Отчет

по индивидуальному заданию

по курсу Нейронные сети

# «New York City Taxi Trip Duration»

выполнил

студент 2 курса магистратуры

Буланов Дмитрий

Оглавление

[Постановка задачи 3](#_Toc504550726)

[Используемые методы и инструменты 3](#_Toc504550727)

[Анализ данных 3](#_Toc504550730)

[Предобработка данных и выделение признаков 5](#_Toc504550731)

[Создание и обучение модели 7](#_Toc504550738)

[Заключение 8](#_Toc504550739)

# **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**

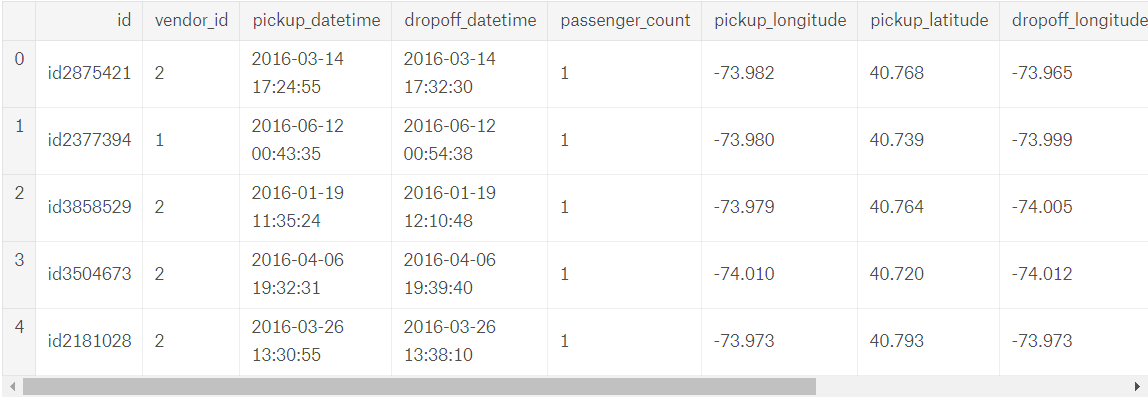
Данные для задачи представлены в виде текстового файла в формате csv.  
train.csv – данные для обучения (1458644 записей)

test.csv – данные для тестирования (625134 записей)

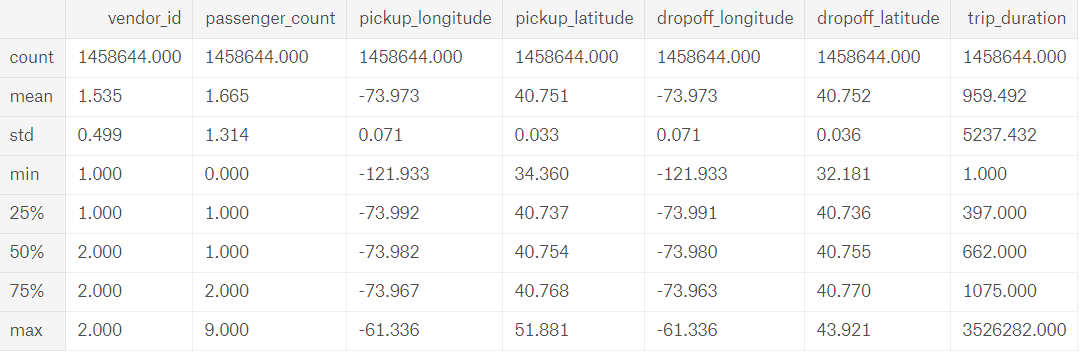
## Поля данных:

* 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:



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

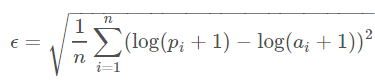


Формат выходного файла:

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).



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

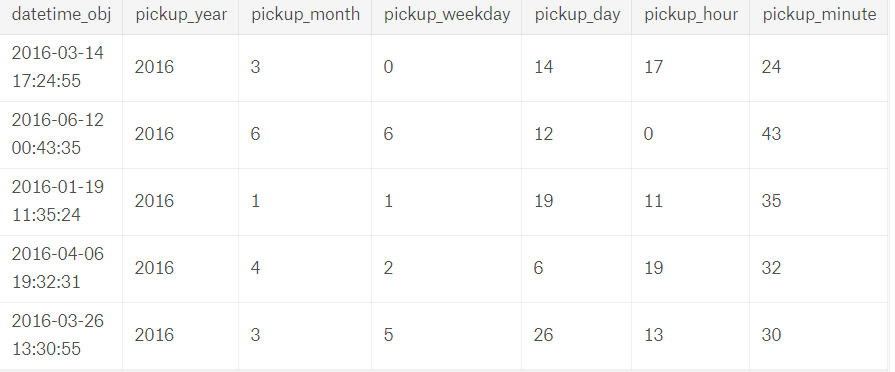
data['pickup\_weekday'] = data['datetime\_obj'].dt.weekday

