Финальная работа

Machine Learning Engineer

Сессии и целевые действия

В таблице ga_sessions.csv хранятся реквизиты сессии: когда и откуда пришёл клиент на сайт. А действия пользователей хранятся в таблице ga_hits.csv.

```
hits = pd.read_csv('E:/Code/Skillbox/ga_hits.csv')
```

```
ta_hits = hits[hits.event_action.isin(['sub_car_claim_click',
                                       'sub_car_claim_submit_click',
                                       'sub_open_dialog_click',
                                       'sub_custom_question_submit_click',
                                       'sub_call_number_click',
                                       'sub_callback_submit_click',
                                       'sub_submit_success',
                                       'sub_car_request_submit_click'])].copy()
```

```
sess_events = ta_hits.groupby(by='session_id', as_index=False).hit_number.min()
```

```
sessions = pd.read_csv('E:/Code/Skillbox/ga_sessions.csv')
```

```
sessions['hit_number'] = sessions.session_id.map(sess_events.set_index('session_id')['hit_number']
                                                 ).fillna(0).astype('int16')
sessions['hit'] = 0
sessions['hit'] = sessions['hit'].astype('int8')
sessions.loc[sessions.hit_number != 0, 'hit'] = 1
```

Теперь имеем единую таблицу с фичами и поле hit – 1/0 показывает было ли совершенно целевое действие или нет.

Data Understanding

Входные данные

Полученная таблица

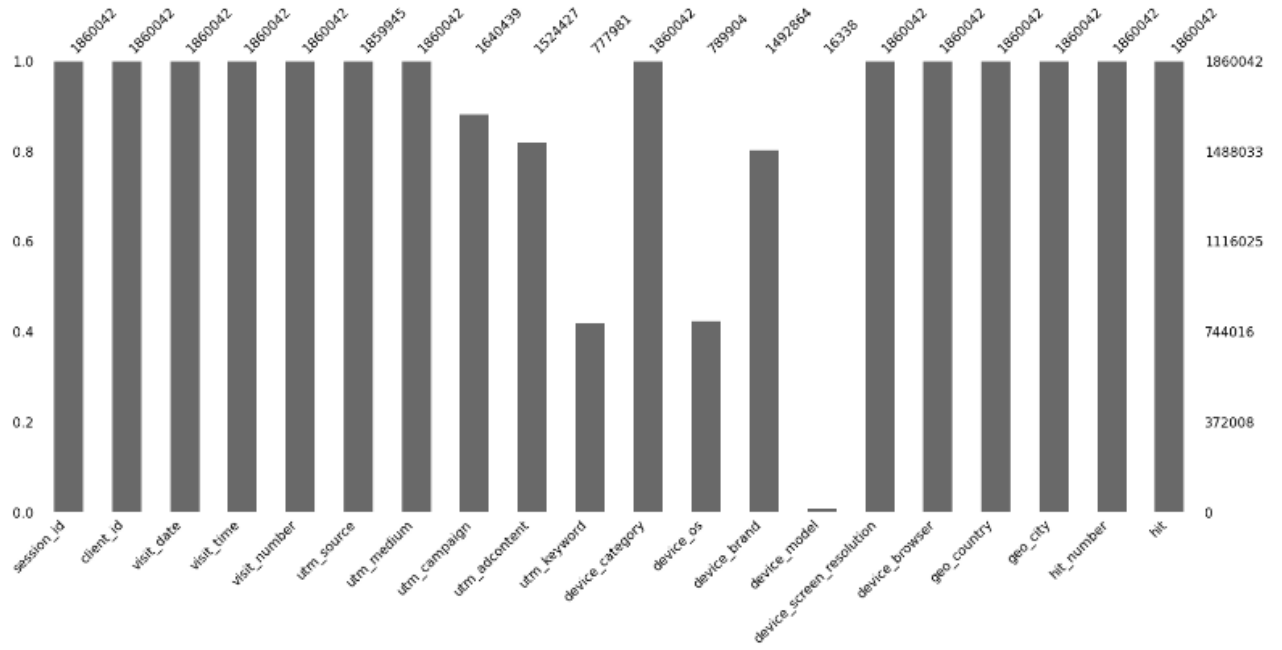
```
sessions.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1860042 entries, 0 to 1860041
Data columns (total 20 columns):
 #   Column                     Dtype 
---  ------                     ----- 
 0   session_id                 object
 1   client_id                  object
 2   visit_date                 object
 3   visit_time                 object
 4   visit_number               int64 
 5   utm_source                 object
 6   utm_medium                 object
 7   utm_campaign               object
 8   utm_adcontent              object
 9   utm_keyword                object
 10  device_category            object
 11  device_os                  object
 12  device_brand               object
 13  device_model               object
 14  device_screen_resolution   object
 15  device_browser             object
 16  geo_country                object
 17  geo_city                   object
 18  hit_number                 int16 
 19  hit                        int8  
dtypes: int16(1), int64(1), int8(1), object(17)
memory usage: 260.8+ MB
```

