KARPOV.COURSES >>>
КОНСПЕКТ

> Конспект > 5 урок > Лекция: Применение не распределенных моделей МО на больших данных

> Оглавление

> Spark UDF

UDF (User Define Function) - это функция Spark, которая позволяет пользователям определять свои собственные функции.

Пример создания UDF:

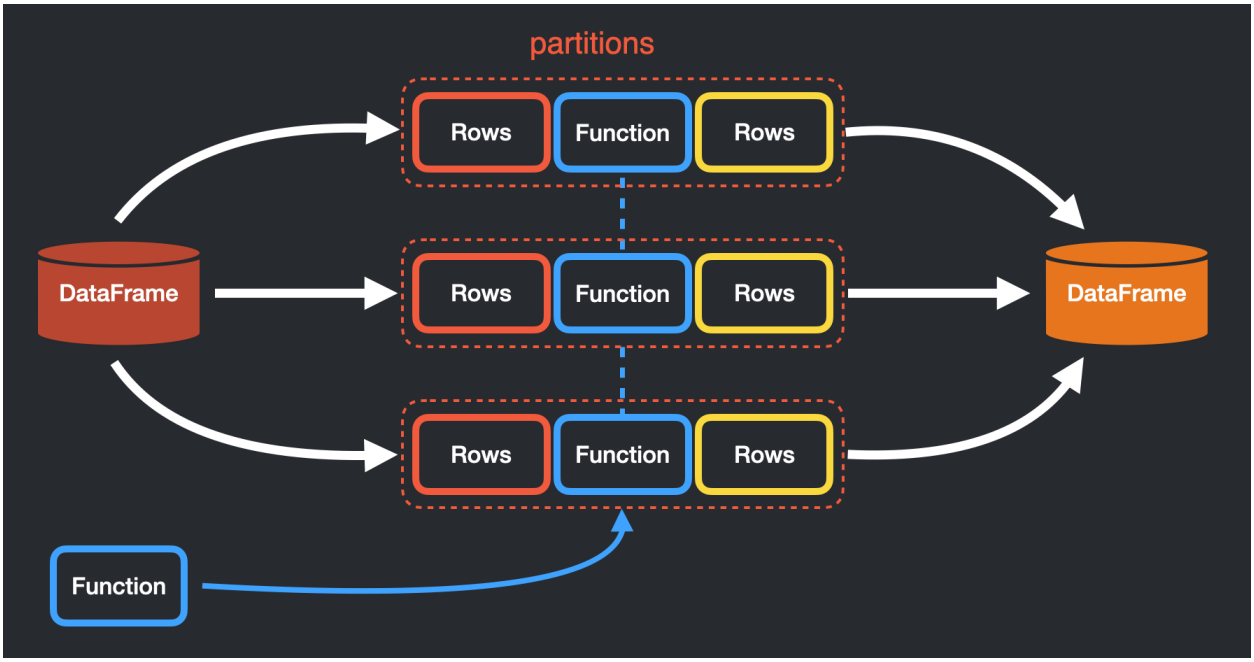
```
from pyspark.sql.functions import udf

@udf('double')      # тип возвращаемого значения
def plus_one(v):
    return v + 1
```

```
df.withColumn('plus_one', plus_one(df.value))
```

Разберем, как UDF применяется к данным.

На входе мы имеем датафрейм, который разделен на партиции. Партиции состоят из определенного числа строк. Чтобы применять UDF к датафрейму, требуется доставить ее на каждую из партиций, где UDF будет применена к каждой строчке партиции. В результате применения функции мы получаем новую строку, такие строки затем будут объединены в новый набор строк определенной партиции, из которых будет состоять новый датафрейм.



> Pandas UDF

Еще один вариант создания UDF - это использование pandas UDF.

Apache Arrow - это столбчатый формат данных (InMemory), который используется в Spark для эффективной передачи данных между процессами JVM и Python.

Pandas UDF (User Define Function) - функции на основе Apache Arrow, предоставляют возможность полностью определять высокопроизводительные пользовательские функции с низкими накладными расходами на Python.

