DATASKAI

Фреймворк для AI/ML-проектов

(R&D-версия)

Руководство пользователя Junior Data Scientist

Skoltech

Веб-страница проекта: http://dataskai.com

Содержание

1 Роли пользователей

В настоящем руководстве мы придерживаемся разделения всех дата саентистов (DS далее в этом документе), связанных с проектом проектом, на три группы:

Junior DS

Типичные задачи:

Выполнить исследовательский анализ данных (EDA),

создать и представить работу над моделями для задач DS.

Middle DS

Типичные задачи:

– проверка моделей,

– создание и проверка экстракторов функций,

– развертывание служб DATASKAI.

Senior DS

Типичные задачи:

– управление определениями задач DS,

– создание и проверка моделей предметной области,

– отслеживание результатов задач.

2 Типичные задачи Junior DS

Для поддержания соревновательной модели, применяемой в data science, большинство задач Junior DS выполняются на конкурентной основе. Этот способ решения задач по data science, используемый многими платформами, предлагает соревнования по машинному обучению (Kaggle и тому подобное).

Базовый цикл решения задачи состоит из следующих задач:

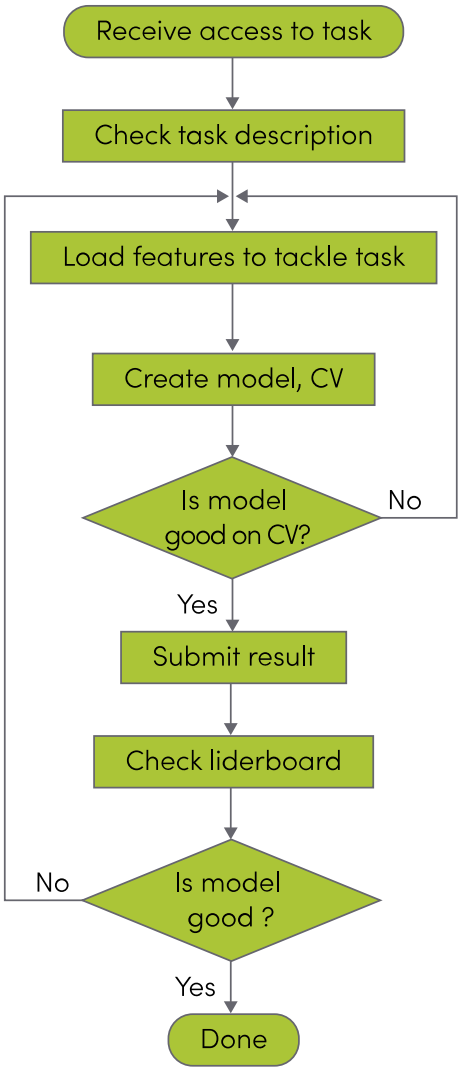

Рис. 1. Типичные задачи для Junior DS

2.1 Доступ к контейнеру с задачей

Вся работа, выполняемая специалистом по обработке данных, должна выполняться в предоставленном док-контейнере. Обратитесь к своему старшего коллеге для предоставления ссылки на него.

Получив ссылку, вставьте ее в свой браузер. Стандартный блокнот Jupyter будет загружен и отображен в браузере.

Рекомендации

Мы рекомендуем вам использовать переадресацию портов с командой «autossh» на вашем локальном компьютере для подключения к вашим контейнерам. Это даст некоторые преимущества, а именно:

– Автоматически подключаться к ноутбукам Jupyter в случае потери соединения;

– Нет необходимости запоминать IP-адреса вашего внутреннего сервера.

Используйте флаги -L -N для команды ”autossh”:

```
autossh -N -L 8888: some_serv_ip: 8989 some_serv_ip
```

2.2 Файловая структура проекта

Помните, что работа над задачей в проекте data science - это длительная работа, выполняемая совместно с вашей командой, поэтому вы должны хорошо организовать свой проект. Файловая структура проекта по умолчанию позволяет хранить весь необходимый код для задачи и проекта в одном месте:

```
project_name
├── .git
├── configs
├── dockerfiles
├── data
│ ├── external
│ ├── processed
│ │
│ └── raw
│
├── documents
├── modules
│ ├── subject_domain
│ ├── evaluation_tools
│ ├── metrics
│ └── feature_extractors
├── notebooks
├── scripts
├── tools
├── .env
└── README.md
```

2.3 Детализация структуры

project_name – название вашего проекта

.git – стандартный каталог репозитория git

configs – файлы конфигурации, которые будут использоваться в качестве отправной точки для инициализации инструментов DATASKAI

dockerfiles – docker-файлы для контейнера проекта DS

data – папка для данных

external – папка для небольших объемов внешних данных (каталоги, ссылки, данные), прикрепленных к git

processed – папка для обработанных данных, полученных из raw и загруженных с raw_data_loader (должен использоваться как папка tmp для обработанных данных, фактические данные здесь хранить не следует)

raw – папка для необработанных данных, загруженных извне и используемых в проекте (должна использоваться как папка tmp для обработанных данных, фактические данные здесь хранить не следует)

documents – важные для проекта pdf-файлы или документы других форматов

modules – модули Python, используемые в проекте

Subject_domain – модели предметной области для проекта cookiecutter.project_name, написанного на python

evaluation_tools – инструменты DATASKAI

metrics – Python-код метрик, используемый metrics_service для расчета метрик для проекта

feature_extractors – Feature_extractor-модули текущего проекта

notebooks – ноутбуки с некоторыми исследованиями и моделями, создаваемыми DS в ходе работы

scripts – различные вспомогательные сценарии

tools – бинарные инструменты, необходимые для данных

.env – файл окружения DATASKAI с переменными для клиента и сервера

README.md – файл помощи проекта

Папка project_name контейнера также содержит стандартную директорию .git с внутренностями git, поэтому проект по науке о данных также является репозиторием git.

2.4 Каталог ноутбуков Jupyter

Мы рекомендуем вам использовать вложенные папки с вашим именем в каталоге ноутбука проекта:

```
project_name - имя вашего проекта
├── notebooks - каталог для данных
│ ├── elon.musk
│ ├── andrew.ng
│ └── stanislav.semenov
...
```

Также мы рекомендуем вам пронумеровать свои записные книжки и добавить к их названию как минимум два числа, например:

```
01__Simple_EDA.ipynb
02__First_model.ipynb
03__First_CV_model.ipynb
...
```

Такое наименование поможет вам и вашим коллегам перемещаться по каталогу вашей записной книжки.

2.5 Загрузка задачи в ноутбук

Создайте первый ноутбук внутри своего каталога и загрузите в него класс TaskLoader:

```
import sys
sys.path.append('../../modules/evaluation_tools/')
from task_loader import TaskLoader
```

После загрузки TaskLoader вы можете получить доступ к определению задачи через конфигурацию задачи по умолчанию и имя задачи:

```
TASK_CONFIG = json.load(open('../../configs/
tasks_default_config.json'))

TASK_NAME = 'raw_data__aero__fw_classification_v1'

task_loader = TaskLoader(TASK_NAME, TASK_CONFIG['mongo_config'])
```

Чтобы эти команды выполнялись, проект должен содержать конфигурацию по умолчанию и определение задачи.

2.6 Отображение описания задачи

Поскольку объект task_loader создан, вы можете проверить описание задачи с помощью встроенной функции display_description_in_jupyter:

```
task_loader.display_description_in_jupyter()
```

Выполнение этой команды в вашем блокноте jupyter приведет к отображению описания задачи в markdown:

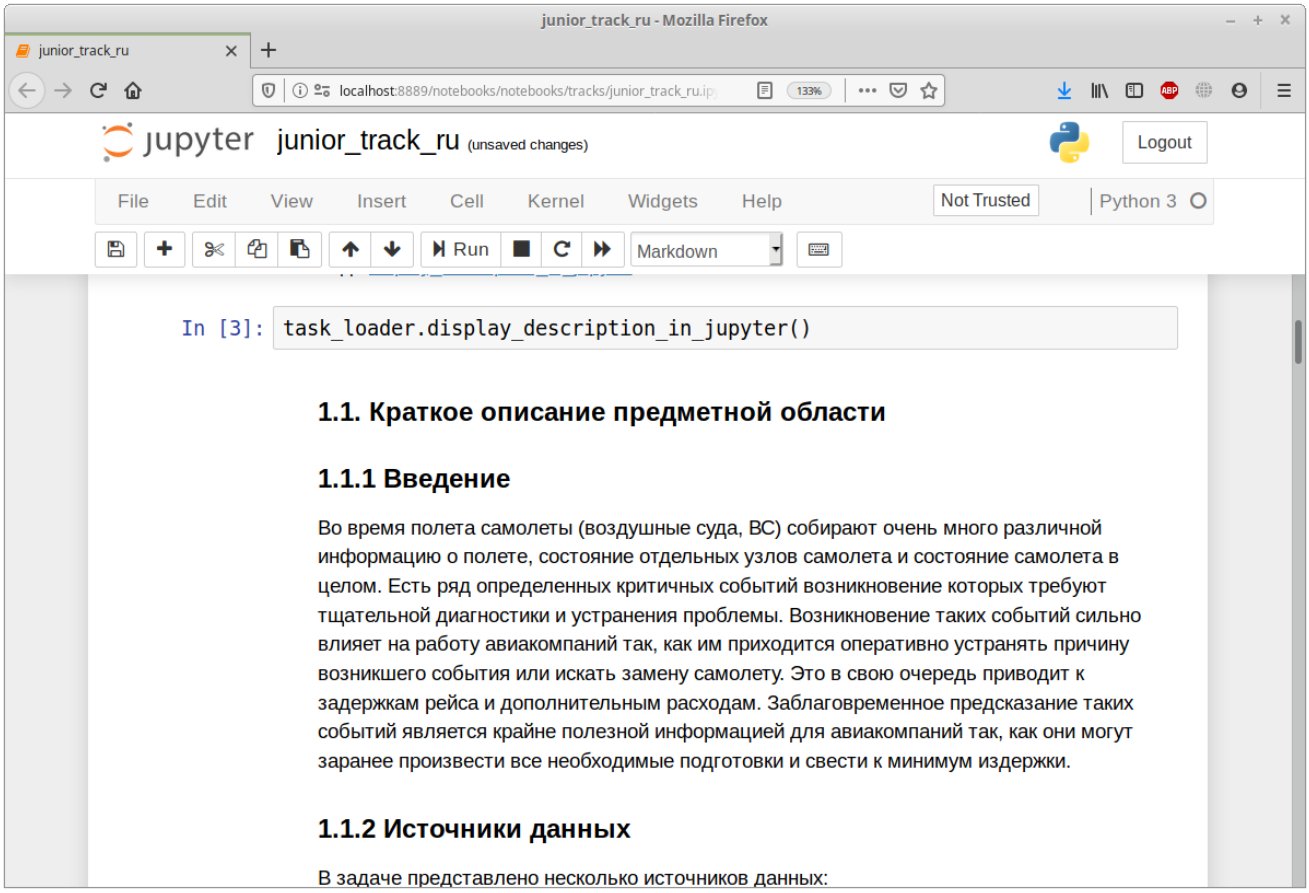

Рис. 2. Описание задачи

2.7 Загрузка необработанных данных

Чтобы загрузить необработанные данные для анализа (например, анализ EDA и т. Д.), Используйте компонент raw_data_loader из task_loader:

```
raw_data_loader = task_loader.raw_data_loader
```

Чтобы получить список всех доступных данных из raw_data_loader, используйте этот фрагмент кода:

```
feature_records = [x['feature_manager_config']
['feature_records'] for x in raw_data_loader.list_configs()]

record_names = [x['name'] for x in feature_records[0]]
record_names
```

Переменные record_names теперь содержат имена всех доступных источников данных в текущей задаче.

