Отчет
по индивидуальному заданию
по курсу Нейронные сети
«New York City Taxi Trip Duration»

выполнил

студент 2 курса магистратуры

Буланов Дмитрий

Оглавление

Постановка задачи

New York City Taxi Trip Duration (Продолжительность поездки такси в Нью-Йорке), в этом соревновании Kaggle бросает вызов нам построить модель, которая предсказывает общую продолжительность поездки такси в Нью-Йорке. Основной набор данных - один, выпущенный двумя компаниями Нью-Йорк такси и Комиссия лимузина, который включает время начала поездки, геокоординирование, количество пассажиров и несколько других переменных.

Используемые методы и инструменты

При решении использовались следующие библиотеки: Scikit-Learn для нескольких операций машинного обучения, Pandas - используется для обработки данных, Numpy - для вычислений в Python, XGBoost - алгоритм классификации, используемый для окончательных прогнозов.

Анализ данных

Данные для задачи представлены в виде текстового файла в формате csv.
train.csv – данные для обучения (1458644 записей)
test.csv – данные для тестирования (625134 записей)

Поля данных:

- id – уникальный идентификатор поездки
- vendor_id – код, указывающий поставщика услуг
- pickup_datetime – дата и время, когда был включен счетчик
- dropoff_datetime - дата и время, когда был выключен счетчик
- passenger_count – количество пассажиров
- pickup_longitude – долгота, когда был включен счетчик
- pickup_latitude - широта, когда был включен счетчик
- dropoff_longitude - долгота, когда был выключен счетчик
- dropoff_latitude - широта, когда был выключен счетчик
- store_and_fwd_flag – этот флаг указывает, хранилась ли запись поездки в памяти транспортного средства, Y= хранилось, N= не сохранялось
- trip_duration – продолжительность поездки в секундах

Рассмотрим более подробно данные в виде таблицы:

	id	vendor_id	pickup_datetime	dropoff_datetime	passenger_count	pickup_longitude	pickup_latitude	dropoff_longitude
0	id2875421	2	2016-03-14 17:24:55	2016-03-14 17:32:30	1	-73.982	40.768	-73.965
1	id2377394	1	2016-06-12 00:43:35	2016-06-12 00:54:38	1	-73.980	40.739	-73.999
2	id3858529	2	2016-01-19 11:35:24	2016-01-19 12:10:48	1	-73.979	40.764	-74.005
3	id3504673	2	2016-04-06 19:32:31	2016-04-06 19:39:40	1	-74.010	40.720	-74.012
4	id2181028	2	2016-03-26 13:30:55	2016-03-26 13:38:10	1	-73.973	40.793	-73.973

А также в виде статистических данных, где можно заметить максимальные, минимальные и средние значения по каждой из переменной:

	vendor_id	passenger_count	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	trip_duration
count	1458644.000	1458644.000	1458644.000	1458644.000	1458644.000	1458644.000	1458644.000
mean	1.535	1.665	-73.973	40.751	-73.973	40.752	959.492
std	0.499	1.314	0.071	0.033	0.071	0.036	5237.432
min	1.000	0.000	-121.933	34.360	-121.933	32.181	1.000
25%	1.000	1.000	-73.992	40.737	-73.991	40.736	397.000
50%	2.000	1.000	-73.982	40.754	-73.980	40.755	662.000
75%	2.000	2.000	-73.967	40.768	-73.963	40.770	1075.000
max	2.000	9.000	-61.336	51.881	-61.336	43.921	3526282.000

Формат выходного файла:

Для каждой строки в наборе данных, выходной файл должен содержать два столбца: id и trip_duration. Идентификатор соответствует столбцу этого идентификатора в test.csv. Файл должен содержать заголовок и иметь следующий формат:

```
id,trip_duration
id00001,978
id00002,978
id00003,978
id00004,978
...
```

Метрика оценки для этого соревнования - Root Mean Squared Logarithmic Error (RMSLE).

$$\epsilon = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(p_i + 1) - \log(a_i + 1))^2}$$

Где,
ε - значение RMSLE (оценка)
nn - общее количество данных
pi - наш прогноз продолжительности поездки
ai - фактическая продолжительность поездки
log (x) - натуральный логарифм от x

