

Министерство науки и высшего образования Российской Федерации
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТОМСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (НИ ТГУ)
Институт прикладной математики и компьютерных наук

Отчет по проекту по предмету
«Введение в интеллектуальный анализ данных»
**«Обучение модели для предсказания продолжительности поездки на
такси по Нью-Йорку»**

Выполнили:

студент группы № 932210 Е.Г. Чегодаева

студент группы № 932210 Д.А. Королев

Проверил:

д-р техн. наук Замятин А.В.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	Ошибка! Закладка не определена.
1 Описание проекта	Ошибка! Закладка не определена.
2 Цель, задачи и инструменты.....	Ошибка! Закладка не определена.
3 Шаги для достижения результатов	5
3.1 Разбор и загрузка данных.....	5
3.2 Анализ данных	6
3.3 Обработка выбросов	10
3.4 Моделирование	12
4 Результаты.....	15
ЗАКЛЮЧЕНИЕ.....	15
СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ.	Ошибка! Закладка не определена.

ВВЕДЕНИЕ

Машинное обучение концентрируется на разработке таких компьютерных программ и алгоритмов, которые сами учатся расти и адаптироваться при подаче новых данных. Этот процесс не похож на процесс интеллектуального анализа данных. Обе системы проходят через предоставленные им данные или собираются в поисках шаблонов. Однако в приложениях для интеллектуального анализа данных, данные извлекаются для понимания человеком, в то время как алгоритмы машинного обучения используют эти данные для поиска шаблонов в данных и соответственно изменения действий программы.

Машинное обучение возникло из-за стремления к искусственному интеллекту. В первые дни искусственного интеллекта уже как интеллектуального поля исследователи были очень заинтересованы в том, чтобы машины учились на данных. Поэтому они пытались подойти к проблеме с помощью различных символических методов, а также из методов, которые в то время назывались нейронными сетями, обычно это были только модели, которые впоследствии были обнаружены для переупаковки общих линейных моделей вероятности и статистики.

Машинное обучение стало отдельным полем и начало расширяться в 1990-х годах. Линия изменила свою цель - достичь ИИ, пытаясь решить разрешимые проблемы более практического характера. Затем поле отодвинуло его внимание от символических методологий, которые оно унаследовало от искусственного интеллекта, и вместо этого перешло к методам и моделям, взятым из вероятности и статистики.

1 Описание проекта

Для повышения эффективности электронных систем диспетчеризации такси важно иметь возможность прогнозировать, как долго водитель будет занят в своем такси. Если бы диспетчер приблизительно знал, когда водитель такси завершит свою текущую поездку, он мог бы лучше определить, какого водителя назначить для каждого запроса на доставку.

2 Цель, задачи и инструменты

Цель этой работы — предсказать продолжительность поездки на такси по Нью-Йорку на основе соответствующих данных.

Задачи:

1. Представить историю поездок, а также данные, содержащие погодные условия.
2. Провести анализ, удалить выбросы.
3. Провести обучение модели LightGBM.
4. Сделать выводы.

Инструменты для выполнения работы:

1. Язык программирования: python.
2. Загрузка данных: pandas.
3. Визуализация: seaborn.
4. Модель: lightgbm.
5. Подбор оптимальных параметров: optuna.

3 Шаги для достижения результатов

3.1 Разбор и загрузка данных

Для начала подготовим данные. Данные включает информацию о времени начала и окончания поездки, географических координатах, количествах пассажиров и др.

```
# Добавим тренировочные данные
train = pd.read_csv('/kaggle/input/nyc-taxi-trip-duration/train.zip', compression='zip')
# Добавим тестовые данные
test = pd.read_csv('/kaggle/input/nyc-taxi-trip-duration/test.zip', compression='zip')
# Добавим данные о погоде в 2016 году
weather = pd.read_csv("/kaggle/input/weather-data-in-new-york-city-2016/weather_data_nyc_centralpark_2016(1).csv")
# Добавим данные о маршрутах из OpenStreetMap
routes_1 = pd.read_csv('/kaggle/input/nyfastestroutes/fastest_routes_train_part_1.csv')
routes_2 = pd.read_csv('/kaggle/input/nyfastestroutes/fastest_routes_train_part_2.csv')
```

- `id` — уникальный идентификатор для каждой поездки;
- `vendor_id` — код, указывающий поставщика, связанного с записью о поездке;
- `pickup_datetime` — дата и время включения счетчика;
- `dropoff_datetime` — дата и время отключения счетчика;
- `passenger_count` — количество пассажиров в транспортном средстве (значение, введенное водителем);
- `pickup_longitude` — долгота, где был задействован счетчик;
- `pickup_latitude` — широта, на которой был задействован счетчик;
- `dropoff_longitude` — долгота, где счетчик был отключен;
- `dropoff_latitude` — широта, на которой счетчик был отключен;
- `store_and_fwd_flag` — этот флаг указывает, хранилась ли запись о поездке в памяти транспортного средства перед отправкой продавцу, потому что у транспортного средства не было соединения с сервером — Y = сохранить и переслать; N = не промежуточная поездка
- `trip_duration` — продолжительность поездки в секундах;