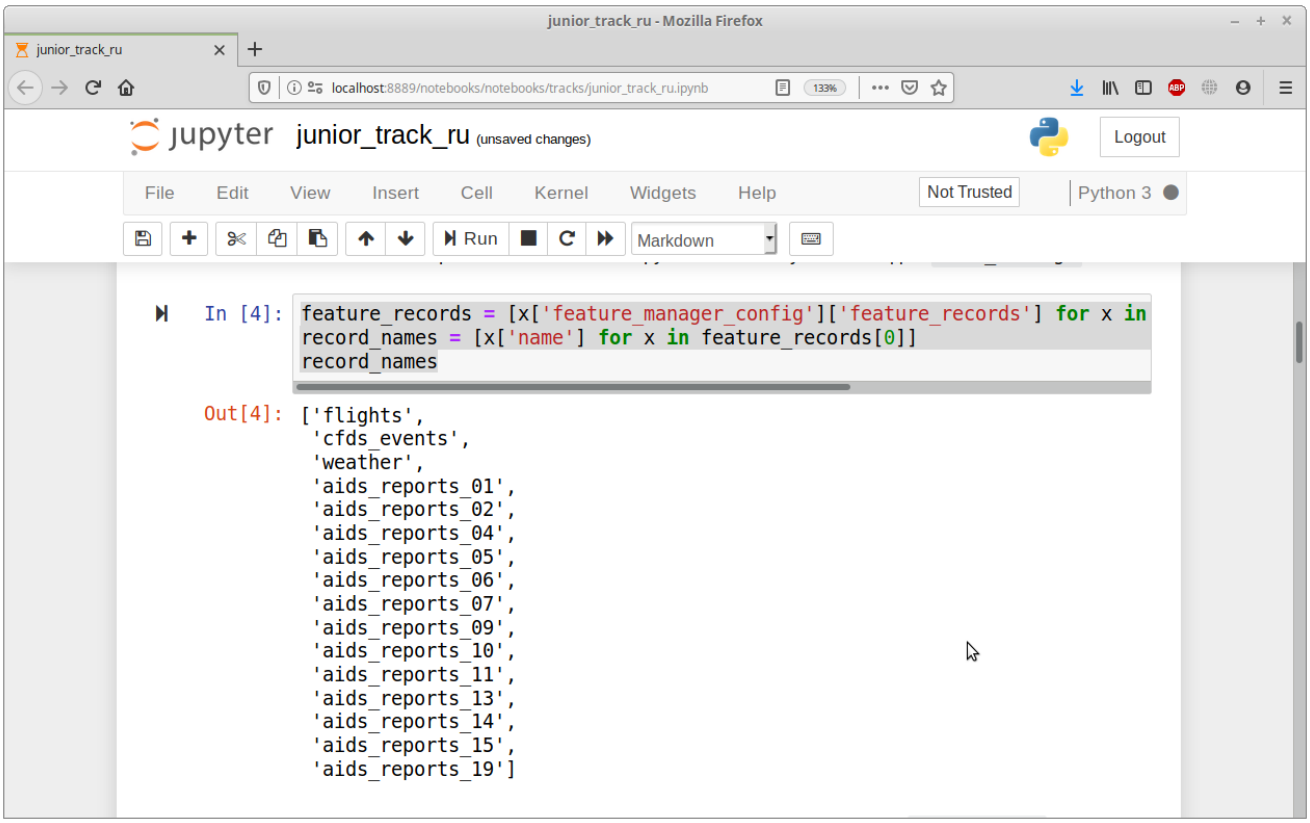

Рис. 3. Список источников необработанных данных

Чтобы получить необработанные данные из именованного источника, используйте:

```
some_raw_data = raw_data_loader.load_data_one_record
('some_available_source')
```

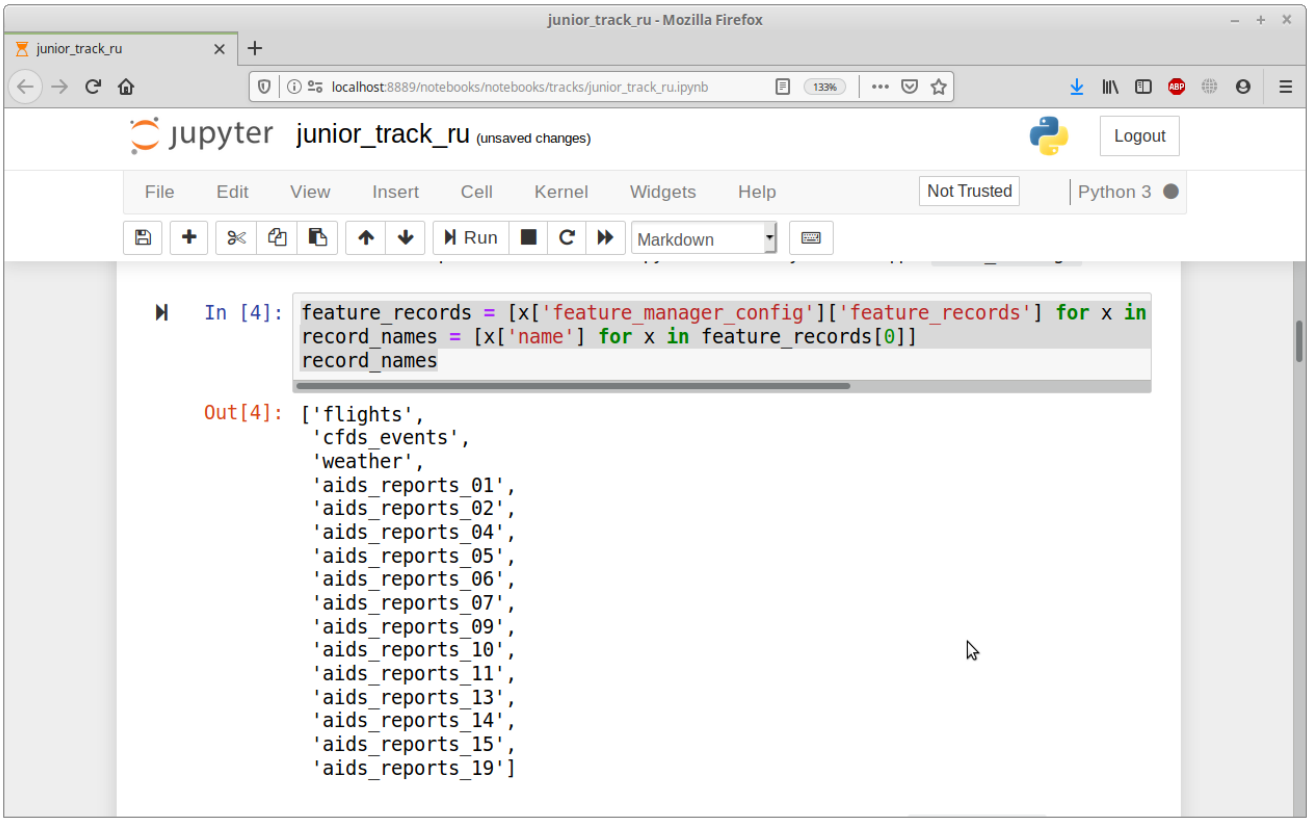

Рис. 4. Загрузка необработанных данных в ноутбук jupyter

Внимание

Значения в ячейках в фрейме необработанных данных обычно загружаются как пустая строка (объекты). Это будет занимать много памяти, не забудьте удалить ненужные кадры данных.

2.8 Загрузка функций DataFrame

Чтобы загрузить фрейм данных, используйте инструмент feature_loader:

```
feature_loader = task_loader.feature_loader
```

Этот инструмент является клиентом для функции хранилища / local_fs_storage.

Это позволяет вам получить доступ к любой доступной функции и подмножеству функций через интерфейс фильтрации.

Для этого вам нужно сначала создать функцию фильтрации, а затем передать ее в качестве аргумента методу load_features в качестве именованного аргумента feature_selector:

```
def feature_filtering_func(feature_name):
    return feature_name.startswith('cur_val__aids_rep_01')
    or feature_name.startswith('event__cfds')

features = feature_loader.load_features(
feature_selector=feature_filtering_func)
```

Возвращенные функции хранятся в одном кадре данных Pandas, разделение на части train / test отсутствует.

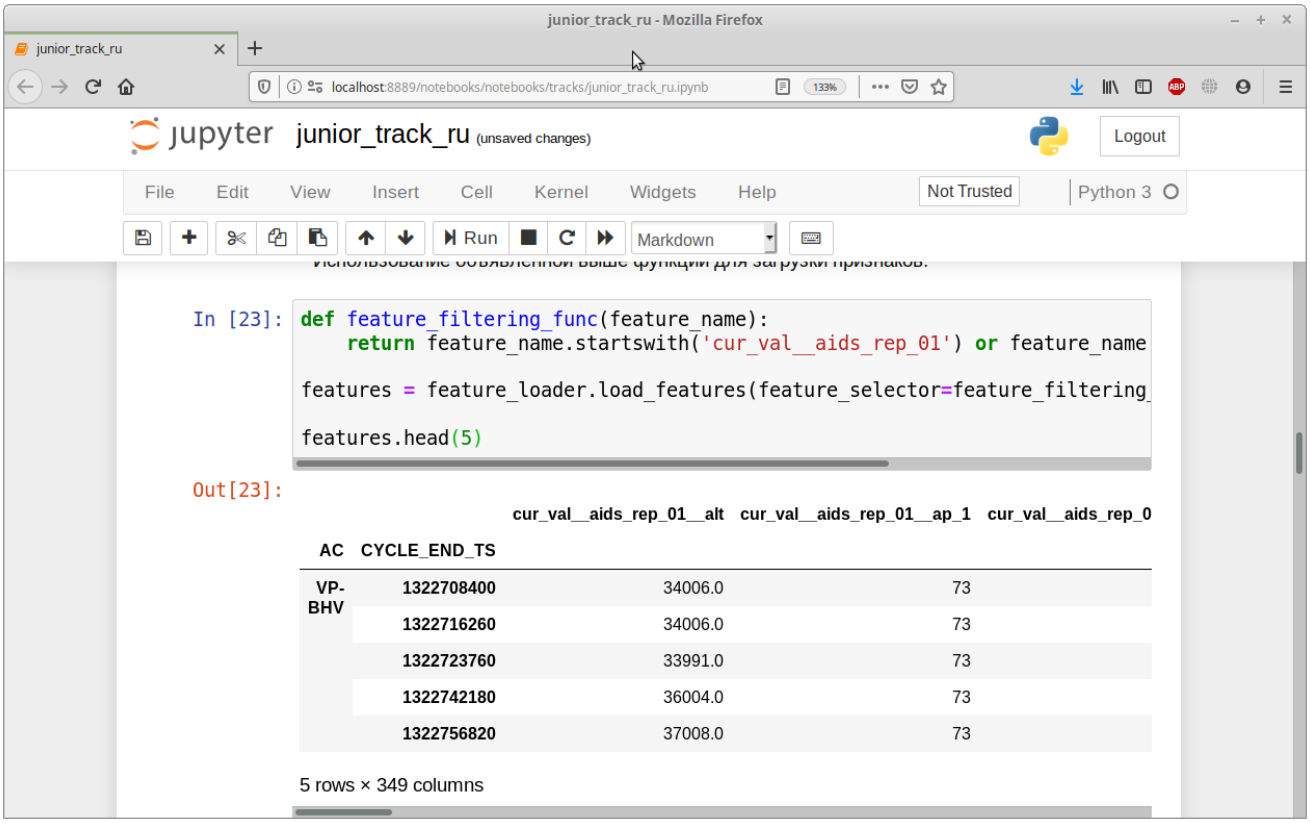

Рис. 5. Загрузка функций с помощью feature_loader

Один большой возвращаемый фрейм данных – не случайность, а результат наиболее распространенной ошибки: многие новички в DS продолжают делать ошибки при применении преобразований для обучения детали и забывают (или теряют) преобразования для тестовой детали и попадают в проблемы отладки.

