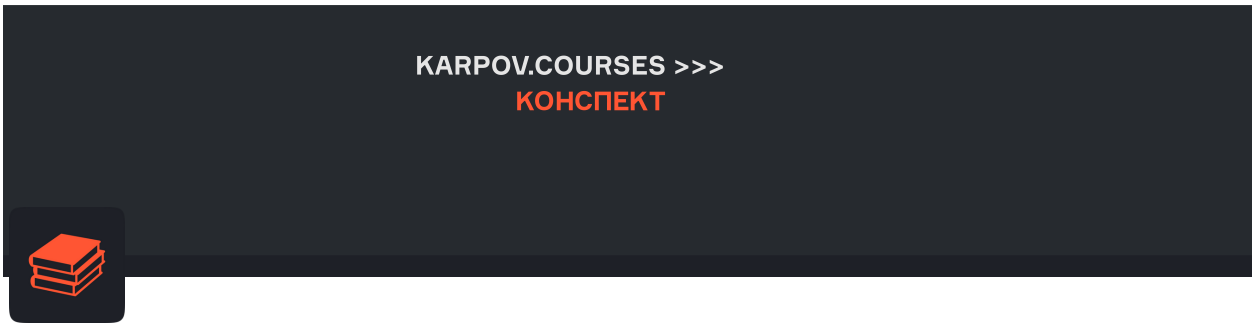


> Конспект > 3 урок > Spark ML и распределенное машинное обучение

> Оглавление

> Spark ML

Spark ML - это один из компонентов Spark, который позволяет работать с моделями машинного обучения, применять алгоритмы для обучения и производить оценку.

В данный момент под модулем Spark ML часто подразумевают один их двух вариантах:

- Spark MLlib (для Spark < 2.0, основная структура данных - RDD) - первый модуль МО, который был создан в спарке и существует до сих пор, но уже не развивается.
- Spark ML (для Spark ≥ 2.0, основная структура данных - DataFrame) - актуальная библиотека. Она более гибкая, позволяет эффективнее и проще применять модели к структурированным данным.

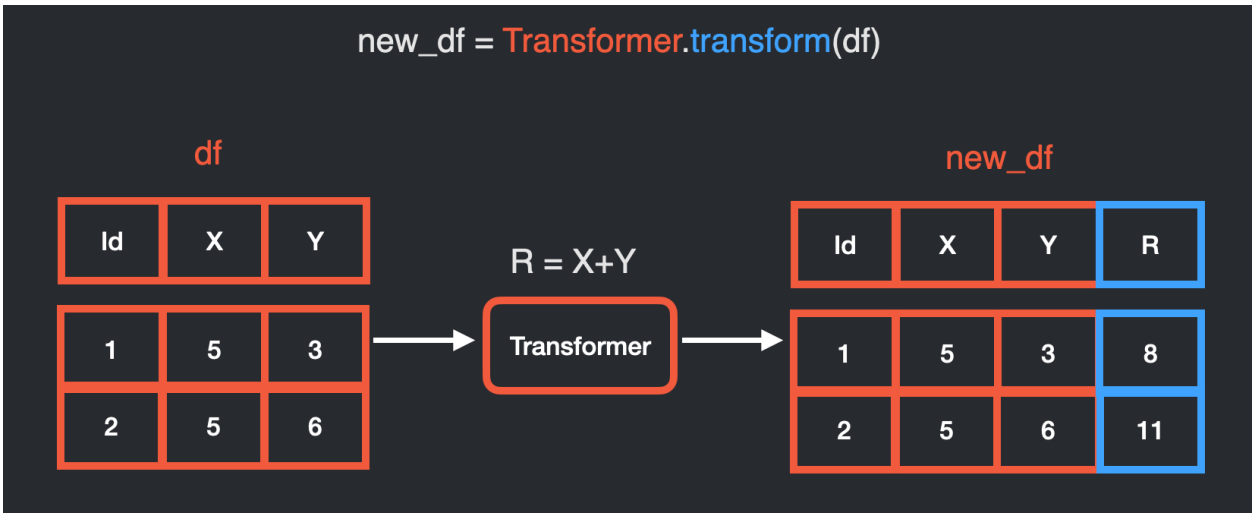
Далее мы будем изучать именно Spark ML.

