Flight Direction prediction

https://github.com/TheodorrodeohT/flight-direction-prediction

APP IN THE AIR

Задачи

1. Предложить модель, предсказывающую для конкретного пользователя его **следующий полёт** (топ-5 вариантов)
2. По имеющимся данным **train** натренировать модель и сделать предсказание для **test**
3. Оценить качество предсказания

Classification

Одним из используемых подходов была классификация.

Изначально были удалены нерелевантные признаки, в итоге в датасете остались следующие признаки: **user_id**, **date**, **carrier**, **city_code_dep**, **city_code_arr**. Затем весь датасет был отсортирован по признаку **date**.

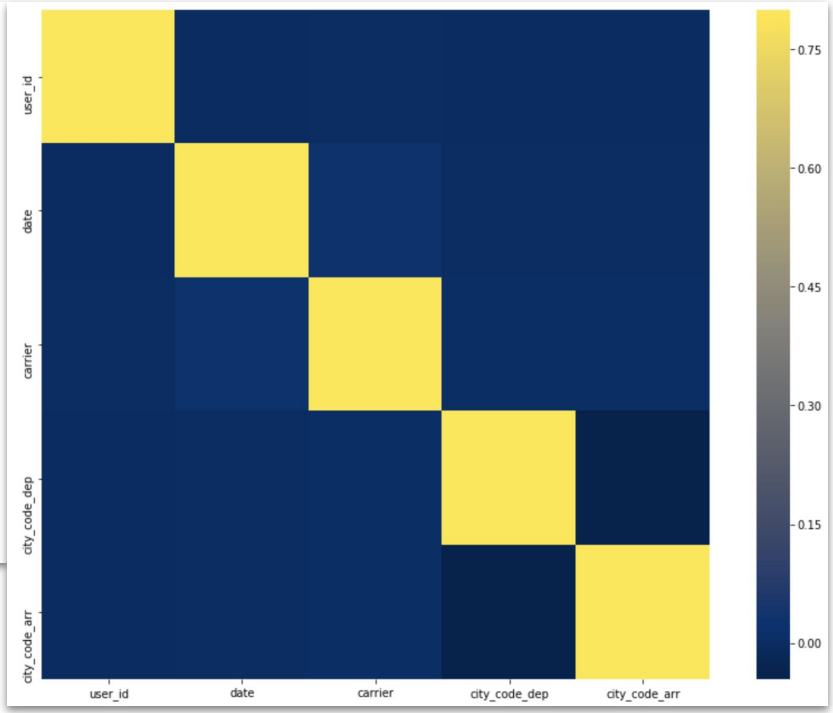
Далее шла обработка категориальных признаков. В признаке **date** были оставлены только месяца. Признаки **carrier**, **city_code_dep**, **date** и **city_code_arr** были закодированы в числовые переменные с помощью **LabelEncoder**.

Было принято решение не объединять дубликаты **user_id**. Также, из-за большого размера датасета, использовались данные только за **2015** год. Дополнительные признаки в конечном итоге тоже не добавлялись, все так же из-за размера датасета.

Касательно моделей были рассмотрены **Logistic Regression Classifier**, **CatBoost Classifier** и **XGB Classifier**. **Последняя модель** дала наилучший результат.

Features and Correlation Matrix

	user_id	date	carrier	city_code_dep	city_code_arr
391183	13643115200653456530	1	168	318	231
391115	3864152860678403525	1	168	318	231
390999	2289478520792205525	1	168	318	231
408837	14550263533831971003	1	41	192	48
12112	7514257410118127424	1	138	318	330
279673	12595761759477122490	1	16	232	546
351342	11628554629795960123	1	16	419	155
984576	12020148716044257436	1	138	332	439
72576	16597752047613988848	1	138	476	69
226202	6595510818733314592	1	16	332	538