Примечания

1. Загрузчик функций также поддерживает постоянный индекс.

2. Загрузчик компонентов применяет типы в соответствии с конфигурацией задачи, поэтому вам не нужно беспокоиться о возвращаемых типах данных и потреблении памяти (оно уже уменьшено).

3. Загрузчик объектов также отслеживает ваши последние вызовы метода load_features и создает кэш на диске с общими загруженными фреймами данных.

2.9 Соглашения о присвоении имен

Мы рекомендуем вам использовать такое соглашение об именах функций:

```
[oper_N]__ ... __[oper_2]__[oper_1]__[src]__[parameter]
```

Имя следует читать справа налево, с двумя символами подчеркивания в виде разделителей. Оно показывает, что эта функция получена в следующем порядке: взять параметр [параметр] из источника [src], применить к нему [oper_1], затем взять результат и применить к нему [oper_2] и продолжать этот процесс до тех пор, пока не будет применен последний [oper_N] ,

Например:

```
cur_val__aids_rep_01__alt
```

Следует читать как «взять параметр alt из источника aids_rep_01 и применить к нему операцию cur_val».

Заметка

Обратите внимание, что имя функции является уникальным и распознается системой как одна функция, независимо от того, какая конфигурация записи содержит ее. В случае, если более чем одна конфигурация записи содержит имя функции, не существует строгого правила для определения, из какой функции конфигурации записи будет считываться.

Внимание

Всегда приводите имена своих функций в соответствие с действующим соглашением об именах, которое позволяет вам и вашим коллегам:

– Эффективно искать и использовать функции для решения задач DS;

– Создавать понятную систему именования функций для легкой интерпретации и общения с экспертами;

– Создавать автоматические сервисы по интерпретации и объяснению особенностей для экспертов и менеджеров.

2.10 Загрузка таргетных фреймов данных

Чтобы получить ответы для контролируемого обучения и фрейм данных, чтобы заполнить ответы для тестового набора, вам нужно взять объект target_loader:

```
target_loader = task_loader.target_loader
```

Затем требуется вызвать метод get_train_test:

```
y_train, y_test = target_loader.get_train_test()
```

Этот метод возвращает два кадра данных Pandas. Первый содержит ответы для контролируемого обучения, используйте его для создания схем проверки и моделей обучения: Использование target_loader для получения y_train и y_test

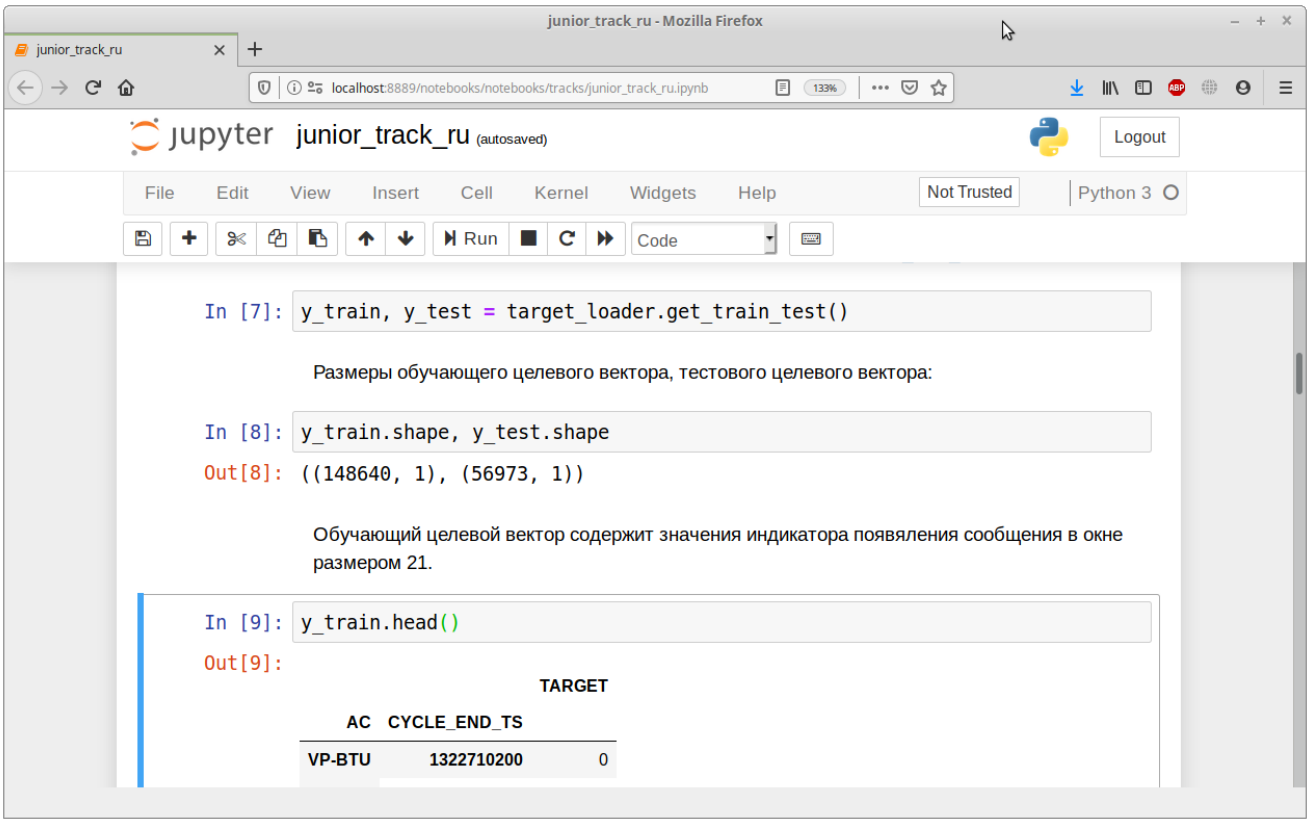

Рис. 6. Использование target_loader для получения y_train и y_test

Второй фрейм данных используется в качестве заполненной формы для ответов на тестовом наборе, поэтому значения целевой переменной будут удалены:

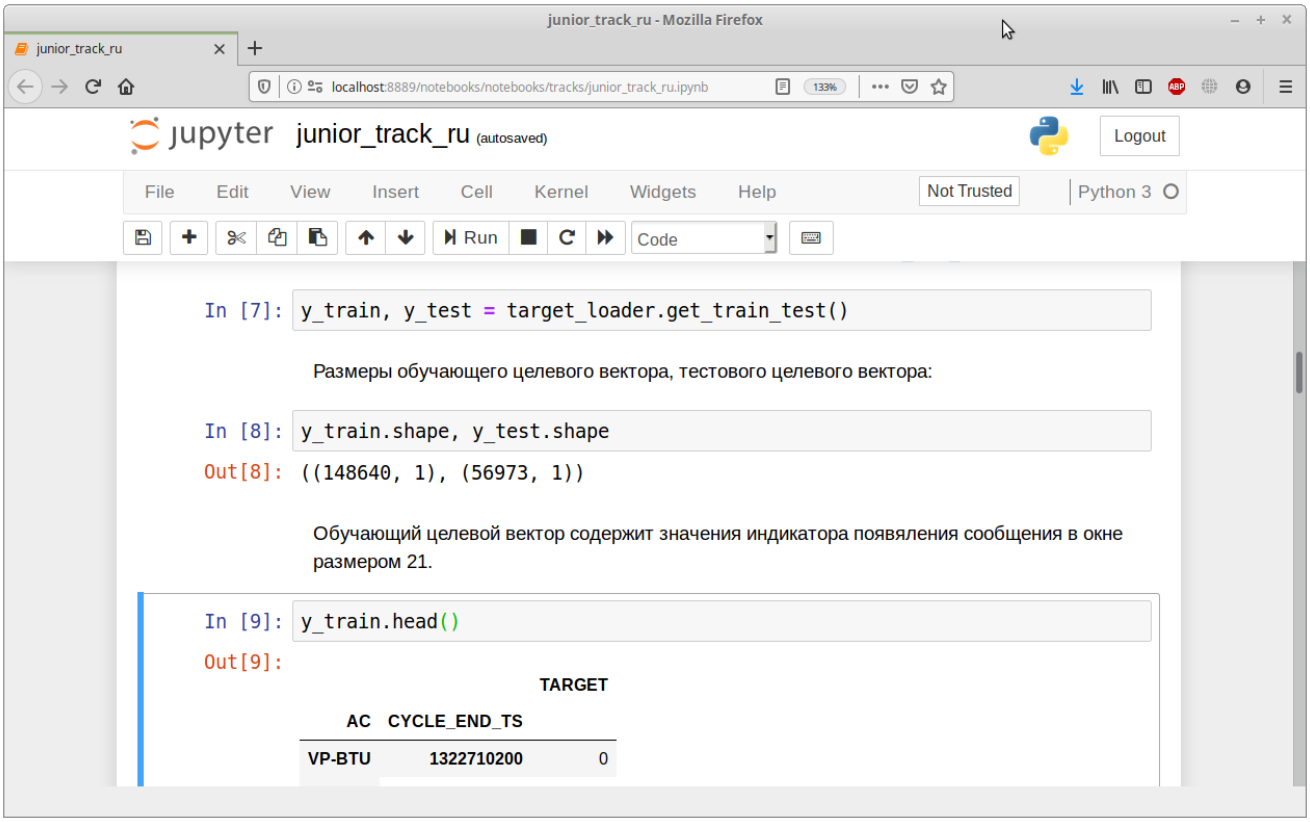

Рис. 7. Тест целевых значений удален системой

Примечания

Целевой загрузчик сохраняет согласованный индекс для DataFrames.

Целевой загрузчик использует кэширование в базе данных конфигурации задач для хранения целевых фреймов данных поезда / теста.

В случае, если поезд / тестовые наборы для задачи не найдены в кеше или повреждены, target_loader перестроит кеш, используя определение предметной области из вашего проекта.

2.10.1 Использование валидаций из пакета model_selection

DATASKAI содержит свои собственные инструменты для поддержки перекрестной проверки. Большая их часть не представлена в библиотеке scikit-learn, но может быть полезна в различных исследованиях.

Эти инструменты представлены в виде классов Python и собраны в отдельный модуль с именем model_selection.

Классы разбивают входные данные на фолды согласно определенному правилу и возвращают эти фолды пользователю. В то же время входные данные сортируются по времени (или должны быть уже отсортированы пользователем, но мы исправим это в ближайшем будущем), поэтому в сгибах результатов содержатся данные, близкие по времени.

Также в некоторых классах мы используем группирование данных по сущности (класс объекта, категория или что-то подобное), а затем строим фолды так, чтобы данные каждого объекта равномерно распределялись между различными сгибами.

Существует пять базовых классов:

TimeRangeBase,

TimeRangeBaseExtraFold,

TimeRangeByEntityBase,

TimeRangeByEntityBaseExtraFold и

TimeRangeConsecutiveOnesBase.

Каждый из них предназначен для предварительного разделения входных данных на складки, которые затем объединяются в поезд и тестируются с использованием метода разделения дочерних классов.

На диаграмме также показан класс SignalAggregationKFold, логика которого аналогична TimeRangeConsecutiveOnesBase, но охватывает дополнительные функции.

Работа всех классов будет обсуждаться более подробно ниже.

