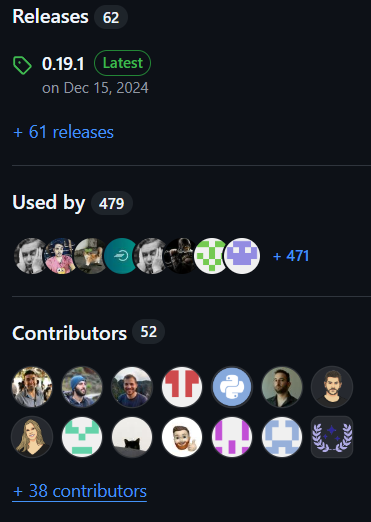
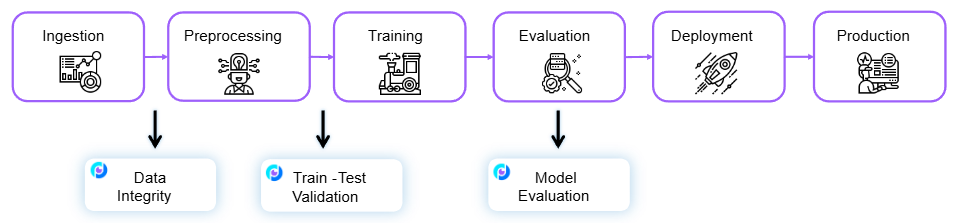
# DeepChecks for Tabular

## Введение

Репозиторий живой



В Deepchecks есть множество встроенных проверок и наборов, которые могут помочь в проверке различных моментов на протяжении всего этапа исследования. Разработчики определили, что существует несколько повторяющихся сценариев, каждый из которых имеет свои собственные потребности и характеристики.



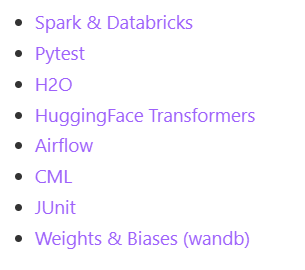
**Новые данные**: Проверка единого набора данных. Когда вы начинаете работать с новым набором данных, у вас есть только один набор данных (без разделения на обучение и тестирование), и у вас, вероятно, нет модели. В рамках вашего EDA вы хотите обеспечить целостность ваших данных и подготовить их к вашим потребностям. Например, вы хотите знать, есть ли много повторяющихся выборок, проблемы со строковыми или категориальными признаками, значительные отклонения, несогласованные метки и т.д. Для этих целей вы можете использовать пакет deepchecks.tabular.suites.data\_integrity().

**После разделения данных**: Проверка train-test. Когда вы разбиваете свои данные, у вас есть два или более отдельных набора данных, однако у вас может еще не быть модели. Непосредственно перед тем, как продолжить работу с данными, вы должны убедиться, что разбиения действительно являются репрезентативными, какими вы хотите их видеть. Например, вы хотите убедиться, что классы сбалансированы одинаково, что нет существенных изменений в распределении объектов или меток в каждом из классов, что нет потенциальной утечки данных, которая может повлиять на вашу модель или предполагаемые результаты, и т.д. Для этих целей вы можете использовать пакет deepchecks.tabular.suites.train\_test\_validation().

**После обучения модели**: анализ и проверка. На этом этапе у вас есть обученная модель, которую вы хотите оценить. Таким образом, вы, вероятно, захотите изучить несколько показателей производительности, сравнить их с различными эталонными показателями и составить четкое представление о производительности модели. Возможно, вы также захотите выяснить, где она работает недостаточно эффективно, и выяснить, есть ли какие-либо идеи, которые можно было бы использовать для повышения производительности модели. Для этих целей вы можете использовать пакет deepchecks.tabular.suites.model\_evaluation().

**Общий обзор**: Полный набор. Здесь вы хотите получить краткий обзор проекта и получить всю необходимую информацию, которую вы можете получить, учитывая конкретное состояние модели и данных. Для этой цели вы можете использовать функцию deepchecks.tabular.suites.full\_suite().

