NYC TAXI FARE AMOUNTS

Владислав Бояр
Александра Ивойлова
Мария Мичурина

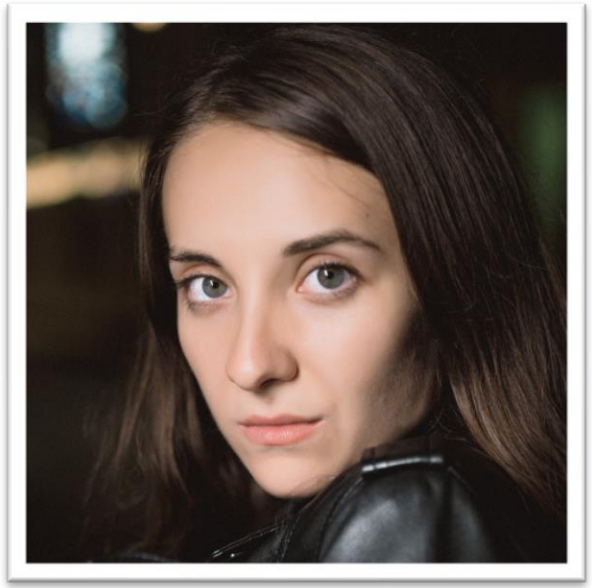
Наша команда

Влад

Обрабатывает фичи

Саня

Бряцает железом



Маша

Ловит баги, заливает результаты на Kaggle

Обработка характеристик

Проанализируем датасет. Видим, что, во-первых, бывают отрицательные цены за поездки – это явно какой-то баг. Во-вторых, количество пассажиров бывает больше 200, анриал ☺

	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
count	5.542386e+07	5.542386e+07	5.542386e+07	5.542348e+07	5.542348e+07	5.542386e+07
mean	1.134505e+01	-7.250968e+01	3.991979e+01	-7.251121e+01	3.992068e+01	1.685380e+00
std	2.071083e+01	1.284888e+01	9.642353e+00	1.278220e+01	9.633346e+00	1.327664e+00
min	-3.000000e+02	-3.442060e+03	-3.492264e+03	-3.442025e+03	-3.547887e+03	0.000000e+00
25%	6.000000e+00	-7.399207e+01	4.073493e+01	-7.399140e+01	4.073403e+01	1.000000e+00
50%	8.500000e+00	-7.398180e+01	4.075265e+01	-7.398015e+01	4.075316e+01	1.000000e+00
75%	1.250000e+01	-7.396708e+01	4.076713e+01	-7.396367e+01	4.076810e+01	2.000000e+00
max	9.396336e+04	3.457626e+03	3.408790e+03	3.457622e+03	3.537133e+03	2.080000e+02

Обработка характеристик

Сами по себе координаты нам не много что дают. От чего зависит стоимость поездки? От расстояния – и от того, куда (откуда) едет человек. В аэропорт и из него ехать дороже. Координаты аэропортов нам известны...

Обработка характеристик

Также можем посчитать, на какое расстояние сместился автомобиль (разница по модулю широты и долготы), и распарсить дату и время.

```
# 1. Разница по модулю широты и долготы
df['abs_diff_longitude'] = (df.dropoff_longitude - df.pickup_longitude).abs()
df['abs_diff_latitude'] = (df.dropoff_latitude - df.pickup_latitude).abs()

# 2. Признаки даты и вермени: год, месяц, день, день недели. Данные берём из столбца key.
df['key_added'] = pd.to_datetime(df.key, format="%Y-%m-%d %H:%M:%S")
df.key.value_counts()
df['year'] = df.key_added.dt.year
df['month'] = df.key_added.dt.month
df['day'] = df.key_added.dt.day
df['day_of_the_week'] = df.key_added.dt.dayofweek
df["hour"] = df.key_added.dt.hour
```

Обработка характеристик

Одна из самых важных характеристик – это итоговое расстояние: чем дальше проехали, тем выше оплата. Можно тоже посчитать по координатам.