Диаграмма классов представлена как:

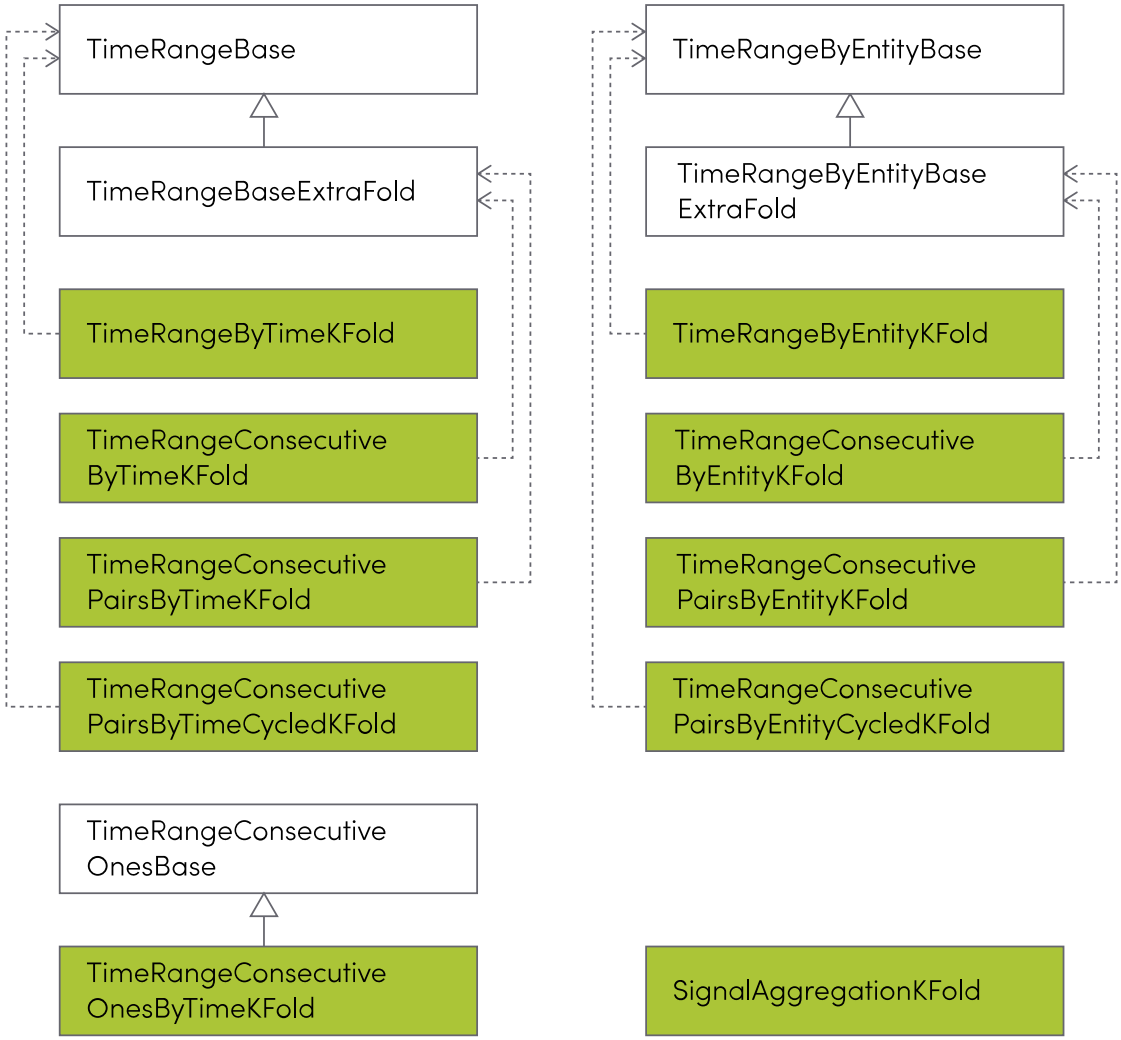

Рис. 8. Диаграмма классов

2.11 Базовые классы

Базовые классы используются для построения фолдов данных, из которых затем расщепления будут обрабатываться с использованием различных алгоритмов. Первичные фолды содержат примерно одинаковое количество данных.

2.11.1 TimeRangeBase

Класс TimeRangeBase работает как показано ниже на схеме. Он принимает входные данные, сортирует их и выполняет разделение на n фолдов (пользователь может настроить n параметров при создании экземпляра класса).

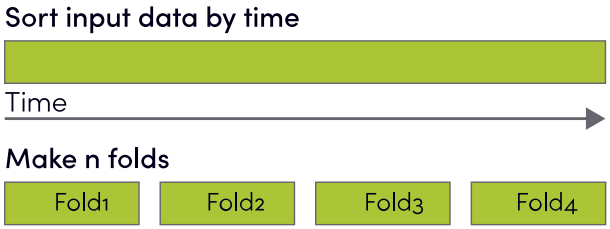

Рис. 9. Схема TimeRangeBase

2.11.2 TimeRangeBaseExtraFold

TimeRangeBaseExtraFold работает как предыдущий класс, но создает +1 дополнительный фолд. Это происходит потому, что классы TimeRangeConsecutiveByTimeKFold и TimeRangeConsecutivePairsByTimeKFold, которые являются наследниками класса TimeRangeBaseExtraFold, требуют +1 начального фолда для создания ожидаемого количества разбиений.

2.11.3 TimeRangeByEntityBase

Другой базовый класс - TimeRangeByEntityBase. Помимо сортировки по времени, он также выполняет группирование данных по имени объекта. Таким образом, мы гарантируем, что данные каждой сущности будут распределены по всем фолдам как можно более равномерно.

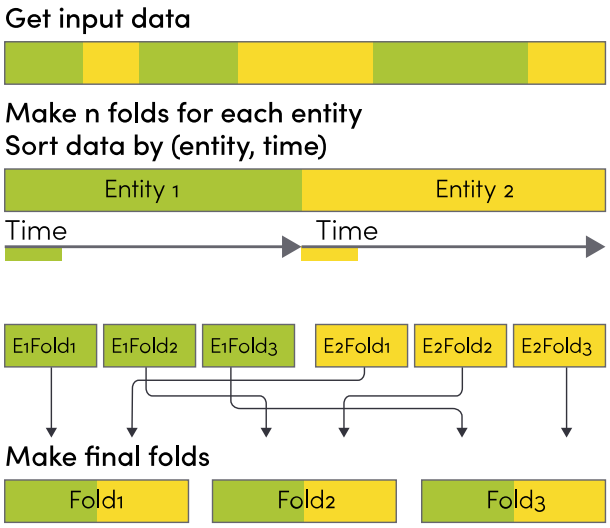

Рис. 10. Схема TimeRangeByEntityBase

2.11.4 TimeRangeByEntityBaseExtraFold

TimeRangeByEntityBaseExtraFold работает как предыдущий класс, но создает +1 дополнительный фолд. Это происходит потому, что классы TimeRangeConsecutiveByEntityKFold и TimeRangeConsecutivePairsByEntityKFold, которые являются наследниками класса TimeRangeByEntityBaseExtraFold, требуют +1 начальный фолд для создания ожидаемого количества разбиений.

2.11.5 TimeRangeConsecutiveOnesBase

Третий базовый класс - TimeRangeConsecutiveOnesBase. Ожидается, что этот класс будет использоваться в случае, когда входные данные содержат функцию в виде вектора нулей и единиц, где один означает возникновение какого-либо события, а ноль означает его отсутствие.

После разбиения каждая складка данных содержит последовательность единиц и предшествующих во времени нулей. Например, последовательность [1, 1, 0, 0, 1, 1, 0, 1, 1, 0] будет разделена на 4 раза со следующими индексами: [0 1] [2 3 4 5] [6 7 8] [9].

Внимание

В текущей реализации входные данные должны быть уже отсортированы (сущность, время).

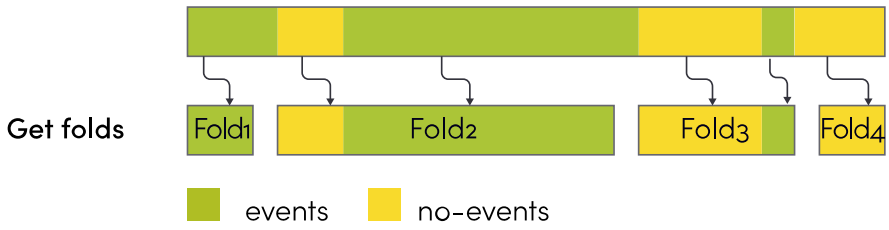

Рис. 11. Схема TimeRangeConsecutiveOnesBase

Ниже приведено описание четырех пар классов генераторов складок. Классы одной пары работают по аналогичным алгоритмам, но один из классов группирует данные по имени объекта, а другой - нет.

Принципы работы классов TimeRangeConsecutiveOnesBase и SignalAggregationKFold будут приведены в конце этого раздела.

2.12 Создание KFold

Классы TimeRangeByTimeKFold и TimeRangeByEntityKFold делают примерно одинаковые фолды из данных и создают разбиения для проверки путем последовательного выбора каждой из частей тестового образца. Например, если вы используете n_splits = 4 в базовом классе и имеете 4 фолда данных (1, 2, 3, 4), после генерации вы получите следующие сплиты: (2 + 3 + 4, 1), (1 + 3) + 4, 2), (1 + 2 + 4, 3), (1 + 2 + 3, 4).

2.12.1 TimeRangeByTimeKFold

В случае использования TimeRangeByTimeKFold (унаследованного от класса TimeRangeBase) расщепление выполняется с сквозной сортировкой данных по времени.

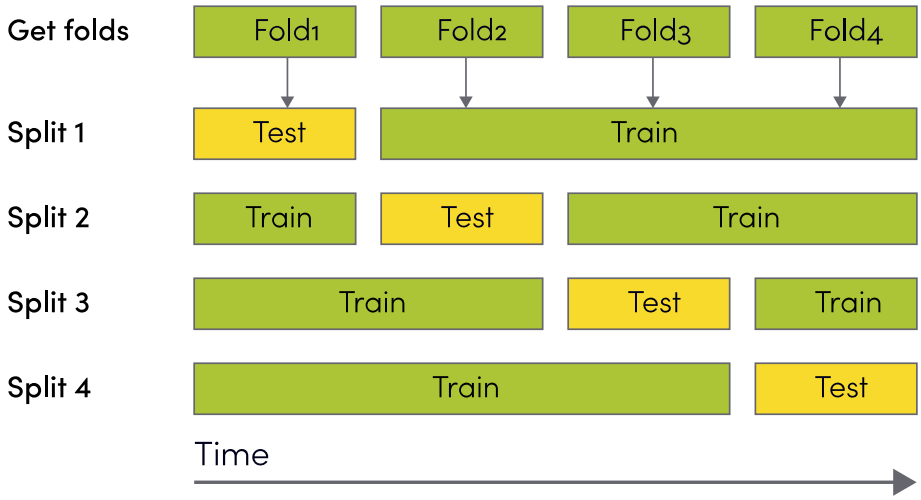

Рис. 12. Схема TimeRangeByTimeKFold

2.12.2 TimeRangeByEntityKFold

Если вы используете TimeRangeByEntityKFold, происходит дополнительное деление: данные для каждой сущности равномерно распределяются по начальным фолдам (см. Базовый класс TimeRangeByEntityBase). Затем первичные фолды объединяются для фолдов проверки / обучения. Все данные отсортированы по времени в подмножествах каждого объекта.

Внимание

В текущей реализации входные данные должны быть уже отсортированы (сущность, время).

