Предиктивная аналитика

Описание прогнозной модели

Прогнозные модели и скрипты размещены на сервере `izb-ally-nodered02` (IP: 10.1.241.244).

Скрипты написаны на **Python** с использованием библиотеки **ENTA**

```
python3.10 main.py --help
Usage: main.py [OPTIONS] COMMAND [ARGS]...

Options:
  --help  Show this message and exit.

Commands:
  copy      copy time_series from one DB to another
  fit       Fits model with collected lines from DB
  forecast  Makes forecast for the next 7 days and saves it to DB
  test      Evaluates forecast metrics
  view      Prints one segment used for forecast
```

Наилучшие результаты при прогнозировании показала модель **CatBoost**.

Перед использованием модель необходимо обучить. После обучения она может применяться для построения прогноза.

Данные для обучения поступают из различных источников, а результаты прогнозов сохраняются в гипертаблицу `time_series`.

Для обучения используются погодные данные, которые ежедневно загружаются с сайта rp5.ru в таблицы `weather` и `weather_stations`. Загрузка осуществляется с помощью скрипта `rp5_weather`, который настраивается через файл `static/cities.txt`. Этот файл содержит список метеостанций по всей стране.

В таблице `weather` также хранятся исторические данные о погоде, начиная с 2005 года.

Кроме исторических данных, для прогнозирования требуется актуальный прогноз погоды. Он ежедневно загружается с сайта api.met.no через **Node-RED** в таблицу `weather`.

Также в прогнозе используется данные табель календаря из таблицы calendar и календарь Православных Христианских праздников.

Также при построении прогноза учитываются данные производственного календаря из таблицы `calendar` и календаря православных праздников.

Попытка включить мусульманские праздники в расчёт прогноза не улучшила точность.

Прогноз может строиться на заданное количество дней вперёд и с различной степенью дискретизации.

Работа с данными для прогнозов

Мы получаем исходные данные из базы данных MS SQL с помощью SQL-запросов.

Получение данных о заказах

Для получения данных о заказах составляется отчет по заказам с фильтрацией по дате заказа и наличию даты поставки.

Источником получения данных служит таблица Report.[report_zakaz_tbl]

Запрос извлекает данные о заказах из таблицы отчетов за указанный период времени. Запрос фокусируется на заказах, где дата заказа находится в заданном диапазоне, и дата поставки не является NULL. Дополнительно вычисляется поле с датой поставки, увеличенной на 15 минут.

Текст запроса

Ниже представлен запрос для получения информации о заказах:

```
SELECT
    id_order as id,
    ShopNo as shopno,
    order_type,
    gettype,
    date_order,
    date_supply,
    date_supply_untill,
    date_posted,
    date_collect_start,
    date_collected,
```

```
    date_delivery_start,
    date_delivered,
    sum_paid,
    sum_paid_coupon,
    count_pos,
    latitude,
    longitude,
    client_type,
    order_weight,
    delivery_hottime,
    collecting_await_dur,
    collecting_dur,
    delivery_await_dur,
    delivery_dur,
    completed_agg,
    distance,
    DATEADD(Minute, 15, date_supply) as date_supply_vv
FROM Report.[report_zakaz_tbl] rzt (NOLOCK)
WHERE date_order BETWEEN '{{payload.from}}' AND '{{payload.to}}'
AND date_supply IS NOT NULL
```

Поля запроса

Запрос возвращает 27 полей. Ниже приведена таблица с описаниями:

Поле SELECT	Исходное поле	Тип данных
id	id_order	INT
shopno	ShopNo	INT
order_type	order_type	TinyInt
gettype	gettype	TinyInt
date_order	date_order	DateTime
date_supply	date_supply	DateTime

date_supply_untill	date_supply_untill	DateTime
date_posted	date_posted	DateTime
date_collect_start	date_collect_start	DateTime
date_collected	date_collected	DateTime
date_delivery_start	date_delivery_start	DateTime
date_delivered	date_delivered	DateTime
sum_paid	sum_paid	FLOAT
sum_paid_coupon	sum_paid_coupon	FLOAT
count_pos	count_pos	SmallInt
latitude	latitude	Decimal(19,16)
longitude	longitude	Decimal(19,16)
client_type	client_type	NVarChar(100) COLLATE Cyrillic_General_CI_AS
order_weight	order_weight	Decimal(15,3)
delivery_hottime	delivery_hottime	DateTime
collecting_await_dur	collecting_await_dur	INT
collecting_dur	collecting_dur	INT
delivery_await_dur	delivery_await_dur	INT
delivery_dur	delivery_dur	INT
completed_agg	completed_agg	INT
distance	distance	Real
date_supply_vv	date_supply	(вычисляемое)

Фильтры запроса

В запросе присутствуют 2 фильтра:

- date_order BETWEEN '{{payload.from}}' AND '{{payload.to}}' — Фильтр по дате заказа. Включает заказы, где дата заказа >= '{{payload.from}}' и <= '{{payload.to}}'.
- date_supply IS NOT NULL — Исключает заказы без указанной даты поставки.

