BÜYÜK VERİ KAGGLE PROJESİ

AD SOYAD:ASUDE YURT

NO:G211210035

Bu projede kullanılan veri seti, Portekiz'de iki farklı lisede okuyan öğrencilerin akademik başarıları, demografik bilgileri, aile yapıları, sosyal alışkanlıkları ve okul performansı ile ilgili bilgiler içeren bir öğrenci performansı veri setidir. Dataset, özellikle öğrencilerin matematik dersindeki başarılarını etkileyen faktörleri analiz etmek amacıyla hazırlanmıştır.

Veri seti toplamda 33 öznitelik (kolon) içermekte olup her bir satır bir öğrenciyi temsil eder.

Veri Setinde Yer Alan Temel Kolonlar

- **Demografik Bilgiler:**
  age, sex, address, famsize, Pstatus
- **Aile ve Eğitim Bilgileri:**
  Medu (anne eğitim durumu), Fedu (baba eğitim durumu),
  Mjob, Fjob, guardian (veli), famsup (aile desteği)
- **Okul İçi Faktörler:**
  studytime, failures, schoolsup, absences
- **Sosyal ve Kişisel Bilgiler:**
  romantic, activities, internet,
  goout (arkadaşlarla dışarı çıkma),
  Dalc (günlük alkol tüketimi), Walc (haftalık alkol)
- **Başarı Notları:**
  G1, G2, G3 (dönem içi performans ve final notu)

Veri setinin hedef değişkeni (label) genellikle **G3 – final notu** olarak kabul edilmektedir. Ancak bu projede Spark kullanılarak yeni türetilmiş değişkenler (ör. total_score = G1+G2+G3) ile farklı analizler de yapılmıştır.

Hadoop ve Hive Kullanımı

- **Amaç:** Büyük veri dosyalarını HDFS üzerinde depolamak ve Hive ile sorgulamak.

HDFS: CSV dosyasını /user/asude/student/ dizinine yükledim.
Dosyaların HDFS üzerinde doğru şekilde listelenip listelenmediğini **hdfs**

dfs -ls komutu ile kontrol ettim.

```
asude@ubuntu:~$ start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [ubuntu]
asude@ubuntu:~$ jps
4928 DataNode
5286 Jps
5110 SecondaryNameNode
4798 NameNode
asude@ubuntu:~$ start-yarn.sh
Starting resourcemanager
Starting nodemanagers
asude@ubuntu:~$ hdfs dfs -mkdir -p /user/asude/student
```

```
asude@ubuntu:~$ hdfs dfs -put /home/asude/student/student-mat.csv /user/asude/st
udent/
asude@ubuntu:~$ hdfs dfs -ls /user/asude/student
Found 1 items
-rw-r--r--   1 asude supergroup      56993 2025-12-02 04:32 /user/asude/student/
student-mat.csv
```

Hive: Database oluşturduktan sonra CSV dosyasını Hive'da external table olarak oluşturdum ve Hive üzerinden temel sorgular gerçekleştirdim.

```
asude@ubuntu:~$ hive
Hive Session ID = e5cff36c-b0c6-4680-a91e-3c9f9c72a71e

Logging initialized using configuration in jar:file:/home/asude/hive-3.1.3/lib/h
ive-common-3.1.3.jar!/hive-log4j2.properties Async: true
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versio
ns. Consider using a different execution engine (i.e. spark, tez) or using Hive
1.X releases.
Hive Session ID = 85447705-c549-4421-96e0-04c89f03f1b6
hive> CREATE DATABASE IF NOT EXISTS student_db;
OK
Time taken: 1.89 seconds
hive> USE student_db;
OK
```

```
hive> CREATE EXTERNAL TABLE student_mat (
    >     school STRING,
    >     sex STRING,
    >     age INT,
    >     address STRING,
    >     famsize STRING,
    >     Pstatus STRING,
    >     Medu INT,
    >     Fedu INT,
    >     Mjob STRING,
    >     Fjob STRING,
    >     reason STRING,
    >     guardian STRING,
    >     traveltime INT,
    >     studytime INT,
    >     failures INT,
    >     schoolsup STRING,
    >     famsup STRING,
    >     paid STRING,
    >     activities STRING,
    >     nursery STRING,
    >     higher STRING,
    >     internet STRING,
    >     romantic STRING,
    >     famrel INT,
    >     freetime INT,
    >     goout INT,
    >     Dalc INT,
    >     Walc INT,
    >     health INT,
    >     absences INT,
    >     G1 INT,
    >     G2 INT,
    >     G3 INT
    > )
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ';'
    > STORED AS TEXTFILE
    > LOCATION '/user/asude/student/';
OK
```

```
hive> SHOW TABLES;
OK
student_mat
Time taken: 0.354 seconds, Fetched: 1 row(s)
```

Sorgular: Basit ve karmaşık sorguları test ettim.