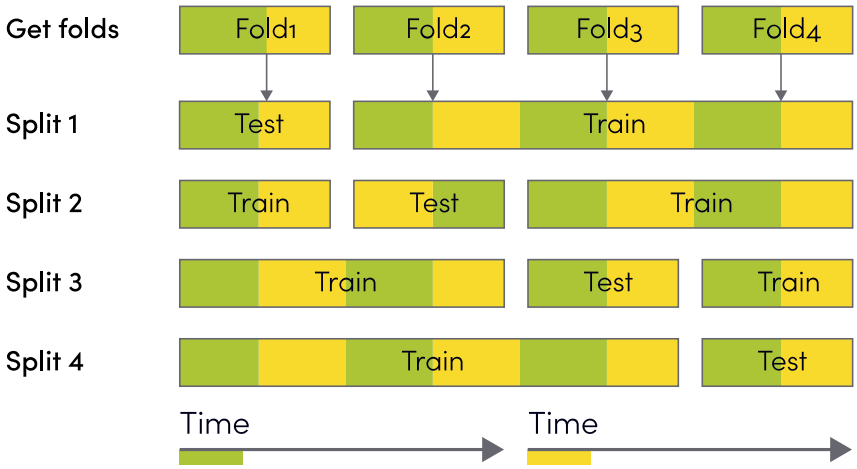

Рис. 13. Принцип действия класса TimeRangeByEntityKFold.

2.13 Разделение временных рядов

Классы этого типа делают расщепления, в которых тестовая складка всегда выходит за пределы железнодорожной складки вовремя. На первой итерации тестовый фолд представляет собой вторую часть входных данных, затем третью, четвертую и т. Д. Например, если вы разделите данные на 4 раза (1, 2, 3, 4), на выходе вы получите следующие разбиения: (1, 2), (1 + 2, 3), (1 + 2 + 3, 4)

2.13.1 TimeRangeConsecutiveByTimeKFold

При использовании TimeRangeConsecutiveByTimeKFold (унаследованном от класса TimeRangeBase) вы получите разделение с сквозной сортировкой данных по времени, как показано ниже:

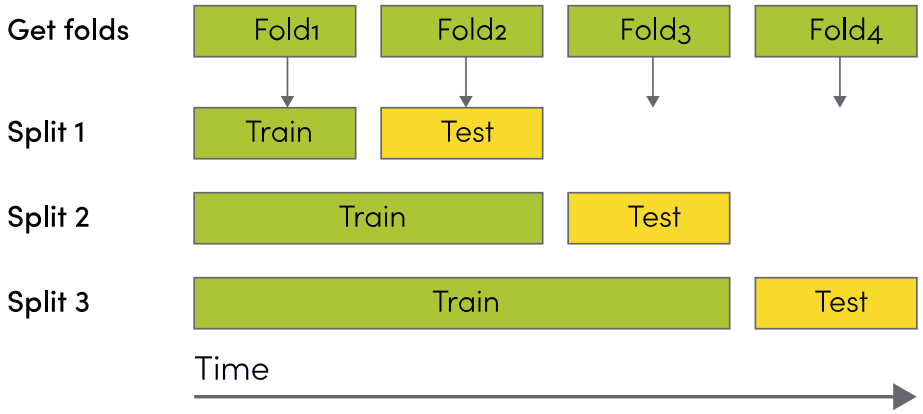

Рис. 14. Принцип действия TimeRangeConsecutiveByTimeKFold

2.13.2 TimeRangeConsecutiveByEntityKFold

Также вы можете равномерно распределить данные каждой сущности по разным фолдам, используя TimeRangeConsecutiveByEntityKFold (унаследованный от класса TimeRangeByEntityBase). Все данные отсортированы по времени отдельно для каждого объекта.

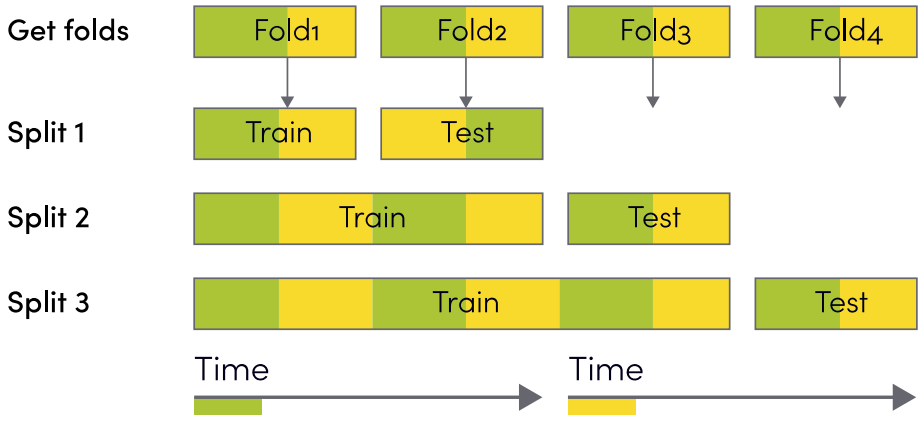

Рис. 15. Принцип действия класса TimeRangeConsecutiveByEntityKFold

2.14 Последовательные пары

Классы TimeRangeConsecutivePairsByTimeKFold и TimeRangeConsecutivePairs-ByEntityKFold создают наборы, которые содержат пары соседних частей данных - одну в качестве обучения, а другую - тестовую складку. Например, если вы разделите данные на 4 раза (1, 2, 3, 4), на выходе вы получите следующие разбиения: (1, 2), (2, 3), (3, 4)

2.14.1 TimeRangeConsecutivePairsByTimeKFold

Класс TimeRangeConsecutivePairsByTimeKFold (унаследованный от класса TimeRangeBase) возвращает разбиения с общей сортировкой по времени.

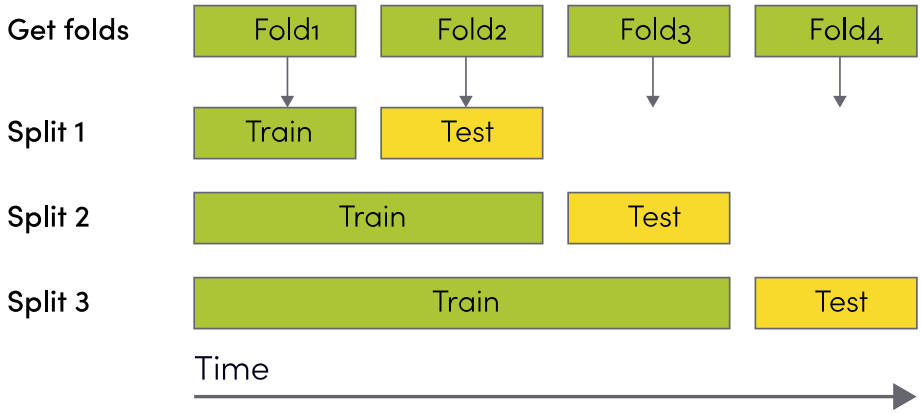

Рис. 16. Принцип работы класса TimeRangeConsecutivePairsByTimeKFold

2.14.2 TimeRangeConsecutivePairsByEntityKFold

Класс TimeRangeConsecutivePairsByEntityKFold (унаследованный от класса TimeRangeByEntityBase) дополнительно распределяет данные каждого объекта по разным фолдам, данные сортируются по времени отдельно для каждого объекта.

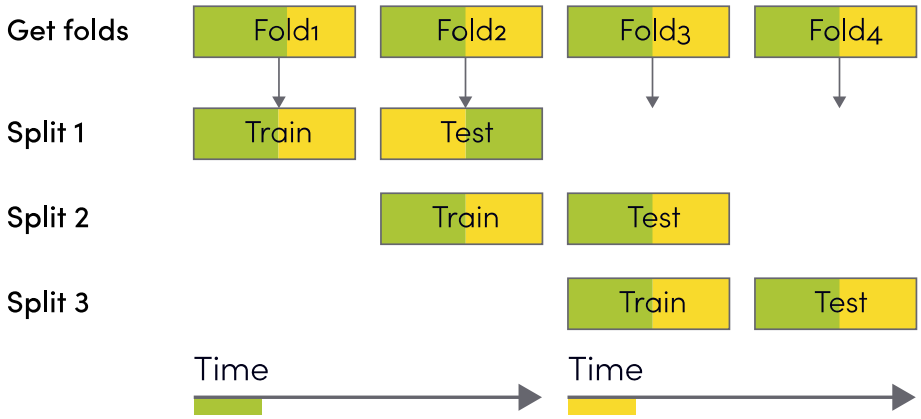

Рис. 17. Принцип работы класса TimeRangeConsecutivePairsByEntityKFold

2.15 Циклические последовательные пары

Классы TimeRangeConsecutivePairsByTimeCycledKFold и TimeRangeConsecutivePairsByEntityCycledKFold возвращают пары, состоящие из двух соседних фолдов данных - одного в качестве обучения, а другого - тестового сгиба, и когда достигается последнее свертывание данных, последовательность повторяется, создавая пару из последнего и первого в сгибах последовательности , Например, если вы разделите данные на 4 раза (1, 2, 3, 4), на выходе вы получите следующие разбиения: (4, 1), (1, 2), (2, 3), (3 4).

2.15.1 TimeRangeConsecutivePairsByTimeCycledKFold

TimeRangeConsecutivePairsByTimeCycledKFold (унаследованный от класса TimeRangeBase) возвращает циклические пары с общими данными сортировки по времени.

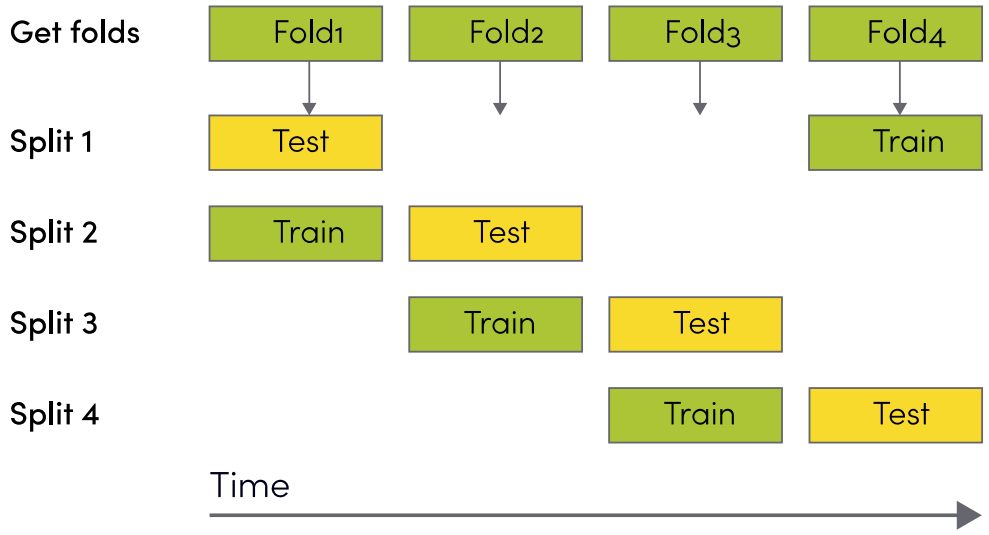

Рис. 18. Принцип работы класса TimeRangeConsecutivePairsByTimeCycledKFold

Также вы можете настроить количество фолдов данных, которые будут объединены в сгиб тренировки. Например, если этот параметр равен 2, то в 4 сгибах данных (1, 2, 3, 4) у вас будет (3 + 4, 1), (4 + 1, 2), (1 + 2, 3), (2 + 3, 4) результат разделения, как показано ниже.

