C2. Предсказание отмены такси

Ozon Masters ML2
Студент: Арешин Станислав Олегович

Постановка задачи

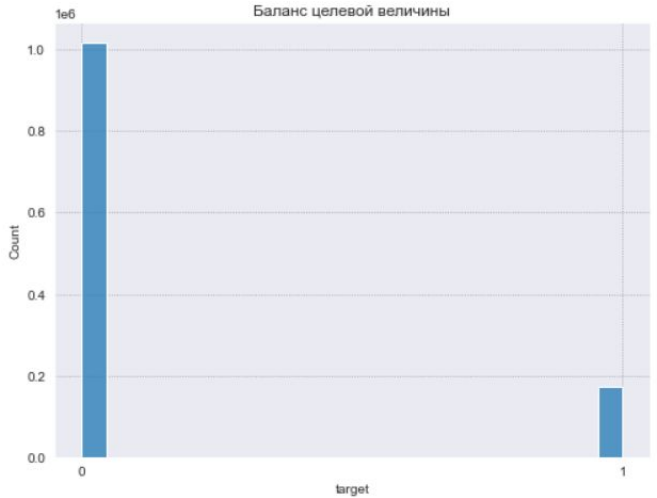
Цель соревнования - научиться предсказывать "сгоревшие" (отменённые до того, как на них откликнулся водитель) заказы сервиса такси.

Формат данных:

- due — на какое время была заказана машина (UTC).
- fclass, sclass, t_class — Дополнительные опции заказа. в приложении такси можно выбирать классы машин (например, "эконом или бизнес"). Это соответственно первый, второй и третий элементы списка.
- lon, lat — точка, куда делали заказ.

В обучении 1187461 данных, в тесте 510937.

Баланс таргета представлен на графике

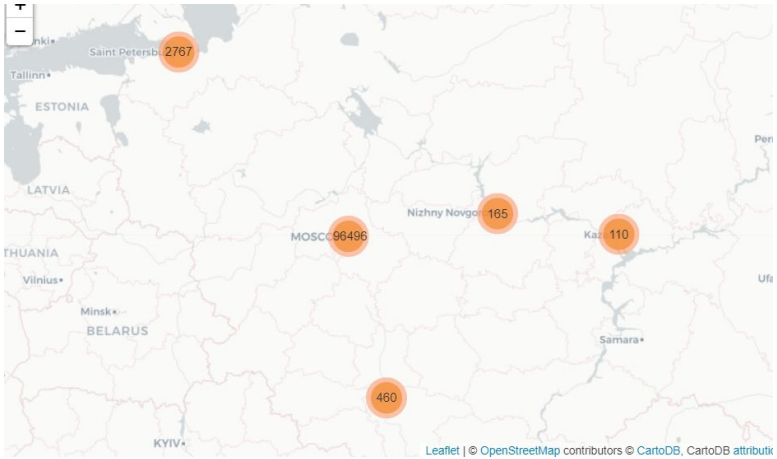


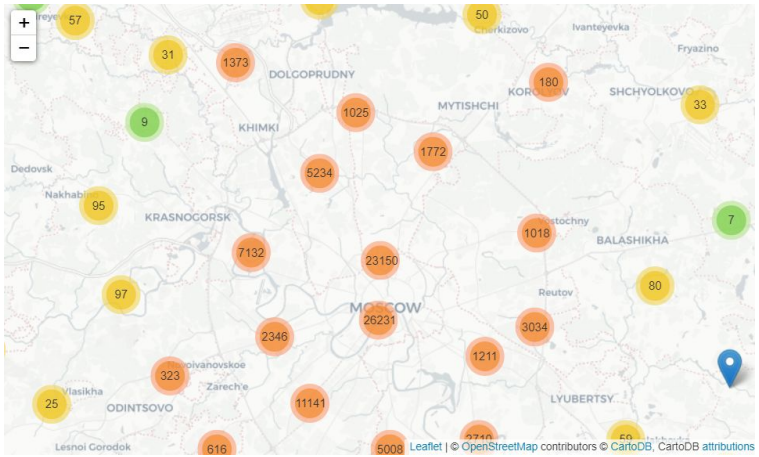
Подтягиваем карту

Классные карты можно построить с помощью библиотеки folium. Я по очереди визуализировал по 1/4 датасета. Ноутбук с парсингом на карту я выложил в открытый доступ на kaggle : https://www.kaggle.com/code/areshinstanislav/creating-map-and-getting-geodata