Для наглядности представлены первые строки данных:

[127]:

```
weather[:3]
```

[12...]

	date	maximum temperature	minimum temperature	average temperature	precipitation	snow fall	snow depth
0	1-1-2016	42	34	38.000	0.00	0.0	0
1	2-1-2016	40	32	36.000	0.00	0.0	0
2	3-1-2016	45	35	40.000	0.00	0.0	0

▶

```
test[:3]
```

[12...]

	id	vendor_id	pickup_datetime	passenger_count	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	store_and_fwd_flag
0	id3004672	1	2016-06-30 23:59:58	1	-73.988	40.732	-73.990	40.757	N
1	id3505355	1	2016-06-30 23:59:53	1	-73.964	40.680	-73.960	40.655	N
2	id1217141	1	2016-06-30 23:59:47	1	-73.997	40.738	-73.986	40.730	N

▶

```
train[:3]
```

[12...]

	id	vendor_id	pickup_datetime	dropoff_datetime	passenger_count	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	store_and_fwd_flag	trip_duration
0	id2875421	2	2016-03-14 17:24:55	2016-03-14 17:32:30	1	-73.982	40.768	-73.965	40.766	N	455
1	id2377394	1	2016-06-12 00:43:35	2016-06-12 00:54:38	1	-73.980	40.739	-73.999	40.731	N	663
2	id3858529	2	2016-01-19 11:35:24	2016-01-19 12:10:48	1	-73.979	40.764	-74.005	40.710	N	2124