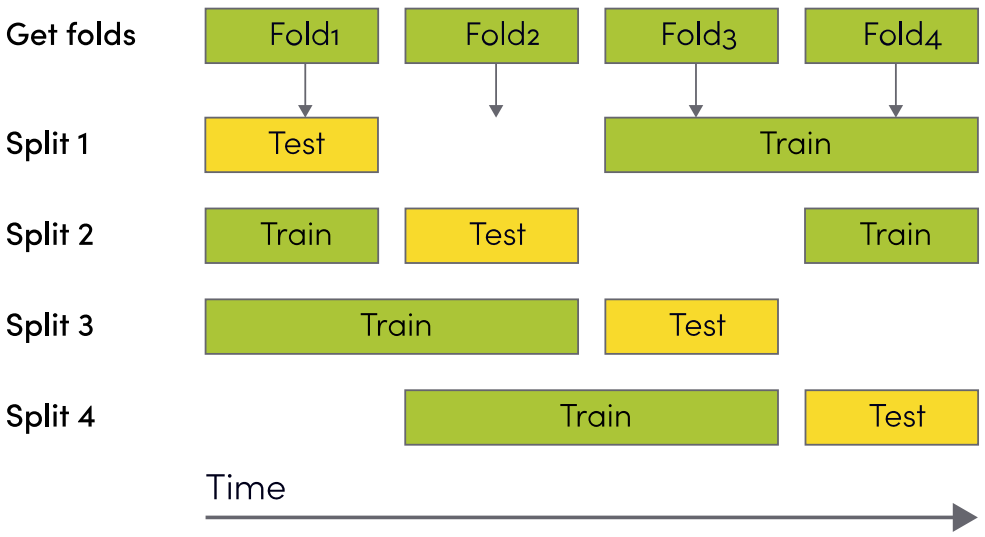

Рис. 19. TimeRangeConsecutivePairsByTimeCycledKFold с агрегацией двух соседних фолдов данных

2.15.2 TimeRangeConsecutivePairsByEntityCycledKFold

TimeRangeConsecutivePairsByEntityCycledKFold (унаследованный от класса TimeRangeByEntityBase) дополнительно распределяет данные каждого объекта по разным фолдам, данные сортируются по времени отдельно для каждого объекта.

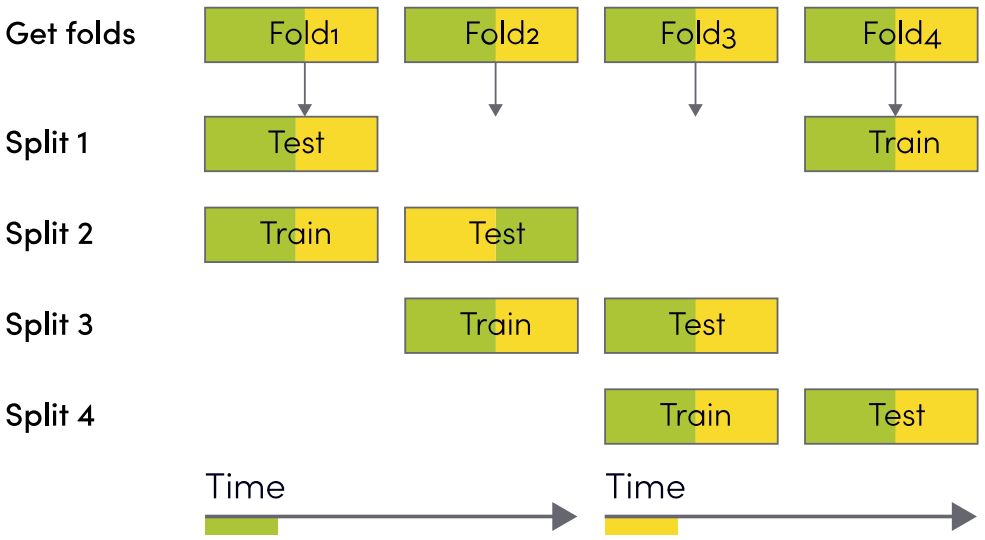

Рис. 20. Принцип работы класса TimeRangeBase

Схема работы для TimeRangeConsecutivePairsByEntityCycledKFold с объединением двух соседних фолдов данных:

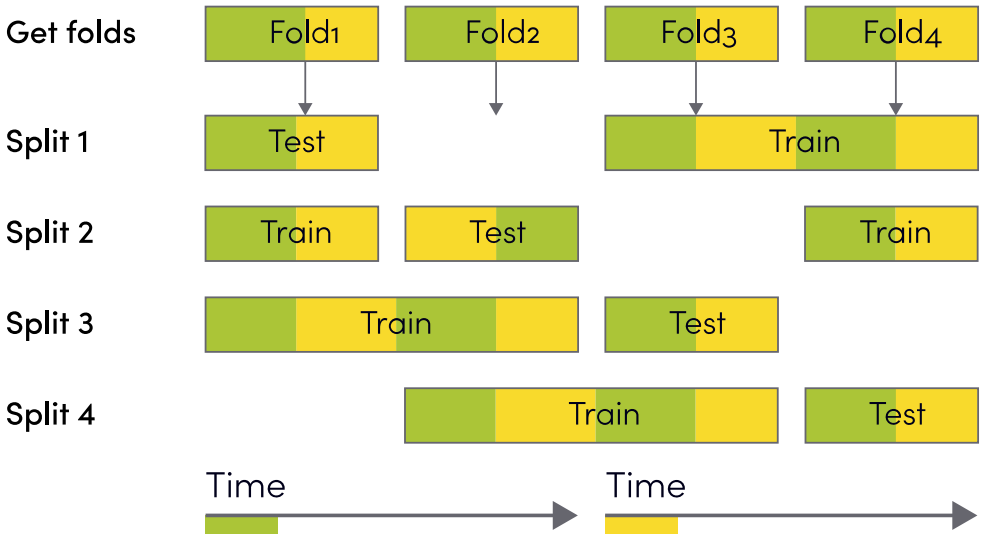

Рис. 21. Принцип работы класса TimeRangeConsecutivePairsByEntityCycledKFold с агрегацией двух соседних фолдов данных

2.16 Последовательные события

2.16.1 TimeRangeConsecutiveOnesByTimeKFold

Класс TimeRangeConsecutiveOnesByTimeKFold использует алгоритм для выделения групп последовательных событий и основан на классе TimeRangeConsecutiveOnesBase. При получении окончательных фолдов используется следующая логика: контрольный сгиб всегда впереди по времени. Например, если вы разделите данные на 4 раза (1, 2, 3, 4), на выходе вы получите следующие разбиения: (1, 2), (1 + 2, 3), (1 + 2 + 3, 4)

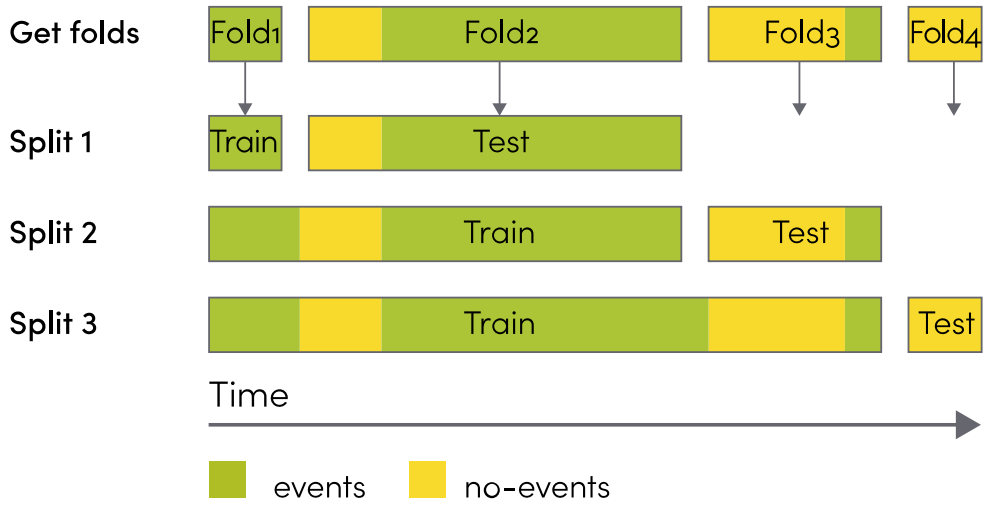

Рис. 22. Принцип действия класса TimeRangeConsecutiveOnesByTimeKFold

2.16.2 SignalAggregationKFold

Класс SignalAggregationKFold также работает на основе распределения групп событий (единиц), но вы можете настроить количество разбиений и ширину окна, которая представляет собой число предыдущих и последующих значений для каждой группы событий.

SignalAggregationKFold позволяет равномерно распределять данные по группам. Если у вас есть разрыв ставок между группами событий (длинная нулевая последовательность), то вы гарантированно не получите фолд без событий. Параметр h, ширина окна, действует как контекст, который показывает, как развивается процесс до и после группы событий.

Таким образом, этот класс обнаруживает группы единиц, добавляет h предыдущих и h следующих (для каждой группы) значений к себе, распределяет группы по сгибам n_splits (параметры n_splits и h настраиваются пользователем) и удаляет повторяющиеся значения. Затем он расширяет фолды данных с еще не

задействованными нулевыми (без событий) значениями входных данных. Данные в каждой отдельной сортировке сортируются по времени.

Если количество требуемых разбиений равно количеству отдельных групп событий, оно работает так, как показано на диаграмме. Обратите внимание, что тренировочный фолд, по сути, представляет собой массив входных данных, из которых был исключен тестовый фолд с сохранением начальной сортировки.

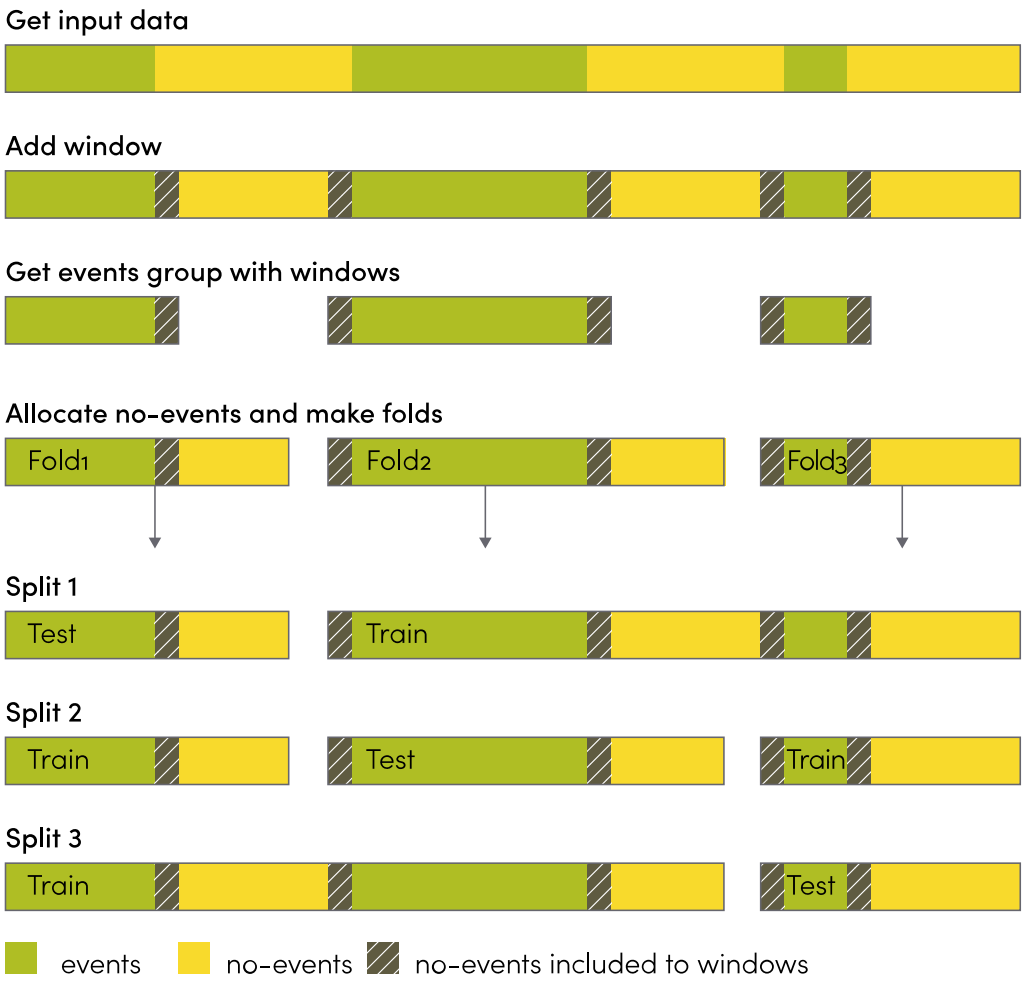

Рис. 23. Принцип действия класса SignalAggregationKFold

В случае, если количество разделений (параметр n_splits) меньше количества групп событий, группы агрегируются по правилу

```
number_of_groups % number_of_splits
```