Пропуски в данных

```
msno.bar(sessions)
```

```
<AxesSubplot:>
```



Сразу визуально можно оценить поля, которые надо выкинуть из анализа
Выведем дополнительно % пропусков

```
(sessions.isna().sum() / sessions.shape[0]).sort_values()
```

```
session_id                 0.000000
geo_city                   0.000000
geo_country                0.000000
device_browser             0.000000
device_screen_resolution   0.000000
device_category            0.000000
hit_number                 0.000000
hit                        0.000000
utm_medium                 0.000000
visit_number               0.000000
visit_time                 0.000000
visit_date                 0.000000
client_id                  0.000000
utm_source                 0.000052
utm_campaign               0.118063
utm_adcontent              0.180434
device_brand               0.197403
device_os                  0.575330
utm_keyword                0.581740
device_model               0.991216
dtype: float64
```

По показателям – более 20% – исключаем из анализа 3 поля.

```
sessions = sessions.drop(columns=[
                                  'device_os', 'utm_keyword', 'device_model'])
```

Дата и время визита

Преобразовываем дату время

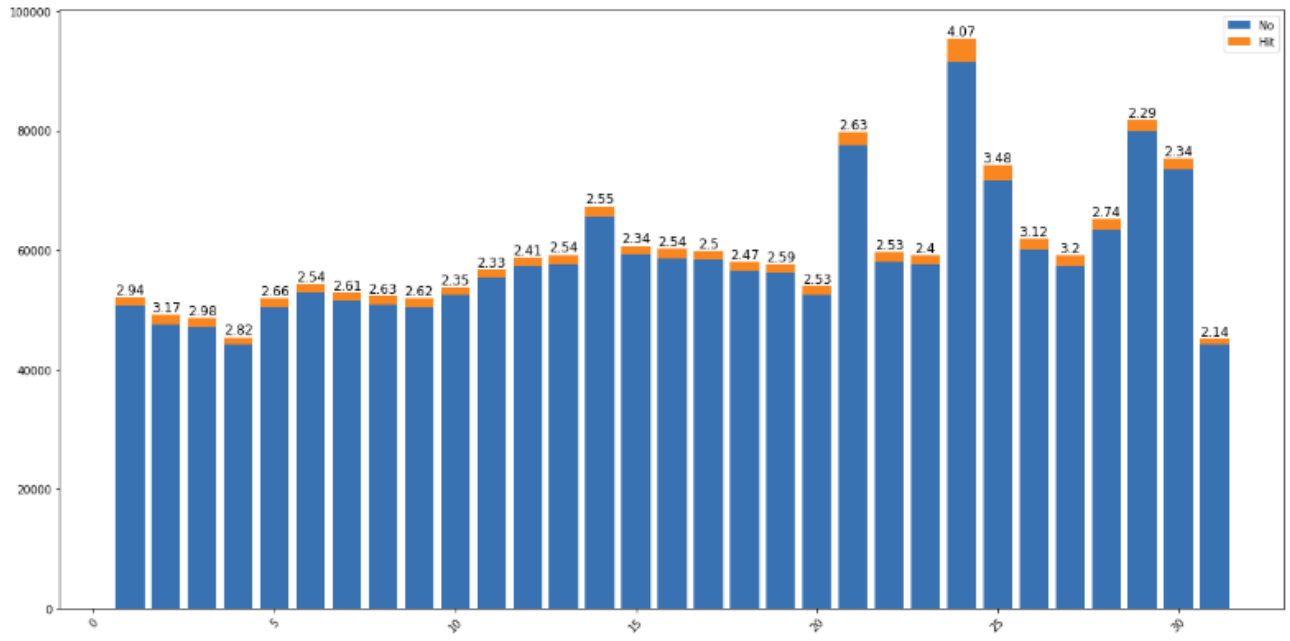
```
sessions['v_date'] = pd.to_datetime(sessions.visit_date)
sessions['day'] = sessions.v_date.dt.day.astype('int8')
sessions['week_day'] = (sessions.v_date.dt.day_of_week + 1).astype('int8')
```

```
sessions['month_frombeg'] = (sessions.v_date.dt.year - 2021) * 12
                          + sessions.v_date.dt.month.astype('int8') - 5
```