Пример создания pandas UDF:

```
from pyspark.sql.functions import pandas_udf

@pandas_udf('double')
def plus_one(v):
    return v + 1

df.withColumn('plus_one', plus_one(df.value))
```

Стоит помнить о том, что в случае создания обычной UDF функции мы работает с одной строчкой, а в случае работы с pandas_udf - с pandas series, то есть с набором значений, что позволяет за раз преобразовывать целую пачку данных.

Pandas UDF предоставляет возможность использовать функции для работы с pandas series, это позволяет реализовывать функции более эффективно.

Пример использования функции `mean`

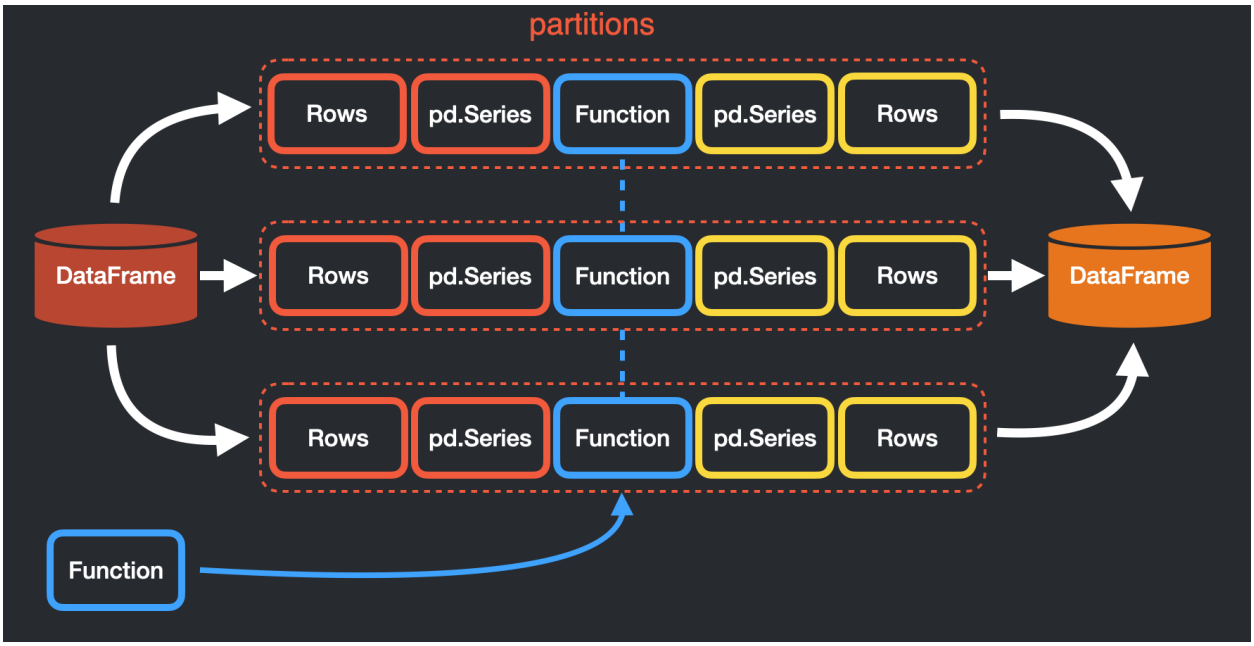
```
from pyspark.sql.functions import pandas_udf, PandasUDFType

@pandas_udf('double', PandasUDFType.SCALAR)
def pudf_mean(v: pd.Series) -> float:
    return v.mean()

df.withColumn('mean', pudf_mean(df.value))
```

Разберем, как pandas UDF применяется к данным.

Написаная pandas_udf доставляется на каждую из партиций входного датафрейма. Строки преобразуются в pandas серию, которая подается на вход udf. Результатом выполнения функции является pandas серия. Строки из таких серий затем объединяются для получения нового датафрейма.



> Типы Pandas UDF

Два важных типа pandas_udf - это скалярные и группировочные функции.

