Финальная работа по курсу Skillbox “Введение в Data Science”

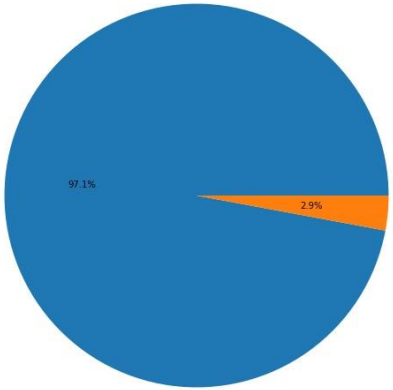
Специализация Machine Learning Engineer

Данные:	Сайт сервиса СберАвтоподписка
Задача:	Предсказать совершение целевого действия пользователем на основе характеристик сессии
Целевая метрика:	ROC AUC >= 0.65

Особенности датасета:

- 1732266 строк в рабочем датасете

- Несбалансированное соотношение классов целевой переменной в датасете



- Преобладание категориальных признаков со множеством значений внутри категорий

Образец записи в исходном датасете

session_id	905544597018549464.1636867290.1636867290
client_id	210838531.163687
visit_date	2021-11-14
visit_time	08:21:30
visit_number	1
utm_source	MvfHsxITijuriZxsqZqt
utm_medium	cpm
utm_campaign	FTjNLDyTrXaWYgZymFkV
utm_adcontent	xhoenQgDQsgfEPYNPwKO
utm_keyword	IGUCNvHlhfHpROGclCit
device_category	mobile
device_os	Android
device_brand	Samsung
device_model	NaN
device_screen_resolution	385x854
device_browser	Samsung Internet
geo_country	Russia
geo_city	Moscow
target	0
Name: 1, dtype: object	

Основные сложности при подготовке модели

Вычислительные ресурсы

Не допускать чрезмерного раздувания датасета за счет нерелевантных признаков

Использовать доступные облачные ресурсы (Yandex Datasphere, Google Colab)

Контролировать потребление вычислительных ресурсов и время обработки в процессе разработки и обучения

Преобладание категориальных признаков

Сократить количество значений внутри категорий

Создать новые числовые признаки на основе категорийных

Несбалансированность выборки

Использовать инструменты и алгоритмы предназначенные для работы с несбалансированными данными

Feature Engineering

Геоданные

Признаки на основе частотности вхождения в датасет

Признаки с использованием Isolation Forest Classifier

Нетекстовые категориальные признаки

Кодирование текстовых значений категориальных признаков

Стандартизация числовых признаков

Геоданные

IN

geo_city

1. Пропуски, неизвестные и невалидные значения заполняем значением столицы соответствующей страны из предобработанной таблицы
2. Чтобы избавиться от категориальных данных, из предобработанной таблицы загружаем географические координаты, создаем признаки из географических координат lat и lng
3. Оставшиеся пропуски заполняем с помощью Geolocator

geo_country

1. Считаем, что по умолчанию пользователи из России, заполняем пропуски и неизвестные значения. Пока не отбрасываем пользователей из других стран, т.к. пользователи могут использовать VPN или временно находиться за границей
2. Чтобы сократить количество значений в категории, с помощью предобработанной таблицы добавляем признак geo_region