> Компонент Transformer

Transformer - алгоритм преобразования одного набора данных в другой.

То есть трансформер - это функция, которая работает с колонками датафрейма и может формировать новую колонку.

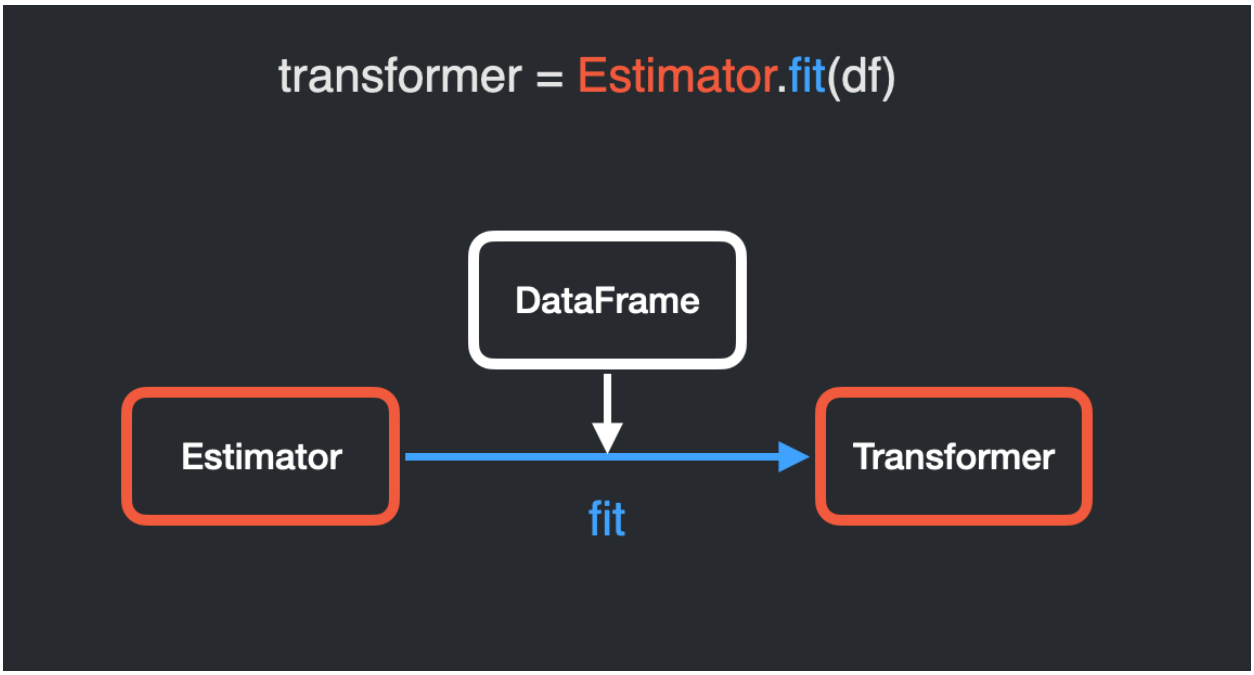
На примере видим некий датафрейм `df`, к которому применяемся трансформер. В итоге получается новый датафрейм `new_df`, в котором присутствует новая колонка `R`.



> Компонент Estimator

Estimator - алгоритм создания Transformer на основе данных.

У эстиматора есть метод `fit`, куда передается набор данных, результатом является трансформер.