На картах видно, что данные сконцентрированы в основном в крупных городах России: Москва, Санкт-Петербург, Казань, Нижний Новгород и Воронеж. Встречаются данные, удаленные от этих городов, но их мало. Также есть значения широты и долготы 0,0.

В тесте были найдены перепутанные широта и долгота, это необходимо обработать.

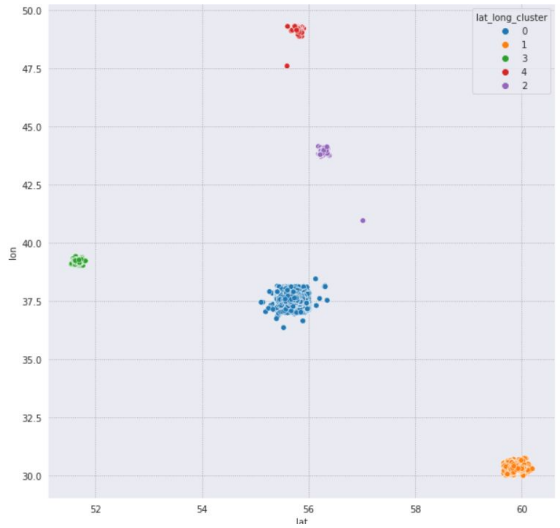




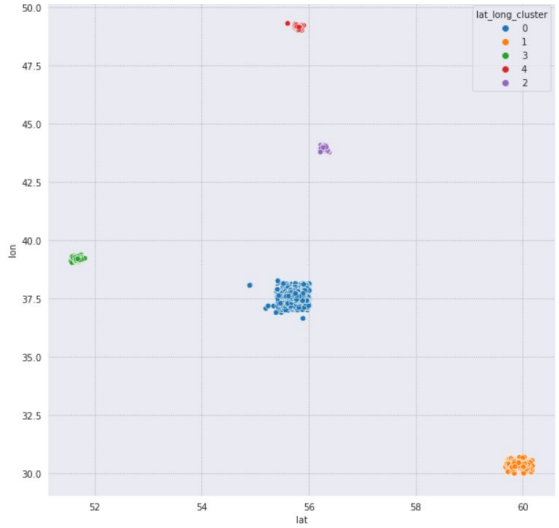
Развлекаемся с кластеризацией

Я попробовал кластеризовать данные алгоритмом Kmeans. В итоговом решении я данные результаты не использовал, так как в условиях задачи это тоже самое, что статичные рамки городов, но работа интересная, с ней можно ознакомиться по ссылке https://www.kaggle.com/code/areshinstanislav/lat-lon-clustering, либо в ноутбуке lat_lon_clustering.ipynb в репозитории.

Обучение



Тест



Что ещё есть в данных?

- Признак dist содержит много значений -1 - это пропуски, которые были заполнены до предоставления данных
- В признаке dist есть аномальные значения
- В категориальных признаках f_class, s_class, t_class есть пропуски, их заполним отдельной категорией
- В данных есть время заказа - это однозначно нужно использовать

```
Data columns (total 14 columns):
 #   Column              Non-Null Count    Dtype  
---  ------              --------------    -----  
 0   dist                1187461 non-null  float64
 1   due                 1187461 non-null  object 
 2   f_class             1175550 non-null  object 
 3   lat                 1187461 non-null  float64
 4   lon                 1187461 non-null  float64
 5   s_class             348920 non-null   object 
 6   t_class             18388 non-null    object 
```