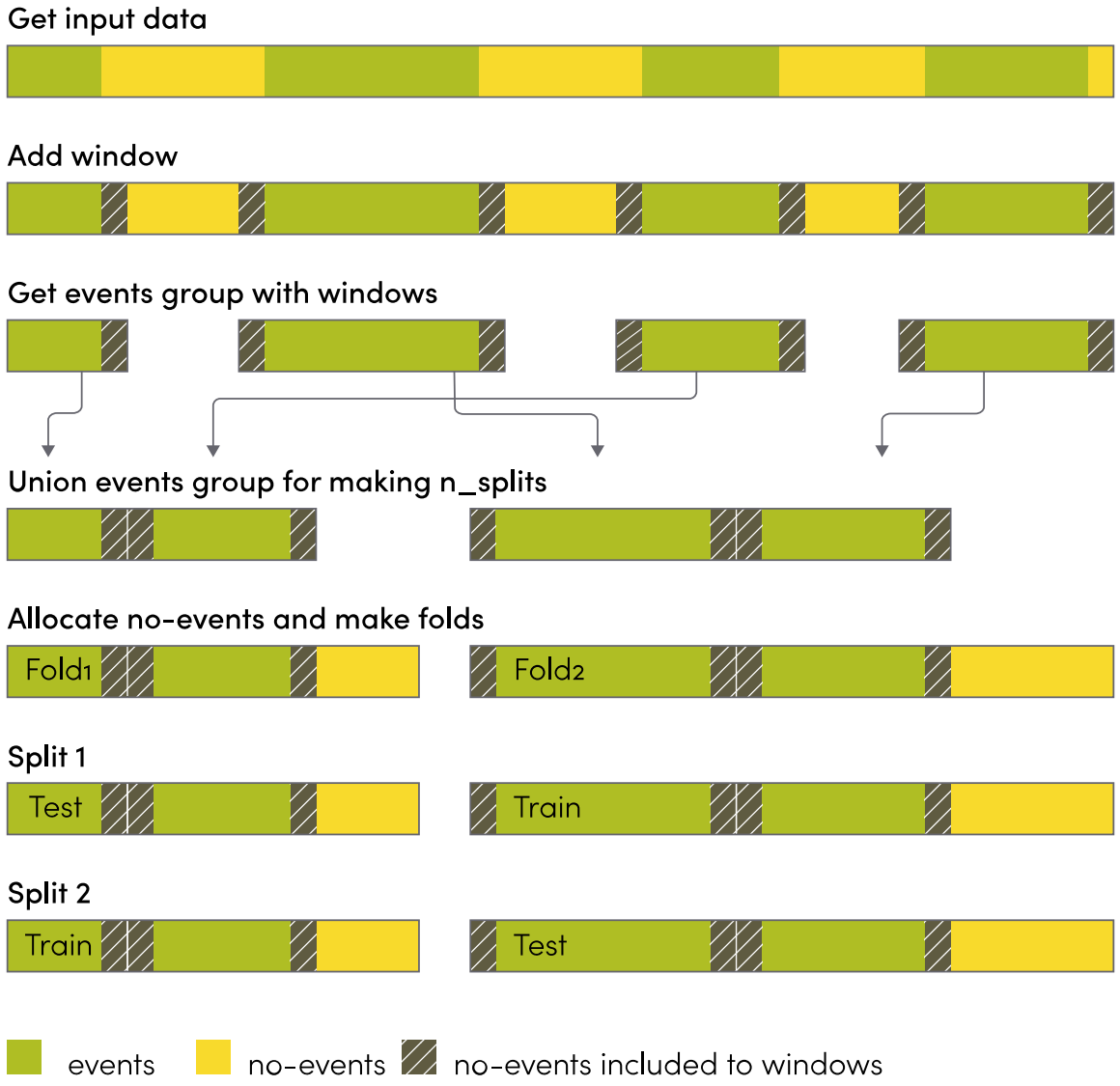

Рис. 24. Принцип работы класса SignalAggregationKFold с созданием определенного количества разбиений

2.17 Отправка ноутбука с результатами

Когда модель создается и проверяется с помощью процедуры CV, следующее, что нужно сделать, это представить результаты в таблице лидеров.

DATASKAI предлагает компонент Submitter для быстрой отправки прогнозов на тестовом наборе, а также много дополнительной информации в подсистеме хранения данных:

```
submitter = task_loader.submitter
```

После того, как вы заполнили значения в y_test предсказанными ответами для набора тестов, вам нужно вызвать метод submit_results:

```
submitter.submit_results(
    results_df=y_test,
    author_name=AUTHOR_NAME,
    model=logistic_regression,
    model_name='Logistic Regression',
    tags=['baseline'],
    model_feature_columns=feature_columns.tolist(),
    model_fill_na_values=[0] * len(feature_columns),
    model_version='1.0'
)
```

Этот метод содержит множество именованных аргументов, каждый со своей целью:

results_df - (обязательный, pandas dataframe) - Pandas Dataframe

author_name - (обязательно, строка) - имя автора, создавшего модель

model - (обязательно, объект) - экземпляр объекта с изученной моделью

model_name - (обязательно, строка) - название модели для отображения в таблице результатов теги - (обязательно, список строк) - ключевые слова для быстрого поиска в подсистеме результатов

model_feature_columns - (обязательно, список строк) - упорядоченный список используемых функций

model_fill_na_values - (обязательно, список с плавающей точкой) - числа, используемые по умолчанию для значений признаков в случае его отсутствия

model_version - (обязательно, строка) - версия модели, для быстрого поиска и сортировки в таблице результатов

Примечание

Отправитель автоматически сохраняет текущее состояние записной книжки при вызове submit_results. Также Submitter упаковывает блокнот jupyter в файл и отправляет его в хранилище отправлений вместе с указанными аргументами. Поэтому вам не нужно отслеживать свой блокнот в git или называть его другим именем после отправки результатов, просто используйте тот же блокнот и

отправьте результаты снова в следующий раз при изменении кода.

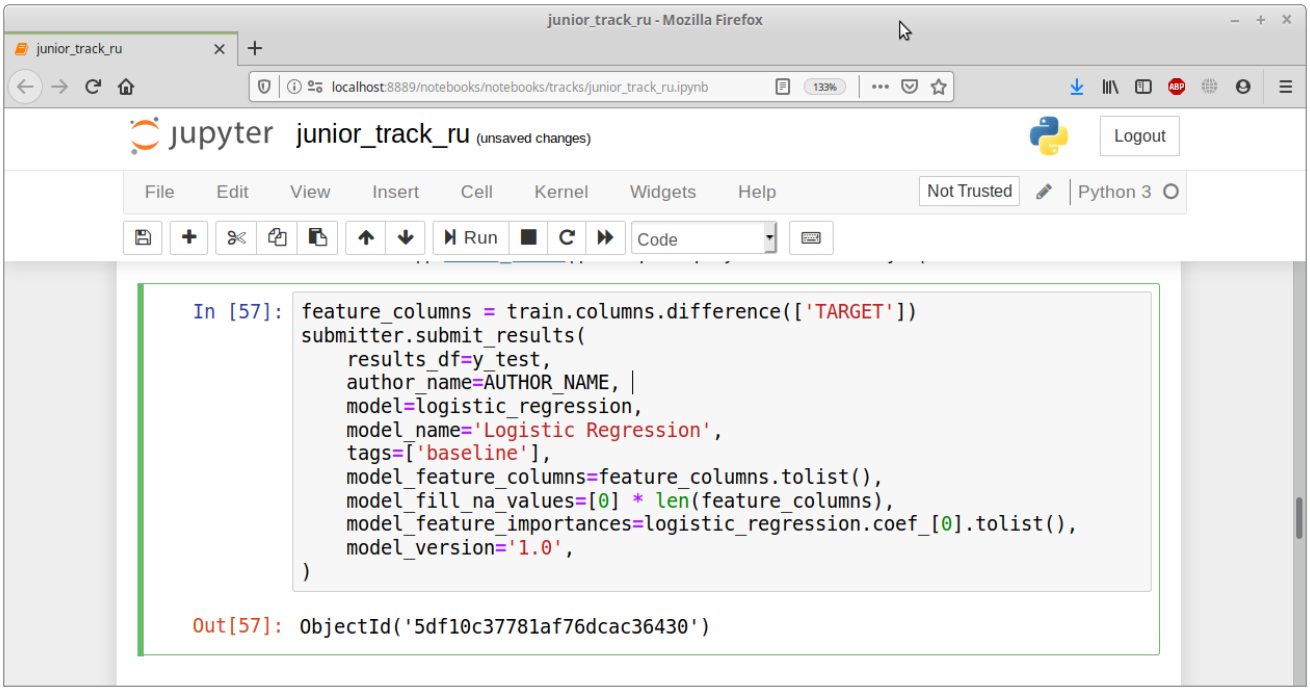

Рис. 25. Отправка сабмитов

Метод submit_results также возвращает идентификатор Id в результате правильной отправки. Этот идентификатор можно использовать в качестве ключевого слова для поиска в графическом интерфейсе результатов.

Метод submit_results также содержит необязательные параметры, которые могут использоваться в некоторых случаях:

notebook_name - (необязательно, строка) - имя текущего файла блокнота, из которого вызывается метод submit_results. Обычно вам не нужно передавать этот параметр, поскольку Submitter автоматически определяет имя записной книжки, из которой он был вызван.

data_scaler - (необязательно, объект) - Sklearn Scaler объект, в случае, если вы используете один model_feature_importances - (необязательно, список с плавающей точкой) - веса объектов (зависит от используемой вами модели), некоторые модели (XGBoost, sklearn RandomForest и т. д.) поддерживаются Submitter по умолчанию, и вам не нужно передавать этот параметр

Внимание

Помните, что тщательно заполненные данные об отправке могут сэкономить много времени и усилий для вас и вашей команды на последующих этапах решения проблем или на следующем этапе жизненного цикла модели (например, проверка модели, модели упаковки и т. д.).

