**CCЫЛКА НА ПРОЕКТ ЦЕЛИКОМ:** [**https://drive.google.com/drive/folders/1-6w-eYeTDt4Sg3uUY5hjvFOTLL-XmesF?usp=sharing**](https://drive.google.com/drive/folders/1-6w-eYeTDt4Sg3uUY5hjvFOTLL-XmesF?usp=sharing)

**РЕПОЗИТОРЙ ПРОЕКТА:** [**https://github.com/TeamLight2023/Task6**](https://github.com/TeamLight2023/Task6)

**ОСНОВНЫЕ ФАЙЛЫ**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**opt\_route\_1.py** - результат работы над задачей "поиск оптимального числа броневиков".

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Класс OptRouteFinder представляет собой алгоритм поиска оптимальных маршрутов при необходимости выполнения заданных в ТЗ условий:

1) обязательно обслуживать терминал, если терминал сегодня будет переполнен

2) обязательно обслуживать терминал раз в 14 дней

Использовался на ФАКТИЧЕСКИХ ДАННЫХ для подбора параметров функции штрафа, которая бы позволила выполнить все условия

с минимальным числом броневиков и издержками.

Было протестировано число броневиков 6 (успешно), 5 (успешно), 4 (не успешно).

АЛГОРИТМ РЕАЛИЗОВАН ПОСЛЕДОВАТЕЛЬНО ПО ДНЯМ, НЕ БЫЛО НЕОБХОДИМОСТИ РАЗБИВАТЬ НА ТРЕНИРОВОЧНЫЙ И ТЕСТОВЫЙ НАБОР.

ЕСЛИ АЛГОРИТМ ПРОШЕЛ ТРЕНИРОВОЧНЫЙ ПЕРИОД, ОН МОГ БЫ ДАЛЕЕ НЕ ПРОЙТИ ТЕСТОВЫЙ. И НАОБОРОТ.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**opt\_route\_2.py - ИТОГОВЫЙ алгоритм поиска оптимальных маршрутов для N броневиков (в нашем случае 5) на ПРОГНОЗИРУЕМЫХ данных по приросту наличности.**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Добаботка класса OptRouteFinder с учетом того, что работаем с прогнозами.

АЛГОРИТМ УСПЕШНО ОТРАБОТАЛ на 5 броневиках, подтвердив стабильность определенной в opt\_route\_1 функции штрафа и числа броневиков.

1. Ввиду неуверенности в прогнозах добавили работу с выбросами. Определили на обучающем периоде терминалы, для которых характерны большИе выбросы

(более 300000/500000 руб и отклонением больше 1 сигмы от среднего в бОльшую сторону) и бинарно отметили их для алгоритма.

Добавили параметры:

outlier\_limit=300000, # с какой суммы начинаем учитывать терминалы с выбросами в функции штрафов

koef\_outlier = 0.1): # коэф. при параметре "терминалы с фактами выбросов" в функции штрафов

И мотивировали алгоритм почаще заезжать в такие терминалы через функцию штрафов и koef\_outlier .

2. В случае, если факт прироста все же сильно превысил наш прогноз, и точку мы не обсужили (на конец дня случилось переполнение),

на след. день она попадает в список ОБЯЗАТЕЛЬНЫХ.

3. Добавили расчет финансовых издержек на факте уже после отработки части "определения оптимального маршрута"

4. Стали сохранять больше промежуточных результатов.

ОСНОВНЫЕ методы :

.balance # баланс

.all\_terms\_set # все терминалы

.bags # кейсы, когда в точку обязательно нужно было заехать, но алгоритм не направил туда броневик (не заехали)

.costs\_without\_vehicle\_sum # общие издержки без броневиков

.cycle\_time # время отработки на 1 дне (соотв на 91 дне - cycle\_time\*91)

.day\_df\_dict # промежуточные данные по дням

.distance\_matrix # матрица времени/расстояния для графа

.num\_veh # число машин

.over\_balance\_limit # терминал не был переполнен, но к концу дня переполнился (на след. день попадает в обязательные)

.predictions # прогнозы прироста от модели

.problem\_tid\_lst # терминалы, определенные как "нестабильные", с выбросами

.route\_vehicle # маршруты броневиков по дням и машинам (ВАЖНО!! терминалы - это "порядковый номер -1", не ID)

.terms\_dict # терминалы, которые обсуживаем по дням

.time\_vehicle # время в пути (с обслуживанием) броневиков

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**cash\_predict.py** - результат работы над подзадачей "прогноз приростов наличности в терминалах".

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Класс CashPredict реализует модель прогноза временных рядов SARIMA для всех представленных в данных банковских терминалов. Для каждого терминала строится своя авторегрессионная модель. На этапе обучения модели (методы .fit\_model, .fit\_models) проводится кросс-валидационный поиск наилучший модели для данного ряда из пула моделей с различным сочетанием параметров сезонной авторегрессии (p=[1..5], d=[0..2], q[1..5])(P[1..5], D[0..1], Q[1..5], S=7) по критерию Акаике. На этапе исследования данных была проверена стационарность исходных данных - максимальной порядок дифференцирования до достижения стационарности составил d=2 (подробности в файле cash forecast.ipynb).