Погодные данные. Join

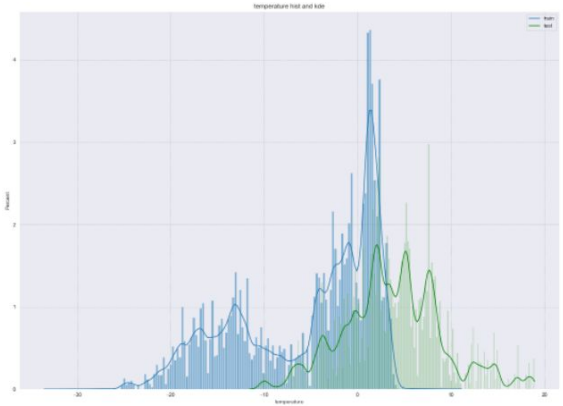
Дополнительно к соревнованию были предоставлены погодные данные по крупным городам России. Для джоина были взяты статичные рамки городов из открытого ноутбука, выделены временные признаки 'month', 'day', 'hour', а далее просто используется pd.merge.

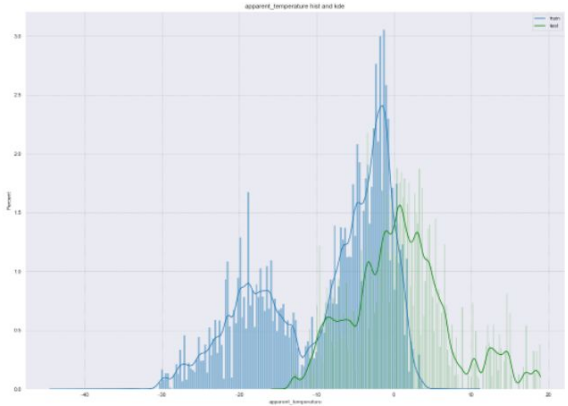
Данная реализация гораздо быстрее представленной в ноутбуке на kaggle, отрабатывает за секунду вместо указанных 40 минут. Ознакомиться можно в ноутбуке weather_data_eda.ipynb в репозитории.

Погодные данные. Анализ. Непрерывные признаки 1

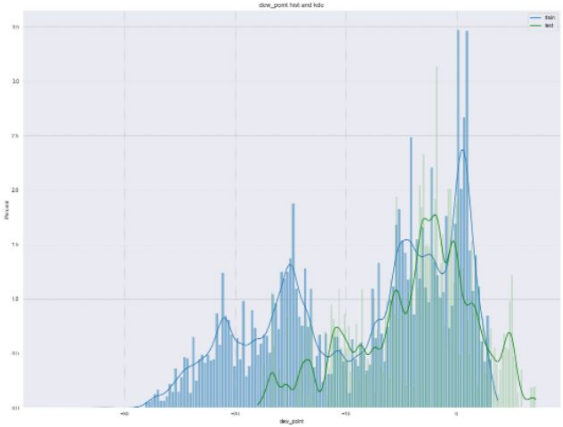
Синим цветом обозначена обучающая выборка, зеленым тестовая. Распределения в процентах.

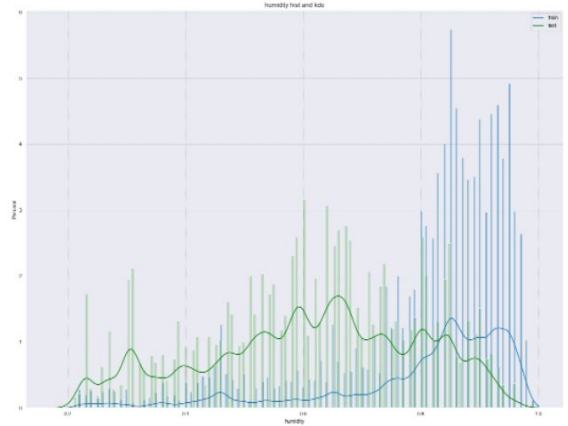
- 'temperature', 'apparent_temperature' сильно смещены на тесте, наступила весна.