Вычисляемые поля

В запросе есть вычисляемое поле:

- DATEADD(Minute, 15, date_supply) as date_supply_vv — Функция SQL Server для добавления 15 минут к полю date_supply.

Получение данных о геозонах

Для получения данных о геозонах составляется отчет по зонам с фильтрацией по магазинам.

Источником получения данных служит таблицы [GeoReports].[Analytics].[EffectiveZonesOnlineServices] и [Geo].[geo].[tt]

Запрос объединяет данные из двух таблиц для получения информации о зонах, услугах и временных интервалах.

Текст запроса

Ниже представлен запрос для получения информации о заказах:

```
SELECT
    tt.N AS id_tt,
    id_online_service,
    id_poly,
    date_add,
    geo,
```

```
    payway,
    time_start,
    time_end
FROM [GeoReports].[Analytics].[EffectiveZonesOnlineServices] (NOLOCK) z
JOIN [Geo].[geo].[tt] tt ON tt.id_TT = z.id_tt
WHERE tt.name_TT LIKE '%ДС[_]%
```

Поля запроса

Запрос возвращает 8 полей. Ниже приведена таблица с описаниями:

Поле SELECT	Исходное поле	Тип данных
id_tt	tt.N	
id_online_service	id_online_service	
id_poly	id_poly	
date_add	date_add	
geo	geo	
payway	payway	
time_start	time_start	
time_end	time_end	

Фильтры запроса

В запросе присутствуют 2 фильтра:

- INNER JOIN на tt.id_TT = z.id_tt — Это объединяет записи только если есть совпадение по идентификатору точки (id_tt). Если в z нет соответствующей записи в tt, она не попадет в результат.
- WHERE tt.name_TT like '%ДС[_]%' — Фильтр по имени точки (магазина) в таблице tt.

Таблицы хранения полученных данных

Исходные данные, используемые для построения прогноза, размещены в следующих таблицах:

- vv_orders_ts — гипертаблица с информацией о заказах. Включает поля, заполняемые на основе исходной таблицы Report.[report_zakaz_tbl], а также ряд дополнительных полей.
- test_vv_points — географические зоны, связанные с торговыми точками (ТТ).
- weather — данные о погоде, загруженные из сервиса https://api.met.no.
- weather_stations — погодные станции с сервиса rp5.ru. Применялись для загрузки исторических метеоданных.
- calendar — табель-календарь, заполняемый посредством системы репликации.

Кроме того, на базе этих таблиц создаются материализованные представления:

- vv_orders_ts_hash_hourly — заказы в географической зоне за определенный час, исключая заказы с типом gettype = 6 (доставка через Яндекс).
- vv_lines_ts_hash_hourly — количество собранных строк для географической зоны за конкретный час.

Ниже приведено описание этих таблиц и представлений, в которым перечислены поля использующиеся для составления прогноза.

vv_orders_ts

Поле	Тип данных
id	int4
shopno	int4
order_type	int4
gettype	int4
date_order	timestamp
date_supply	timestamp
date_supply_untill	timestamp

date_posted	timestamp
date_collect_start	timestamp
date_collected	timestamp
date_delivery_start	timestamp
date_delivered	timestamp
sum_paid	float8
sum_paid_coupon	float8
count_pos	int2
latitude	numeric(19, 16)
longitude	numeric(19, 16)
client_type	varchar(100)
order_weight	numeric(15, 3)
delivery_hottime	timestamp
collecting_await_dur	int4
collecting_dur	int4
delivery_await_dur	int4
delivery_dur	int4
completed_agg	int4
distance	float4
date_supply_vv	timestamp
details	jsonb
tt_id	int4

geohash	varchar(20)

test_vv_points

Поле	Тип данных	Описание
geohash	varchar(20)	Геохэш области
tt_id	numeric	ИД иорговой точки
coeff	int4	Вероятность того, что заказ в данной области попадет в данную ТТ. Определяется на основе статистики заказов за предыдущие 3 дня