Предобработка данных и выделение признаков

Первым делом, преобразуем переменную дата в другой формат и выделим отдельно год, месяц, день недели, часы и минуты для каждой поездки.

```
data['datetime_obj'] = pd.to_datetime(data['pickup_datetime'])
data['pickup_year'] = data['datetime_obj'].dt.year
data['pickup_month'] = data['datetime_obj'].dt.month
data['pickup_weekday'] = data['datetime_obj'].dt.weekday
data['pickup_day'] = data['datetime_obj'].dt.day
data['pickup_hour'] = data['datetime_obj'].dt.hour
data['pickup_minute'] = data['datetime_obj'].dt.minute
```

В результате получим следующие поля:

datetime_obj	pickup_year	pickup_month	pickup_weekday	pickup_day	pickup_hour	pickup_minute
2016-03-14 17:24:55	2016	3	0	14	17	24
2016-06-12 00:43:35	2016	6	6	12	0	43
2016-01-19 11:35:24	2016	1	1	19	11	35
2016-04-06 19:32:31	2016	4	2	6	19	32
2016-03-26 13:30:55	2016	3	5	26	13	30

Затем, поработаем с флагом записи поездки. Преобразуем значения Y, N к числовому типу:

```
col = 'store_and_fwd_flag'
data_dict = {'Y':1, 'N':0}
data_tf = data[col].map(data_dict)
data[col].update(data_tf)
```

После, удалим неиспользуемые столбцы, такие как pickup_datetime и datetime_obj:

```
data.drop('pickup_datetime', axis=1, inplace=True)
data.drop('datetime_obj', axis=1, inplace=True)
```

Следующей командой заметим, что все данные приведены за 2016 год:

```
combine_data_tf['pickup_year'].value_counts()
```

```
2016    2083778
Name: pickup_year, dtype: int64
```

В результате можно избавиться от столбца pickup_year:

```
combine_data_tf.drop('pickup_year', axis=1, inplace=True)
```

Следующим шагом, было преобразование координат начала поездки и конца (широта и долгота) в расстояние, для этого используется функция haversine, вычисляющая расстояние между двумя точками на сфере, учитывая их долготу и широту. *(источник: https://stackoverflow.com/questions/15736995/how-can-i-quickly-estimate-the-distance-between-two-latitude-longitude-points)*

$$a = \sin^2(\Delta\varphi/2) + \cos\varphi_1 \cdot \cos\varphi_2 \cdot \sin^2(\Delta\lambda/2)$$

$$c = 2 \cdot \operatorname{atan2}(\sqrt{a}, \sqrt{(1-a)})$$

$$d = R \cdot c$$

Где, φ широта, λ долгота, R радиус земли (средний радиус = 6,371км); углы в радианах.

```
def haversine(lon1, lat1, lon2, lat2):
    # градусы в радианы
    lon1, lat1, lon2, lat2 = map(radians, [lon1, lat1, lon2, lat2])
    # формула haversine
    dlon = lon2 - lon1
    dlat = lat2 - lat1
    a = sin(dlat/2)**2 + cos(lat1) * cos(lat2) * sin(dlon/2)**2
    c = 2 * asin(sqrt(a))
    km = 6367 * c
    return km
```

Применим нашу функцию для заданных координат и получим новый столбец расстояние (distance) :

```
data['distance'] = combine_data_tf.apply(lambda row: haversine(row['pickup_latitude'], row['pickup_longitude'], row['dropoff_latitude'], row['dropoff_longitude']), axis=1)
```

distance
1.948753
2.130839
3.356930
0.475220
0.328255

На следующем шаге было решено применить к продолжительности поездки логарифмирование:

```
target_log = np.log(train_set[label].values)
```

Затем разобьем выборку на 2 подвыборки:

```
X_train, X_test, Y_train, Y_test = train_test_split(
    data, target_log, train_size=0.85, random_state=1234)
```

Создание и обучение модели

Для модели был выбран класс XGBRegressor с указанными параметрами:

```
model = XGBRegressor(n_estimators=500, max_depth=5,
                     learning_rate=0.1, min_child_weight=1, n_jobs=-1)
```

Функция fit для тренировки с параметрами:

```
early_stopping_rounds = 50
model.fit(
    X_train, Y_train, eval_set = [(X_test, Y_test)],
    eval_metric="rmse", early_stopping_rounds=early_stopping_rounds,
    verbose=early_stopping_rounds
)
```

Где, ранняя остановка равна 50.