- 'dew_point', 'humidity' тоже едут на тесте

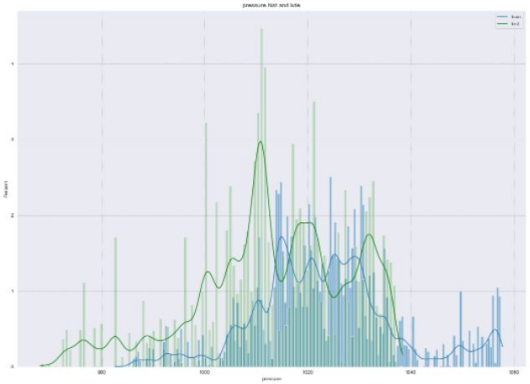


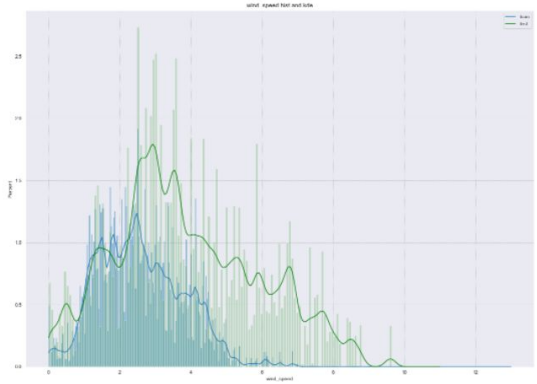


Погодные данные. Анализ. Непрерывные признаки 2

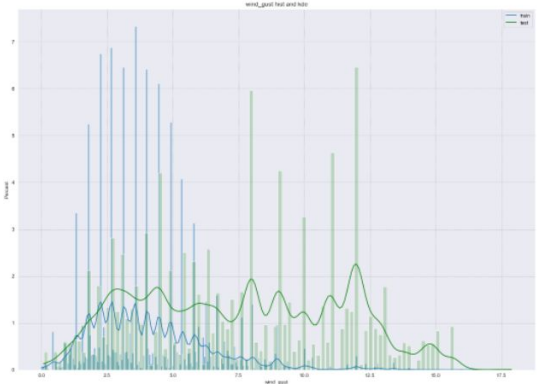
Синим цветом обозначена обучающая выборка, зеленым тестовая. Распределения в процентах.

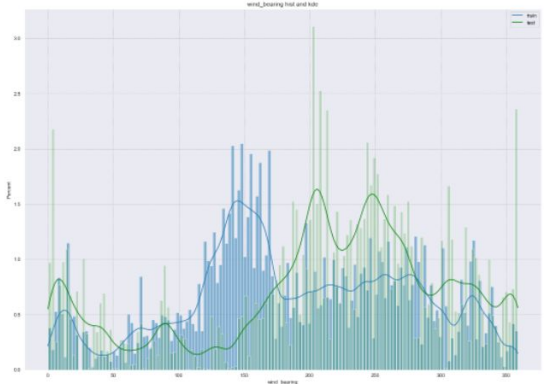
- 'pressure' смещено, а вот 'wind_speed' пусть с смещен, но не так сильно и можно попробовать добавить как фичу





- 'wind_gust', 'wind_bearing' сильно смещены





Погодные данные. Анализ. Категориальные признаки

Аналогично синим цветом обозначена обучающая выборка, зеленым тестовая. Распределения в процентах. Признаки 'summary', 'icon', 'precip_type' соответсвенно.

Все признаки разбалансированны, плюс плохое разделение по таргету (см. ноутбук weather_data_eda.ipynb)

Для модели использовать не будем.

Baseline 2. Решение

Расскажу про решение, побившее второй бейслайн. (Скор решения при использовании всех данных на паблике 0.74958 при бейслайне 0.69283). В целом это решение отличается от решения, которое побило первый бейслайн набором сгенерированных признаков.

В решении первого бейслайна использовались только временные признаки и ohe hot кодирование категорий с тем же catboost.

Посмотреть код с комментариями данной части повествования можно в ноутбуке baseline2_solution.ipynb.

