного количества признаков и сжатого времени разработки провести статистический анализ признаков не удалось. Это и не потребовалось, т.к. при сжатии методом PCA всех нумерованных признаков целевая метрика классификатора не ухудшилась, поэтому решил их сжать.. Анализ сжатых признаков показал, что распределение второй компоненты на трейне и тесте различны, пришлось ее удалить. Вообще полученные компоненты для классификатора оказались мало значимыми и любые их преобразования на метрики никак не влияли.

Выбросы

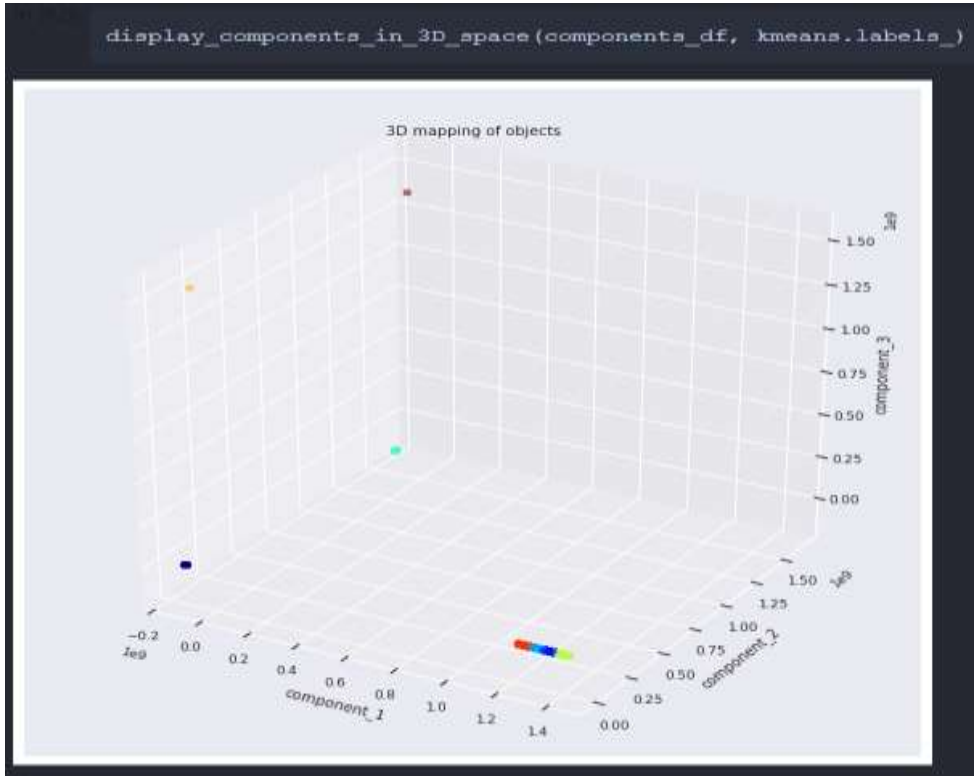
```
isf = IsolationForest(n_estimators=20, random_state=42, contamination=0.02)
preds = isf.fit_predict(train.drop(columns=['target', 'id', 'date']))

train["iso_forest_outliers"] = preds
#train["iso_forest_scores"] = isf.decision_function(train.drop(columns=['target', 'id', 'date', "iso_forest_outliers"]))

print(train["iso_forest_outliers"].value_counts())
train_outliers = train.loc[train['iso_forest_outliers']== -1]
train_out = train.loc[train['iso_forest_outliers']== 1].drop(columns=['iso_forest_outliers', "iso_forest_outliers"])
train_out.head()
```

```
X does not have valid feature names, but IsolationForest was fitted with feature names
 1    280393
-1      5713
Name: iso_forest_outliers, dtype: int64
```

Помимо выбросов во временном ряде, есть выбросы (примерно 2%) по строкам датасета.

Метрики с удаленными выбросами не изменилась. Метрики вообще почти ни на что не реагируют хоть в лучшую хоть в худшую сторону. Поэтому оставил выбросы без изменений.