weather

Поле	Тип данных	Описание
weather_station_id	int4	ИД погодной станции
date	timestamp	Время
temperature	numeric(3, 1)	Температура
humidity	int4	Влажность
wind_speed	int4	Скорость ветра

weather_stations

Поле	Тип данных	Описание
id	serial4	ИД погодной станции
active	bool	Признак активности

calendar

Поле	Тип данных	Описание
id	serial4	ИД
date	date	Дата
type	int4	Тип (2 - суббота, 3 - воскресенье, 4 - предпраздничный день, 5 - праздник)

vv_orders_ts_hash_hourly

Для составления представления формируется запрос который формирует представление для хранения агрегированных данных о количестве заказов, сгруппированных по геохешу и часовым интервалам. Исключаются заказы с типом получения gettype = 6. Представление обновляется автоматически.

Источником получения данных служит таблица vv_orders_ts.

Текст запроса:

```
CREATE MATERIALIZED VIEW vv_orders_ts_hash_hourly
WITH (timescaledb.continuous) AS
SELECT
    geohash,
    time_bucket('1 hour', ts.date_supply_vv) AS bucket,
    COUNT(*) AS cnt
FROM vv_orders_ts ts
WHERE ts.gettype != 6
GROUP BY geohash, bucket;
```

Поля запроса:

Поле	Тип данных	Описание
geohash	varchar(20)	
bucket	timestamp	

cnt	int8	

vv_lines_ts_hash_hourly

Для составления представления формируется запрос который формирует представление для хранения агрегированных данных о суммарном количестве позиций (строк) в заказах, сгруппированных по геохешу и часовым интервалам.

Источником получения данных служит таблица vv_orders_ts.

Текст запроса:

```
CREATE MATERIALIZED VIEW vv_lines_ts_hash_hourly
WITH (timescaledb.continuous) AS
SELECT
    geohash,
    time_bucket('1 hour', ts.date_supply_vv) AS bucket,
    SUM(ts.count_pos) AS cnt
FROM vv_orders_ts ts
GROUP BY geohash, bucket;
```

Поля запроса:

Поле	Тип данных	Описание
geohash	varchar(20)	
bucket	timestamp	
cnt	int8	

Хранение результатов обработки исходных данных

Данные для прогнозов, сами прогнозы и результаты их анализа хранятся во временных рядах в таблице time_series:

Поле	Тип данных	Описание
id	bigserial	
tstamp	timestamptz	
type	int4	Описывает тип данных
restaurant_id	int4	
user_id	int4	
value	float8	Значение прогноза
details	jsonb	

Тип прогноза, для которого сформирован результат обработки, определяется значением поля type. Возможные числовые значения поля и их интерпретация приведены ниже. Сформированный прогноз по часам помещается в таблицу под типами 4-7

type	Описание	Подразделение	Тип	Источник
1	Данные по доставке	Последняя миля	Исходные данные	Репликатор
2	Данные по сборке	Последняя миля	Исходные данные	Репликатор
4	Прогноз доставка	Последняя миля	Прогноз	Python
5	Прогноз сборка	Последняя миля	Прогноз	Python
6	Прогноз по геозонам доставка	Последняя миля	Прогноз	Python
7	Прогноз по геозонам сборка	Последняя миля	Прогноз	Python
51	Оценка трудоемкости розница	Розница	Исходные данные	Node-RED
52	Прогноз суммарной трудоемкости Розница	Розница	Прогноз	Python
53	Прогноз трудоемкости кассиров Розница	Розница	Прогноз	Python
55	Факт суммарной выработки Розница	Розница	Аналитика	pgAgent

56	Факт выработки кассиров Розница	Розница	Аналитика	pgAgent
154	Курьеры факт	Последняя миля	Аналитика	Node-RED
155	Курьеры план	Последняя миля	Аналитика	Node-RED
156	Курьеры прогноз	Последняя миля	Аналитика	Node-RED
157	Заказы план	Последняя миля	Аналитика	Node-RED

Запросы для составления прогнозов

Данные о доставке и сборке используются в качестве основного прогнозируемого ряда, в то время как остальные в качестве регрессионных данных.

Зональный прогноз доставки

Для зонального прогноза доставки извлекаются и агрегируются данные рассчитываемые как сумма количества заказов (cnt), умноженного на коэффициент из геоточки (coeff), сгруппированных по идентификатору торговой точки (tt_id) и часовому интервалу (bucket).