Baseline 2. Генерация признаков

- **get_dists_from_cities** - вычисление евклидовых расстояний до центров городов и минимального расстояния до города: 'dist_to_MOSCOW', 'dist_to_SPB', 'dist_to_KAZAN', 'dist_to_N_NOVGOROD', 'dist_to_VORONEZH', 'min_dist_to_city'
- **get_datetime_features** - признаки, выделенные из даты и времени : 'month', 'day', 'weekday', 'is_weekend', 'hour', 'day_period', 'total_seconds', 'total_minutes'
- **get_geo_features** - выделение городов по статичным рамкам
- Категориальные признаки кодируются с помощью **OrdinalEncoder**
- Иии.. самое важное на следующем слайде.

Baseline 2. Спасательный признак. Описание

Самая важная функция данного решения. Благодаря этим признакам был побит второй бейслайн. Давайте подумаем, почему человек мог отменить заказ до того, как его взял таксист? Ниже несколько вариантов:

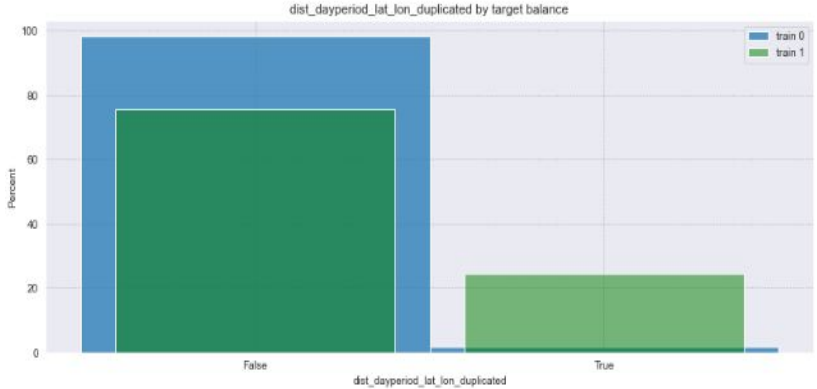
- Человек заказал случайно в другое место, потом отменил и заказал в нужное.
- Человек резко передумал ехать именно сейчас и поехал позже (например отговорили друзья или появились срочные дела). Тогда он закажет и поедет позже : через несколько минут, несколько часов, в другое время суток и тд.

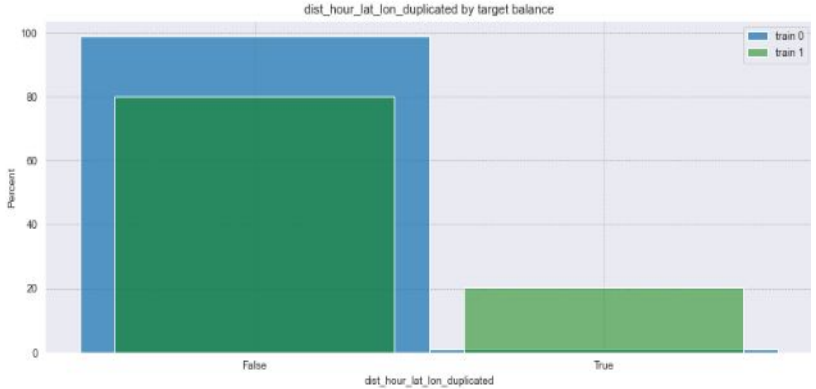
Напрашивается идея искать похожие по таким критериям заказы, приняв последний похожий заказ как не дубликат, а остальные помечать как дубликаты (для этого сортируем данные по due). Из этой идеи получается прекрасный признак. Можно придумать несколько комбинаций, по которым будем искать почти дубликаты, и нагенерировать дополнительные признаки, которые помогут итоговой модельке.

