Департамент образования и науки города Москвы
Государственное автономное образовательное учреждение
высшего образования города Москвы
«Московский городской педагогический университет»
Институт цифрового образования
Департамент информатики управления и технологий

Мареев Георгий Александрович БД-241м

Практическая работа 3.1 Анализ и визуализация больших данных. Машинное обучение на больших данных с использованием Apache Spark MLlib

Вариант 13

Направление подготовки/специальность
38.04.05 - Бизнес-информатика
Бизнес-аналитика и большие данные
(очная форма обучения)

Руководитель дисциплины:
Босенко Т.М., доцент департамента
информатики, управления и технологий,
кандидат технических наук

Москва
2025

Введение

Цели и задачи:

- Познакомиться с понятием «большие данные» и способами их обработки;
- Познакомиться с инструментом Apache Spark и возможностями, которые он предоставляет для обработки больших данных.
- Получить навыки выполнения разведочного анализа данных использованием pyspark.

Основная часть

Блокнот доступен на GitHub по *ссылке*

Теоретические вопросы

1. Фреймворк обработки больших данных Apache Spark, его назначение, функции и отличия от Hadoop MapReduce.

Apache Spark — фреймворк для распределенной обработки больших данных. Он оптимизирован для скорости (в памяти), поддерживает пакетную обработку, потоковую аналитику (Spark Streaming), машинное обучение (MLlib), обработку графов (GraphX).
Функции:

- In-memory computing: данные хранятся в оперативной памяти между этапами обработки, что ускоряет итеративные алгоритмы (например, ML).
- Поддержка SQL: работа с структурированными данными через DataFrames/Datasets.
- Интеграция с экосистемой Hadoop: чтение/запись данных из HDFS, HBase, Hive.
- API на Java, Scala, Python, R .

Отличия от Hadoop MapReduce: быстрее (за счет in-memory), модель вычислений DAG, поддерживает операции трансформации и действия, потоковая обработка, встроенная поддержка кэширования.

2. Понятие устойчивого распределенного набора данных (RDD). Понятие раздела RDD (partition). Способы создания RDD. Трансформации (transformations) и действия (actions). Кэширование (cache) данных в Spark.

RDD – основная абстракция Spark, неизменяемый распределенный набор данных, разбитый на партиции (partitions), которые обрабатываются на разных узлах кластера.
Способы создания: параллельные коллекции, чтение из внешних источников.
Трансформации: ленивые операции (map(), filter(), flatMap(), groupByKey(), reduceByKey()).
Действия: запускают выполнения DAG и возвращают результат (count(), collect(), reduce(), saveAsTextFile()).
Кэширование: rdd.cache() или rdd.persist() сохраняют RDD в памяти (или на диске) для повторного использования.

3. RDD и PairRDD: понятие, назначение. Основные трансформации (transformations) и действия (actions) над ними.

PairRDD — RDD, состоящий из пар (ключ, значение). Используется для агрегаций и соединений.
Трансформации:

- reduceByKey(func): агрегация по ключу.
- groupByKey(): группировка значений по ключу.
- join(otherRDD): соединение двух RDD по ключу.
- mapValues(func): применение функции к значениям.

Действия:

- countByKey(): подсчет количества элементов по ключу.
- lookup(key): поиск всех значений для ключа.

4. Реализация концепции MapReduce в фреймворке Spark. Функции map, flatMap, mapValues, mapPartitions, reduce, reduceByKey.

Функции:

- map(func): преобразует каждый элемент.
- flatMap(func): возвращает 0 или более элементов на каждый входной.
- mapValues(func): применяет функцию к значениям в PairRDD.
- mapPartitions(func): применяет функцию к целому партиции.
- reduce(func): агрегирует все элементы с помощью ассоциативной функции.
- reduceByKey(func): агрегирует значения по ключу.

5. Модели запуска Spark-приложений (YARN, Standalone, Kubernetes). Понятие драйвера (driver) и исполнителей (executors). Понятия задания (job), этапа (stage) и задачи (task). Модель ленивых вычислений (lazy) и ее применение в Spark. Понятие ориентированного ациклического графа (Directed Acyclic Graph, DAG).

Модели: Standalone, YARN, Kubernetes.
Компоненты:

- Драйвер (Driver): управляет приложением (создает SparkContext).
- Исполнители (Executors): выполняют задачи и хранят данные.