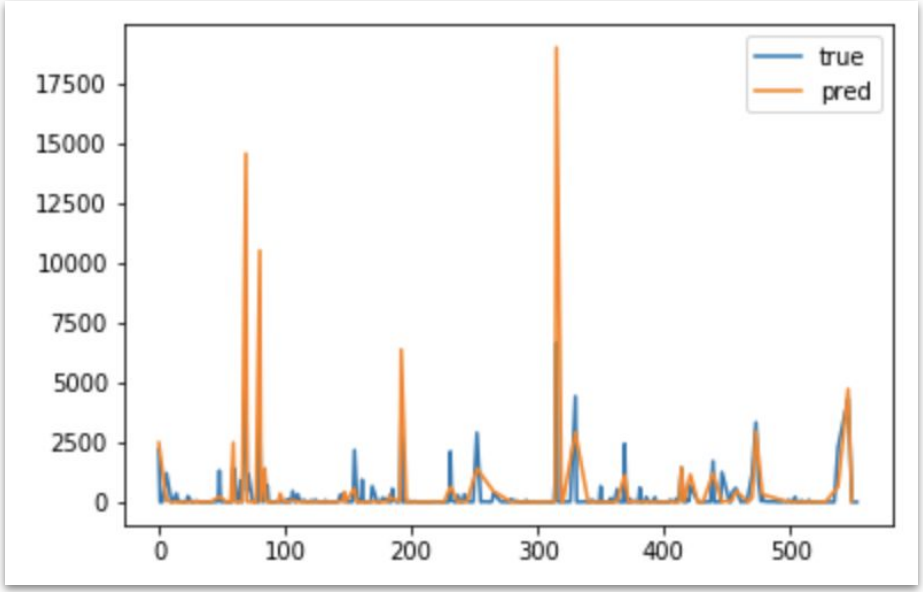
Результаты классификации

Как мы можем видеть, ярко выражены неточности в предсказаниях примерно в 4 местах, при этом в целом модель показывает схожие результаты. Данные ошибки могут быть связаны с неправильной выборкой, так как на тестовых данных были города, которых не было на обучающей. Была идея выкинуть данные города, но это не соответствовало бы реальной ситуации.

UPD: выкинув данные города ничего не изменилось

Результаты:

Accuracy_score: `0.22837230442906925`



Markov Model

Для начала необходимо сделать выбор между тем, какую из моделей взять за основу:

- AdaBoost Markov Model - одной из составляющих частей модели является выделение important locations, что входит в задачу другого проекта, поэтому эта модель на текущем этапе не подходит
- MMC ((n-/temporal) Mobility Markov Chains) - достаточно неочевидный процесс кластеризации трипов пользователя
- Hidden Markov Model - по сути схожая с n-MMC модель с несколькими вариантами декодирования цепи событий
- Mixed Markov Models - не удалось найти достаточно информации, которая бы помогла адаптировать модель к нашему случаю

Конечный выбор остановился на **Hidden Markov Model**.

Hidden Markov Model: decoding algorithm

Кандидатами для алгоритма декодирования hidden chain были следующие варианты:

- Viterbi
- Forward-Backward
- Baum-Welch - не под нашу задачу
- Expectation-Maximization - не под нашу задачу

Были испробованы Viterbi и Forward-Backward, но второй показал время работы практически в два раза хуже с приростом в точности в пределах погрешности. В итоговой реализации оставлен алгоритм **Viterbi**.

Hidden Markov Model: работа с данными

Для построения самой модели нам важны только признаки **city_code_dep**, **city_code_arr** и самодельный признак **trips** -- пара начального и конечного городов. Данные отсортированы по возрастанию даты полёта, так как иначе последовательность obervations рискует оказаться некорректной.

Предсказание для каждого пользователя строится независимо от остальных. В качестве hidden states используются все возможные города из датасета, в качестве observable states - города, в которые пользователь уже летал (**city_code_arr**).

Распределение вероятностей для городов строилось по количеству поездок в данный город. Те города, в которых пользователь не был, имели равную изначальную вероятность, зависящую от настраиваемого параметра **p_thresh**.

Для работы над вероятностями изначально проводились попытки кластеризовать пользователей по схожим паттернам trip-ов: изучение зависимости схожих путей, времени полётов и количества trip-ов, но это не удалось по нескольким причинам. Во-первых, для такого анализа было необходимо рассмотреть всех пользователей, а для этого пришлось бы перебирать все пары уникальных юзеров (C_250k^2). Конечно, можно кластеризовать для начала по количеству поездок или по частоте перелётов в определённый сезон, но тогда всё равно присутствовал риск не захватить большую часть данных для кластера. Во-вторых, это бы повлияло на выбор hidden states -- был бы смысл в качестве состояний принять кластеры, на которые были поделены все пользователи, и затем следующий город можно было бы определить по наиболее частому городу из выбранного кластера.

Hidden Markov Model: результаты предсказания топ-1 и топ-5

Для тестирования у каждого пользователя будем брать n-1 trip-ов в качестве **train** данных и **city_code_arr** последнего **trip**-а в качестве теста. Таким образом мы будем выстраивать цепь последовательно расположенных полётов и в конце делать предсказание на следующий маршрут. Так как предсказание проводится независимо от остальных пользователей, то у клиента с len(trips) == 1 будет равновероятное попадание во все города кроме того, в котором он уже побывал, поэтому в данном случае топ-5 будет более актуален. Измерять точность предсказание будем с помощью accuracy.

В случае топ-5 мы смотрим, попал ли тестовый город в наше топ-5 предсказание. Очевидно, получаем прирост в вероятности.

Таким образом, наше предсказание работает для чуть более чем трети пользователей. В jupyter notebook с основным кодом можно ознакомиться с данным набором юзеров.

	top1 accuracy	top5 accuracy
100	45.00	51.00
1000	34.30	40.40
2500	35.80	41.60
5000	33.64	39.88