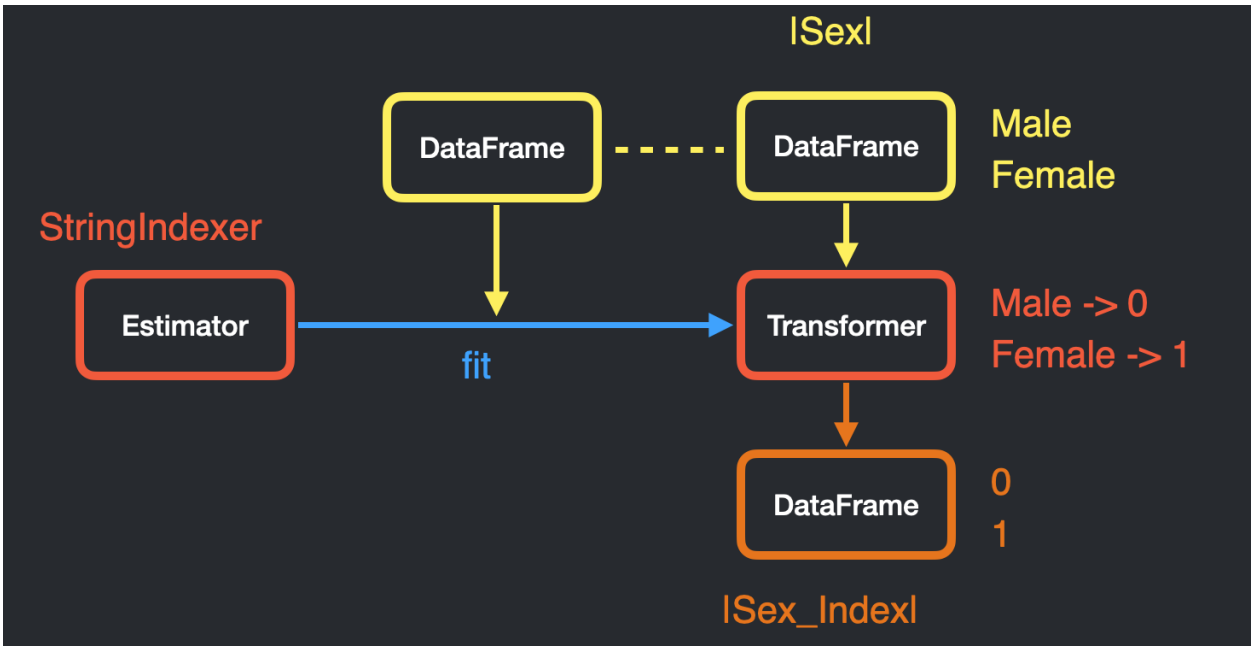


В спарке присутствует большое число эстиматоров. К примеру, там есть эстиматоры для энкодинга.

Encoding - процесс, с помощью которого признаки преобразуются в подходящую форму.

Представим, что в данных есть признак `sex`, в котором описан пол объектов. Пусть признак `sex` может принимать всего два значения - `male`, `female`. Чтобы модель могла работать с таким данными их нужно преобразовать в числовые значения. Операция энкодинга как раз и будет преобразовывать наши категориальные данные к числовым. Сам алгоритм преобразования нам не важен, но важно, чтобы он запомнил, как именно преобразовывал данные.

То есть нам нужно задать энкодер (вид эстиматора), указать какую колонку мы хотим преобразовать, передать датафрейм в метод `fit`. Эстиматор понимает какие возможны категории у данного признака, и определит алгоритм для энкодинга таких данных, то есть построит функцию, такую что `male → 0, female → 1`.