2.18 Проверка результатов теста через GUI

После завершения отправки вы можете проверить результат отправки в графическом интерфейсе. Попросите Middle DS или Senior DS предоставить вам ссылку на интерфейс GUI:

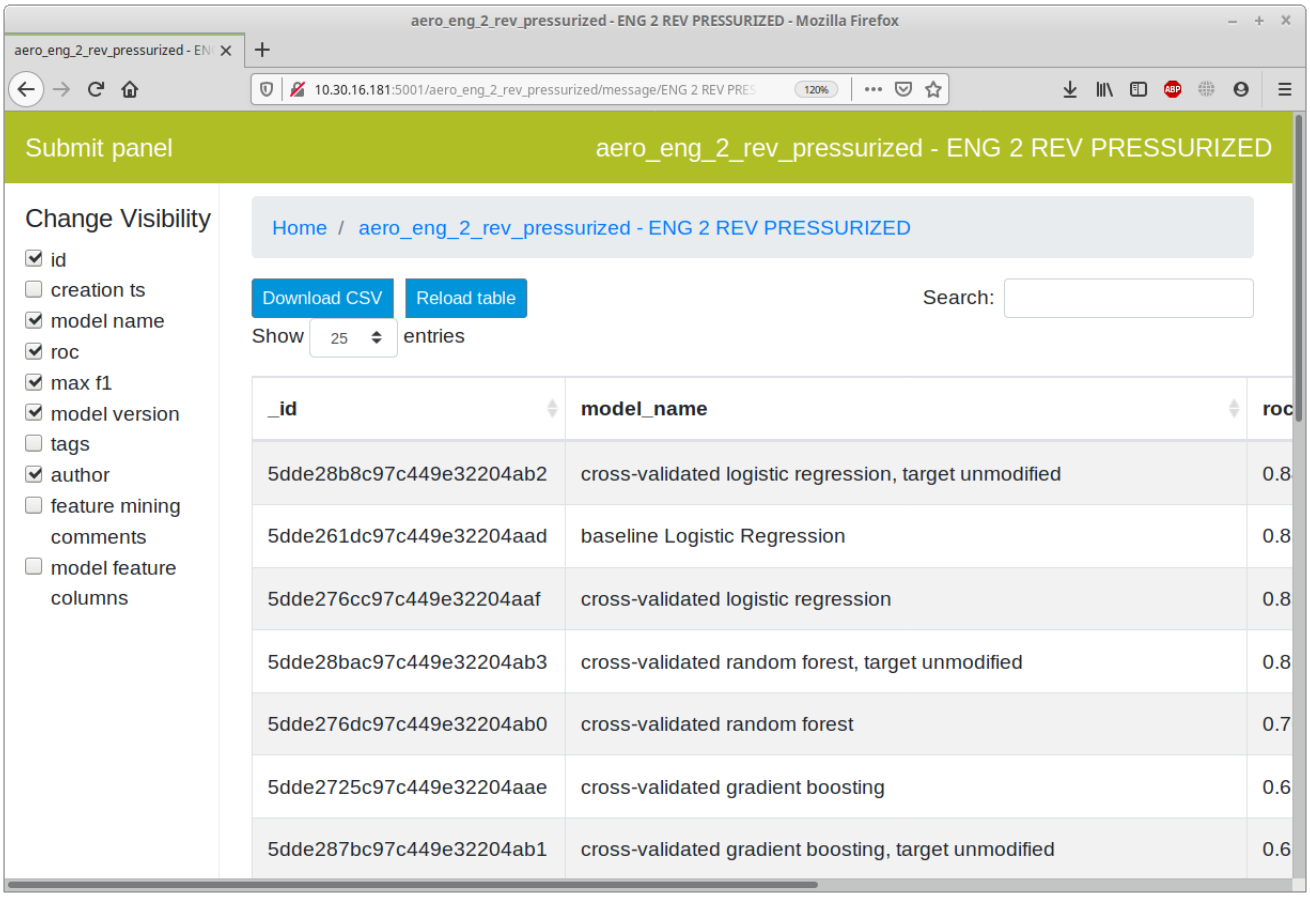

Рис. 26. Проверка метрик для отправки

Этот интерфейс позволяет проверять метрики для всех представленных решений по текущей задаче, искать лучшие / худшие значения метрик, выполнять быстрый текстовый поиск для представлений, скрывать некоторые поля и т. д. Кроме того, меню в верхней части окна позволяют переключаться на все доступные в графическом интерфейсе задачи и проверить оценки по нему:

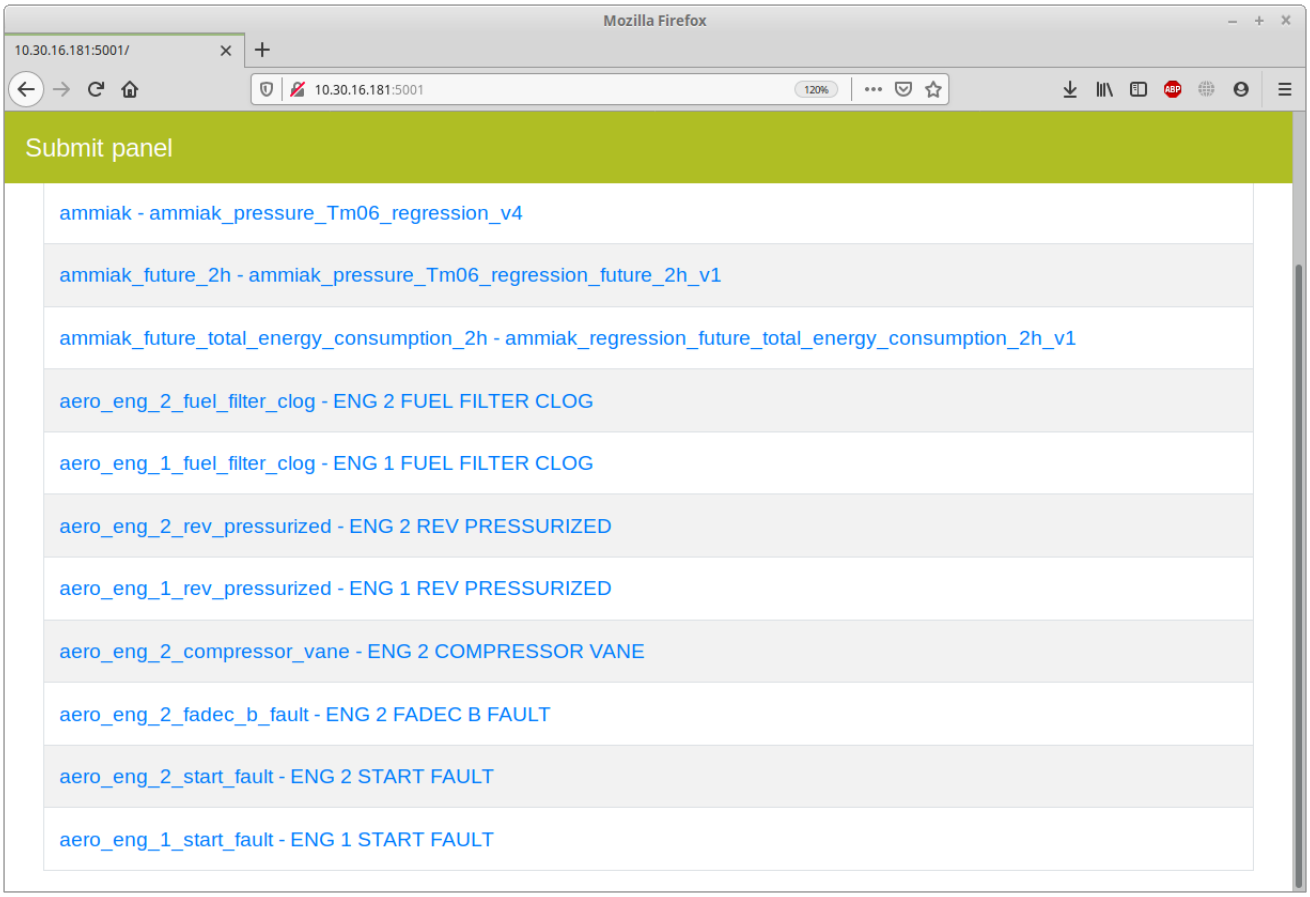

Рис. 27. Проверка метрик

Внимание

Имейте в виду, что позитивная конкурентная среда в проекте является неотъемлемой частью любого процесса решения проблем DS, поэтому следите за тем, чтобы ваши результаты были хорошо организованы, и тщательно заполняйте все необходимые поля, чтобы сохранить таблицу в удобочитаемой форме.

2.19 Загрузка ноутбука из ранее представленного к проверке

Можно загрузить любой представленный к проверке ноутбук обратно из списка лидеров. Для Junior DS может оказаться полезным пересмотреть подход к машинному обучению другого участника или воспользоваться функцией в случае потери любого оригинального ноубтука. Также это необходимо Middle Data Scientist для проверки воспроизводимости моделей машинного обучения перед упаковкой их в производственную систему. Для таких нужд существует класс NotebookDownloader.

Мы начнем с выбора идентификатора интересующей нас заявки в таблице лидеров. Затемиспользуем класс NotebookDownloader и скачааем ноубтук и все, что связано с выбранным сабмитом:

```
from notebook_downloader import NotebookDownloader

notebook_id = '5df7731cf8427b16f81b4dbb'

nd = NotebookDownloader(TASK_CONFIG['mongo_config'])
filename, submit = nd.download_notebook_to_file
(notebook_id=notebook_id,
ilename='downloaded_notebook.ipynb',
overwrite=True)
```

Загруженный блокнот будет находиться в текущем рабочем каталоге.

2.20 Использование инструментов из пакета утилит

2.20.1 AdversarialTrainTestValidator

AdversarialTrainTestValidator из модуля utils можно использовать в качестве предварительного шага перед построением реальной модели ML.

Это метод выбора функций, основанный на степени сходства тренировочных / тестовых моделей с точки зрения распределения функций. Основная идея состоит в том, чтобы попытаться обучить двоичный вероятностный классификатор для различения примеров обучения / тестирования. Если их трудно различить, вероятно, они были получены из одного и того же распределения.

AdversarialTrainTestValidator работает итеративно, удаляя самую сильную функцию (функцию с наивысшим коэффициентом важности), пока не будет достигнут заданный приемлемый порог показателя.

Например, возьмите классификатор ”случайный лес”(метод машинного обучения на основе комитета регрессионных деревьев принятия решений), 5-кратную перекрестную проверку, 0,7 ROC AUC в качестве приемлемого метрического порога.

Как мы видим, в начале метрика разделения выше, чем 0,7 ROC AUC - что настоятельно свидетельствует о том, что обучающие / тестовые образцы легко различить даже для такой слабой модели (наша модель случайного леса имеет 5 деревьев и максимальную глубину дерева, равную 2).

Требуется 5 исключающих шагов, чтобы достичь приемлемого порогового значения метрики, и дополнительные stopping_rounds = 5 для достоверности.

Наконец, чтобы увидеть функции, определенные для удаления, мы получаем доступ к атрибуту remove_features. В результате стоит попытаться удалить эти функции из тренировочной / тестовой модели.

In [34]:

```python
from utils import AdversarialTrainTestValidator
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import KFold

clf = RandomForestClassifier(
    n_estimators=5,
    max_depth=2,
    random_state=17,
)

kf = KFold(
    n_splits=5,
    shuffle=True,
    random_state=17,
)

validator = AdversarialTrainTestValidator(
    classifier=clf,
    scoring='roc_auc',
    cv=kf,
    threshold=0.7,
    stopping_rounds=5,
    verbose=2,
)

validator.validate(x_train, x_test)

print(validator.removed_features)
```

```
Step: 1, Score: 0.7256764465201794, removed feature: cur_val__aids_rep_01__alt
0
Step: 2, Score: 0.7292390885492711, removed feature: cur_val__aids_rep_01__n1c_2
0
Step: 3, Score: 0.711758268472457, removed feature: cur_val__aids_rep_01__n1c_1
0
Step: 4, Score: 0.7081490764252288, removed feature: cur_val__aids_rep_01__tn_2
0
Step: 5, Score: 0.6893923395665462, removed feature: cur_val__aids_rep_01__pha_sc_1
1
Step: 6, Score: 0.6899058776320618, removed feature: cur_val__aids_rep_01__gle_2
2
Step: 7, Score: 0.6787174804398723, removed feature: cur_val__aids_rep_01__t25_2
3
Step: 8, Score: 0.6854036827438927, removed feature: cur_val__aids_rep_01__n2_2
4
Step: 9, Score: 0.6664119107243437, removed feature: cur_val__aids_rep_01__vl_sd_1
5
['cur_val__aids_rep_01__alt', 'cur_val__aids_rep_01__n1c_2', 'cur_val__aids_rep_01__n1c_
1', 'cur_val__aids_rep_01__tn_2', 'cur_val__aids_rep_01__pha_sc_1']
```

Рис. 28. Пример использования AdversarialTrainTestValidator