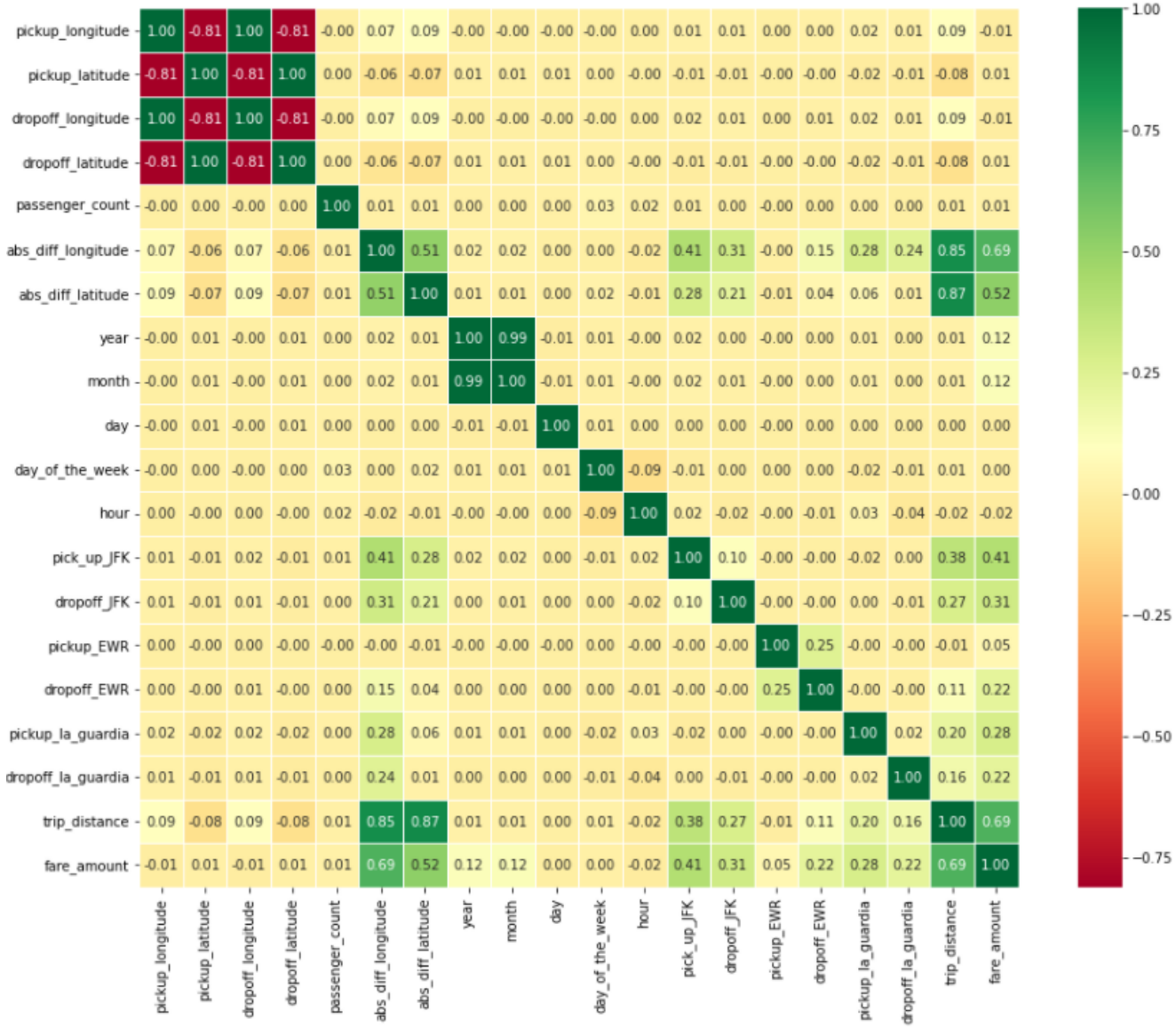
```python
# 4. Дистанция поездки
def trip_distance(lat1, lat2, lon1, lon2):
    p = 0.017453292519943295  # Pi/180
    a = 0.5 - np.cos((lat2 - lat1) * p)/2 + np.cos(lat1 * p) * \
        np.cos(lat2 * p) * (1 - np.cos((lon2 - lon1) * p)) / 2
    return 0.6213712 * 12742 * np.arcsin(np.sqrt(a))

df['trip_distance'] = df.apply(lambda row: trip_distance(
        row['pickup_latitude'], row['dropoff_latitude'], row['pickup_longitude'], row['dropoff_longitude']), axis=1)
return df
```