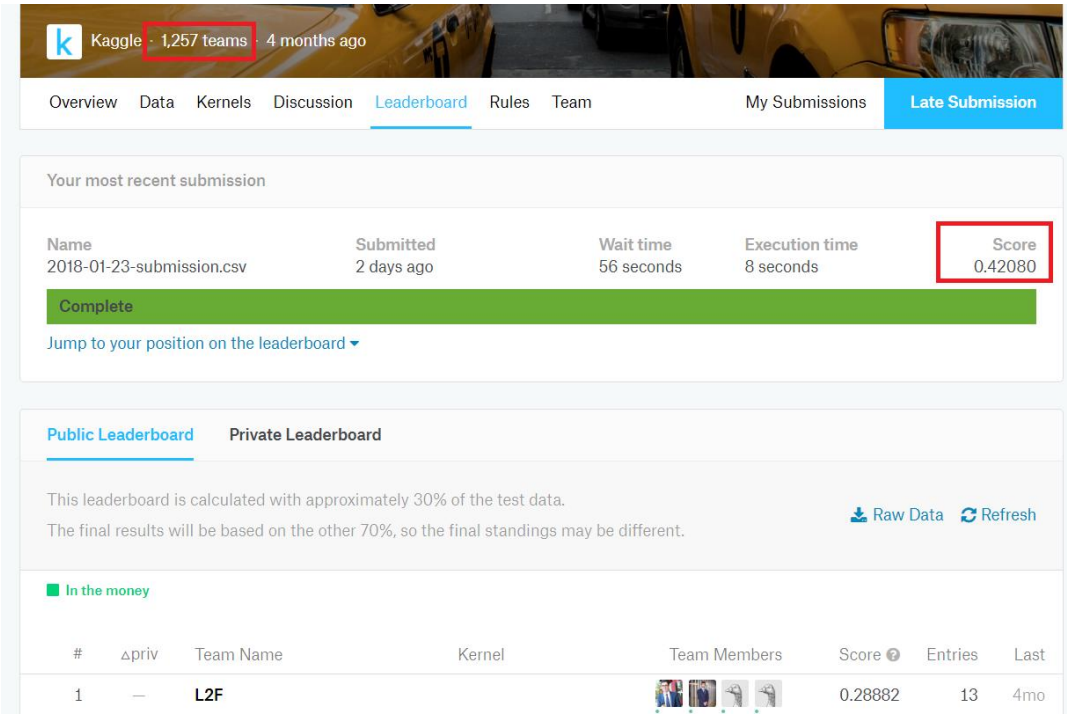
После вычислений над тестовой выборкой, на выходе получим следующие данные:

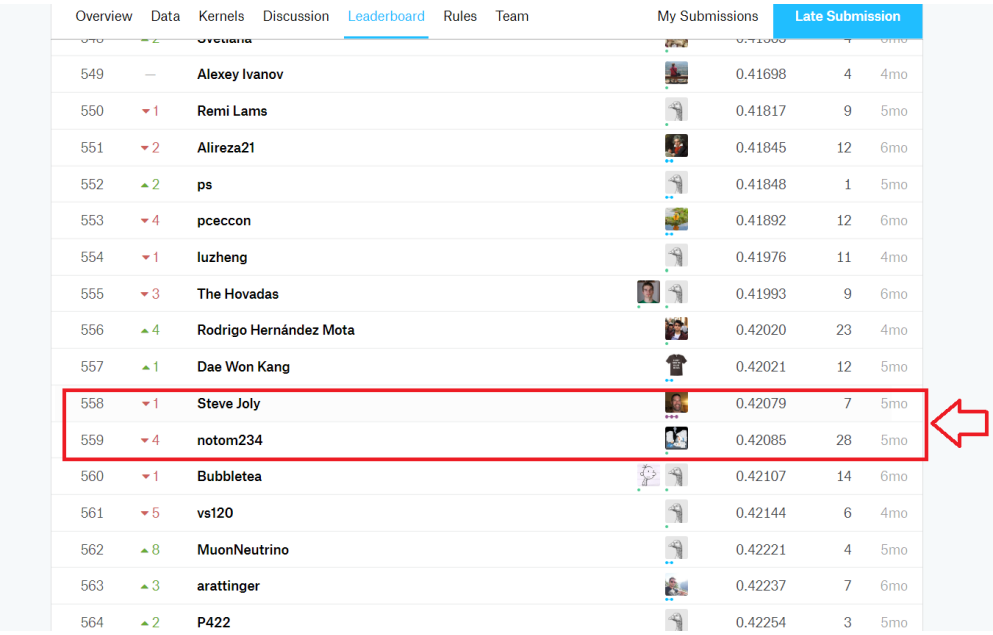
	id	trip_duration
0	id3004672	64.597374
1	id3505355	57.711868
2	id1217141	74.613304
3	id2150126	113.438652
4	id1598245	57.516701