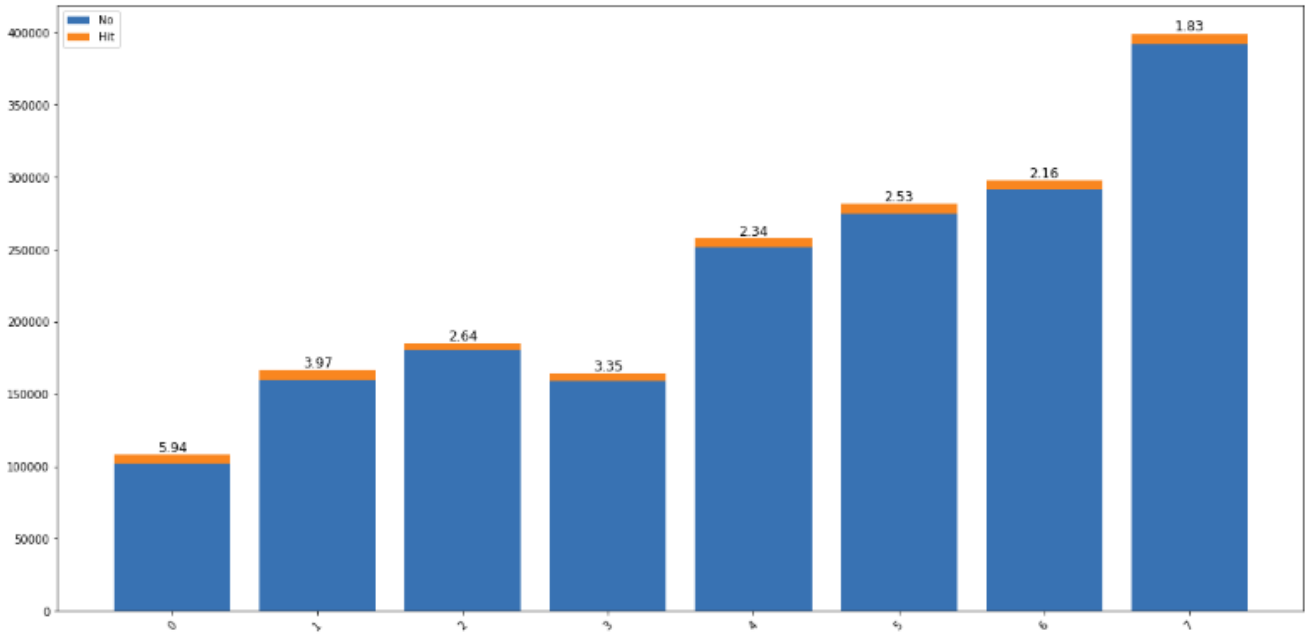
```
sessions['v_time'] = sessions.visit_time.apply(lambda x: x[:2]).astype('int8')
```