**Доступна интеграция с**



## Основные структурные элементы

**Check (проверка)**

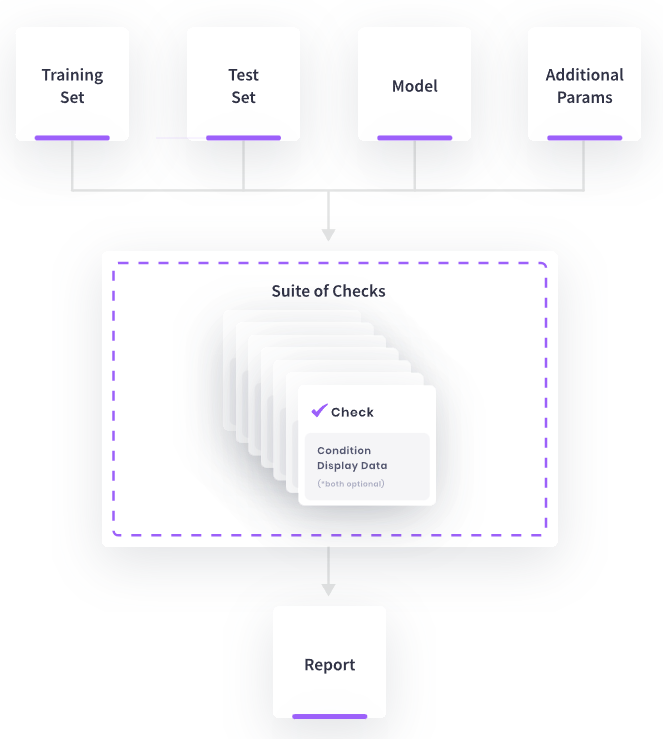
Каждая проверка позволяет проверить определенный аспект данных и моделей. Они являются базовым компонентом пакета, охватывающего все виды распространенных проблем, таких как: низкая производительность сегментов, смещение фичей, отклонения свойств изображения, аспекты, связанные с матрицей ошибок. Каждая проверка может иметь два типа результатов: визуальный результат, предназначенный для отображения (например, рисунок или таблица), либо возвращаемое значение, которое может быть использовано для проверки ожидаемых результатов проверки (проверка обычно выполняется путем добавления “условия” к проверке).

**Condition (условие)**

Условие - это функция, которая может быть добавлена к проверке, которая возвращает pass ✓, fail ✖ или warning ! – результат, предназначенный для проверки возвращаемого значения Check. Большинство проверок содержат предварительно реализованные условия, которые можно добавить к проверке, а их параметры можно настроить. В одной и той же проверке могут быть объединены разные условия. Оценка будет выполняться отдельно для каждого условия. Каждое условие имеет тип ConditionCategory, который указывает, будет ли невыполненное условие классифицироваться как fail (✖) или как warning (!).

**Suite (набор)**

Упорядоченный набор проверок, к которым могут быть добавлены условия. Набор позволяет отображать итоговый отчет по всем выполненным проверкам. Существующие наборы включают в себя условия по умолчанию, добавленные для большинства проверок. Вы можете отредактировать предварительно настроенные наборы или создать свой собственный набор с набором проверок и необязательных условий.



## Checks for Tabular

Доступно создание своих проверок, в документации указан туториал.

**data\_integrity**Модуль содержит все проверки целостности данных.

Классы

* ColumnsInfo  
  Возвращает роль и логический тип каждого столбца.
* MixedNulls  
  Поиск различных типов значений NULL, включая строковые представления NULL.
* StringMismatch  
  Обнаружение различных вариантов строковых категорий (например, разные написания одного и того же значения).
* MixedDataTypes  
  Обнаружение столбцов, содержащих смесь числовых и строковых значений.
* IsSingleValue  
  Проверка наличия столбцов, которые содержат только одно уникальное значение во всех строках.
* SpecialCharacters  
  Поиск в столбце(ах) значений, состоящих только из специальных символов.
* StringLengthOutOfBounds  
  Обнаружение строк, длина которых значительно больше/меньше определенной "нормальной" длины строк.
* DataDuplicates  
  Проверка наличия повторяющихся образцов в наборе данных.
* ConflictingLabels  
  Поиск образцов, у которых одинаковые значения признаков, но разные метки.
* ClassImbalance  
  Проверка, является ли набор данных несбалансированным, путем анализа распределения целевой переменной.
* OutlierSampleDetection  
  Обнаружение выбросов в наборе данных с использованием алгоритма LoOP.
* FeatureLabelCorrelation  
  Возвращает PPS (Predictive Power Score) всех признаков относительно метки.
* FeatureFeatureCorrelation  
  Проверка парной корреляции между признаками.
* IdentifierLabelCorrelation  
  Проверка, могут ли идентификаторы (Индекс/Дата) использоваться для предсказания метки.
* PercentOfNulls  
  Процент значений 'NULL' в каждом столбце.

**train\_test\_validation**  
Модуль содержит проверки для валидации обучающего и тестового наборов данных.

Классы

* CategoryMismatchTrainTest  
  Поиск новых категорий в тестовом наборе данных.
* NewCategoryTrainTest  
  Поиск новых категорий в тестовом наборе данных.
* DatasetsSizeComparison  
  Проверка размера тестового набора данных по сравнению с обучающим набором данных.
* DateTrainTestLeakageDuplicates  
  Проверка наличия тестовых дат в обучающих данных.
* DateTrainTestLeakageOverlap  
  Проверка тестовых данных, датированных ранее самой поздней даты в обучающем наборе.
* IdentifierLabelCorrelation  
  Проверка, могут ли идентификаторы (Индекс/Дата) использоваться для предсказания метки.
* IndexTrainTestLeakage  
  Проверка наличия тестовых индексов в обучающих данных.
* NewLabelTrainTest  
  Поиск новых меток в тестовом наборе данных.
* FeatureLabelCorrelationChange  
  Возвращает Predictive Power Score всех признаков для оценки способности каждого признака предсказывать метку.
* StringMismatchComparison  
  Обнаружение различных вариантов строковых категорий между одним и тем же категориальным столбцом в двух наборах данных.
* TrainTestFeatureDrift  
  Проверка TrainTestFeatureDrift устарела и будет удалена в версии 0.14.
* FeatureDrift  
  Вычисление дрейфа между обучающим и тестовым наборами данных для каждого признака с использованием статистических мер.
* TrainTestLabelDrift  
  Проверка TrainTestLabelDrift устарела и будет удалена в версии 0.14.
* LabelDrift  
  Вычисление дрейфа меток между обучающим и тестовым наборами данных с использованием статистических мер.
* TrainTestSamplesMix  
  Обнаружение образцов в тестовых данных, которые также присутствуют в обучающих данных.
* MultivariateDrift  
  Вычисление дрейфа между всеми обучающими и тестовыми наборами данных с использованием модели, обученной различать их.
* WholeDatasetDrift  
  Вычисление дрейфа между всеми обучающими и тестовыми наборами данных с использованием модели, обученной различать их.

