

Описание структуры проекта:

"Анализ поведения пользователей сайта "СберАвтоподписка" и разработка ML-модели для прогнозирования целевых действий"

Содержание

1. [**Описание проекта**](#)
 - 1.1 [Основные файлы проекта](#)
 - 1.2 [Директории проекта](#)
2. [**Запуск API-сервиса**](#)
 - 2.1 [PowerShell](#)
 - 2.2 [Postman](#)
 - 2.3 [Примеры POST-запросов](#)
3. [**Скриньки работающего API-сервиса**](#)

Основные файлы проекта:

1. Файл Jupyter Notebook (Lab)

```
part_1_data_preprocessing_and_analysis.ipynb
```

Подготовка основного датасета для моделирования и Анализ данных

- Реализованы следующие этапы:
 - Подготовка данных:
 - Загрузка исходных данных
 - Очистка от дубликатов и пропущенных значений
 - Типизация данных
 - Создание новых признаков (Feature Engineering):
 - Создание новых признаков из сырых данных:
 - Временные признаки из дат визитов
 - Категориальные признаки (utm-метки, device-устройства)
 - Географические признаки (geo)
 - Анализ динамики, распределений и зависимостей:
 - Источники трафика
 - Данные о визитах

- Типы устройств
- География пользователей
- Взаимосвязи между признаками (матрицы корреляций)
- **Визуализация:**
 - Динамика основных показателей
 - Гистограммы распределений
 - Heatmap и матрицы корреляций

2. Файл Jupyter Notebook (Lab)

`part_2_model_selection.ipynb`

Выбор и построение модели для предсказания целевого действия

- **Реализованы следующие этапы моделирования:**
 - Загрузка данных и создание основного датасета.
 - Создание Pipeline для предобработки данных.
 - Разделение данных на обучающую и тестовую выборки с применением undersampling для балансировки классов.
 - Обучение и сравнение моделей
(`LogisticRegression`, `RandomForest`, `MLPClassifier`, `XGBoost`, `LightGBM`, `CatBoost`).
 - Подбор гиперпараметров для лучшей модели.
 - Подготовка финальной модели, обучение на всем датасете.
 - Сохранение и загрузка финальной модели.
 - Проверка загруженной модели, тестирование на выборке из основного датасета.
- **Результаты моделирования представлены в виде:**
 - Таблиц с метриками моделей (ROC-AUC, precision, recall, F1, accuracy).
 - Визуализация важности признаков и ROC-кривых.

3. Файлы Python:

- `main_api_sber.py`
 - **API-сервис (FastAPI)**
 - **Реализованы запросы:**
 - **GET:**
 - `status`
 - `version`
 - **POST:**

- predict
- pipeline_model_sber.py
 - **Реализованы следующие этапы моделирования:**
 - Построение pipeline обработки исходных данных.
 - Обучение модели на тестовом датасете.
 - Кроссвалидация, оценка качества модели (cross_val_score).
 - Обучение модели на полном датасете.
 - Запись модели в папку models.
- testing_on_examples.py
 - Тестирование модели на json-файлах, примерах из папки examples.
 - В папке examples – 100 файлов (50 – с target_action = 1 и 50 – с target_action = 0).
 - Результаты тестирования записываются в папку test_results в текстовый файл вида test_results_YYYY-mm-dd_HH-MM-SS.txt.
 - Время тестирования на 100 файлах порядка 30 секунд.

Директории проекта:

- data/
 - Содержит файлы с исходными данными:
 - ga_sessions.pkl
 - ga_hits.pkl
 - Ссылка на оригиналы файлов с данными ([исходные данные](#))
- models/
 - В нее сохраняются обученные модели, pkl-файлы вида model_prediction_sber*.pkl.
 - Также здесь находится файл locations.csv с гео-координатами (широты и долготы).
 - Координаты получены с помощью пакета geopy и сохранены в этот файл.
 - Данные из файла locations.csv используются при создании новых признаков (geo_latitude, geo_longitude, geo_distance_to_city_km).
 - Есть отдельный режим получения координат налету, но он долго работает, порядка 40 минут.
- examples/
 - Содержит 100 примеров, json-файлов для тестирования работы моделей и API-сервиса.
 - Пример данных одного из файлов:

```
{  
    "session_id": "2500012602493962943.1638337216.1638337216",
```

```
        "client_id": "582079543.1638337215",
        "visit_date": "2021-12-01",
        "visit_time": "08:40:16",
        "visit_number": 1,
        "utm_source": "ZpYI0DJMcFzVoPFsHGJL",
        "utm_medium": "banner",
        "utm_campaign": "LEoPHuyFvzoNfnzGgfcd",
        "utm_adcontent": "vCIpmraGBnIQhyYNkXqp",
        "utm_keyword": "puhZPIYqKXeFPaUviSjo",
        "device_category": "mobile",
        "device_os": "Android",
        "device_brand": "Samsung",
        "device_model": null,
        "device_screen_resolution": "412x869",
        "device_browser": "Chrome",
        "geo_country": "Russia",
        "geo_city": "Bryansk",
        "target_event_action": 1
    }
```

