 About and Documentations

 Модель кредитного риск-менеджмента.

Проблема.

Техническое задание.

Виртуальное окружение.

Логирование.

Структура проекта.

Функции в модулях.

Новые признаки.

Особенности подготовки данных.

Моделирование и обучение pipeline.

Предсказания.

 Проблема, которую предстоит решить:

В рамках проекта необходимо решить задачу — оценить риск неуплаты клиентом по кредиту.

Сервис на базе обученной модели, позволит банку или другой кредитной организации оценить текущий риск по любым выданным займам и кредитным продуктам. И с большой долей вероятности предотвратить неисполнение кредитных обязательств клиентом. Таким образом, банк меньше рискует понести убытки.

 Техническое задание:

1. Оценить важность признаков.

2. Сгенерировать новые признаки.

3. Собрать итоговый датафрейм. состоящий из признаков для обучения модели.

4. Сделать предсказания на тестовом датасете.

5. Подготовить автоматизированный пайплайн, который по вызову fit будет готовить данные и обучать модель, а по вызову predict — делать предсказания на заданном наборе данных.

6. Обучить пайплайн подготовки данных и обучения модели и сохранить результат обучения в бинарном формате pickle.

 Виртуальное окружение:

Проект реализован в PyCharm и запускается в виртуальном окружении. Зависимости отлаженного проекта находятся в файле requirements.txt. Запись requirements.txt производится - modules.my\_requirements

 Логирование:

В проекте настроено логирование каждого этапа. Запись логов производится в директорию: data.logs.

 Структура проекта:

 - dags # модуль AirFlow.

 - data # директория данных.

 - - parquet # директория начальных данных.

 - - models # директория модели.

 - - pipe # директория обученного pipeline.

 - - target #  директория целевой переменной.

 - - to\_predict # директория файлов запроса.

 - - train # директория подготовленых данных.

 - - predictions # директория предсказаных запросов.

 - - logs # директория логов.

 - modules # модули проекта.

 - - main # основной модуль проекта.

 - - module\_api # модуль для api запросов.

 - - my\_requirements # модуль записи зависимостей.

 - - preprocessing # модуль обработки данных.

 - - take\_parquet # модуль агрегации данных.

 - jupyter

 - - scoring\_credit\_default # ноутбук с исследованиями.

Запуск проекта осуществляется в main.pipeline.

 Функции модулей:

main:

def compute\_class\_weights - Функция для вычисления весов классов.

def objective - Функция подбора гиперпараметров и лучшей модели.

def pipeline - Основная функция для подготовки данных и обучения лучшей модели.

def predictor - Функция предсказаний.

preprocessing:

def new\_features - Создание новых признаков.

def high\_corr - Удаление признаков с низкой корреляцией.

def important - Удаление признаков с нулевой значимостью.

def duplicates - Удаление дубликатов большего класса.

def preparations - Запуск обработки данных.

def aggregator - Функция агрегирования данных запроса для предсказания.

def preparations\_predict - Подготовка данных для предсказания.

def fileout - Создание имени файла для записи предсказания.

def take\_data\_to\_predict - Подготовка тестового файла для предсказаний.

Новые признаки:

1. Отношение планового количества дней до закрытия кредита к количеству просроченых платежей - 'planned\_to\_zero\_loans\_530'.

2. Сумма просроченых платежей от 30 до 60 дней и от 60 до 90 дней - 'suspended\_loans\_3060\_6090'.

3. Процент использования кредита при отсутствии просрочек - 'utilization\_no\_overdue'.

4. Сумма просрочек по каждому типу кредита - 'overdue\_by\_credit\_type'.

5. Отношение максимальной просрочки к использованию кредита - 'max\_overdue\_to\_utilization'.

6. Сумма кодов платежей - 'enc\_payment\_sum'.

 Особенности подготовки данных:

1. Первоначальные данные собираются и агрегируются из 12 файлов parquet, к ним добавляется целевая переменная.

2. Удаление признаков с низкой корреляцией.

3. Для удаления признаков с нулевой значимостью применяются возможности библиотеки shap.

4. Удаление дубликатов большего класса.

 Конвейер и моделирование:

Для моделирования применяются:

XGBClassifier

CatBoostClassifier

LGBMClassifier

Для выбора лучшей модели и подбора гиперпараметров применяются возможности библиотеки optuna.

Предсказания:

Для предсказаний применяется функция main.predictor.

def predictor:

Принимает на вход:

данные для предсказания,

в случае df=None - данные для предсказания берутся из - data.to\_predict, в случае нескольких файлов, данные объединяются в один запрос.

Файл с предсказаными значениями записывается в - data.predictions.

На выход:

Подается df - содержащий id клиента и процентную вероятность дефолта.

Спасибо за внимание.