Кластеризация



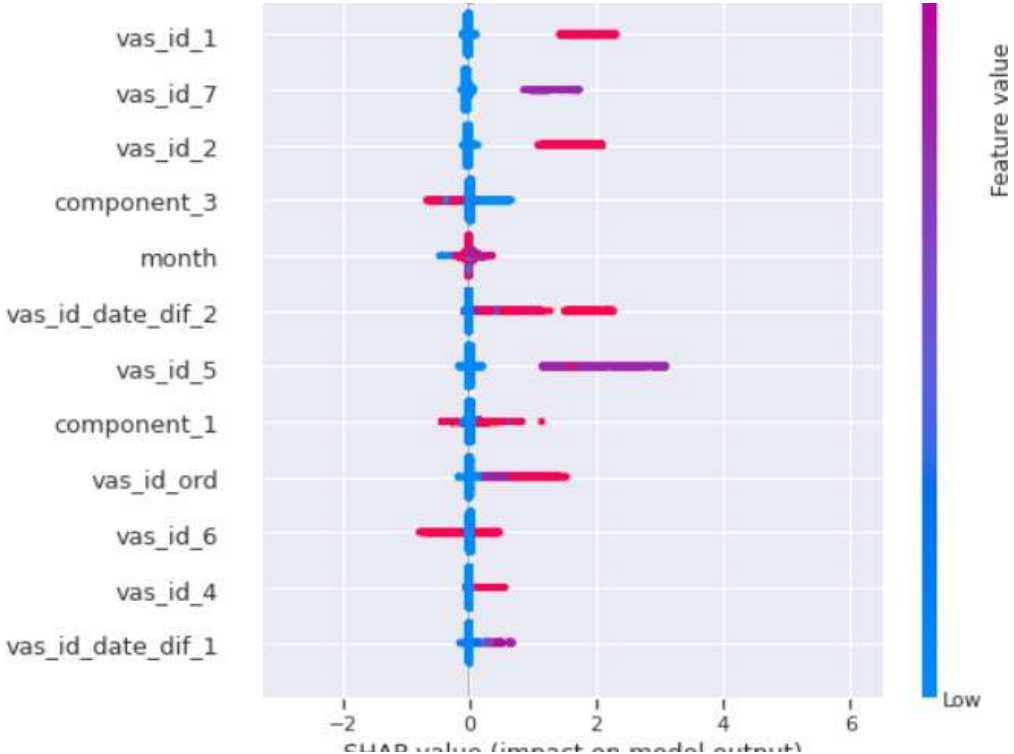
С помощью обычного kmeans удалось кластеризовать скомпонованные признаки по клиентам на 9 кластеров. Но ни как признак, не разбиение датасета на 9 шт. увеличению метрики не помогло. Странный датасет.

FeatureGeneration

```
def buy_time_change(X):
    """Функция преобразует Unix time в другие форматы и ищет новизну предложения"""
    from datetime import datetime, date, time, timedelta
    import holidays
    from pandas.tseries.holiday import USFederalHolidayCalendar as calendar


    X['date'] = list(map(datetime.fromtimestamp,X['buy_time_train']))
    X['month'] = X['date'].apply(lambda x: x.timetuple()[1])
    X['day'] = X['date'].apply(lambda x: x.timetuple()[7])
    # неделя года
    X['weekofyear'] = X['buy_time_train'].apply(lambda x: pd.to_datetime(date.fromtimestamp(x)).weekofyear)
    X['time_max'] = X.buy_time_train.max()
        # Новизна предложения
    X['novelty'] = X['time_max'] - X['buy_time_train']
    # эти признаки оказались бесполезными
    #df_all['weekday'] = df_all['date'].apply(lambda x: x.timetuple()[6])
    # data['year'] = data['date'].map(lambda x: x.year)
    #df_all['hour'] = df_all['date'].map(lambda x: x.hour)
    #df_all['is_holiday'] = df_all['date'].map(lambda x: x in holidays.RU())
    return X
```