Ниже представлен запрос который делает все это и объединяет данные о заказах с географическими точками для анализа по сегментам (торговым точкам или зонам) в заданном диапазоне дат:

```
SELECT
    h.bucket AS timestamp,
    vp.tt_id AS segment,
    SUM(cnt * coeff) AS target
FROM vv_orders_ts_hash_hourly h
JOIN test_vv_points vp ON vp.geohash = h.geohash
WHERE bucket BETWEEN '{from_date}' AND '{to_date}'
GROUP BY vp.tt_id, h.bucket;
```

Запрос берет значения из:

- vv_orders_ts_hash_hourly — представление которое содержит агрегированные данные о количестве заказов по геохешу и часам.
- test_vv_points — таблица, содержащая географические зоны, связанные с торговыми точками.

Запрос формирует следующие поля:

Поле в SELECT	Исходное поле	Описание
timestamp	h.bucket	Временной интервал, начало часа для агрегации данных о заказах.
segment	vp.tt_id	Идентификатор торговой точки, к которой привязан geohash.
target	(Вычисляемое)	Сумма (cnt * coeff), где cnt — количество заказов, coeff — коэффициент из геоточки.

Зональный прогноз сборки

Для зонального прогноза сборки извлекаются и агрегируются данные рассчитываемые как сумма количества строк в заказах (cnt), умноженного на коэффициент (coeff), сгруппированных по идентификатору торговой точки (tt_id) и часовому интервалу (bucket).

Ниже представлен запрос который который делает все это и объединяет данные о строках заказов с географическими точками для зонального анализа в заданном диапазоне дат.

```
SELECT
    h.bucket AS timestamp,
    vp.tt_id AS segment,
    SUM(cnt * coeff) AS target
FROM vv_lines_ts_hash_hourly h
JOIN test_vv_points vp ON vp.geohash = h.geohash
WHERE bucket BETWEEN '{from_date}' AND '{to_date}'
GROUP BY vp.tt_id, h.bucket;
```

Запрос берет значения из:

- vv_lines_ts_hash_hourly — представление которое содержит агрегированные данные о количестве заказов по геохешу и часам.
- test_vv_points — таблица, содержащая географические зоны, связанные с торговыми точками.

Запрос формирует следующие поля:

Поле в SELECT	Исходное поле	Описание

timestamp	h.bucket	Временной интервал, начало часа для агрегации данных о заказах.
segment	vp.tt_id	Идентификатор торговой точки, к которой привязан geohash.
target	(Вычисляемое)	Сумма (cnt * coeff), где cnt — количество строк в заказах, coeff — коэффициент из геоточки.

Данные о погоде

В прогнозах используются данные о погоде, такие как температура, влажность и скорость ветра из активных погодных станций в заданном диапазоне дат.

Ниже представлен запрос данных о погоде с дополнением последними значениями на конечную дату.

```
WITH t AS (
    SELECT
        w.weather_station_id AS station_id,
        w.date,
        w.temperature,
        w.humidity,
        w.wind_speed,
        ROW_NUMBER() OVER (PARTITION BY w.weather_station_id ORDER BY w.date DESC) AS rn
    FROM weather w
    JOIN weather_stations st ON st.id = w.weather_station_id
    WHERE st.active AND date BETWEEN '{from_date}' AND '{to_date}'
)
SELECT
    station_id,
    date,
    temperature,
    humidity,
    wind_speed
FROM t
UNION
SELECT
    station_id,
```

```
    '{to_date}',
    temperature,
    humidity,
    wind_speed
FROM t
WHERE t.rn = 1
ORDER BY station_id, date;
```

Запрос берет значения из:

- weather — содержит исторические данные о погоде по станциям и датам.
- weather_stations — содержит информацию о погодных станциях.

Запрос формирует следующие поля:

Поле в SELECT	Исходное поле	Описание
station_id	w.weather_station_id	Идентификатор погодной станции.
date	w.date	Дата измерения погоды.
temperature	w.temperature	Температура
humidity	w.humidity	Влажность воздуха
wind_speed	w.wind_speed	Скорость ветра

На основе метеорологических данных рассчитывается эквивалентная температура, которая используется в качестве входных данных для регрессионного анализа.

```
37 - (37 - temperature) / (0.68 - 0.0014 * humidity] + 1 / (1.76 + 1.4 * pow(wind_speed, 0.75))) - 0.29 * temperature * (1 - humidity / 100)
```

В этой формуле:

- Вычисляется разница между 37°C и фактической температурой: (37 - temperature). Это базовый "дефицит тепла".
- Вычисляется фактор сопротивления: (0.68 - 0.0014 * humidity + 1 / (1.76 + 1.4 * pow(wind_speed, 0.75))). Он увеличивается при высокой влажности (меньше охлаждения) и уменьшается при сильном ветре (больше охлаждения).
- Делится разница на фактор сопротивления и вычитается из 37: это даёт основную ощущаемую температуру с учетом конвекции.