Обработка характеристик

Не забудем удалить все лишнее: избыточные столбцы, нулевые значения, отрицательную стоимость поездок

```python
# Удаляем избыточный столбец pickup_datetime, поскольку добавили признаки в функции add_new_features
if 'pickup_datetime' in df:
    df.drop("pickup_datetime", axis = 1, inplace=True)

# удаляем строки с нулевыми значениями
df = df.dropna(how = 'any', axis = 'rows')

# удаляем строки со стоимостью поездки меньше 0
if 'fare_amount' in df:
    df = df[df['fare_amount'] > 0]

# удаляем строки, где кол-во пассажиров меньше 1 или больше 9
df = df[(df['passenger_count'] > 0) & (df['passenger_count'] <= 9)]

# Учитывая, что поездки совершаются в рамках одного города, разница широты и долготы не должна превышать 1
# (1 градус долготы это приблизительно 69 миль). Следовательно, стоит исключить из датасета строки с слишком
# большим показателем разницы широты или долготы.
df = df[(df.abs_diff_longitude < 5.0) & (df.abs_diff_latitude < 5.0)]
```

Обработка характеристик



Получаем какую-то такую картинку. Как кажется, совершенно на итоговый результат не влияют чистые ширина и долгота и день поездки, но мы пробовали их дропать – качество предсказания тоже резко дропается ☹

Модели

- Linear Regression;
- BaggingRegressor;
- RandomForestRegressor;
- XGBoost;
- CatBoost;
- LGBM

GridSearchCV применялся только к стандартным склерновским моделям и с опаской (на сэмпле в 10 тыс. строк, потому что сперва запустили на всю ночь... а результатов так и не дождались)

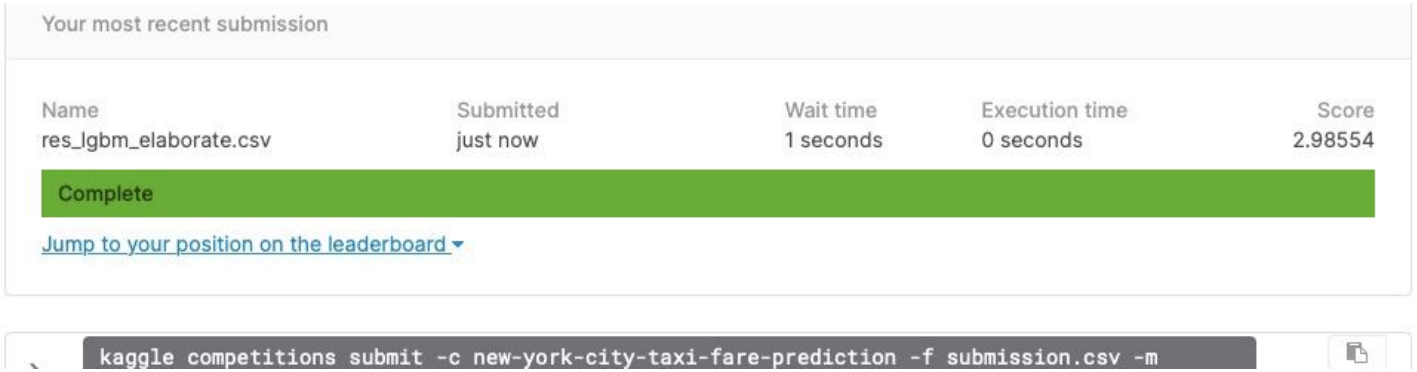
Модели

Модель	RMSE Train (best run)	RMSE Test (best run)
Linear Regression (baseline)	6.06	NA
Bagging Regressor	1.76	3.18
Random Forest Regressor	1.67	3.197
XGBoost	3.57	3.04
CatBoost	4.09	3.20
LGBM	**2.9**	**2.98**

Победитель - LGBM

```
hyper_params = {
        'boosting_type':'gbdt',
        'objective': 'regression',
        'num_leaves': 31,
        'learning_rate': 0.05,
        'max_depth': -1,
        'bagging_fraction' : 1,
        'max_bin' : 5000 ,
        'bagging_freq': 20,
        'colsample_bytree': 0.6,
        'metric': 'rmse',
        'min_split_gain': 0.5,
        'min_child_weight': 1,
        'min_child_samples': 10,
        'scale_pos_weight':1,
        'zero_as_missing': True,
        'seed':0,
        'num_iterations': 50000
    }
```

Гонялся на 5, 6 миллионах строк



Подведение итогов

- Линейная регрессия (скорее всего), бэггинг и случайный лес переобучались;
- Успех во многом зависит от грамотной работы с фичами;
- Успех вообще зависит от случайности: если случайно подобранный датасет удачен, то оценка выше;
- А это уже заставляет задуматься о том, насколько хороша исключительно автоматическая оценка результата;
- Стандартные склерновские модели очень жрут оперативу (не удалось обучить больше чем на 3 млн строк), а XGBoost, LGBM и CatBoost распараллелены и забивают процессор, за счет чего быстры (относительно).