OUT

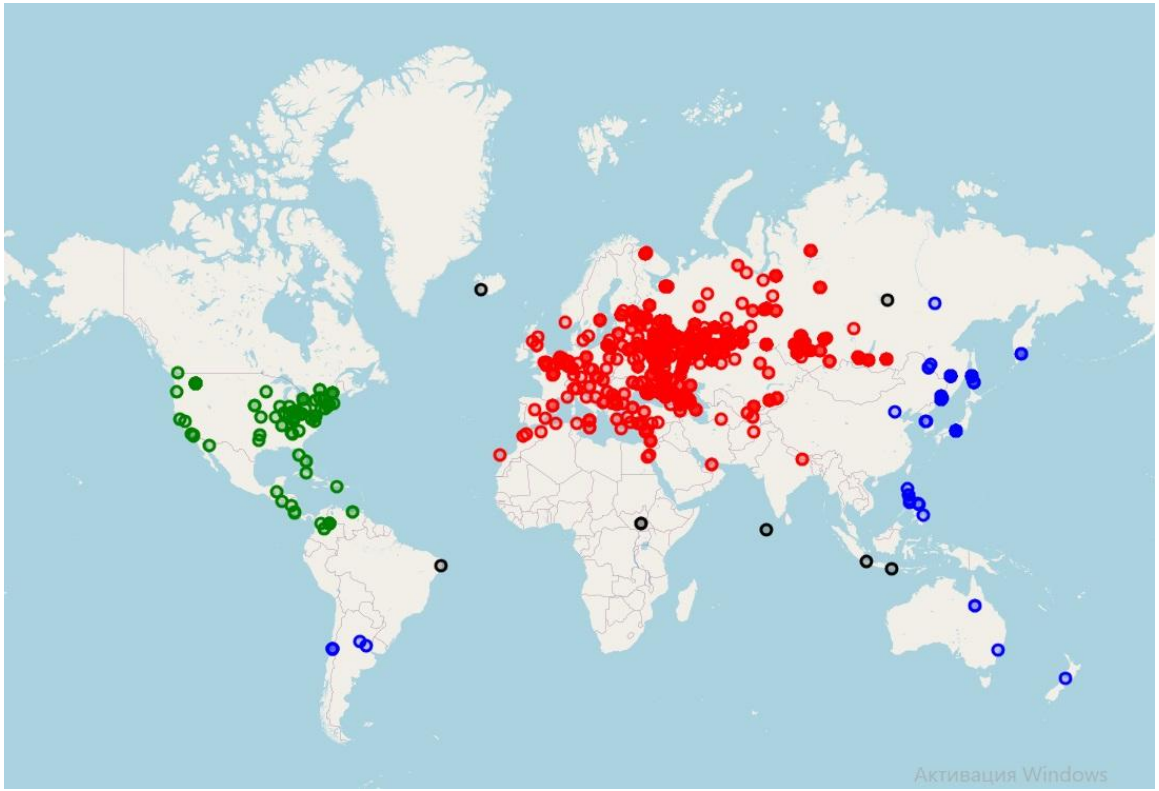
lat

lng

geo_region

Визуализация пользователей по кластерам

В качестве промежуточного эксперимента на полученных данных была обучена модель DBSCAN. Результаты кластеризации планировалось использоваться в качестве одного из признаков. Признак оказался не очень полезным для и в финальную версию дополнительная кластеризация не вошла, чтобы не утяжелять модель. Здесь визуализация приведена в качестве иллюстративного материала для интерпретации полученных признаков координат.



Работа с нетекстовыми категориальными признаками

IN

visit_date

visit_time

device_screen_resolution

*данные предоставлены за один неполный год, беглый анализ не выявил сезонности, поэтому признаки год и месяц считаем нерелевантными

- Габариты экрана преобразованы в площадь
- С помощью метода интерквартильного размаха обработаны выбросы

OUT

visit_day

visit_dayofweek

visit_hour

device_screen_resolution

Работа с текстовыми категориальными признаками

- Выявление лидеров в отдельных категориях
- Объединение редко встречающихся значений в одно
- Бинаризация значений категории по признаку вхождения в заданное множество
  - utm_medim - organic/paid, множество значений organic было описано в глоссарии к задаче
  - utm_source - social net/other, множество social net также было описано в глоссарии
  - utm_campaign - ввиду отсутствия информации о значениях в глоссарии, в этом случае для бинаризации использовался прихзнак вхождения в число трех лидирующих по частотности значений категории.
- Применение OneHotEncoder для кодирования оставшихся текстовых значений категорий

Признаки на основе частотности вхождений

Предварительные эксперименты с обучением на тренировочной выборке показали, что все алгоритмы позитивно реагируют на увеличение количества числовых признаков в обучении, при этом добавление бинарных признаков (0, 1), (-1, 1)дает более слабый эффект.