u.m.д

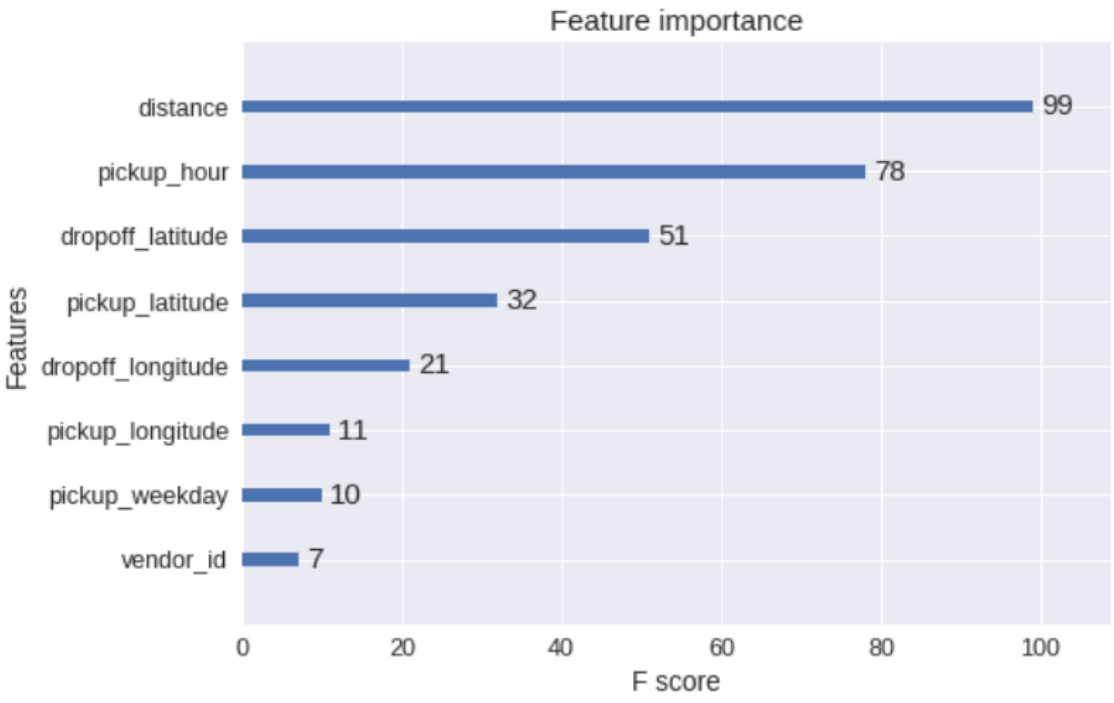
Заключение

Удалось получить модель, для которой значение RMSLE после обучения составляет 0.42080. Из 1257 участников, этот результат находится на 558 месте. Таким образом, мы попали в топ 45% лидеров.





Выделим наиболее важные признаки для реализованной модели:



Естественным образом, наиболее важным является признак расстояние.

Ссылка на конкурс: https://www.kaggle.com/c/nyc-taxi-trip-duration
Ссылка на исходный код: https://github.com/dmitriybulanov/KaggleProject