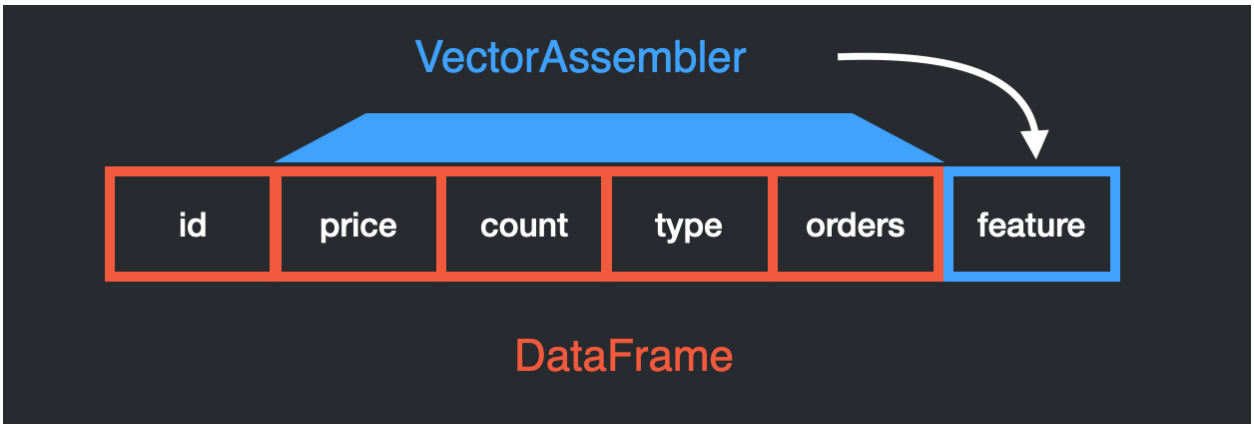


Таким образом для обучения моделей применяются эстиматоры, а для применения - трансформеры.

> Векторизация

Векторизация (embedding) - векторное представление данных.

Векторизация позволяет из данных строить некий признак. Например, мы можем применить трансформер VectorAssembler, который построит вектор `feature` из переданных данных. Этот `feature` - вектор потому будет удобно использовать в качестве входных параметров для модели.



> Компонент Pipeline

Pipeline - конвейер, объединяющий любое количество Transformer и Estimator для создания процесса машинного обучения.

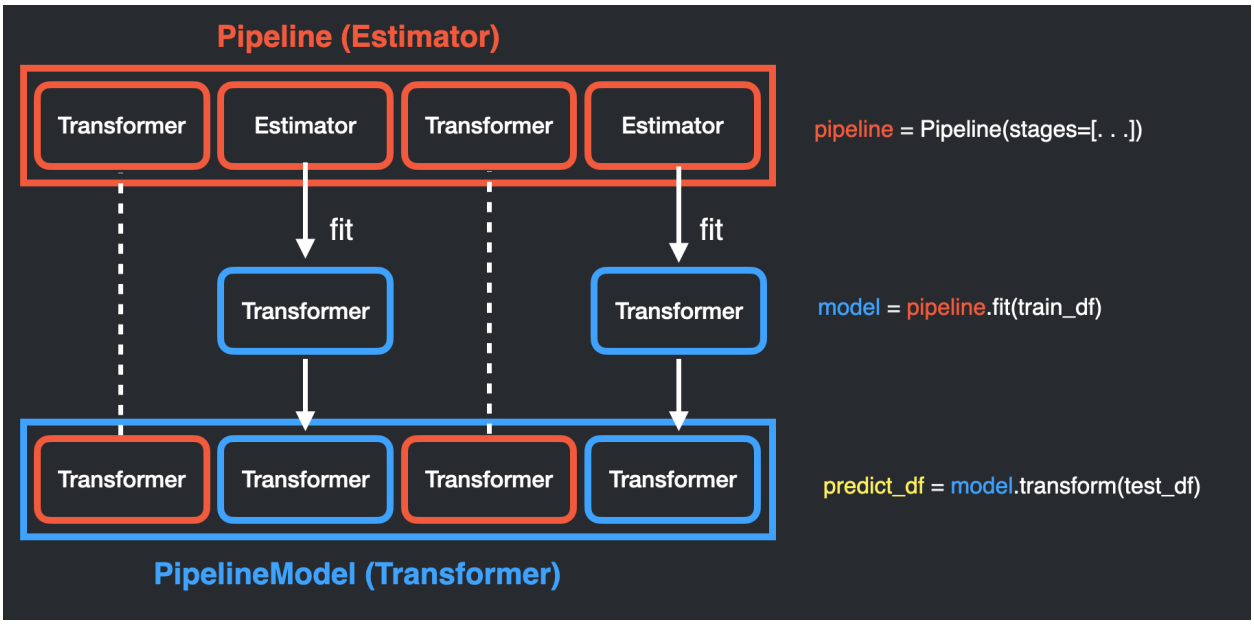
Пайплайны упрощают внедрение моделей, так как позволяют делать преобразования над данными частью самой модели.