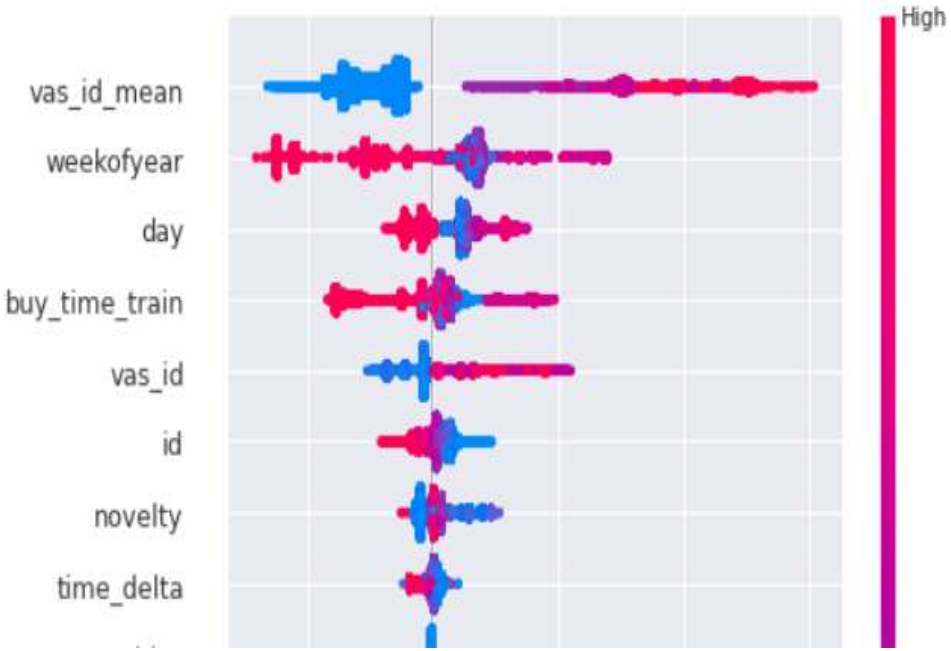
На первом этапе сгенерировал временные фичи из признаков buy_time_train и buy_time. Не все из них оказались полезными. Праздничные и предпраздничные дни на продажи вообще никак не влияют. Отчеты по продажам сгенерированы в один и тот же день недели в одно и то же время.

FeatureGeneration



По мнению SHAP хорошей разделяющей способностью для классификатора являются фичи логирования продаж по каждому клиенту. Но логирование по каждому клиенту привело к резкой потере целевой метрики, поэтому логирование провел по клиентам с несколькими предложениями. Логика такая: чем больше было клиенту предложений услуги и чем больше прошло времени, тем меньше вероятность что он ее подключит. Но ввиду очень низкого в датасете количества повторных предложений, эти признаки не столь значимы и они не сильно увеличили метрику.. Тем не менее когда накопится по ним достаточно статистики, они непременно «выстрелят». Для поддержки этих признаков оставил в датасете id клиентов. И хотя через id происходит определенный слив информации, информацию от этого слива когда будет много предложений возьмут на себя эти признаки.

FeatureGeneration



Модель на столько не чувствительна к новым признакам, что даже таргет-энкодинг особо не помог. Только таргет-энкодинг по vas_id немного увеличил метрику. Логарифм от этого признака показал еще большую разделяющую способность, но к сожалению я не сохранил его диаграмму.

Остальные методы генерации признаков не помогли.