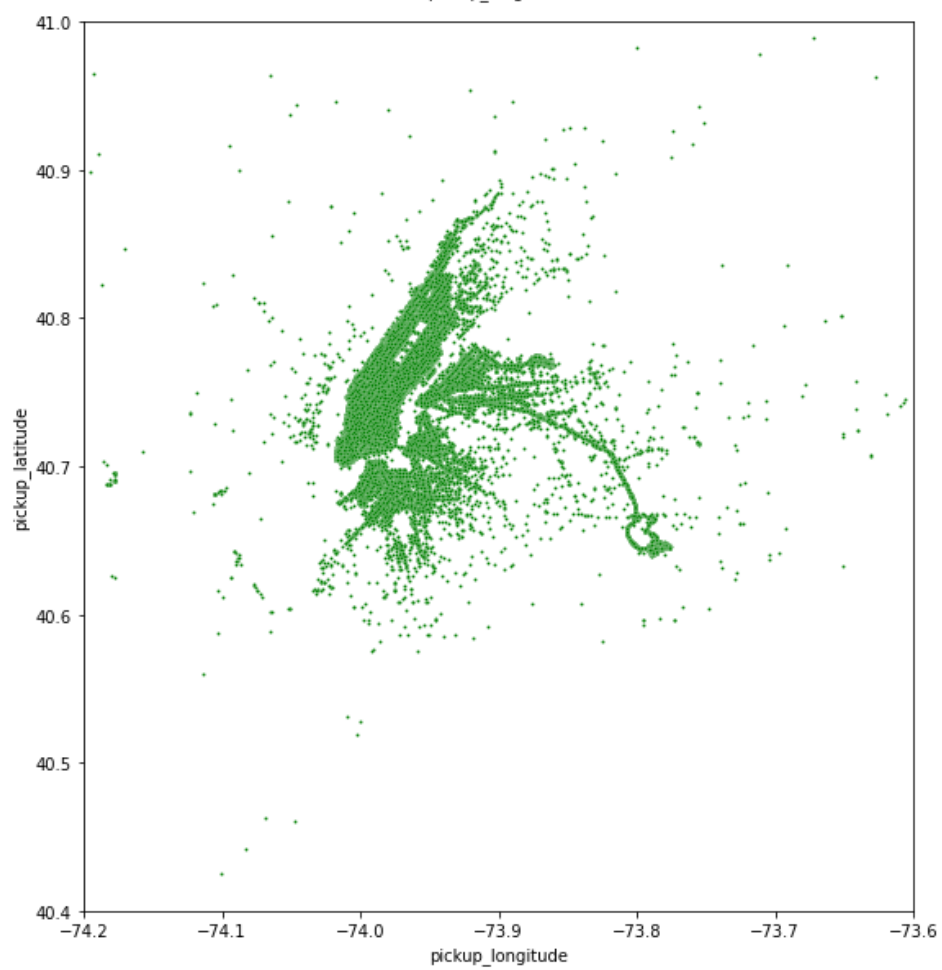
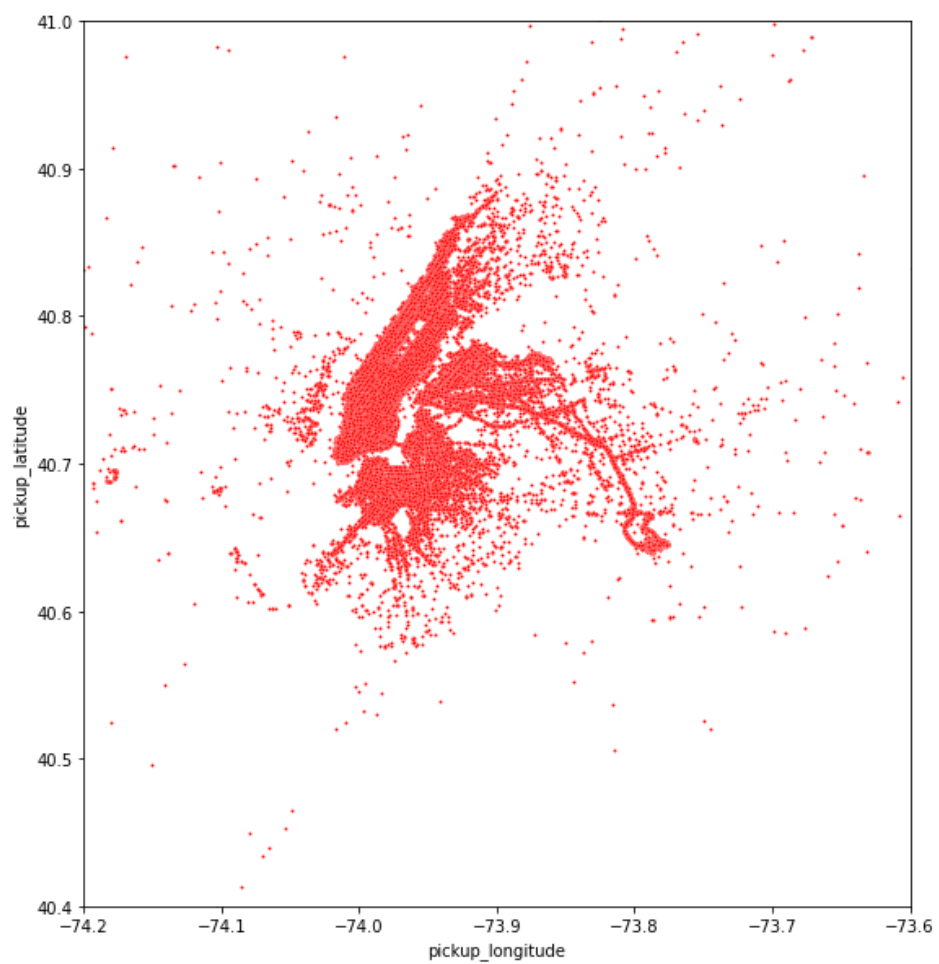
3.2 Анализ данных

Анализ маршрутов:

```
fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(20, 10))

# график тренировочных данных
sns.scatterplot(train['pickup_longitude'], train['pickup_latitude'], s=4, ax=ax[0], color='red')
ax[0].set_xlim([-74.2, -73.6])
ax[0].set_ylim([40.4, 41.0])

# график тестовых данных
sns.scatterplot(test['pickup_longitude'], test['pickup_latitude'], s=4, ax=ax[1], color='green')
ax[1].set_xlim([-74.2, -73.6])
ax[1].set_ylim([40.4, 41.0])
```



```
[134]: # Статистика по каждому числовому признаку
pd.set_option('display.float_format', lambda x: '%.3f' % x)
train.describe()
```

```
[13...      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
```

