Lab 3 PySpark

Большие данные и рекомендательные системы
сдается до 7 ноября!

Требования к заданию:
- Отчёт сдаётся в виде jupyter notebook + выслать тетрадку в формате html/pdf. Ноутбук должен быть аккуратным и читаемым (а не черновиком, где ячейки запущены в произвольном порядке, а результаты не воспроизводятся)

Исходные данные:
Используется датасет MovieLens 1M Dataset (1 million ratings from 6000 users on 4000 movies).

Доступ: ml-1m.zip
Нам понадобятся файлы: movies.dat, ratings.dat.

Задание -1: Теория:

Развернуто ответьте на вопросы (мне важно не чтобы вы просто вбили это в ИИ и скопировали текст, а чтобы хотя бы два раза его прочитали с целью осознания минимальной теории):

- Что такое PySpark и в чем его главное отличие от Pandas?
- Что такое PySpark MLlib и в чем ее главное отличие от Sklearn?
- В чем концепция "Ленивых вычислений" (Lazy Evaluation) в PySpark?
- В чем разница между "Трансформациями" (Transformations) и "Действиями" (Actions)?
- Что такое RDD, DataFrame и SparkSession?
- Как устроена базовая архитектура Spark-приложения (Driver и Executors)?
- Что такое Apache Parquet и зачем он нужен?

Задание 0: Подготовка:

- Установите PySpark (pip install spark) и сделайте импорты:

```python
import pyspark
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, avg, count, split, explode, lit
from pyspark.sql.types import StructType, StructField, StringType, IntegerType,
FloatType
```

- Создайте Spark сессию:

```python
spark = ( SparkSession.builder
.appName("MovieLens Lab")
.master("local[*]")
.getOrCreate()
)
```

- Отключите излишнее логирование:

```python
spark.sparkContext.setLogLevel("ERROR")
```

Задание 1: Предобработка:

- Загрузите файлы movies.dat и ratings.dat в Spark DataFrame (Не пугайтесь непонятного формата файла, так все и трэба, вам просто нужно явно указать разделитель (`::`) и задать схему данных (названия колонок), так как в файлах нет заголовков. Используйте StructType и StructField. Это лучшая практика по сравнению с inferSchema);
- После загрузки и перед началом любых трансформаций, закэшируйте DataFrame: ratings_df.cache(). **Объясните, зачем это нужно;**
- Столбец “Жанры” в movies имеет неудобный формат: "Adventure|Animation|Children". Намного удобнее работать, если

каждый жанр будет в отдельной строке. Разделите строку на массив по разделителю и “взорвите” массив.

Задание 2: Аналитика:

- Найдите 10 самых популярных жанров;
- Найдите еще топ 10 чего-то на свой вкус.

Задание 3: Объединение данных (Join):

- Найти 10 самых популярных фильмов по количеству оценок;
- Найдите 10 лучших фильмов (самый высокий средний рейтинг), у которых не менее 500 оценок;
- Придумайте как вы можете осмысленно показать разницу между видами джоинов (inner, left, right, outer и cross). Используйте для этого маленькие тестовые таблицы.

Задание 4: Spark SQL:

- Создайте временные таблицы из DataFrame;
- Выполните задание (топ-10 фильмов с >= 500 оценок), используя spark.sql().

Задание 5: Spark MLlib:

PySpark имеет встроенную библиотеку машинного обучения (MLlib). Мы используем самый популярный алгоритм для рекомендаций — ALS (Alternating Least Squares, Метод попеременных наименьших квадратов).

- Создайте модель ALS:

```python
als = ALS(
maxIter=5,  # Количество итераций
```

```
regParam=0.01,  # Параметр регуляризации
userCol="userID", # Название условное
itemCol="movieID", # Название условное
ratingCol="rating", # Название условное
coldStartStrategy="drop" # Игнорировать "холодных" пользователей/фильмы в тесте
)
```

- Обучите модель на обучающей выборке (80%), сделайте предсказание на тестовой выборке (20%);
- Оцените качество по среднеквадратичной ошибке.

Задание 6: Персонализация рекомендаций:

- Выберите пользователя (id) и получите топ 10 фильмов для него. Используйте метод .recommendForUserSubset(... , 10).

ЗАВЕРШИТЕ РАБОТУ СПАРК СЕССИИ!!! и расскажите зачем это делать.