Услуга № 6

```
0.0    0.562549
1.0    0.437451
Name: target, dtype: float64
0.0    0.569698
1.0    0.430302
Name: target, dtype: float64
TRAIN

              precision    recall  f1-score   support

         0.0       0.63      0.69      0.66     16571
         1.0       0.55      0.49      0.52     12886

    accuracy                           0.60     29457
   macro avg       0.59      0.59      0.59     29457
weighted avg       0.60      0.60      0.60     29457

TEST

              precision    recall  f1-score   support

         0.0       0.63      0.69      0.66      7242
         1.0       0.53      0.45      0.49      5470

    accuracy                           0.59     12712
   macro avg       0.58      0.57      0.57     12712
weighted avg       0.58      0.59      0.58     12712

CONFUSION MATRIX

col_0     0.0   1.0
target
0.0      5004  2238
1.0      2986  2484
```

Из всех услуг низкой целевой метрикой резко выделяется услуга № 6. Поэтому попытался классифицировать эту услугу в отдельном ноутбуке, что не привело к положительному результату.

Поскольку это часто продаваемая услуга для ее классификации должен «выстрелить» uplift-метод. Он должен помочь.

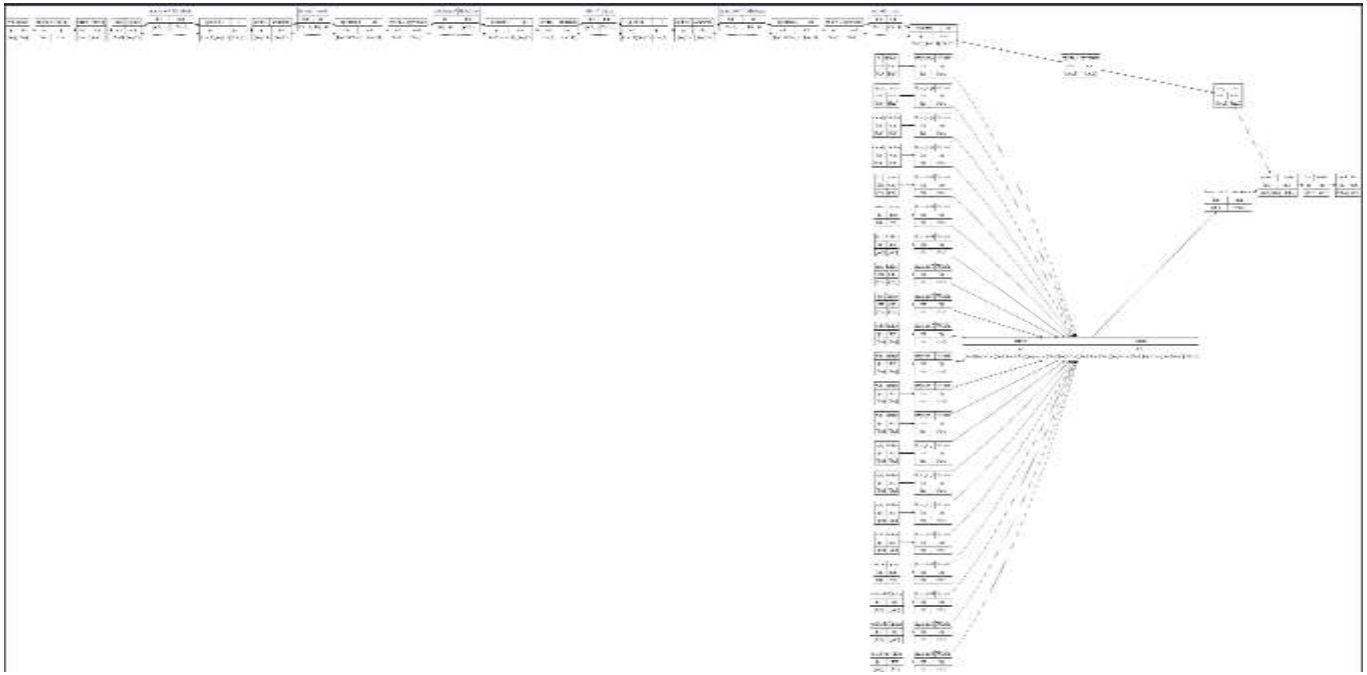
Выбор класификатора

```
0.0    0.90925
1.0    0.09075
Name: target, dtype: float64
0.0    0.909255
1.0    0.090745
Name: target, dtype: float64
TRAIN

              precision    recall  f1-score   support

         0.0       1.00      0.85      0.92    260142
         1.0       0.40      0.96      0.56     25964

    accuracy                           0.86    286106
   macro avg       0.70      0.91      0.74    286106
weighted avg       0.94      0.86      0.89    286106

TEST

              precision    recall  f1-score   support

         0.0       0.99      0.85      0.92    111491
         1.0       0.39      0.95      0.56     11127

    accuracy                           0.86    122618
   macro avg       0.69      0.90      0.74    122618
weighted avg       0.94      0.86      0.89    122618

CONFUSION MATRIX

col_0     0.0    1.0
target
0.0     95290  16201
1.0       586  10541
```