- Вычитается корректировка на испарение: 0.29 * temperature * (1 - humidity / 100), которая дополнительно охлаждает в сухих условиях.

Календарные данные

В прогнозах используются данные производственного календаря из таблицы и календарь Православных Христианских праздников.

Ниже представлен запрос календарных данных:

```
SELECT
    date,
    type AS holiday
FROM calendar
WHERE date BETWEEN '{from_date}' AND '{to_date}'
ORDER BY date;
```

Запрос берет значения из:

- calendar — табель-календарь, заполняемый через систему репликации

Запрос формирует следующие поля:

Поле в SELECT	Исходное поле	Описание
date	date	Дата календарного события
holiday	type	Тип события

Данные о православных праздниках хранятся в файле: calendar.csv

Логика прогнозов

Прогноз для последней мили:

Прогнозы по сборке и доставке строятся с шагом 1 час и охватывают период в 14 дней вперёд.

Обычный прогноз.

Основан на данных о сборке и доставке, где каждый заказ привязан к ТТ.

Данные загружаются в БД Ally из БД MS SQL через **Node-RED**.

Зональный прогноз.

Основан на данных о сборке и доставке, где каждый заказ привязан к ТТ, и на статистике распределения адресов по зонам той или иной ТТ.

На основе геозон ТТ формируются сводные буферы, по которым рассчитываются вероятности выполнения заказов конкретной ТТ, исходя из статистики за последние 7 дней.

Вся история заказов автоматически сопоставляется с текущими геозонами, что позволяет обучать модель так, как будто эти зоны всегда были такими. Это также даёт возможность строить прогнозы для новых ТТ с недостаточной историей данных.

Прогноз для розницы

Прогноз по трудоёмкости строится с шагом 1 день и охватывает 21 день вперёд.

Основан на оценке трудоёмкости, которая загружается во временной ряд 51 через **Node-RED**.

Расчёт требуемого количества сборщиков

Прогноз трудоёмкости сборщиков автоматически пересчитывается в требуемое количество сборщиков. Расчёт выполняется по следующему алгоритму:

1. **Оценка скорости сборки сотрудников**
   Для каждого сотрудника рассчитывается средняя скорость сборки на основе фактических данных за последний месяц. Скорость выражается в количестве строк, которые сотрудник собирает в час.
2. **Расчёт плановой производительности**
   Для каждого часа прогнозируемого периода рассчитывается ожидаемое количество собранных строк на основе списка запланированных сборщиков и их индивидуальной скорости работы.
3. **Сравнение с прогнозом**
   Вычисляется разность между прогнозируемым количеством строк на сборку и плановой производительностью сборщиков.
4. **Корректировка численности**
   Разность преобразуется в дополнительное или избыточное количество сотрудников. Для этого используется средняя скорость сборки по всем сборщикам на данной торговой точке.
5. **Вывод итогового количества**
   Полученное число используется для формирования рекомендаций: сколько сотрудников необходимо добавить или убрать из смены в конкретный час.

Для подсчета скорости сборки заказов используется подход средней скорости сборки которая рассчитывается как количество собранных строк в течение одного часа (строки / час).

Расчет требуемого количества курьеров

Прогноз по требуемому количеству курьеров можно описать формулой:

$$R_t = N_t + \left\lceil \frac{D_t - \sum_{i=1}^{N_t} v_i}{\overline{v}} \right\rceil$$

Где:

t — ?????????? ???, ? ??????? ???? ????

Dt — ??????? ??????? ?? ???????? ????????? ? ???? ???

Nt — ??????? ????? ??? ????????? ????? ?? ???? ???

v_i — ??????? ???????? ?????? i-?? ????????, ????????????? ? ?????

$\overline{v}$ — ??????? ???????? ?? ???? ???????? ?? ?????

Rt — ??????? ?????????? ????? ??????? ? ????? ? ???? ???