Свойства:

- Задается в виде последовательности из Transformer или Estimator. Порядок важен;
- Любой Transformer созданный в результате работы Estimator, автоматически становятся частью Pipeline. То есть сам пайплайн на самом деле является более сложным эстиматором;
- Все компоненты являются Stateless, т.е не хранят состояние (пайплайн представляет из себя некий набор функций, которые будут последовательно применены к данным).

На картинке изображен пример пайпалайна.

Видим, что при его создании необходимо указывать `stages` - массив из трансформеров и эстиматоров. Так как пайплайн - это эстиматор, то у него мы можем вызвать метод `fit`, передавая входные данные. Таким образом мы получим желаемую модель, которая представляет из себя трансформер. Чтобы применить обученную модель над тестовыми данными мы вызываем метод `transform`.



Применение pipeline

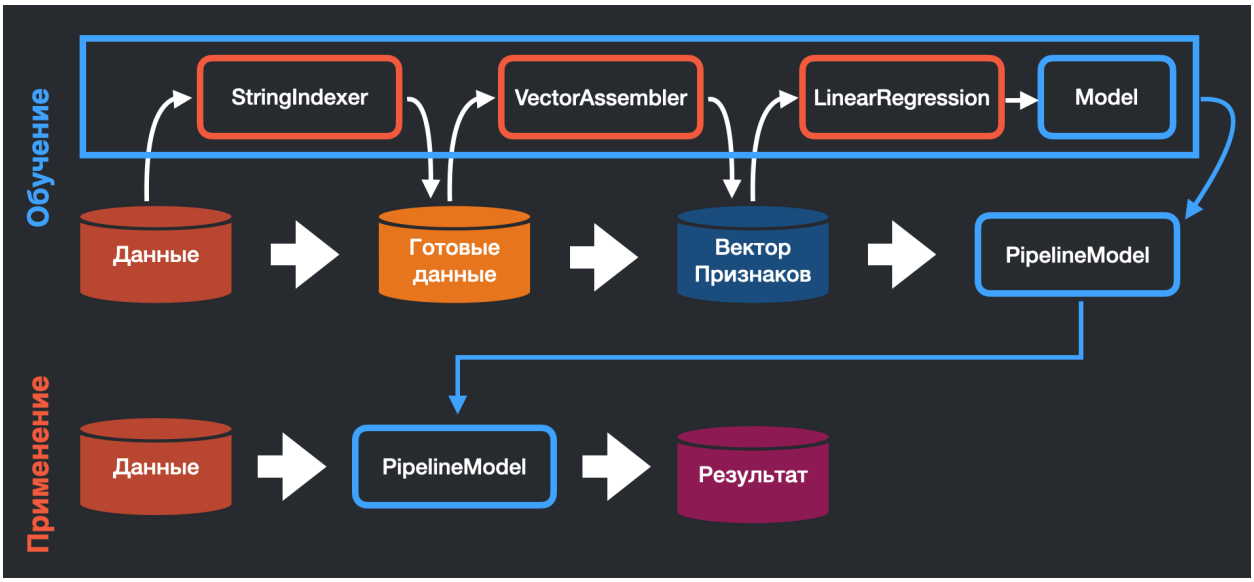
Имеем 2 процесса: процесс обучения и применения модели.