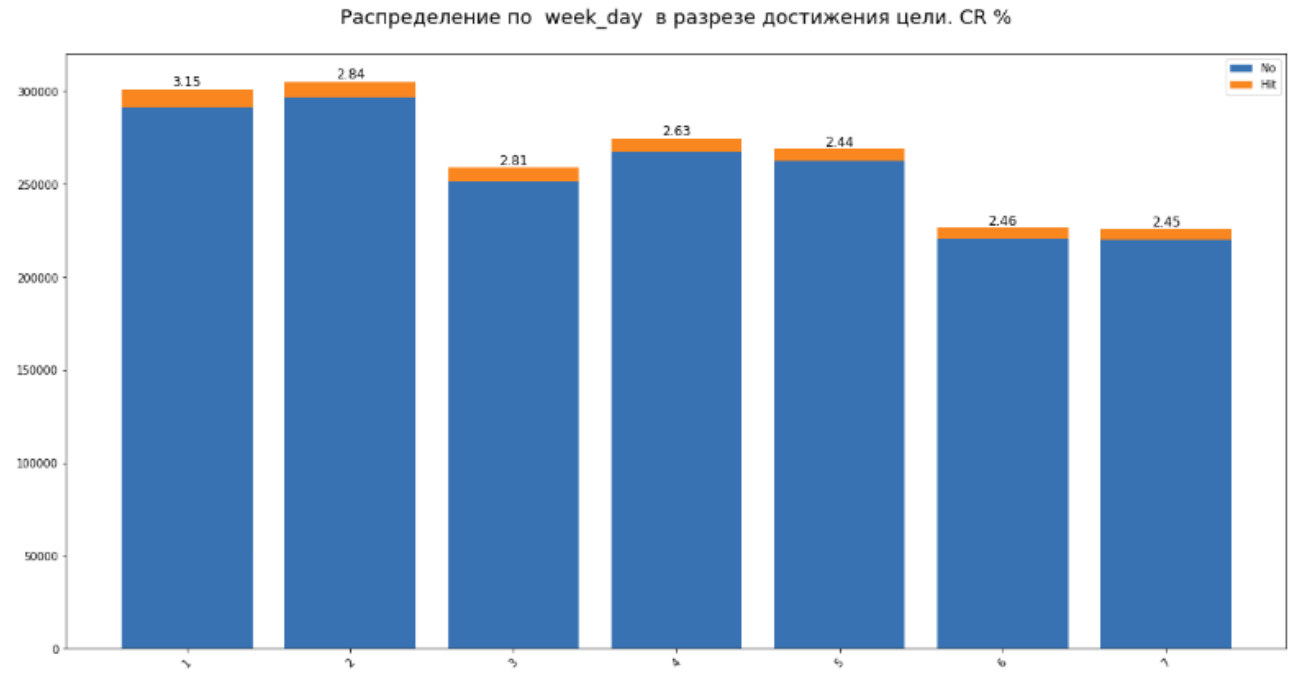
Проанализируем распределение по переменным с оценкой Conversion Rate

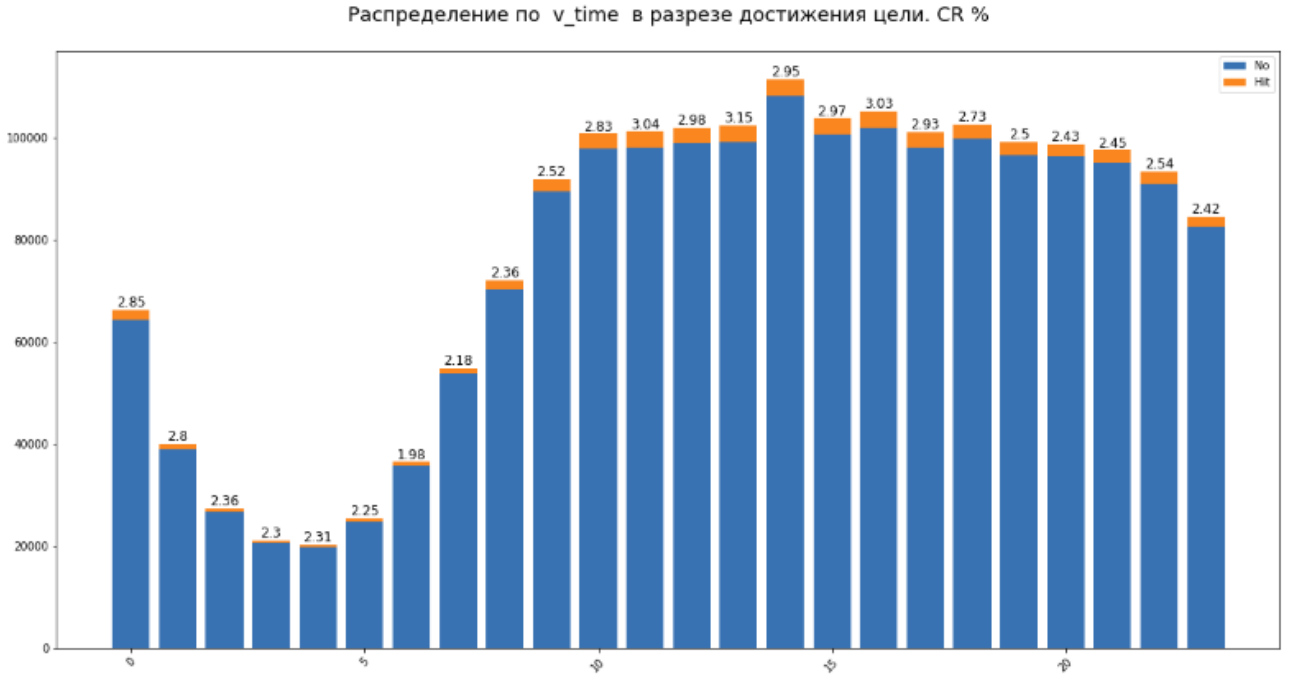
Распределение по day в разрезе достижения цели. CR %



Распределение по month_frombeg в разрезе достижения цели. CR %







На графиках видно, что есть зависимость конверсии от дня месяца, дня недели. Соответственно, данные переменные добавляем в финальную модель.