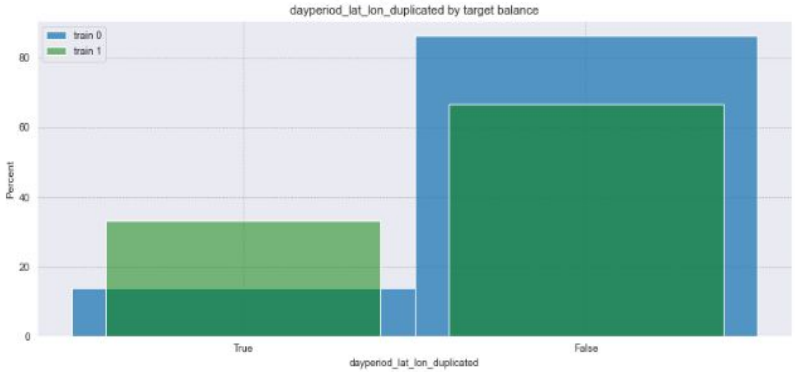
Для генерации признаков реализована функция.find_almost_dupclicates. Получены признаки: 'lat_lon_duplicated', 'dist_lat_lon_duplicated', 'dist_lat_lon_classes_duplicated', 'day_lat_lon_duplicated', 'dayperiod_lat_lon_duplicated', hour_lat_lon_duplicated', 'minutes_lat_lon_duplicated', 'dist_day_lat_lon_duplicated', 'dist_dayperiod_lat_lon_duplicated', 'dist_hour_lat_lon_duplicated'

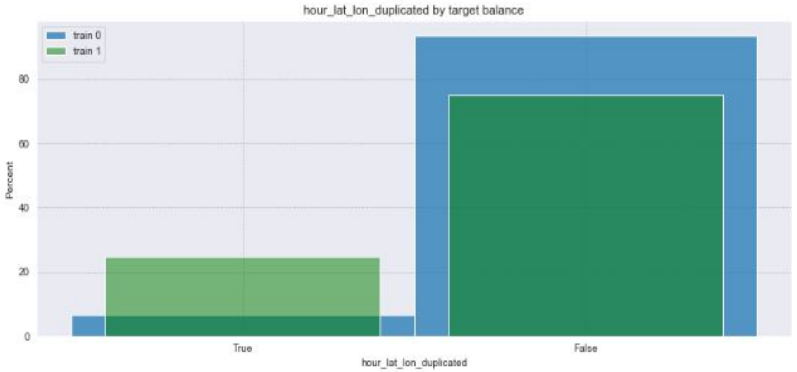
Baseline 2. Спасательный признак. Результат

На данном слайде представлены несколько графиков разделения по таргету для сгенерированных признаков. True - почти дубликат, False - уникальный, либо последнее значение в группе почти дубликатов. Данные в процентах.









Baseline 2. Модель

Для решения я выбрал CatBoostClassifier как при преодолении первого бейслайна, так и второго. Модель практически не тюнилась, несколько параметров вручную были подобраны по валидации. Но получилось все равно хорошо.

Baseline 2. Лучшее решение

Ну и пару слов об итоговом лучшем решении. Это блендинг. Данный подход прибавил чуть-чуть в точности и позволи закрепиться в топ 10. Если кратко, то:

Делим обучающую выборку на две части train1 и train2. Признаки те же, что и в предыдущем решении.

Модель следующая:

- Метапризнаки из двух отличающихся LGBMRegressor (обучаем на train1, получаем признаки для train2 и test)
- Метапризнаки из двух отличающихся CatBoostRegressor (аналогично обучаем на train1, получаем признаки для train2 и test)
- Метапризнаки из двух отличающихся RandomForestRegressor (аналогично обучаем на train1, получаем признаки для train2 и test)
- На метапризнаках train2 обучаем CatBoostClassifier
- Делаем предикт по метапризнакам test
- Итоговый скор на паблике 0.75779, место 8/67.

Итог

- Соревнование было сложным, было много попыток пробовать разные модели и ансамбли моделей в надежде, что в этом ключ, но оказалось, что ключ в генерации хороших признаков и это будет уроком на будущее)
- Я познакомился с новыми библиотеками, это хороший опыт
- Касательно моделей, можно было их потюнить и получить чуть выше скор, но этим не успел заняться, поэтому все так, как есть
- Нужно правильно выделять валидационную часть выборки. По началу валидация меня очень сильно обманывала, так как была рандомной. А когда отсортировал по времени, все стало честно)