Выполнение:

- Job: задача, порожденная действием (например, count()).
- Stage: этапы, разделенные shuffle.
- Task: отдельная работа на партици

Ленивые вычисления: Spark откладывает выполнение до первого действия. План выполнения строится как DAG (ориентированный ациклический граф).

6. Операция перемешивания данных (shuffle): причины возникновения, влияние на производительность. Класс Partitioner. Пути повышения производительности.

Shuffle — перемещение данных между узлами (например, при groupByKey или join).
Из проблем можно выделить: высокую задержку из-за сериализации/десериализации, сетевой трафик.
Оптимизировать можно путем увеличения числа партиций, использованием broadcast для маленьких RDD, кэшированием промежуточных результатов, настройкой spark.sql.shuffle.partitions.

7. Понятие датафрейма в Spark. Основные операции над датафреймами. Скалярные функции, агрегирующие функции, оконные функции. Оптимизация запросов. Catalyst.

DataFrame — распределенная коллекция данных с схемой (структура и типы данных).
Операции:

- Скалярные функции: select("column"), filter("age > 30").
- Агрегирующие функции: groupBy("category").agg(sum("price")).
- Оконные функции

Catalyst — оптимизатор запросов, который:

- Анализирует логический план.
- Применяет оптимизации (например, предикативная выборка).
- Генерирует физический план выполнения.

8. Клиентский (client) и кластерный (cluster) режимы работы Apache Spark.

Client режим: драйвер запускается на машине, откуда отправлена задача, подходит для отладки.
Cluster режим: драйвер запускается внутри кластера (на одном из узлов), используется в production.

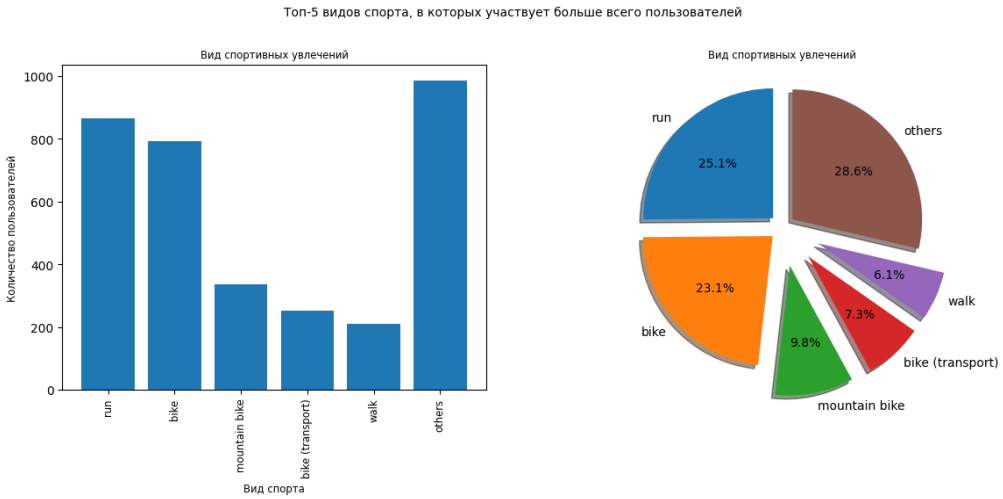
Индивидуальное задание. Вариант 13

1. Как вы думаете, почему в данных присутствует пол 'unknown' (раздел 3)? Предложите 2-3 возможные причины.
2. Сравните гистограммы PerWorkoutRecordCount для 'running' и 'indoor cycling' (раздел 5). Какие выводы можно сделать о типичной плотности записи данных для этих активностей?
3. Представьте, что Endomondo хочет запустить таргетированную кампанию для пользователей, занимающихся несколькими видами спорта. Какие данные из анализа в разделе 6 помогут сегментировать этих пользователей?
4. Напишите код PySpark, чтобы отобрать 10 случайных строк из датафрейма df (используйте sample или limit после orderBy(rand()))
5. Можно ли использовать Spark MLlib для построения рекомендательной системы (например, на основе коллаборативной фильтрации), которая предлагает пользователям виды спорта, популярные среди похожих на них пользователей? Какие данные (userId, sport, возможно, duration) для этого нужны?