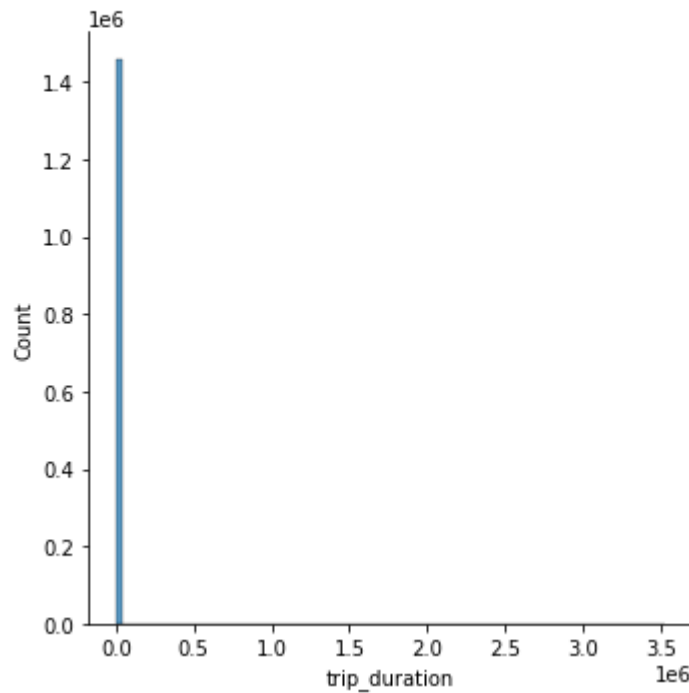
```
[135]: test.describe()
```

```
[13...      vendor_id  passenger_count  pickup_longitude  pickup_latitude  dropoff_longitude  dropoff_latitude
count  625134.000      625134.000      625134.000      625134.000      625134.000      625134.000
mean        1.535          1.662         -73.974         40.751         -73.973         40.752
std          0.499          1.311          0.073          0.030          0.073          0.036
min           1.000          0.000       -121.933         37.390       -121.933         36.601
25%           1.000          1.000       -73.992         40.737       -73.991         40.736
50%           2.000          1.000       -73.982         40.754       -73.980         40.755
75%           2.000          2.000       -73.967         40.768       -73.963         40.770
max           2.000          9.000        -69.249         42.815        -67.497         48.858
```

Очевидные выбросы в кол-ве пассажиров (0 и 9 человек) и продолжительности поездки (1 секунда). Также обратим внимание на долготу и широту, так как нам необходим только город Нью-Йорк, то тогда нужно ограничить пределы географических координат.

Теперь посмотрим на распределение времени поездки:

```
[35]: sns.displot(train_all.trip_duration, bins=100);
```



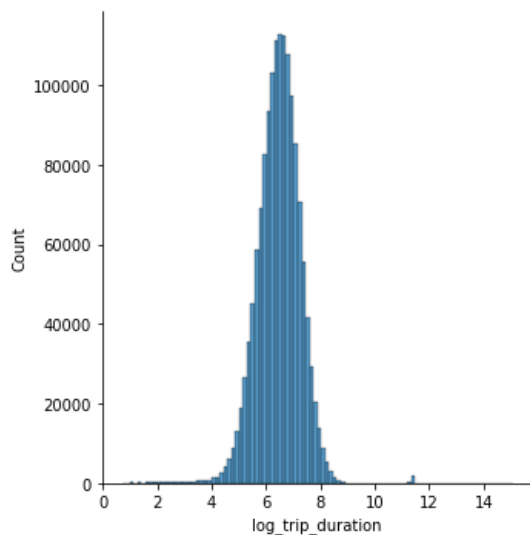
Видим, что есть выбросы, которые обсуждали ранее, прологорифмируем целевую переменную и добавим значение в датасет:

```
[36]: train_all = train_all.assign(log_trip_duration=np.log(train_all.trip_duration + 1))
```

+ Code

+ Markdown

```
[37]: sns.displot(train_all.log_trip_duration, bins=100);
```



Также замечаем, что есть выбросы в скорости (12428 км/ч) и в кол-ве пассажиров (0, 6, 7, 8, 9):

```
[39]: train_all['speed_kmh'].describe()
```