На схеме видим, такие шаги:

1. Применяем `StringIndexer` (нужен, например, для энкодинга данных);
2. Применяем `VectorAssembler` для векторизации подготовленных данных;
3. Применяем `LinerRegression` для обучения модели;

Шаги 1-3 представляют из себя шаги пайплайна. В результате обучения мы получим модель `PipelineModel`.

В результате мы можем применять обученную модель для получения результатов.



> Компонент Evaluator

Evaluator - оценщик качества модели согласно указанному алгоритму.

Типичные метрики для различных задач МО:

- Регрессия - MSE, RMSE, MAE, R2 (Коэффициент детерминации);
- Классификация - Accuracy, Precision, Recall, F-measure, ROC, AUROC, AUPRC;
- Ранжирование - Precision at K, MAP, NDCG.

Все эти метрики уже реализованы в Spark ML.

Для создания кастомных метрик нужно написать свой оценщик.

У оценщика должен быть метод `evaluate`, на вход которому подается датафрейм с колонкой с предсказаниями и колонка с лейблами фичей.

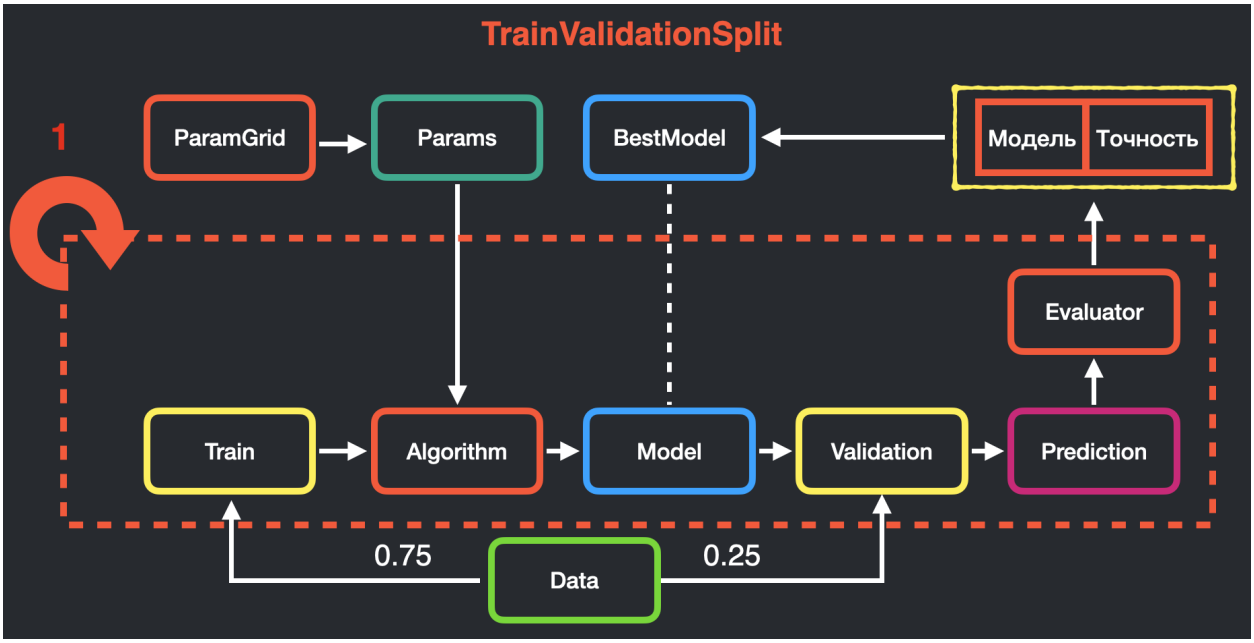
> Оптимизация гиперпараметров

Рассмотрим **TrainValidationSplit -** алгоритм, который рандомно и в заданном отношении делит входные данные на тренировочную и валидационную выборку, а также выполняет процесс оптимизации гиперпараметров.