Обучение моделей производится только на train наборе данных (диапазон всех данных минус self.days\_for\_test=30 при инициировании объекта по умолчанию)

ОСНОВНЫЕ методы :

.fit\_model(tid) # если модель не обучена, запускает процесс обучения и выбора наилучшей модели на имеющемся train наборе данных для одного tid

.fit\_models() # запускает процесс обучения и выбора наилучшей модели для всех терминалов

.predict(id) # формирует прогноз для данного tid, если модели нет, сначала выполняет .fit

.predict\_all() # возвращает результат .predict по всем терминалам

.predict\_out(tid, days\_out\_of\_data) # возвращает результат прогнозирования за пределами предоставленных данных (train + test), при этом модель не переобучается на всех данных, в комментариях есть реализация данного метода, который переобучает модель на всех данных, не производя при этом кросс-валидационного выбора (то есть параметры авторегрессии остаются теми, которые были определены как наилучшие на train данных), соответственно, такой метод будет отрабатывать достаточно быстро и лучше подходит для эксплуатации решения

.calc\_error(tid) # вычисление ошибки прогнозирования - rmse, результаты теста Льюинга-Бокса на остатках

.show\_prediction(tid) # выводит график исходного ряда и прогнозных значений (на train данных результат обучения, на test - результат прогноза)

.show\_acf(tid) # выводит графики ACF/PACF для остатков раздельно для train (показывает качество полученной модели) и для test (показывает качество генерализации выбранного подхода)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ФАЙЛЫ ДЛЯ ЗАПУСКА**

opt\_route\_1\_run.py - запуск расчетов под opt\_route\_1.py и сохранение в формате pickle

**opt\_route\_2\_run.py - запуск расчетов под opt\_route\_2.py и сохранение в формате pickle**

cash\_predict\_fit\_new\_run.py - создание нового объекта класса CashPredict без моделей, позволяет самостоятельно пройти весь путь получения моделей

cash\_predict\_read\_pkl\_run.py - загрузка pickle объекта класса CashPredict, содержащего все модели, обученные на стадии подготовки решения

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**НОУТБУКИ**

opt\_route\_finder\_fact.ipynb - подгрузка решения, рассчитаного на ресурсах СберКлауд с возможностью потрогать результаты + "завернутый" в ноутбук класс OptRouteFinder(opt\_route\_2.py)

opt\_num\_vehicle\_finder.ipynb - подгрузка решения, рассчитаного на ресурсах СберКлауд с возможностью потрогать результаты + "завернутый" в ноутбук класс OptRouteFinder(opt\_route\_1.py)

car\_count\_ideas.ipynb - работа над задачей "поиск оптимального кол-ва броневиков" в разрезе идей (если хочется понять поток мыслей)

outliers.ipynb - анализ выбросов в приростах и в прогнозах

report.ipynb - формирование отчёта

forecast.ipynb - весь анализ данных, обоснование выбранного способа прогнозирование, исследование альтернативных подходов к решению, сравнение результатов, визуализация основных результатов

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ДАННЫЕ**

data - папка с данными для расчетов

CloudRun1 - успешно отработавшие на этапе определения числа броневиков алгоритмы

CloudRun5 - успешно отработавшие на этапе построения маршрутов с прогнозными данными алгоритмы

\* по запросу выгрузим кейсы, не показавшие хорошие результаты

**CCЫЛКА НА КРУПНЫЕ ФАЙЛЫ В ZIP:** <https://drive.google.com/drive/folders/1Ss_Vh3_UIWJ1G88YBHa_DKNEj7kNEW9j?usp=sharing>

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ВАЖНЫЕ МОМЕНТЫ**

Запускали расчёты на ресурсах СберКлауд на cycle\_time=480 (т.е. 12 часов для всего периода).

Уместить все необходимые для объезда точки в 4 машины не вышло.

Для 5 машин сработали параметры:

predict\_cash\_trust=0.9, koef\_priority\_0 = 10000000, koef\_step\_func=1 И НЕСКОЛЬКО ВАРИАНТОВ koef\_nes\_degree - 2.6, 2.8, 3.4, 3.6

Далее только эти параметры использовали для тюнинга класса на прогнозных данных +

outlier\_limit - 300000, 500000 и koef\_outlier - 0, 0.1, 0.2

В результате отработали хорошо (в рамках 5 машин и 12ч рабочего дня без пропуска обязательных точек) только варианты с koef\_outlier = 0.

Таким образом, обязывая машины заезжать в терминалы с выбросами,

(при чем с т.зр незнания завтрашнего дня мы не можем сказать будет выброс завтра или через месяц) мы не умещаемся в 5 машин.

В РЕЗУЛЬТАТЕ БЫЛ ВЫБРАН ВАРИАНТ c наименьшим коэффициентом koef\_nes\_degree, т.к. его увеличение "перетягивает" на себя и делает менее значимым

влияние затрат на обслуживание и фондирование в функции штрафов.

predict\_cash\_trust=0.9,

num\_veh = 5,

cycle\_time=480,

koef\_priority\_0 = 10000000,

koef\_nes\_degree=2.6,

koef\_step\_func=1,

koef\_costs\_without\_vehicle=10000,

outlier\_limit= 500000,

koef\_outlier = 0.

ВСЕ ПАРАМЕТРЫ МОЖНО ПРОВЕРИТЬ при обращении к классу после загрузки соответствующего pickle файла.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**РАЗНОЧТЕНИЯ в отчете и ТЗ**

\*\*ОТЧЕТ - Лист "остатки на конец дня" показывает сумму остатков в устройствах в разрезе дат на конец дня.

Т.е. в случае, если устройство было инкассировано, в ячейке точно должен быть 0

\*\*ТЗ

Принимаем, что остаток в терминале увеличивается на сумму оборота в терминале разово.

Т.е. утром известны остатки в терминалах и они не меняются за время работы бригад броневиков. Остатки на утро = остатки утра вчера + обороты вчера.

Например, для терминала 406136 на утро 02.09 остаток = 250000, если 01.09 терминал не обслужили и = 90000, если терминал 01.09 обслужили.