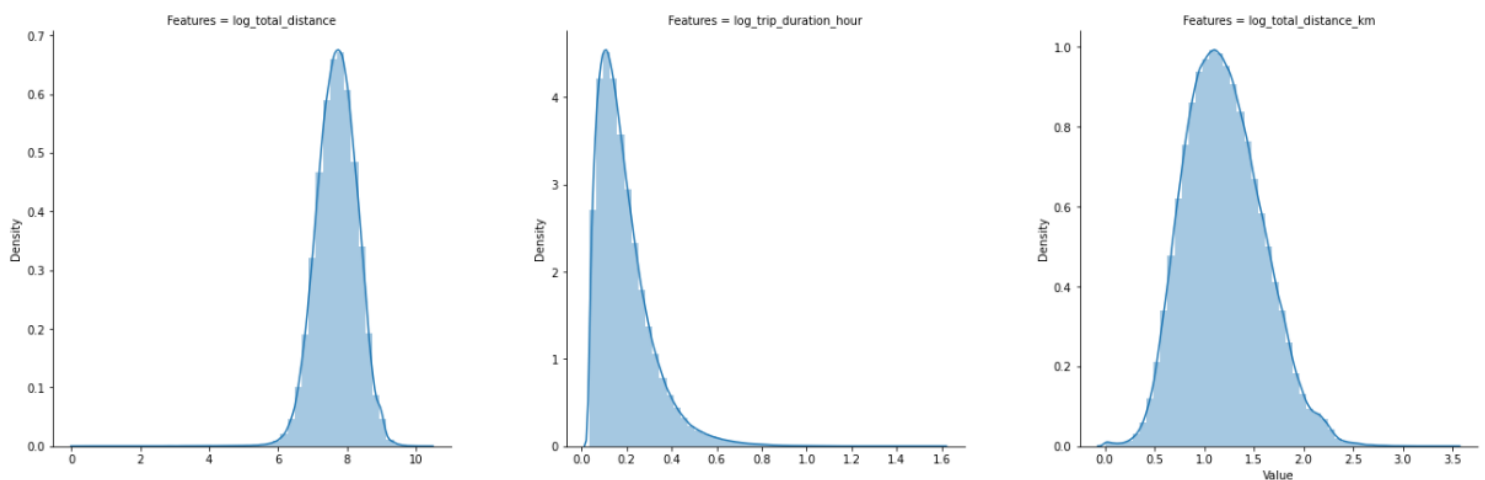
```
[39...] count    1458351.000  
      mean      19.475  
      std       25.220  
      min        0.000  
      25%       12.410  
      50%       17.108  
      75%       23.501  
      max      12428.383  
      Name: speed_kmh, dtype: float64
```

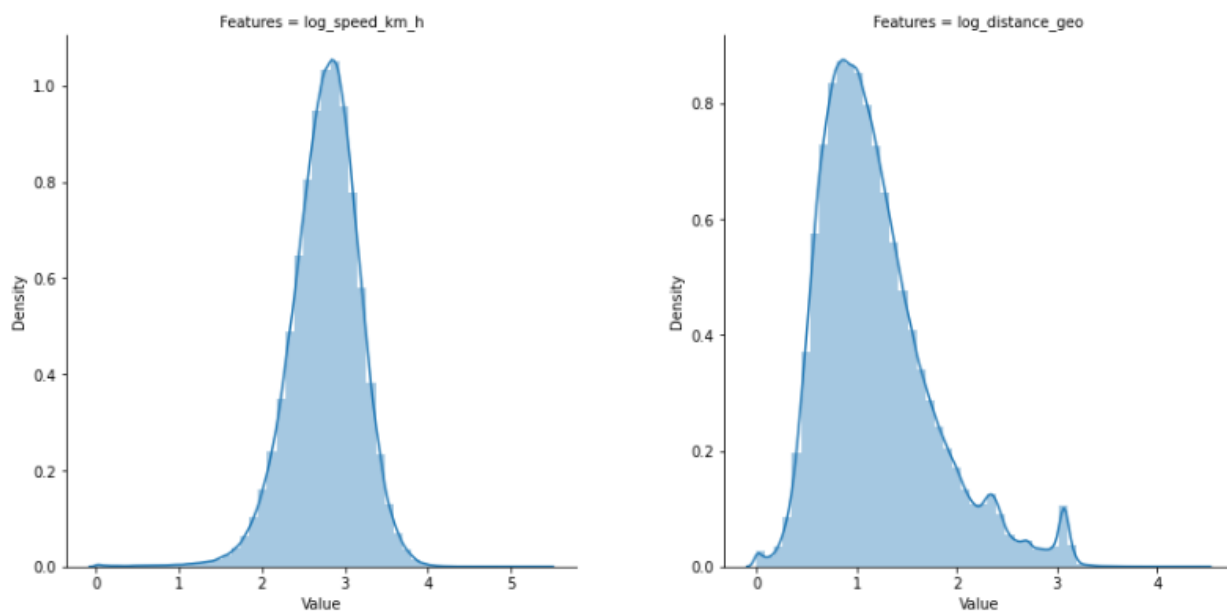
```
train_all['passenger_count'].unique()
```

```
... array([1, 6, 4, 2, 3, 5, 0, 7, 9, 8])
```