Из арсенала ML использовал Catboost, XGBoost и LGBM. Пробовал еще RandomForest, но они вообще ничего не смог сделать. Наименьшее переобучение при высокой метрике показал Catboost. Подбор гиперпараметров осуществлял встроенным методом Катбуста randomized_search. Метрику поднять не удалось, но переобучение полностью исчезло. С помощью Hyperopt пытался подобрать гиперпараметры у LGBM, но полностью убрать переобучение не удалось, а целевая метрика та же самая. С помощью подбора трешхолда удалось увеличить скор на 3%, поэтому в качестве классификатора создал дочерний от Катбуста класс MyCatBoostClassifier .

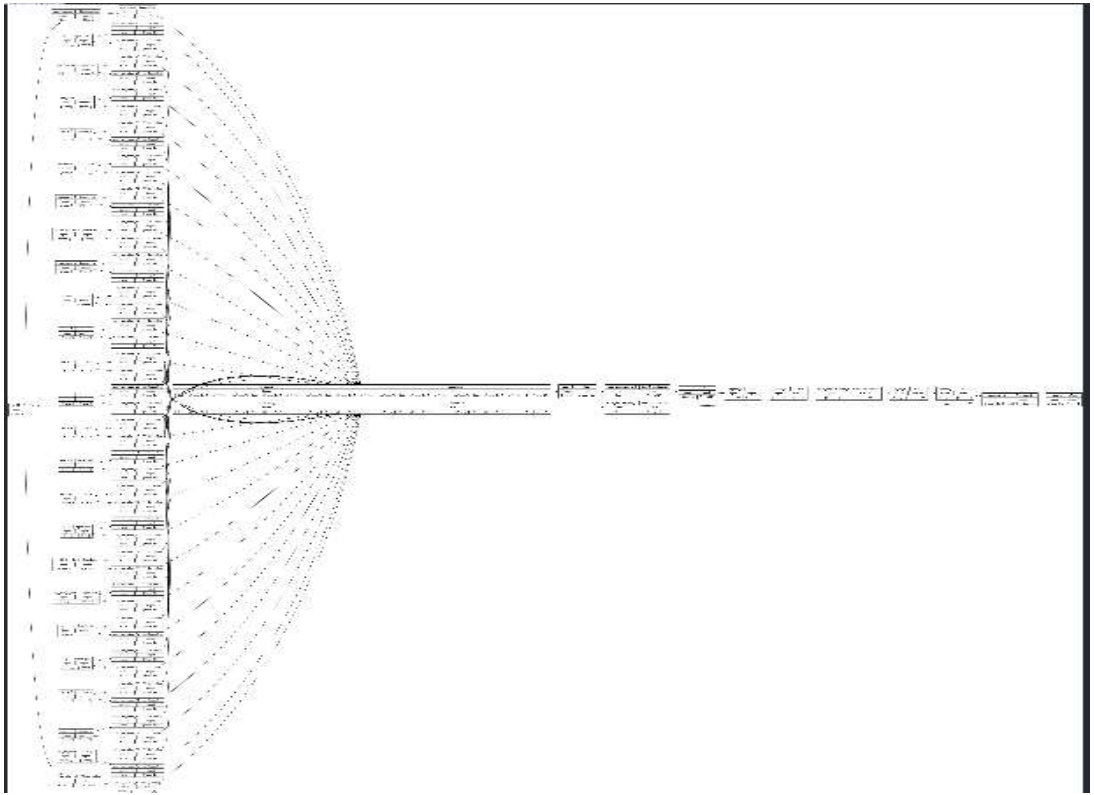
На кроссвалидации f1_score выдает метрику больше 0,9, поэтому ориентировался на валидационную метрику.