Для следующих категорий были рассчитаны значения частотности вхождения каждого значения:

'utm_adcontent',
'utm_source',
'utm_medium',
'geo_region', 'geo_city',
'visit_dayofweek',
'visit_day', 'visit_hour',
'device_browser',
'device_brand',
'device_screen_resolution
'.

*Частотность для значений признаков utm_ и device_screen_resolution рассчитывалась ДО преобразования значений в категориях.

Признаки с использованием Isolation Forest

Добавление серии признаков _freq улучшило метрику, но предел явно еще не был достигнут, поэтому возникла идея поэкспериментировать с IsolationForest.

Почему IsolationForest: из-за несбалансированности классов целевой переменной перед нами стоит фактически не задача бинарной классификации, а задача детектирования аномалий, где аномальным является совершение целевого действия
При этом из-за размера датасета невозможно эффективно применить, к примеру, OneClassSVM, даже на одном признаке время обучения выходило за рамки разумного.
IsolationForest отрабатывал быстро, для основного алгоритма точность была недостаточной, но идея добавить еще одну серию признаков по аналогии с _freq с использованием IsolationForest показалась (и все еще кажется) удачной.
В качестве значений новых признаков использовались score_samples.

Новые признаки были созданы для:
['utm_adcontent_freq', 'utm_source_freq', 'paid_traffic', 'utm_medium_freq', 'top_3_campaigns', 'device_screen_resolution_freq', 'device_browser_freq', 'lat','lng', 'social_net', 'geo_city_freq', 'visit_hour_freq', 'visit_day', 'device_screen_resolution']

Для создания новых признаков были отобраны признаки, лидирующие в списке feature_importances по результатам предварительного обучения GradientBoostClassifier.

Стандартизация числовых признаков

Стандартизации с использованием StandardScaler() были подвергнуты следующие небинарные числовые признаки:

['visit_number', 'device_screen_resolution', 'lat', 'lng', 'visit_dayofweek', 'visit_day', 'visit_hour', 'utm_adcontent_freq', 'utm_source_freq', 'utm_medium_freq', 'geo_region_freq', 'geo_city_freq', 'visit_dayofweek_freq', 'visit_day_freq', 'visit_hour_freq', 'device_browser_freq', 'device_brand_freq', 'device_screen_resolution_freq']

Удаление нерелевантных признаков

Признаки, которые не вошли в датасет для обучения:

'device_model' (более 50% пропусков)
'client_id' (id)
'session_id' (id)
'device_os' (большой % пропусков, при этом OS зависит от марки телефона)
'utm_keyword (большой процент пропусков)'

Признаки, которые были отвергнуты в пользу новых признаков, сконструированных на их основе

'geo_city'
'geo_country',
'visit_time',
'visit_date',
'geo_region',
'utm_adcontent',
'utm_campaign',
'utm_source',
'utm_medium'