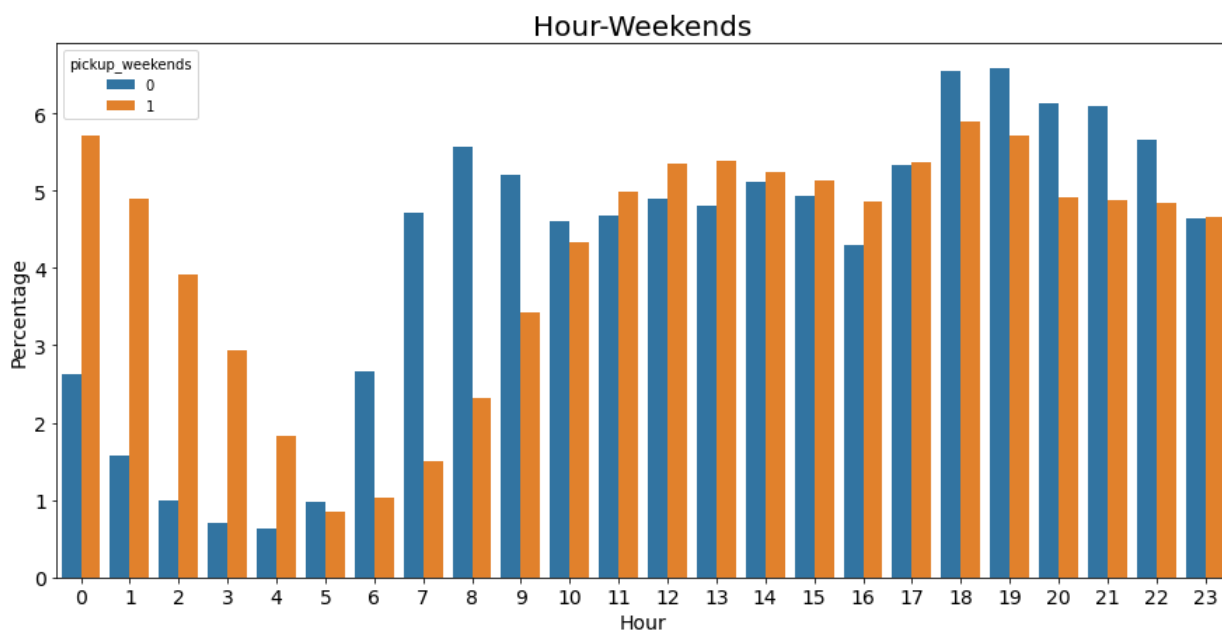
3.3 Обработка выбросов

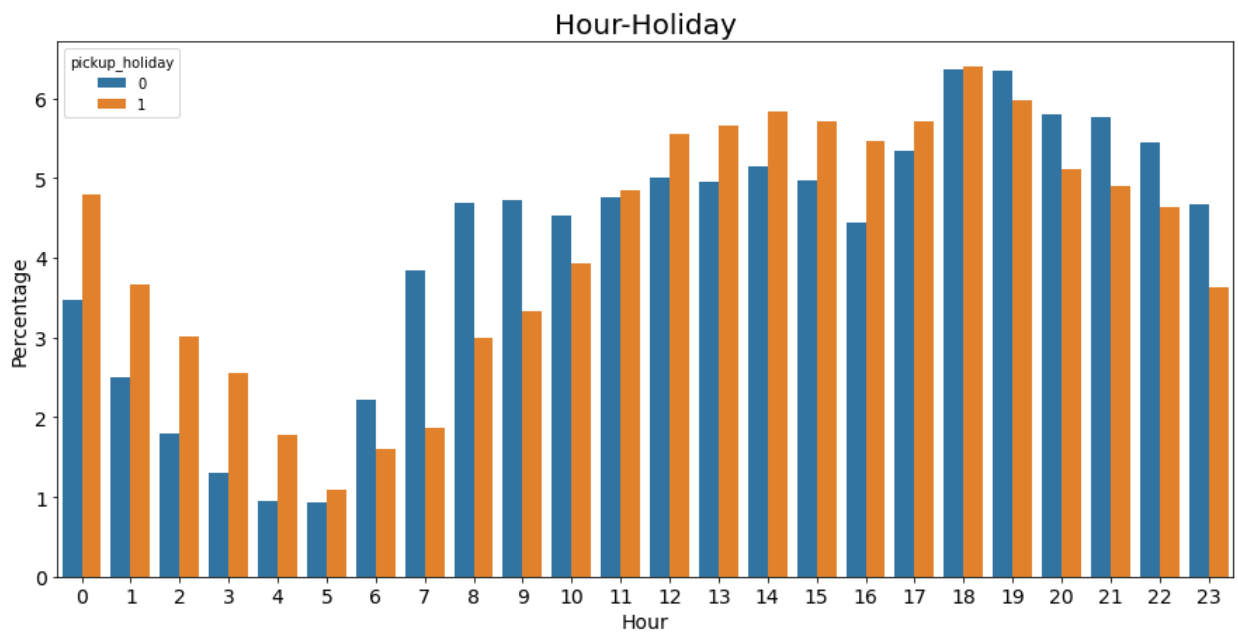
После логарифмизации остальных числовых признаков получаем графики плотностей:





Далее проанализируем, в какие часы, какой процент людей был в выходные и в не выходные дни. На графике видно, что в не выходные дни, (синий) с утра процент пассажиров увеличивается, что говорит о том, что люди спешат на работу, и также процент увеличивается после рабочего дня. В выходные дни люди более активны в ночное время.