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:



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)

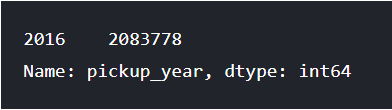
После, удалим неиспользуемые столбцы, такие как pickup\_datetime и datetime\_obj:

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()



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 = sin²(Δφ/2) + cos φ1 ⋅ cos φ2 ⋅ sin²(Δλ/2) |
| c = 2 ⋅ atan2( √a, √(1−a) ) |
| d = R ⋅ 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

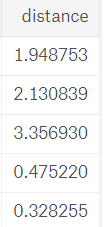
c = 2 \* asin(sqrt(a))

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)



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)],

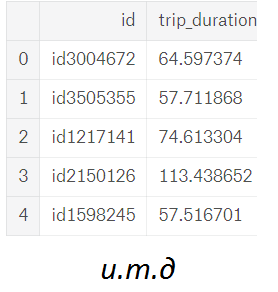
eval\_metric="rmse", early\_stopping\_rounds=early\_stopping\_rounds,

verbose=early\_stopping\_rounds

)

Где, ранняя остановка равна 50.

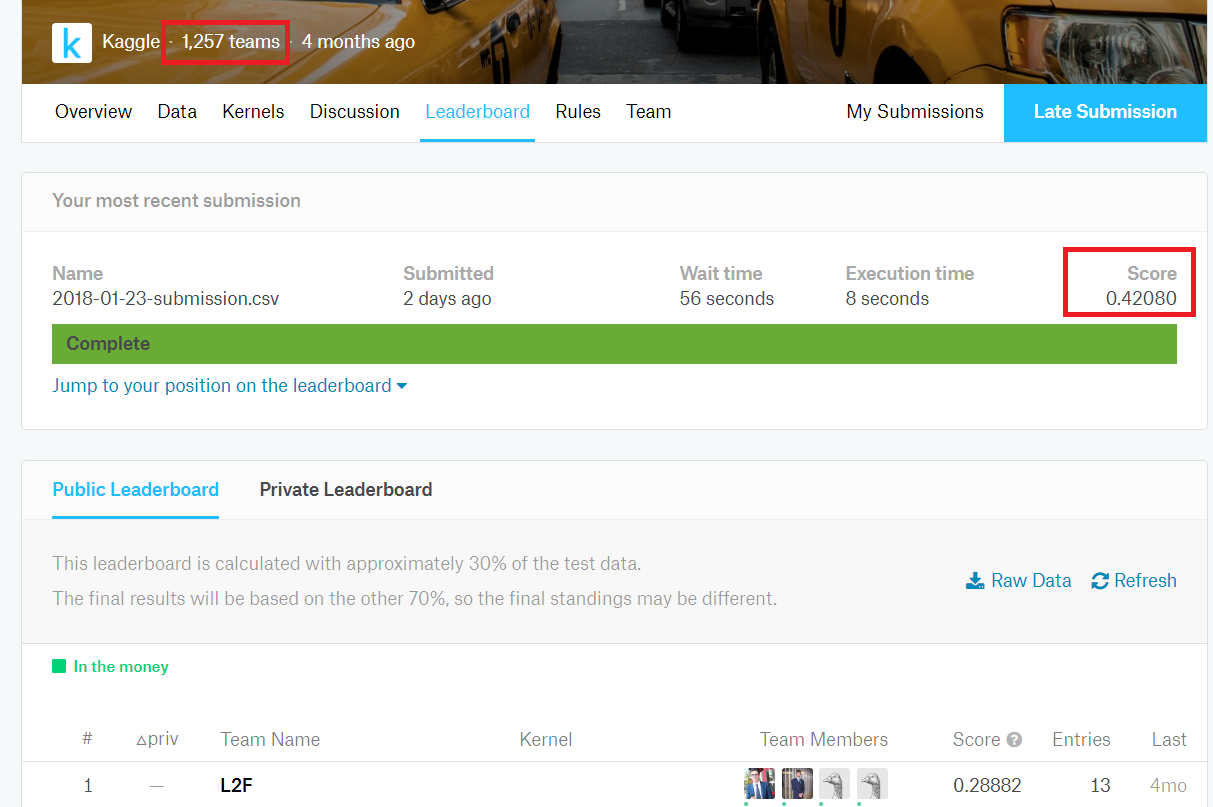
После вычислений над тестовой выборкой, на выходе получим следующие данные:

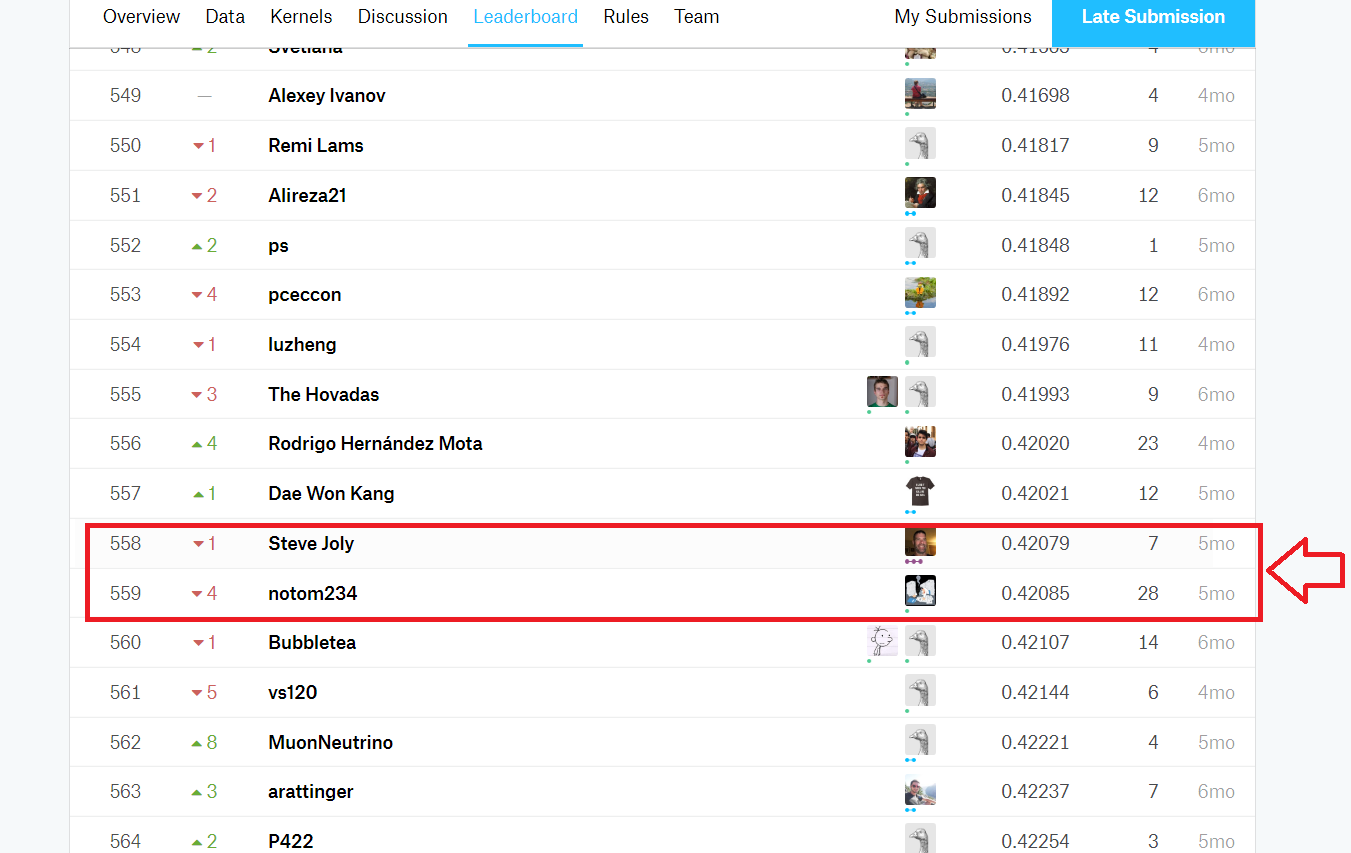


# 

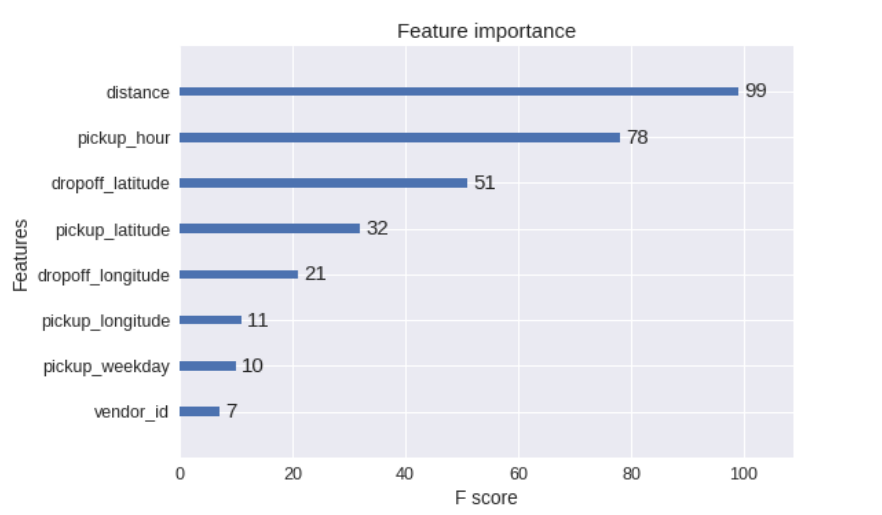
# **Заключение**

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