Задание 1. Пол «unknown»

1) При регистрации пользователям не обязательно заполнять поле «пол»/ есть другие опции помимо «мужского»/ «женского» и им автоматически присвоилось значение «unknown»
2) Произошли технические сбои/сбой и данные были некорректно переданы/сохранены

Задание 2. «Сравните гистограммы PerWorkoutRecordCount для 'running' и 'indoor cycling' (раздел 5). Какие выводы можно сделать о типичной плотности записи данных для этих активностей?»



Можно предположить, что
Running – бег, требует мониторинга множества параметров (GPS-трекинг, пульс, шагомер…)

Indoor cycling – езда на велосипеде в помещении, требует меньшего кол-ва параметров (так как, например, такие параметры как GPS не нужны)
Из чего можно сделать вывод, что running имеет плотность записи выше, чем indoor cycling.

Задание 3. «Таргетированная кампания для пользователей, занимающихся несколькими видами спорта»

Могут помочь данные из раздела «…Сколько людей участвовало более чем в одном виде спорта»

```
min_number_of_sports = 1

sport_df = df \
    .select(df.userId, df.gender, df.sport) \
    .distinct() \
    .groupBy(df.userId, df.gender) \
    .count()

user_more_sports_df = sport_df \
                    .filter(sport_df["count"] > min_number_of_sports) \
                    .orderBy("count", ascending = False) \
                    .toPandas()
user_more_sports_df.rename(columns = {'count':'Sports count'}, inplace = True)
user_more_sports_df.describe().astype(int).T
```

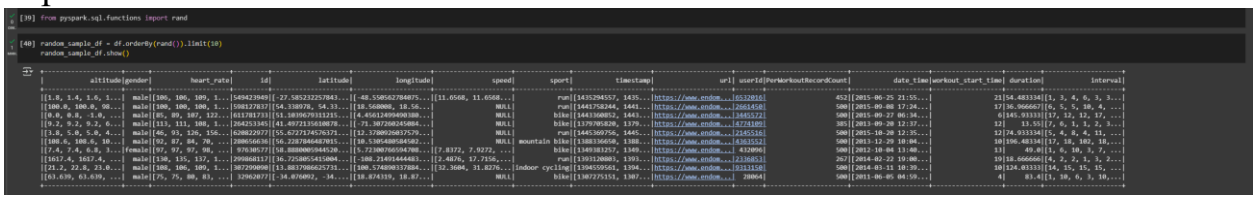
	count	mean	std	min	25%	50%	75%	max
userId	822	4860464	3953412	69	1609606	3730685	7554937	15481421
Sports count	822	3	2	2	2	3	5	16

Где 822 человека занимаются разными видами спорта, минимум 2 вида спорта, среднее 3, максимум 16.

Задание 4. «10 случайных строк из датафрейма df (sample или limit после orderBy(rand()))»

```
from pyspark.sql.functions import rand
random_sample_df = df.orderBy(rand()).limit(10)
random_sample_df.show()
```

Функция rand для создания случайного порядка строк в датафрейме.
orderBy(rand()) – каждая строка получает случайное значение от rand(), а затем происходит сортировка строк в соответствии с этими случайными значениями
limit(10) – ограничивает кол-во строк в результирующем датафрейме до первых 10



Задание 5. «Можно ли использовать Spark MLlib для построения рекомендательной системы (например, на основе коллаборативной фильтрации), которая предлагает пользователям виды спорта, популярные среди похожих на них пользователей? Какие данные (userId, sport, возможно, duration) для этого нужны?»

Да, можно.
Коллаборативная фильтрация, которая подсказывает предпочтения пользователей на основе поведения других пользователей.
Для этого можно использовать ALS (Alternating Least Squares).
Для этого могут понадобиться такие данные, как

- userId – уникальный идентификатор пользователя
- sport – идентификатор/название вида спорта
- duration – продолжительность занятия
- timestamp – время занятия
- rating – оценка, которая отражает вероятность того, что пользователь заинтересуется видом спорта (чем выше значение, тем выше релевантность)

Если мало данных о пользователях, то можно использовать рекомендации на основе характеристик пользователя (возраст, пол) и видов спорта.

Заключение

В результате изучения были достигнуты поставленные цели: сформировано понимание концепции «больших данных» и методов их обработки, освоены базовые возможности Apache Spark как инструмента для распределенных вычислений, а также приобретены практические навыки проведения разведочного анализа данных с использованием PySpark, включая работу с RDD, DataFrame и реализацию алгоритмов машинного обучения