3.4 Моделирование

Optuna – фреймворк подбора оптимальных гиперпараметров, который использует байесовский подход для автоматизации пространства поиска гиперпараметров.

Для начала создаём целевую функцию:

```
# целевая функция
def objective(trial, X, y):
    param_grid = {
        "n_estimators": trial.suggest_categorical("n_estimators", [1000]),
        "learning_rate": trial.suggest_float("learning_rate", 0.01, 0.3),
        "num_leaves": trial.suggest_int("num_leaves", 20, 300, step=10),
        "max_depth": trial.suggest_int("max_depth", 3, 10, step=2),
        "min_child_samples": trial.suggest_int("min_child_samples", 5, 200, step=10),
    }

    cv = KFold(n_splits=N_FOLDS, shuffle=True, random_state=RANDOM_STATE)

    cv_predicts = np.empty(5)
    for idx, (train_idx, test_idx) in enumerate(cv.split(X, y)):
        X_train, X_test = X.iloc[train_idx], X.iloc[test_idx]
        y_train, y_test = y.iloc[train_idx], y.iloc[test_idx]

        pruning_callback = optuna.integration.LightGBMPruningCallback(
            trial, "l2")
        model = lg.LGBMRegressor(objective="regression", **param_grid)
        model.fit(X_train,
                  y_train,
                  eval_set=[(X_test, y_test)],
                  eval_metric="rmsle",
                  early_stopping_rounds=100,
                  callbacks=[pruning_callback],
                  verbose=-1)
        preds = model.predict(X_test)
        cv_predicts[idx] = rmsle(y_test, preds)
    return np.mean(cv_predicts)
```

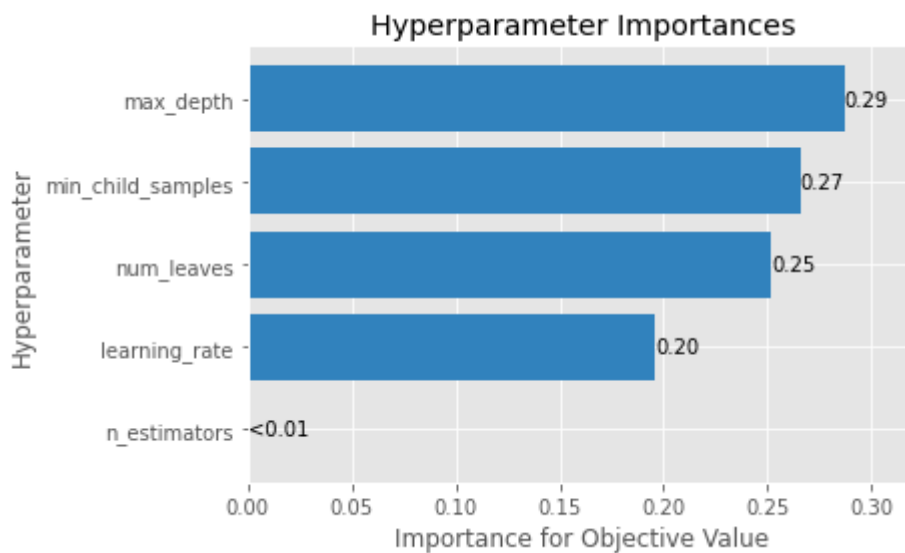
Далее, с помощью Optuna минимизируем эту функцию:

```
study = optuna.create_study(direction="minimize", study_name="LGB")
func = lambda trial: objective(trial, X_train, y_train)
# n_trials - кол-во итераций
study.optimize(func, n_trials=2, show_progress_bar=True)
```

Значения наилучших гиперпараметров:

```
Best value (rmsle): 0.05960
Best params:
  n_estimators: 1000
  learning_rate: 0.12140351113028182
  num_leaves: 250
  max_depth: 7
  min_child_samples: 25
```

Визуализация важности гиперпараметров:



Обучаем модель, воспользовавшись наилучшими параметрами:

```
cv = KFold(n_splits=N_FOLDS, shuffle=True, random_state=RANDOM_STATE)

finish_test_preds = []
cv_predicts = np.empty(5)

for idx, (train_idx, test_idx) in enumerate(cv.split(X_train, y_train)):
    X_train_, X_val = X_train.iloc[train_idx], X_train.iloc[test_idx]
    y_train_, y_val = y_train.iloc[train_idx], y_train.iloc[test_idx]

    model = lg.LGBMRegressor(objective="regression", **study.best_params)
    model.fit(X_train_,
              y_train_,
              eval_set=[(X_val, y_val)],
              eval_metric="rmsle",
              early_stopping_rounds=100,
              verbose=-1)
    preds = model.predict(X_val)
    preds_exp = np.exp(preds) - 1
    y_val_exp = np.exp(y_val) - 1

    cv_predicts[idx] = rmsle(y_val_exp, preds_exp)
    preds_test = model.predict(X_test)
    finish_test_preds.append(preds_test)

    print(f"id = {idx}", cv_predicts[idx], '\n')

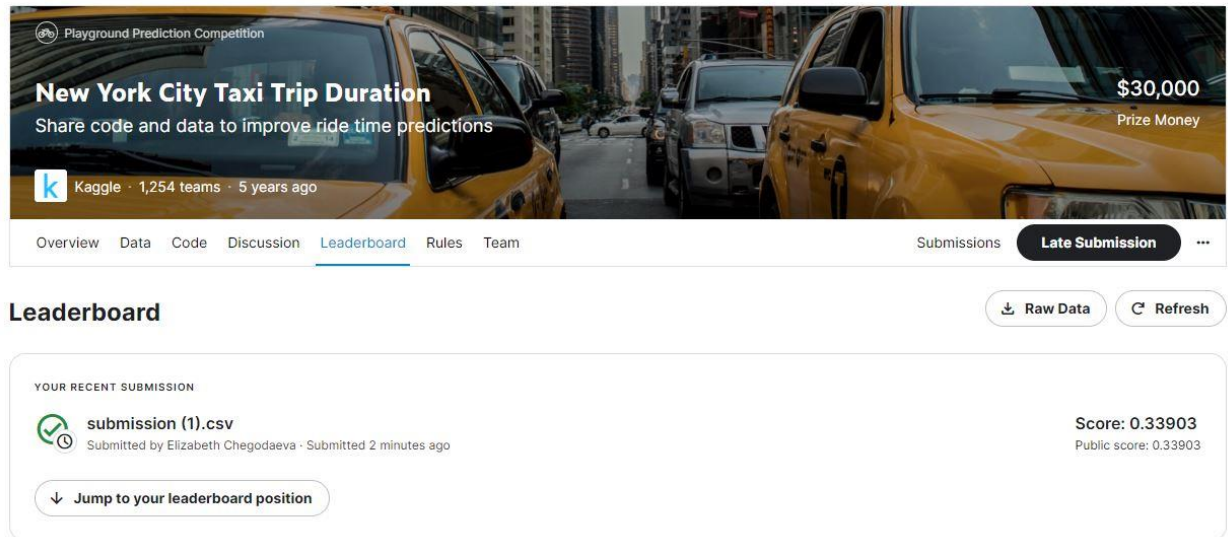
print(np.mean(cv_predicts))
```

OOF score: 0.3393698957254096

HOLDOUT score: 0.3389720108397663

4 Результаты

При участии в соревновании на Kaggle был получен Score: 0.33:



The screenshot shows the Kaggle competition interface for 'New York City Taxi Trip Duration'. The header includes the competition title, a share code and data link, and a prize money of \$30,000. The navigation bar shows 'Overview', 'Data', 'Code', 'Discussion', 'Leaderboard' (selected), 'Rules', and 'Team'. Below the navigation bar, the 'Leaderboard' section displays 'YOUR RECENT SUBMISSION' with a green checkmark icon, the file name 'submission (1).csv', and the submission details: 'Submitted by Elizabeth Chegodaeva · Submitted 2 minutes ago'. The score is shown as 'Score: 0.33903' and 'Public score: 0.33903'. A button 'Jump to your leaderboard position' is also visible.

Были использованы следующие датасеты:

- Годовая история погодных условий за 2016 год ([New York City Taxi Trip Duration](#))
- Годовая история поездок на такси по Нью-Йорку ([Weather data in New York City - 2016](#))
- Данные о протяжённости Нью-Йоркских дорог ([NY Fastest Routes](#))

ЗАКЛЮЧЕНИЕ

Данная работа посвящена изучению методов интеллектуального анализа данных. В рамках этой работы был использован метод градиентного бустинга, а именно, модель LightGBM.

В результате проделанной работы были осуществлены следующие задачи:

- Проведена предварительная обработка данных
- Осуществлено конструирование новых признаков, на основе имеющихся (Feature Engineering)
- Создана модель на основе градиентного бустинга (LightGBM)
- Подобраны оптимальные гиперпараметры модели с помощью фреймворка Optuna
- Предсказана длительность поездки на такси в Нью-Йорке
- Вычислена ошибка предсказания

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. <https://towardsdatascience.com/kagglers-guide-to-lightgbm-hyperparameter-tuning-with-optuna-in-2021-ed048d9838b5?gi=32a22ea8680d>
2. <https://optuna.org>
3. <https://matplotlib.org/stable/index.html>
4. <https://lightgbm.readthedocs.io/en/latest>