Devices

Анализируемые переменные:
dev_list = ['device_category',
'device_brand',
'device_screen_resolution',
'device_browser']

device_category	count	sum	cr
desktop	366863	11513.0	3.138229
mobile	1474871	38379.0	2.602194
tablet	18308	422.0	2.305003

device_brand	count	sum	cr
Apple	551088	14467.0	2.625171
Samsung	332194	10054.0	3.026545
Xiaomi	288367	6592.0	2.285976
Huawei	185853	4519.0	2.431492
Realme	19058	421.0	2.209046
(not set)	17545	454.0	2.587632
OPPO	13504	396.0	2.932464
Vivo	11740	322.0	2.742760
OnePlus	9965	157.0	1.575514
Asus	7929	144.0	1.816118
Nokia	7285	165.0	2.264928

По анализу видна зависимость целевой переменной от типа устройства и бренда.

По переменной device_screen_resolution формируем переменные высота экрана, ширина экрана и пиксели и для каждой из них обрабатываем выбросы. Предел для выбросов формируем так, чтобы % выбросов был не выше 2%.

GEO переменные

2 гео переменные:
geo_country
geo_city

geo_country	count	sum	share	cr
Russia	1800565	49156.0	0.968024	2.730032
United States	11784	64.0	0.006335	0.543109
Ukraine	9012	199.0	0.004845	2.208167
Ireland	4034	1.0	0.002169	0.024789
Belarus	3636	59.0	0.001955	1.622662
Sweden	2694	26.0	0.001448	0.965108
Kazakhstan	2279	54.0	0.001225	2.369460
Germany	2232	78.0	0.001200	3.494624
Turkey	1953	48.0	0.001050	2.457757
Netherlands	1549	38.0	0.000833	2.453196

Как видим 1,8 млн визитов из 1,86 приходится на Россию. Это 96.8%
Выведем количество уникальных городов для каждой страны:

geo_country	nunique
Russia	506
United States	318
Germany	239
Ukraine	209
United Kingdom	91
France	79
Italy	68
Turkey	54
Netherlands	49
Spain	47

Для России очень высокое количество городов, при этом 10 городов занимают 75% визитов

248	Moscow	805329	23629.0	2.934080
351	Saint Petersburg	296788	7113.0	2.396660
0	(not set)	70021	1361.0	1.943703
480	Yekaterinburg	35788	887.0	2.478484
188	Krasnodar	32243	1081.0	3.352666
147	Kazan	29531	1139.0	3.856964
355	Samara	24992	727.0	2.908931
270	Nizhny Novgorod	22227	559.0	2.514959
431	Ufa	21679	639.0	2.947553
286	Novosibirsk	21568	509.0	2.359978
192	Krasnoyarsk	16346	376.0	2.300257
77	Chelyabinsk	15951	397.0	2.488872
424	Tula	15814	438.0	2.769698
462	Voronezh	13908	282.0	2.027610
343	Rostov-on-Don	13886	397.0	2.858995
124	Irkutsk	13532	327.0	2.416494
119	Grozny	12742	401.0	3.147073
40	Balashikha	12679	359.0	2.831454
453	Vladivostok	12325	249.0	2.020284
478	Yaroslavl	9833	238.0	2.420421
387	Sochi	8972	316.0	3.522069

Добавим категорию – регион. Это позволит объединит большое количество например визитов из Подмосковья.

Количество визитов из-за границы 3,2%, и дополнительная переменная города будет лишней.