Выбор класификатора



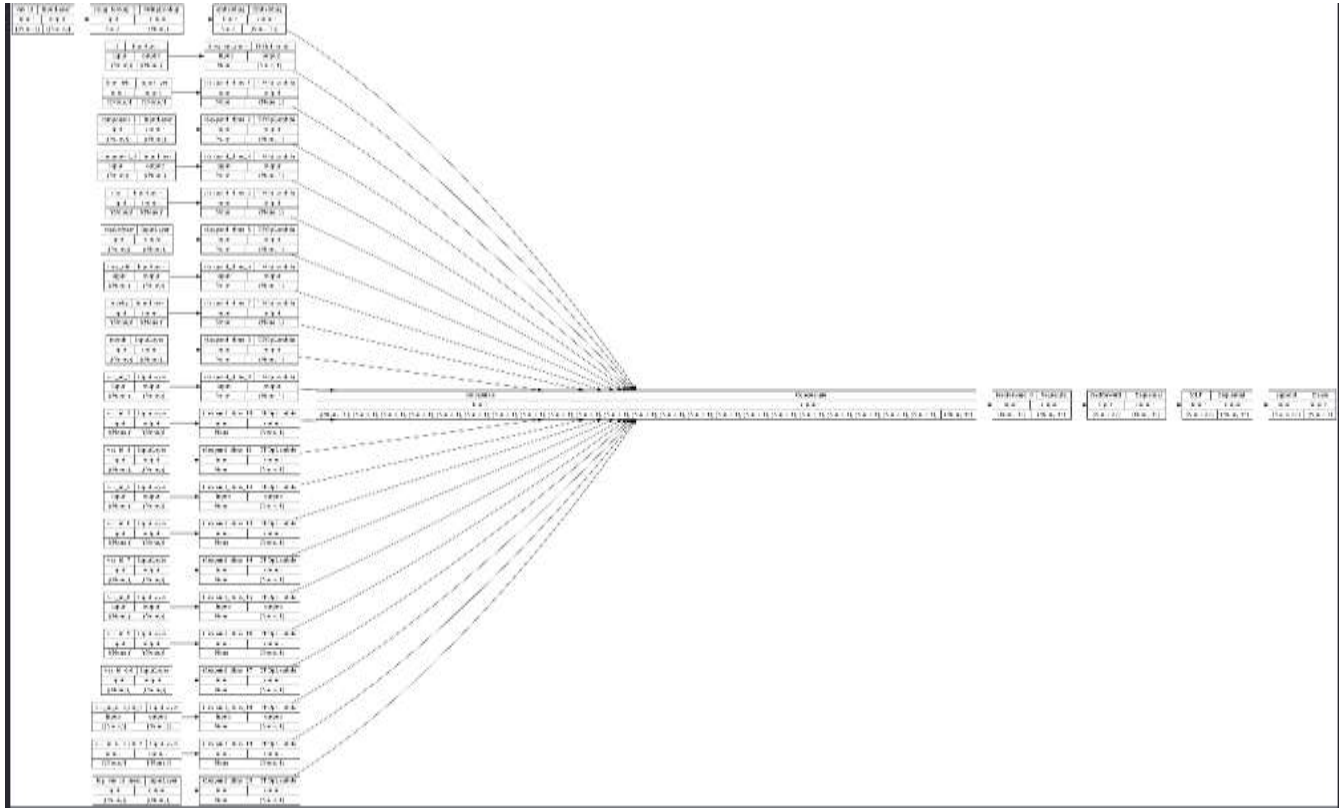
Multi Layer Perceptron не поймали никакой зависимости. Скатывались к предсказанию либо одних нулей либо единиц. Из арсенала Deep Learning сначала попробовал трансформер с такой архитектурой.
Максимальный f1_macro на нем 0,74.