Признаки, использованные для обучения

54 признака

visit_number	float64	device_category_desktop	float64	paid_traffic	int64
device_screen_resolution	float64	device_category_mobile	float64	social_net	int64
lat	float64	device_category_tablet	float64	top_3_campaigns	int64
lng	float64	device_brand_Apple	float64	utm_adcontent_freqisf	float64
visit_dayofweek	float64	device_brand_Huawei	float64	utm_source_freqisf	float64
visit_day	float64	device_brand_Realme	float64	paid_trafficisf	float64
visit_hour	float64	device_brand_Samsung	float64	utm_medium_freqisf	float64
utm_adcontent_freq	float64	device_brand_Xiaomi	float64	top_3_campaignsisf	float64
utm_source_freq	float64	device_brand_other	float64	Device_screen_resolution_freqisf	float64
utm_medium_freq	float64	device_brand_unknown	float64	device_browser_freqisf	float64
geo_region_freq	float64	device_browser_Android Webview	float64	latisf	float64
geo_city_freq	float64	device_browser_Chrome	float64	lngisf	float64
visit_dayofweek_freq	float64	device_browser_Edge	float64	social_netisf	float64
visit_day_freq	float64	device_browser_Firefox	float64	geo_city_freqisf	float64
visit_hour_freq	float64	device_browser_Opera	float64	visit_hour_freqisf	float64
device_browser_freq	float64	device_browser_Safari	float64	visit_dayisf	float64
device_brand_freq	float64	device_browser_Samsung Internet	float64	device_screen_resolutionisf	float64
device_screen_resolution_freq	float64	device_browser_YaBrowser	float64		
device_category_desktop	float64	device_browser_other	float64		

Пайплайн

```
Pipeline(steps=[
        ('preprocessor', Pipeline(steps=
                        [('constructor',  Pipeline(steps=
                                [('get coordinates',  FunctionTransformer(get_coordinates)),  lat,  lng
                                   ('get region', FunctionTransformer(get_region)), добавляется признак geo_region
                                  ('process date', FunctionTransformer(process_date_time)) visit_day, visit_dayofweek, visit_hour
                                   ('add frequencies', FunctionTransformer(add_freqs)) серия признаков _freqs
                                   ('add categories', FunctionTransformer(add_categories)) бинаризация, сокращение значений категорий
                                  ('screen resolution', FunctionTransformer(multiply_res)) преобразование габаритов в площадь
                         )]
                           ('encoder', ColumnTransformer(remainder='passthrough', verbose_feature_names_out=False,
                                   transformers=[('num', numerical_transformer, ['visit_number', 'device_screen_resolution', 'lat', 'lng',   'visit_dayofweek',
                          'visit_day', 'visit_hour', 'utm_adcontent_freq', 'utm_source_freq', 'utm_medium_freq', 'geo_region_freq', 'geo_city_freq',
                          'visit_dayofweek_freq', 'visit_day_freq', 'visit_hour_freq', 'device_browser_freq', 'device_brand_freq', 'device_screen_resolution_freq' ]),
                          StandardScaler()
                                                ('cat', categorical_transformer,  ['device_category', 'device_brand', 'device_browser']),
                                           ]).set_output(transform='pandas') OneHotEncoder()


                           ('iso', FunctionTransformer(add_iso_features) серия признаков с использованием предобученного IsolationForest
                         ('cleaner',  Pipeline(steps=[('outliers', FunctionTransformer(treat_outliers)) выбросы в visit_number и device_screen_resolution
                                           ('filter cols', FunctionTransformer(filter_cols))]))])) удаление колонок
      ('classifier', GradientBoostingClassifier(n_estimators=200))])
```

* Перед применением финального пайплайна для обучения IsolationForest отдельно запускается усеченный вариант (constructor, encoder)

Modelling

Выборки для обучения и тестирования

Обучение с использованием RandomUnderSampling и без

Выбор алгоритма

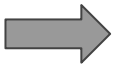
Выводы

Обучение базовых моделей

*Эксперименты с обучением разных моделей проводились еще на этапе Feature Engineering, поэтому к моменту сборки пайплайна уже сложилось определенное понимание того, что можно от них ожидать. В частности, сложилось понимание, что изменение параметров на метрики и качество обучения существенно не влияет, обеспечивая колебания в районе 1-1,5 процентов, сопоставимые с колебаниями при многократном случайном разбиении выборок, при этом самые результативные комбинации приводят к кратному увеличению времени обучения и потребляемых ресурсов. Основной прирост метрики обеспечен добавлением новых числовых признаков на этапе Feature Engineering.