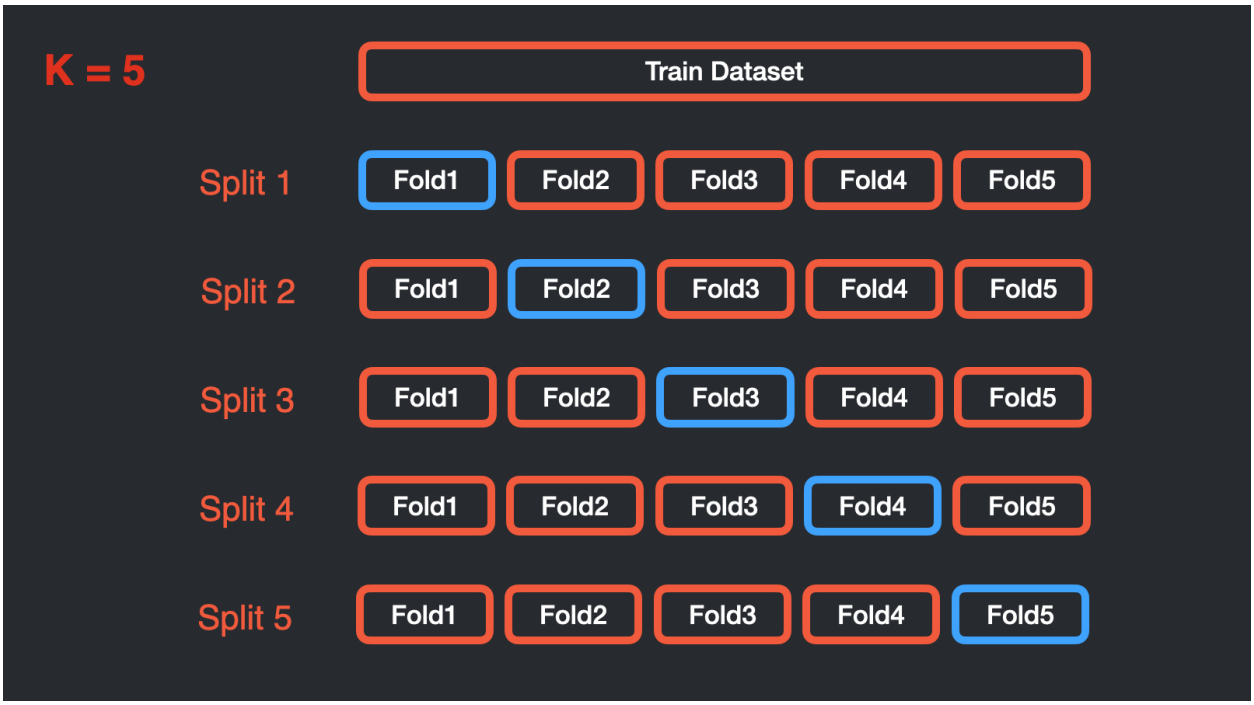
Для оптимизации параметров нам нужна сетка параметров - `ParamGrid`, в которой перечислены наборы параметров для поиска наилучшего.

Из сетки выбирается набор параметров `Params`, который передается в алгоритм для процесса обучения. На вход алгоритму подается сформированная тренировочная выборку. Мы получаем обученную модель и проверяем ее на валидационной выборке. Получаем предсказания и передаем и оценщику для оценки точности.

Проделываем данные шаги для всех наборов параметров из `ParamGrid`. В итоге выбираем набор гиперпараметров, который использовался в модели с наилучшей оценкой точности.

