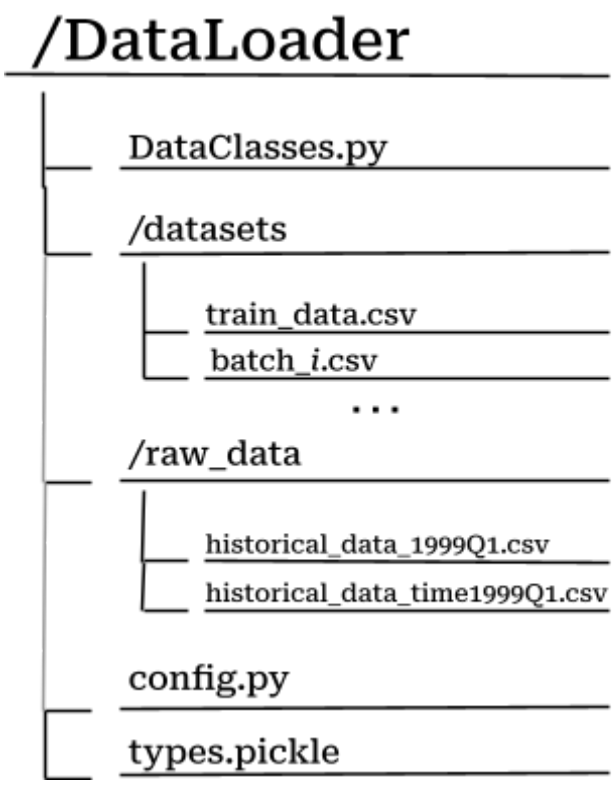
Модуль по работе с данными

Структура модуля



Содержание

>> **class DataLoader**

Предоставляет функционал для работы с данными: обработка исходных наборов данных, разделение на батчи, эмуляция потоковости. Предполагает использование гиперпараметров, хранящихся в config.py:

```
from DataLoader.DataClasses import DataLoader
from DataLoader.config import DATA_LOADER_PARAMS

dl = DataLoader(DATA_LOADER_PARAMS)
```

Публичные методы:

- create_datasets(verbose: Int) - обрабатывает исходные данные: объединяет статическую и динамическую таблицы и сохраняет разделенный на батчи датасет в директорию datasets

- get_data(verbose: Int) - при вызове возвращает следующий батч данных, начиная с тренировочного, в процессе происходит оценка качества данных и их очистка. После вызова функции создаются файлы stats.picke - сохраненные статистики, и stats.txt - отчет о качестве данных.

  Если указан verbose=1, то отчет также выводится в консоль, а если verbose=2, то также выводятся логи промежуточных этапов получения следующего батча.

Реализованные этапы:

- разделение исходного набора на батчи и эмуляция потока
- разработка хранилища сырых данных: файловая система
- расчет метапараметров
- создание конфигурационного файла (.py) с гиперпараметрами сбора:
  - year: год разделения на тренировочную, и дополнительные батчи
  - num_batches: количество дополнительных батчей
- интеграция с несколькими источниками данных (статические + динамические)
- система логирования (вручную), обработка ошибок (проверка корректности гиперпараметров сбора, количества вызовов get_data и количества батчей и тд)

- оценка и хранение показателей качества данных
- базовая очистка данных на основе порогов допустимых значений качества
- генерация отчетов о качестве данных

>> **class DataQualityEvaluator**

Дополнительный класс для оценки качества, используется в DataLoader-е, но при необходимости, можно использовать отдельно.
Функционал класса:
- оценка completeness
- оценка validity
- генерация отчета (сохраняется в stats.txt в папке исполняемой программы)
- базовая очистка данных (метод fix_data)

>> **datasets**

Директория, генерируемая при вызове метода create_datasets класса DataLoader, содержит датасет для обучения, и батчи для эмуляции потоковости:
- train_dataset.csv
- batch_{i}.csv

>>> **raw_data**

Директория, содержащая сырые данные: две таблицы - со статическими данными, и с динамическими.

>>> **сопутствующие файлы**

- *types.pickle* - содержит описания признаков и их типов

- *stats.pickle* - генерируется при вычислении data quality, содержит соответствующие статистики
- *stats.txt* - отчет о качестве данных, также генерируется при вычислении data quality

API Reference

- **class DataLoader** - предоставляет методы для извлечения и управления данными

Параметры:

- *year_to_split* (*int*, optional): Год, используемый для разделения данных на обучающую выборку и батчи. По умолчанию: `2010`.

- num_batches (*int*, optional): Количество батчей для обработки. По умолчанию: `10`.

Методы:

“Приватные”:

_extract(verbose: Int) -> pd.DataFrame

Извлекает данные из таблиц, объединяет их в один DataFrame, разделяет на батчи и сохраняет.