Целевая метрика: ROC AUC >=0.65

Значения ROC AUC,полученные классификаторами (параметры классификаторов при первом проходе не менялись)



	Train	Test
RandomForestClassifier	0.997	0.620
GradientBoostingClassifier	0.672	0.666
DecisionTreeClassifier	0.998	0.516
XGBClassifier	0.668	0.664
MLPClassifier	0.681	0.665

Настройка классификатора

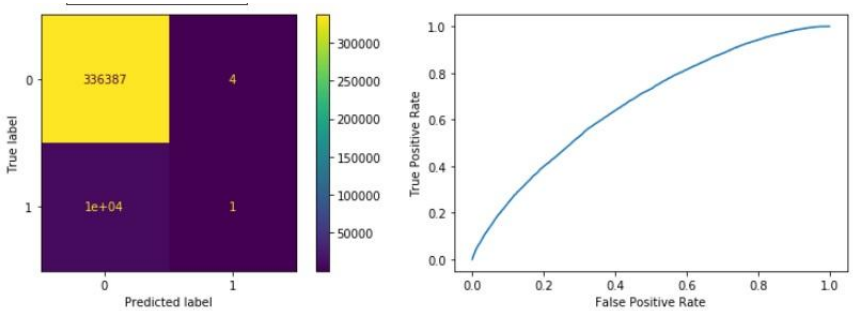
GradientBoostClassifier, XGBClassifier и MLPClassifer даже с параметрами по умолчанию стабильно преодолевают желаемый порог целевой метрики.
Увеличение значений n_estimators, n_iterators, и hidden_layer_sizes позволяет еще немного улучшить метрику, но время обработки и потребление ресурсов возрастают.

	Test	Train
GradientBoostClassifier(n_estimators=200)	0.670	0.679
XGBClassifier(n_iterators=200)	0.664	0.668
MLPClassifier(hidden_layer_sizes=(100, 50)	0.668	0.689

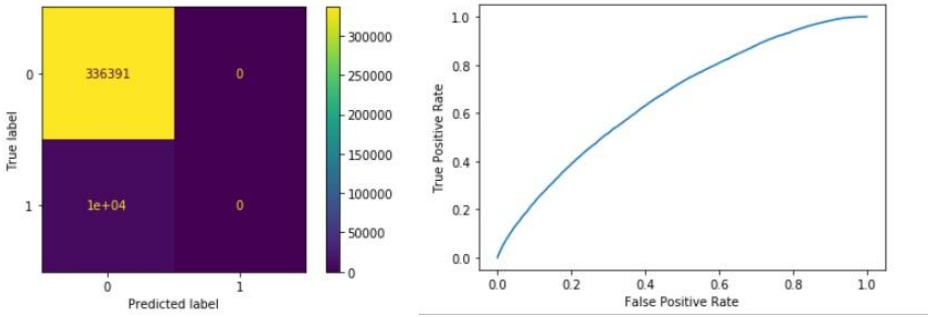
*Если ресурсы позволяют, можно еще увеличить число n_estimators и еще немного подтянуть метрику. В моем конкретном случае 200 предел, после которого обучение становится слишком затратным и не оправдывает полученный прирост качества.
Настройка других параметров существенного эффекта не дала, поэтому ограничиваемся одним параметром.

Матрицы ошибок и построение кривой ROC

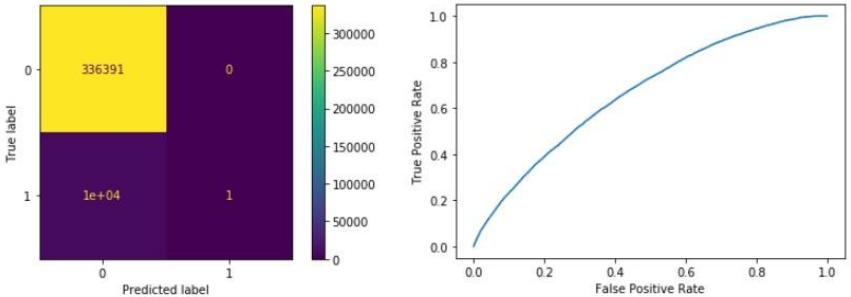
GradientBoostClassifier



XGBClassifier



MLPClassifier



Результаты работы трех отобранных классификаторов практически идентичны. Целевое действие, в имеющейся выборке, по сути являющееся аномалией, не ловит ни один, несмотря на достижение целевого значения метрики.