Итого по гео будут следующие переменные:

Россия / Заграница

Регион – Регион для России, а для заграницы – Страна

Москва как регион остаётся с тем же количеством, при этом на 3е место выходит по сумме визитов Московская область.

region	count	sum	share	cr
Moscow	805330	23629.0	0.432963	2.934077
Saint Petersburg	296788	7113.0	0.159560	2.396660
Moscow Oblast	109702	3106.0	0.058978	2.831307
Krasnodar Krai	45446	1508.0	0.024433	3.318224
Sverdlovsk Oblast	38499	948.0	0.020698	2.462402
Republic of Tatarstan	32869	1241.0	0.017671	3.775594
Samara Oblast	26902	777.0	0.014463	2.888261
Republic of Bashkortostan	23953	696.0	0.012878	2.905690
Nizhny Novgorod Oblast	23519	586.0	0.012644	2.491603
Novosibirsk Oblast	21786	517.0	0.011713	2.373084
Krasnoyarsk Krai	17847	407.0	0.009595	2.280495
Chelyabinsk Oblast	17331	426.0	0.009318	2.458023
Tula Oblast	16429	456.0	0.008833	2.775580
Rostov Oblast	15358	430.0	0.008257	2.799844
Irkutsk Oblast	14549	352.0	0.007822	2.419410
Voronezh Oblast	14054	287.0	0.007556	2.042123
Primorsky Krai	12892	259.0	0.006931	2.008998
Chechen Republic	12809	401.0	0.006886	3.130611
United States	11784	64.0	0.006335	0.543109
Yaroslavl Oblast	10231	247.0	0.005500	2.414231
Ukraine	9012	199.0	0.004845	2.208167
Leningrad Oblast	8776	168.0	0.004718	1.914312

UTM метки

utm_keyword заполнен меньше чем у половины визитов и был удален

Остальные категориальные признаки идут в модель

```
utm_list = ['utm_source',
            'utm_medium',
            'utm_campaign',
            'utm_adcontent'
           ]
```

Кроме того, рассмотрим ещё два булева признака касательно трафика: органический – да/нет и Соцсети – да/нет

Показатели по признаку «Органический трафик»

organic	count	sum	cr
0	1344383	29502.0	0.021945
1	515659	20812.0	0.040360

Показатели по признаку «Трафик из соцсетей»

sm	count	sum	cr
0	1585815	46293.0	0.029192
1	274227	4021.0	0.014663

Pipeline

```
pipe = Pipeline(steps=[
        ('data_preparation',  data_prep),
        ('preprocessor', preprocessor),
        ('classifier', model)
      ])
```

Data_preparation

```
data_prep = Pipeline(steps=[
    ('regions', FunctionTransformer(regions)),
    ('new_features', FunctionTransformer(new_features)),
    ('df_filter', FunctionTransformer(df_filter)),
    ('outliers', FunctionTransformer(outliers))
    ])
```

В соответствии с анализом данных, подготовка данных состоит из:
regions – Добавление федерального субъекта по городу (Для РФ)
new_features – функция формирует "дополнительные" фичи
производные от времени визита:
- день месяца, день недели, номер месяца от запуска сервиса
#по разрешению экрана: ширина, высота, пиксели
Трафик органический / НЕорганический
СоцСети / прочие

df_filter – удаление «лишних» полей
outliers – функция для обработки выбросов по параметрам: высота, ширина экрана, пиксели

Preprocessor

Стандартная обработка

```
numerical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())
    ])

categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('encoder', OneHotEncoder(handle_unknown='ignore'))
    ])

preprocessor = ColumnTransformer(transformers=[
    ('numerical', numerical_transformer,
      make_column_selector(dtype_include=np.number)),
    ('categorical', categorical_transformer,
      make_column_selector(dtype_include=object))
      ])
```

Classifier

На этапе разработки анализируются 2 модели:

```
models = (
    LogisticRegression(C=10.0,
                       class_weight='balanced',
                       max_iter=200,
                       multi_class='ovr',
                       n_jobs=-1,
                       solver='newton-cholesky',
                       tol=0.001),
    MLPClassifier(activation='logistic',
                  alpha=0.001,
                  early_stopping=True,
                  random_state=12,
                  warm_start=True)
    )
```

Best model


```
"name": "Hit target prediction model",
"author": "AK",
"version": 1,
"data": "2023-05-16T11:01:45.212630",
"type": "MLPClassifier",
"roc_auc": 0.7378800461516408
```


Prediction

Модель принимает на вход следующие параметры:

```
class Form(BaseModel):
    session_id: str
    client_id: str
    visit_date: str
    visit_time: str
    visit_number: int
    utm_source: str
    utm_medium: str
    utm_campaign: str
    utm_adcontent: str
    utm_keyword: str
    device_category: str
    device_os: str
    device_brand: str
    device_model: str
    device_screen_resolution: str
    device_browser: str
    geo_country: str
    geo_city: str
```