Глоссарий

Распределённые системы хранят файлы с данными на нескольких компьютерах и предоставляют пользователям доступ к данным. Файлы делятся на фрагменты, причём каждый фрагмент может быть сохранён несколько раз на разных компьютерах. Так гарантируется целостность данных. Распределённая система состоит из нескольких **узлов** (англ. nodes). Это отдельные компьютеры с ресурсами вычисления и хранения данных.

Apache Spark — фреймворк для распределённых вычислений с открытым исходным кодом. Для работы в Python создатели Apache Spark выпустили библиотеку **PySpark**.

RDD (англ. Resilient Distributed Dataset) — отказоустойчивый распределённый набор данных, ключевой тип данных в Spark.

Этапы решения задач машинного обучения

```
from pyspark import SparkContext

sc = SparkContext(appName='appName')
weather_entry = sc.parallelize(['2009-01-01', 15.1, 26.1]) # создание RDD-объекта
print(weather_entry.take(3)) # вывод элементов из RDD
```

Датафрейм в PySpark — таблица, строки которой хранятся в RDD. Это неизменяемый тип данных, работа с которым ведётся по принципу «ленивых вычислений» — вычислений, которые производятся не сразу, а только тогда, когда понадобится результат.

Пример работы датафреймов в PySpark

```
import numpy as np
import pandas as pd
from pyspark.sql import SparkSession

APP_NAME = "DataFrames"
SPARK_URL = "local[*]"

spark = SparkSession.builder.appName(APP_NAME) \
        .config('spark.ui.showConsoleProgress', 'false') \
        .getOrCreate()

taxi = spark.read.load('/datasets/pickups_terminal_5.csv',
                       format='csv', header='true', inferSchema='true') # чтение файла

print(taxi.describe().show()) # выводим информацию о столбцах датафрейма
```

Пример использования SQL-запросов к PySpark-датафрейму

```
from pyspark.sql import SparkSession

APP_NAME = "DataFrames"
SPARK_URL = "local[*]"

spark = SparkSession.builder.appName(APP_NAME) \
        .config('spark.ui.showConsoleProgress', 'false') \
        .getOrCreate()

taxi = spark.read.load('/datasets/pickups_terminal_5.csv',
                       format='csv', header='true', inferSchema='true')

taxi = taxi.fillna(0)

taxi.registerTempTable("taxi") # создание временной таблицы для обращения через SQL

print(spark.sql("SELECT COUNT(*) FROM taxi WHERE pickups > 100").show()) # вывод результата SQL-запроса
```

Пример группировок и агрегаций в PySpark-датафреймах

```
from pyspark.sql import SparkSession

APP_NAME = "DataFrames"
SPARK_URL = "local[*]"

spark = SparkSession.builder.appName(APP_NAME) \
        .config('spark.ui.showConsoleProgress', 'false') \
        .getOrCreate()

taxi = spark.read.load('/datasets/pickups_terminal_5.csv',
                       format='csv', header='true', inferSchema='true')

taxi = taxi.fillna(0)

print(taxi.groupBy("date").mean().select("date", "avg(pickups)").show())
```