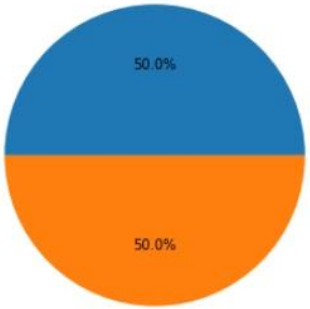
Обучение с использованием RandomUnderSampler

Новая обучающая выборка:

```
from imblearn.under_sampling import RandomUnderSampler
rus = RandomUnderSampler()
x_resampled, y_resampled = rus.fit_resample(x_train, y_train)
```

```
#!c1.8
print('Размер новой обучающей выборки', x_resampled.shape)
print('Соотношение классов целевой переменной в новой выборке')
plt.subplots(figsize=(5, 5))
plt.pie(y_resampled.value_counts(), autopct='%1.1f%%');
```

```
Размер новой обучающей выборки (80502, 54)
Соотношение классов целевой переменной в новой выборке
```

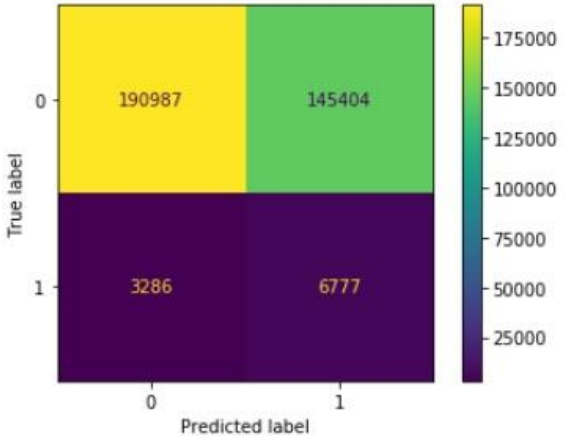


	Train	Test
GradientBoostingClassifier(n_estimators=200)	0.687	0.669
MLPClassifier(hidden_layer_sizes=(100, 50)	0.691	0.668
XGBClassifier(n_iterators=200)	0.676	0.666

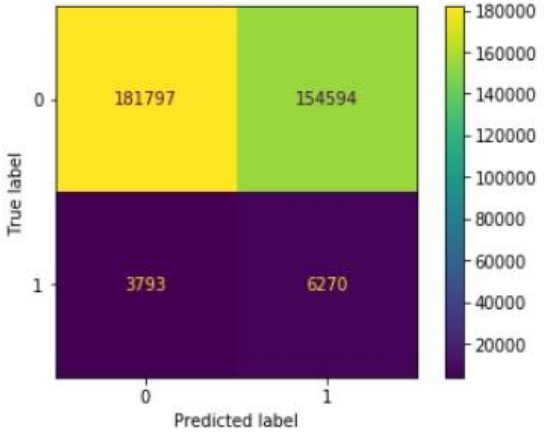
Существенного прироста метрики на сбалансированной по классам целевой переменной выборки нет. Для лучшего классификатора (GradientBoostingClassifier) разница на тестовой выборке составляет 0.001