- **test_results/**

- В нее сохраняются результаты тестирования модели (текстовые файлы) при запуске файла `testing_on_examples.py`.

Запуск API-сервиса:

PowerShell

1. **Запустить PowerShell из командной строки**
2. **Перейти в папку проекта PyCharm**

- Путь на моем компьютере:

```
E:\Projects\Python\SkillBox\final_work
```

3. **Активировать окружение проекта**

```
.\.venv\Scripts\activate
```

4. **Запустить сервис "uvicorn"**

```
uvicorn main_api_sber:app --reload
```

Postman

1. **Запустить Postman локально**
2. **Открыть Postman в браузере**
3. **Перейти в раздел работы с запросами**

4. Выполнить запросы:

- **GET:**
 - 127.0.0.1:8000/status
 - 127.0.0.1:8000/version
 - **POST:**
 - 127.0.0.1:8000/predict
-

Примеры для POST-запросов

(Здесь 3 примера. Больше примеров (100 файлов) – см. в папке `examples`.)

Формат запроса:

- **Body:** raw (JSON)
-

Пример 1:

```
{  
    "session_id": "3432873700966186920.1625224106.1625224106",  
    "client_id": "799278193.1625210792",  
    "visit_date": "2021-07-02",  
    "visit_time": "14:00:00",  
    "visit_number": 3,  
    "utm_source": "bByPQxmDaMXgpHeypKSM",  
    "utm_medium": "referral",  
    "utm_campaign": "LTuZkdKfxRGVceoWkVyg",  
    "utm_adcontent": "JNHcPlZPxEMWDnRiyoBf",  
    "utm_keyword": null,  
    "device_category": "desktop",  
    "device_os": null,  
    "device_brand": "",  
    "device_model": null,  
    "device_screen_resolution": "1920x1080",  
    "device_browser": "Chrome",  
    "geo_country": "Russia",  
    "geo_city": "Moscow",  
    "target_event_action": 1  
}
```

Пример 2:

```
{  
    "session_id": "167161116639790856.1636992776.1636992776",  
    "client_id": "38920230.1636992776",  
    "visit_date": "2021-11-15",  
    "visit_time": "19:12:56",  
    "visit_number": 1,  
    "utm_source": "BHcvLf0aCWvWTykYqHVe",  
    "utm_medium": "cpc",  
    "utm_campaign": null,  
    "utm_adcontent": null,  
    "utm_keyword": "lVHkqXlZapie0XCrCJWG",  
    "device_category": "mobile",  
    "device_os": "Android",  
    "device_brand": "Xiaomi",  
    "device_model": null,  
    "device_screen_resolution": "393x873",  
    "device_browser": "YaBrowser",  
    "geo_country": "Russia",  
    "geo_city": "Moscow",  
    "target_event_action": 1  
}
```

Пример 3:

```
{  
    "session_id": "4087249450691937739.1626108893.1626108893",  
    "client_id": "951636920.1625769419",  
    "visit_date": "2021-07-12",  
    "visit_time": "19:00:00",  
    "visit_number": 9,  
    "utm_source": "kjsLglQLzykiRbcDiGcD",  
    "utm_medium": "cpc",  
    "utm_campaign": null,  
    "utm_adcontent": null,  
    "utm_keyword": null,  
    "device_category": "mobile",  
    "device_os": null,  
    "device_brand": "Apple",  
    "device_model": null,  
    "device_screen_resolution": "414x896",  
    "device_browser": "Safari",  
    "geo_country": "Russia",  
    "geo_city": "Kazan",  
    "target_event_action": 0  
}
```