Параметры:
- *verbose* (*int*, optional): Уровень детализации вывода, по умолчанию *0*

Возвращаемое значение:
- *None*

_add_batch() -> pd.DataFrame

Считывает следующий по счету батч данных, добавляет целевую переменную (интервал между взятием и закрытием кредита)

Параметры:
—

Возвращаемое значение:
- *pandas.DataFrame*: Обновленный набор с добавленными данными.
- *None*: Если шаг превышает количество доступных батчей.

create_time_parameter(df: pandas.DataFrame) -> pandas.DataFrame

Создает временную метку данных (признак ‘time’), рассчитывается как интервал между взятием кредита и последним наблюдением.

Параметры:
- *df (pandas.DataFrame)*: *И*сходный набор данных.

Возвращаемое значение:
- *pandas.DataFrame*: Набор данных с добавленным временным признаком.

_first() -> pandas.DataFrame

Загружает и обрабатывает набор данных для обучения

Параметры:
- *verbose* (*int*, optional): Уровень детализации вывода, по умолчанию *0*

Возвращаемое значение:
- *pandas.DataFrame*: набор с обучающими данными.

_step(df: pandas.DataFrame, verbose: Int) -> pandas.DataFrame

Обновляет данные, оценивает их качество и выполняет очистку.

Параметры:
- *verbose* (*int*, optional): Уровень детализации логов. По умолчанию: *0*.
  - *0*: Без вывода.
  - *1*: Основные статистики.
  - *2*: Подробный вывод.

Возвращаемое значение:
- pd.DataFrame: Обновленный и очищенный DataFrame.

Примечание: *И*спользует класс *DataQualityEvaluator* для оценки качества данных.

“Публичные” методы:

create_datasets(verbose: Int) -> None

Создает обучающие и тестовые наборы данных и сохраняет в директории datasets.

Параметры:
- *verbose* (*int*, optional): Уровень детализации логов. По умолчанию: *0*.
  - *0*: Без вывода.
  - *1*: Основные статистики.
  - *2*: Подробный вывод.

Возвращаемое значение: *-None*

get_data(verbose: Int) -> pandas.DataFrame

Возвращает следующий батч, очищенный и обработанный.

Параметры:
- *verbose* (*int*, optional): Уровень детализации логов. По умолчанию: *0*.
  - *0*: Без вывода.
  - *1*: Основные статистики.
  - *2*: Подробный вывод.

Возвращаемое значение:
- pd.DataFrame: следующий батч

Примечание: Использует класс *DataQualityEvaluator* для оценки качества данных.

- **class DataQualityEvaluator** - предоставляет методы для оценки качества данных, включая проверку полноты, валидности и своевременности

Методы:

completeness(df: pandas.DataFrame) -> Dict[str, float]

Оценивает полноту данных (процент пропущенных значений).

Параметры:
- *df (pandas.DataFrame)*: набор данных для анализа.

Возвращаемое значение:
- Dict[str, float]: Словарь со значениями
  - `"*full*"`: Общий процент пропущенных значений.
  - `"*cols_max*"`: Максимальная доля пропущенных значений в столбцах.
  - `"*rows_max*"`: Максимальная доля пропущенных значений в строках.

validity(df: pandas.DataFrame) -> Dict[str, float]

Проверяет данные на корректность значений.

Параметры:
- *df (pandas.DataFrame)*: DataFrame для анализа.

Возвращаемое значение:
- *Dict[str, bool]*: Словарь, где ключи — названия столбцов, а значения — результаты проверки:
  - *True*: если данные валидны
  - *False*: иначе
  - *None*: если данного столбца нету в DataFrame

check_values(a: pandas.DataFrame) -> List[float]

Проверяет значения в столбцах на соответствие допустимым диапазонам.

Параметры:
- *a (pandas.DataFrame)*: DataFrame для анализа.

Возвращаемое значение:
- *List[float]*: Список с результатами проверки для каждого столбца.

make_stats(df: pandas.DataFrame, p:Bool) -> Dict[str, Any]

Создает статистику качества данных и сохраняет её в файлы

Параметры:
- *df (pd.DataFrame)*: DataFrame для анализа
- *p* (*bool*, optional): вывод статистики в консоль. По умолчанию: *False*.

Возвращаемое значение:
- *Dict[str, Any]*: Словарь с метриками качества данных.

fix_data(a: pandas.DataFrame) -> Tuple[pandas.DataFrame, Int]

Очищает данные, удаляя столбцы с низким показателем полноты или невалидными значениями.

Параметры:
- *a (pandas.DataFrame)*: DataFrame для очистки.

Возвращаемое значение:
- *Tuple[pandas.DataFrame, int]*: Очищенный DataFrame и количество удаленных столбцов.

Пример использования:

Пересоздание датасета и 3 итерации получения следующей порции данных:

```
from DataLoader.DataClasses import DataLoader
from DataLoader.config import DATA_LOADER_PARAMS

data_loader = DataLoader(DATA_LOADER_PARAMS)
data_loader.create_datasets(verbose=1)
for i in range(3):
   next_batch = data_loader.get_data()
   # далее батч подается в модель
   # --- реализация этапов 3-5 ---
```