Матрицы ошибок при обучении на сбалансированной выборке

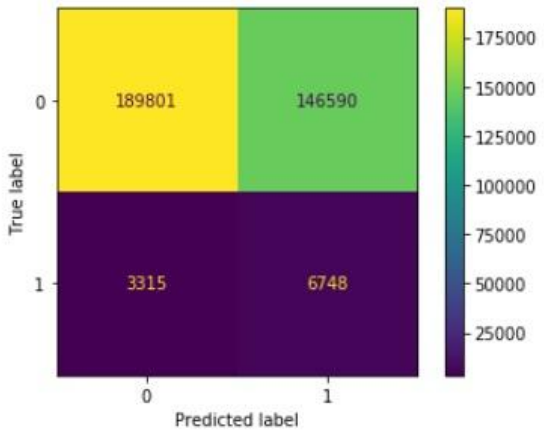
GradientBoostingClassifier



MLPClassifier



XGBClassifier



Ожидаемо, на сбалансированной выборке ошибка смещена в сторону лишних предсказанных единиц. Результаты работы трех алгоритмов при этом мало отличаются.

Выводы

- Целевое значение метрики достигнуто.

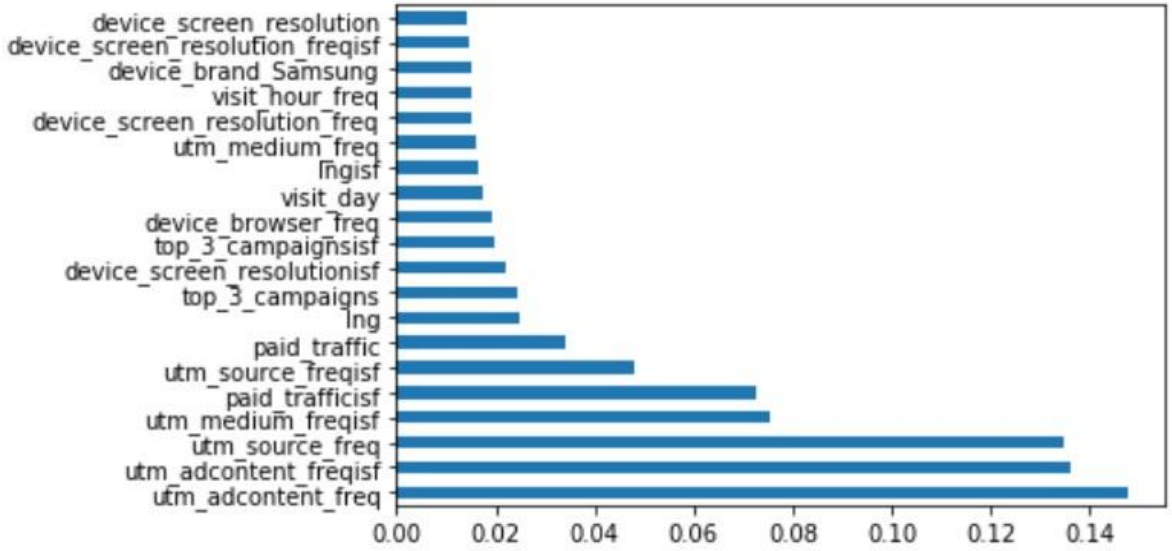
- Задача предсказания совершения целевого действия пользователем на имеющихся данных скорее является задачей детектирования аномалий, нежели задачей бинарной классификации, и требует другого инструментария и подходов.

- В рамках текущего проекта, в зависимости от актуальных бизнес-целей, можно обучить модель на полных данных для лучшего отсечения нецелевых сессий, либо на сбалансированной выборке, если требуется отобрать целевые сессии, и ложно-положительные результаты не являются проблемой.

- В качестве финального классификатора выбран GradientBoostClassifier.
  Финальный классификатор обучался по умолчанию на полном датасете.
  Значение параметра random_state нигде не фиксировалось, поэтому значение метрики на тестовой выборке может колебаться.

Feature Importances

Взглянем на первые 20 признаков, отсортированные по убыванию
Среди них только один категориальный признак из OneHotEncoder, преимущественно же классификатор опирался на признаки, сгенерированные из маркетинговых данных с использованием частотности вхождений в датасет и score_samples.



Localhost Web App

Обернем модель в сервис на базе FastAPI и отправим запрос через Postman