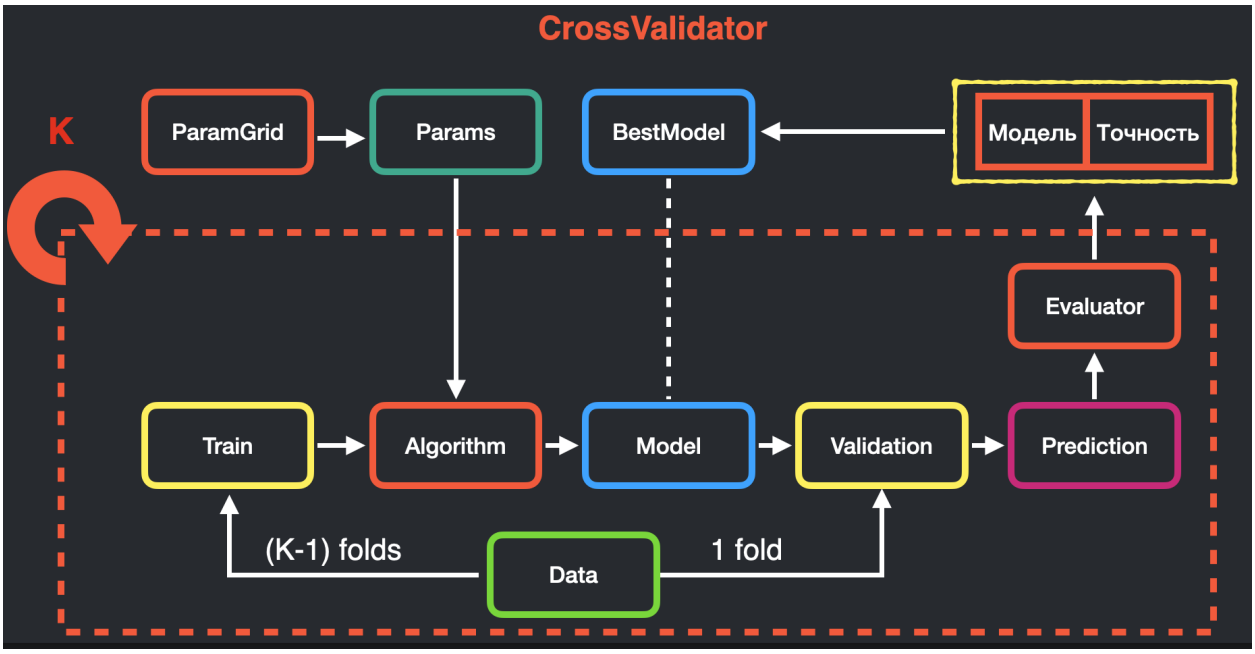


Кросс-валидация - алгоритм, который позволяет разбивать датафрейм на указанное количество фолдов (на картинке это параметр `k`). Один из фолдов (синий) будет выступать в роли валидационной выборки, а остальные в качестве тренировочной. Процесс повторяется до тех пор, пока каждый из фолдов не будет выступать в роли валидационного.



Оптимизация гиперпараметров с использованием кросс-валидации реализован в **CrossValidator**. Обучение моделей с различными параметрами происходит аналогично описанному выше в TrainValidationSplit. Только в этом случае входные данные разбиваются согласно алгоритму кросс-валидации на фолды.

С точки зрения производительности кросс-валидация является более сложным алгоритмом, требует больше ресурсов, но все же применяется достаточно часто, так как выдает высокую точность модели.



> Поддерживаемые классы моделей

Классы задач, которые можно решать с помощью модуля Spark ML:

- Регрессия
- Кластеризация
- классификация
- Деревья решений
- Ансамбли деревьев (GBT)
- Collaborative filtering (ALS)

> Глоссарий

Spark ML - это один из компонентов Spark, который позволяет работать с моделями машинного обучения, применять алгоритмы для обучения и производить оценку.

Компоненты Spark ML:

- Transformer - алгоритм преобразования одного набора данных в другой.

- Estimator - алгоритм создания Transformer на основе данных.
- Pipeline - конвейер, объединяющий любое количество Transformer и Estimator для создания процесса машинного обучения.
- Evaluator - оценщик качества модели согласно указанному алгоритму.