Скалярная функция на вход получает серию и результатом ее выполнение тоже будет pandas серия. Для такой функции необходимо определять результирующий тип.

Группировочный функции на вход получает датафрейм и результатом ее выполнение тоже будет датафрейм. Для такой функции необходимо определять возвращаемую структуру.

	SCALAR	GROUPED_MAP
Input	pandas.Series	pandas.DataFrame
Output	pandas.Series	pandas.DataFrame
Output size	same as input size	any size
Return Size	DataType (double, int ...)	StructType

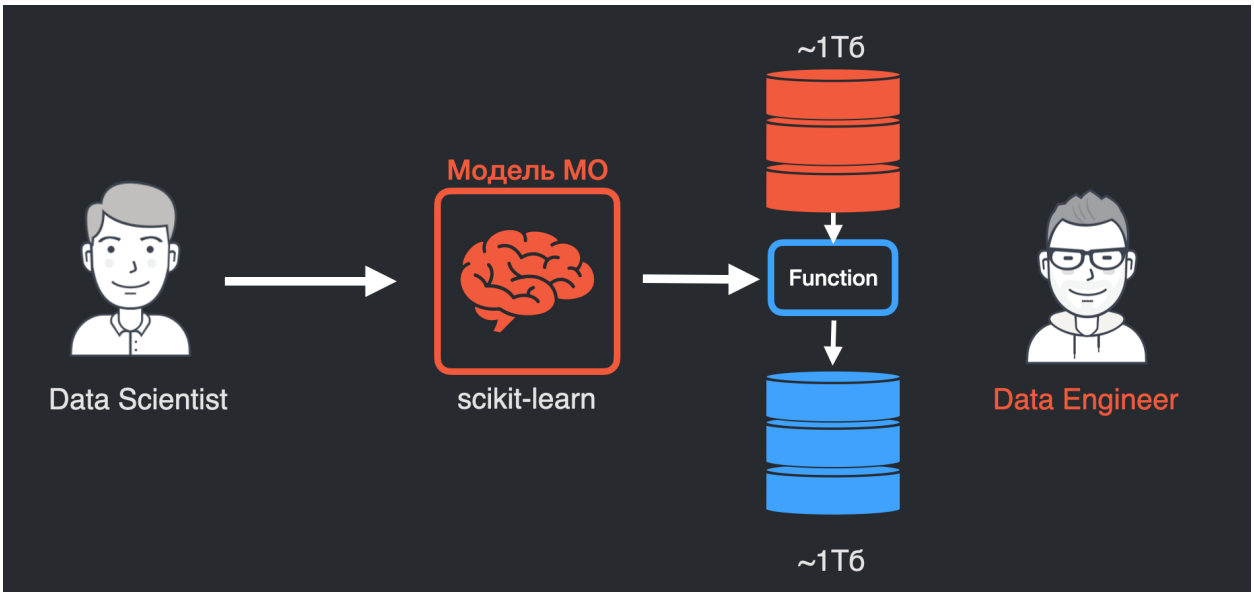
 > Конспект > 1 урок > Введение в машинное обучение

> BIG DATA & ML

Предположим, что у data scientist есть запрос на применение некоторой модели МО к большому объему данных. Сама модель является не распределенной, например, модель из scikit-learn, tensorflow, pytorch...

Для применения такой модели ее нужно завернуть в UDF функцию, где она сможет эффективно применятся к данным.

В данном случае использование pandas_udf может быть эффективнее, так как мы будем обрабатывать сразу серии, и плюсом является то, что pandas серии могут быть легко сконвертированы в numpy array или в tensor.



> Глоссарий

Spark UDF (User Define Function) - это функция Spark, которая позволяет пользователям определять свои собственные функции.

Apache Arrow - это столбчатый формат данных (InMemory), который используется в Spark для эффективной передачи данных между процессами JVM и Python.

Pandas UDF (User Define Function) - функции на основе Apache Arrow, предоставляют возможность полностью определять высокопроизводительные пользовательские функции с низкими накладными расходами на Python.