Скрины работающего API-сервиса:

- GET-запрос /status

The screenshot shows the Postman interface with a successful response to a GET request to `127.0.0.1:8000/status`. The response body contains the string "I'm OK".

```
I'm OK
```

- GET-запрос /version

The screenshot shows the Postman interface with a successful response to a GET request to `127.0.0.1:8000/version`. The response body is a JSON object containing model details.

```
{"name": "SberAutoPodiska: prediction model", "author": "Andrei Mishenkov", "version": 1.0, "date": "2025-07-22 12:52:49", "type": "LGBMClassifier", "roc_auc": 0.7207}
```

- POST-запрос /predict (1-й пример)

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'Collections', 'Environments', 'Flows', and 'History'. The main area displays a 'My Workspace' collection with a single item named 'Your collection'. The 'Body' tab of the request editor is selected, showing a JSON payload with various user-related parameters. The response pane shows a 200 OK status with a timestamp of 501 ms and a size of 210 B. The response body is identical to the request body.

```
1 "session_id": "3432873700966186920.1625224106.1625224106",
2 "client_id": "799278193.1625210792",
3 "visit_date": "2021-07-02",
4 "visit_time": "14:00:00",
5 "visit_number": 3,
6 "utm_source": "bbyQxmBaMKgPheykSM",
7 "utm_medium": "referral",
8 "utm_campaign": "LTUzidKfRGVceoWkVyg",
9 "utm_adcontent": "JNHcPlZPxEMWDnRiyobE",
10 "utm_keyword": null,
11 "device_category": "desktop",
12 "device_os": null,
13 "device_brand": "",
14 "device_model": null,
15 "device_screen_resolution": "1920x1080",
16 "device_browser": "Chrome",
17 "geo_country": "Russia",
18 "geo_city": "Moscow",
19 "target_event_action": 1
20 }
21
22
```

```
1 {
2   "session_id": "3432873700966186920.1625224106.1625224106",
3   "pred": 1,
4   "target_action": 1
5 }
```

- POST-запрос /predict (2-й пример)

This screenshot shows a second POST request to the '/predict' endpoint in Postman. The request body is identical to the one in the previous screenshot. The response is a 200 OK status with a timestamp of 553 ms and a size of 209 B. The response body is identical to the request body.

```
1 "session_id": "167161116639790856.1636992776.1636992776",
2 "client_id": "38920230.1636992776",
3 "visit_date": "2021-11-15",
4 "visit_time": "19:12:56",
5 "visit_number": 1,
6 "utm_source": "BHevLfd0aCwvNTyqHve",
7 "utm_medium": "cpc",
8 "utm_campaign": null,
9 "utm_adcontent": null,
10 "utm_keyword": "lVHkqXlZapieOXCrCJW6",
11 "device_category": "mobile",
12 "device_os": "Android",
13 "device_brand": "Xiaomi",
14 "device_model": null,
15 "device_screen_resolution": "393x873",
16 "device_browser": "YaBrowser",
17 "geo_country": "Russia",
18 "geo_city": "Moscow",
19 "target_event_action": 1
20 }
21
22
```

```
1 {
2   "session_id": "167161116639790856.1636992776.1636992776",
3   "pred": 1,
4   "target_action": 1
5 }
```

- POST-запрос /predict (3-й пример)

127.0.0.1:8000/predict - My Workspace

POST 127.0.0.1:8000/predict

Params Authorization Headers (9) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```

1 {
2   "session_id": "4087249450691937739.1626108893.1626108893",
3   "client_id": "951636920.1625769419",
4   "visit_date": "2021-07-12",
5   "visit_time": "19:00:00",
6   "visit_number": 9,
7   "utm_source": "kjSlg1QLzykiRbcDiGcD",
8   "utm_medium": "cpc",
9   "utm_campaign": null,
10  "utm_adcontent": null,
11  "utm_keyword": null,
12  "device_category": "mobile",
13  "device_os": null,
14  "device_brand": "Apple",
15  "device_model": null,
16  "device_screen_resolution": "414x896",
17  "device_browser": "Safari",
18  "geo_country": "Russia",
19  "geo_city": "Kazan",
20  "target_event_action": 0
21 }

```

Body Cookies Headers (4) Test Results

{ } JSON Preview Visualize

200 OK 580 ms 210 B