Выбор класификатора



Композиция глубоких моделей тоже дала максимальный f1_macro 0,74.

Выбор класификатора



Максимальный f1_macro=0,77 дала такая глубокая модель. Но в отличие от Катбуста она все таки имеет небольшое переобучение. Тем не менее для практики я построил предсказатель и на этой модели.

Достигнутые цели

```
clf.fit(train.drop(columns=['target']),train['target'])
```

```
<__main__.MyCatBoostClassifier at 0x7f98ebd54dd0>
```

```
F1Scores = cross_val_score(MyCatBoostClassifier(random_state=101,
                    iterations=300,
                    max_depth=5,
                    l2_leaf_reg=9,
                    learning_rate=0.1,
                    class_weights=[1, 10.019],
                    cat_features=['vas_id'],
                    eval_metric='F1',
                    early_stopping_rounds=60,
                    silent=True
                    ),
                    train.drop(columns=['target']),train['target'],
                    cv=5,scoring='f1_macro')
F1Scores
```

```
array([0.77721091, 0.77695489, 0.77773293, 0.76963035, 0.7821129 ])
```

```
F1Scores.mean()
```

```
0.77672839466375
```

Удалось перебить baseline по целевой метрике более чем на 10%. Классификатор отлично предсказывает нули и плохо единицы. Я считаю, что это из-за дикого дисбаланса классов.

Пытался выровнять дисбаланс четырьмя методами. Синтетические методы не помогли.

Вероятно можно улучшить результат., если разбить датасет по разным услугам на 8 датасетов. Но я этот проект делаю для учебы а не для того, чтобы на нем умереть.

Сгенерированы новые признаки, определенные SHAPом как самые полезные. Проработаны дополнительные признаки для логирования поведения пользователей, необходимые для создания рекомендательной системы. На основе фреймворка Luigi создан product ready скрипт предсказателя.

Предложения по проекту

- Попробовать качественнее проработать 300 признаков для каждого клиента. Возможно, что нейронная сеть сможет тогда в них найти какие-то закономерности.
- Создать категориальные фичи из вещественных. Часть вещественных перейдет в категориальные при помощи RareLabelEncoder .
- Нагенерировать дополнительных фичей логирования .
- Добавить фичи логирования с «горячим стартом», используя похожесть клиентов.
- Произвести более глубокую настройку моделей на большем количестве параметров.
- Настроить более точный predict._proba для Катбуста с помощью Sklearn.

Идеи по созданию рекомендательной системы

Вообще на курсе по рекомендательным системам мы учили другие метрики и другие методы построения моделей. Но поскольку Мегафон предлагает такую метрику, будем строить систему по этой метрике.

Сразу же надо определиться с vas_id==6. Тут надо либо все таки получить высокую метрику. Либо, поскольку эта услуга достаточно популярна, попробовать uplift-модель. Для остальных вполне нормально зайдет response-модель, особенно когда история по клиентам значительно увеличится.

В связи с почти бесплатной для Мегафона стоимостью СМС, в качестве метрики предлагаю использовать recall, с разрешения клиента на рассылки конечно (чего смски на складе будут пылиться?) либо Fb._score в случае убыточности recall. Коэффициент b следует подобрать в зависимости от окупаемости акции.

Для решения о рассылке одной вероятности подключения будет маловато. В первую очередь конечно надо узнать не подключена ли уже услуга. И представьте что мы клиенту у которого высокая вероятность подключения услуги будем по 10 раз на дню слать приглашение? Для принятия решения о рассылке нужна еще фича или несколько фичей, учитывающих количество посланных ему приглашений , давность последнего приглашения, частота посылаемых приглашений, «старость» клиента, разница по времени между первым и последним приглашением. То есть сгенерированные мною признаки логирования. Другие услуги он покупает по акциям или нет и вообще покупал ли?