**model\_evaluation**  
Модуль содержит проверки для оценки модели.

Классы

* BoostingOverfit  
  Проверка переобучения, вызванного использованием слишком большого количества итераций в градиентно-бустинговой модели.
* CalibrationScore  
  Вычисление кривой калибровки с байесовским показателем (brier score) для каждого класса.
* ConfusionMatrixReport  
  Вычисление матрицы ошибок модели на заданном наборе данных.
* ModelInferenceTime  
  Измерение среднего времени вывода модели (в секундах) на один образец.
* ModelInfo  
  Сводка параметров заданной модели.
* MultiModelPerformanceReport  
  Сводка показателей производительности для нескольких моделей на тестовых наборах данных.
* TrainTestPerformance  
  Сводка производительности заданной модели на обучающем и тестовом наборах данных на основе выбранных метрик.
* RegressionErrorDistribution  
  Проверка систематической ошибки и аномальной формы распределения ошибок регрессии.
* RegressionSystematicError  
  Проверка систематической ошибки регрессии.
* RocReport  
  Вычисление ROC-кривой для каждого класса.
* SegmentPerformance  
  Отображение показателя производительности, сегментированного по двум основным (или заданным) признакам в виде тепловой карты.
* SimpleModelComparison  
  Сравнение заданной модели с простой моделью (в соответствии с указанным типом модели).
* TrainTestPredictionDrift  
  Проверка TrainTestPredictionDrift устарела и будет удалена в версии 0.14.
* PredictionDrift  
  Вычисление дрейфа предсказаний между обучающим и тестовым наборами данных с использованием статистических мер.
* WeakSegmentsPerformance  
  Поиск сегментов с низкими показателями производительности.
* UnusedFeatures  
  Обнаружение признаков, которые практически не используются моделью.
* SingleDatasetPerformance  
  Сводка производительности заданной модели на обучающем или тестовом наборе данных на основе выбранных метрик.
* PerformanceBias  
  Проверка различий в производительности между подгруппами признака с возможностью учета контрольной переменной.

Model  
Deepchecks требует, чтобы модели следовали соглашениям API scikit-learn для вычисления предсказанных значений и вероятностей для каждого класса. Поэтому встроенные классификаторы и регрессоры scikit-learn, а также многие другие популярные типы моделей (например, XGBoost, LightGBM, CatBoost и т.д.), поддерживаются "из коробки".

В частности, Deepchecks требует, чтобы в объекте модели были реализованы следующие методы:

* Метод predict , который принимает массивоподобный объект формы **(n\_samples, n\_features)**, содержащий входные признаки, и возвращает предсказанные значения.
* Метод predict\_proba , который принимает массивоподобный объект формы **(n\_samples, n\_features)**, содержащий входные признаки, и возвращает вероятности для каждого класса. Этот метод является необязательным и актуален только для задач классификации.

Dataset  
Dataset — это контейнер для данных и соответствующих метаданных машинного обучения, таких как специальные роли столбцов (например, метка, индекс, категориальные столбцы). Он позволяет учитывать соответствующий контекст во время валидации и сохранять его удобным образом. Dataset является базовым блоком в deepchecks. Можно создать из Pandas DataFrame или NumPy Array.

**Properties класса:**

* label — Целевые значения, которые модель пытается предсказать.
* cat\_features — Список признаков, которые должны рассматриваться как категориальные. Если они не указаны явно, они будут определены автоматически.
* index\_name — Если набор данных имеет уникальный осмысленный индекс, его определение позволит запустить больше проверок.
* date\_name — Столбец даты, представляющий образец.
* features — Определяет столбцы, используемые моделью для обучения. Используется для указания только подмножества столбцов данных в качестве признаков. Если не указано, то все столбцы, которые явно не определены как метка, дата или индекс, считаются признаками.
* name — Имя набора данных, которое будет отображаться вместо "Train" или "Test".

## Запуск демо

Мой запуск full\_suite проверки в Google Colab

<https://colab.research.google.com/drive/1Ghlgd2TBQcrGxxaVaIamTmPYXA9_wnxn?authuser=1#scrollTo=V80StwGUGZSx>