- Verilerin satır sayısı

```
hive> SELECT COUNT(*) FROM student_mat;
Query ID = asude_20251202043556_0281ace2-8feb-4757-bede-7273dc850cb7
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2025-12-02 04:36:14,623 Stage-1 map = 0%,  reduce = 0%
2025-12-02 04:36:16,670 Stage-1 map = 100%,  reduce = 0%
2025-12-02 04:36:17,720 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local368438878_0001
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 113986 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
396
Time taken: 20.885 seconds, Fetched: 1 row(s)
```

- Cinsiyete göre final not ortalamaları

```
hive> SELECT sex, AVG(G3) AS avg_final_grade
    > FROM student_mat
    > GROUP BY sex;
Query ID = asude_20251202045059_d1013e6c-2f44-482b-b1ed-d6810ea286e0
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Defaulting to jobconf value of: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2025-12-02 04:51:14,641 Stage-1 map = 0%,  reduce = 0%
2025-12-02 04:51:17,811 Stage-1 map = 100%,  reduce = 0%
2025-12-02 04:51:18,880 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local222159307_0001
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 113986 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
"F"     9.966346153846153
"M"     10.914438502673796
sex     NULL
Time taken: 19.248 seconds, Fetched: 3 row(s)
```

- Destek alan ve almayan öğrencilerin not ortalaması

```
hive> SELECT schoolsup, AVG(G3) AS avg_final_grade
    > FROM student_mat
    > GROUP BY schoolsup;
Query ID = asude_20251202050122_28490037-7b16-4c63-b4a0-12328f4f5269
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2025-12-02 05:01:33,903 Stage-1 map = 0%,  reduce = 0%
2025-12-02 05:01:36,938 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local1891507001_0002
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 227972 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
"no"    10.561046511627907
"yes"   9.431372549019608
schoolsup       NULL
Time taken: 14.041 seconds, Fetched: 3 row(s)
```

- Devamsızlığı 10 dan fazla olan öğrenci sayısı

```
hive> SELECT COUNT(*) AS num_students
    > FROM student_mat
    > WHERE absences > 10;
Query ID = asude_20251202050030_1dc3a3a5-6f39-41f0-beea-666043996aa0
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2025-12-02 05:00:40,206 Stage-1 map = 0%,  reduce = 0%
2025-12-02 05:00:41,245 Stage-1 map = 100%,  reduce = 0%
2025-12-02 05:00:42,266 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local107818258_0001
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 113986 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
66
Time taken: 11.98 seconds, Fetched: 1 row(s)
```

Apache Spark

Apache Spark, büyük veri işlemek için kullanılan hızlı ve bellek içi paralel işlem motorudur.

Bu projede Spark ile:

- Veri okuma (Spark DataFrame API)
- yeni kolonlar oluşturma (ör. total_score, passed)
- istatistiksel analizler
- Python ve Scala ile veri işleme

işlemleri yapılmıştır.

Spark hem **PySpark** hem **Scala** ile kullanılmıştır.

- HDFS'den veri okuma

```
>>> df = spark.read \
...     .option("header", "true") \
...     .option("sep", ";") \
...     .option("inferSchema", "true") \
...     .csv("hdfs://localhost:9000/user/asude/student/student-mat.csv")

[Stage 0:>                                                          (0 + 1
```

- Nota göre geçenlerin ve kalanların sayısı

```
>>> df = df.withColumn("pass_fail", when(col("G3")>=10, "pass").otherwise("fail"))
>>> df.groupBy("pass_fail").count().show()
+---------+-----+
|pass_fail|count|
+---------+-----+
|     fail|  130|
|     pass|  265|
```

- Okula göre final ortalamaları

```
scala> df.groupBy("school").avg("G3").show()
+------+------------------+
|school|           avg(G3)|
+------+------------------+
|    MS| 9.847826086956522|
|    GP|10.489971346704872|
+------+------------------+
```

- Yeni kolon eklenerek sorgu

```
>>> from pyspark.sql.functions import col
>>>
>>> # Yeni kolon ekle
>>> df = df.withColumn("total_score", col("G1") + col("G2") + col("G3"))
>>>
>>> # İlk 10 satırı göster
>>> df.select("G1", "G2", "G3", "total_score").show(10)
+---+---+---+-----------+
| G1| G2| G3|total_score|
+---+---+---+-----------+
|  5|  6|  6|         17|
|  5|  5|  6|         16|
|  7|  8| 10|         25|
| 15| 14| 15|         44|
|  6| 10| 10|         26|
| 15| 15| 15|         45|
| 12| 12| 11|         35|
|  6|  5|  6|         17|
| 16| 18| 19|         53|
| 14| 15| 15|         44|
+---+---+---+-----------+
only showing top 10 rows
```

- Seçili sütunların toplam nota göre 10 tanesinin çoktan aza gösterimi

```
scala> df2.select("age", "total_score","internet")
res4: org.apache.spark.sql.DataFrame = [age: int, total_score: int ... 1 more field]

scala>     .orderBy(col("total_score").desc)
res5: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [age: int, total_score: int ... 1 more field]

scala>     .show(10)
+---+-----------+--------+
|age|total_score|internet|
+---+-----------+--------+
| 16|         58|     yes|
| 15|         56|     yes|
| 15|         56|     yes|
| 18|         56|     yes|
| 15|         55|     yes|
| 18|         55|     yes|
| 16|         54|     yes|
| 16|         54|     yes|
| 17|         54|     yes|
| 17|         54|      no|
+---+-----------+--------+
only showing top 10 rows
```

- Seçili sütunların gösterimi

```
scala> df.select("school", "studytime", "age", "G3").show(10)
+------+---------+---+---+
|school|studytime|age| G3|
+------+---------+---+---+
|    GP|        2| 18|  6|
|    GP|        2| 17|  6|
|    GP|        2| 15| 10|
|    GP|        3| 15| 15|
|    GP|        2| 16| 10|
|    GP|        2| 16| 15|
|    GP|        2| 16| 11|
|    GP|        2| 17|  6|
|    GP|        2| 15| 19|
|    GP|        2| 15| 15|
+------+---------+---+---+
only showing top 10 rows
```

NOT: Not: Projenin kapsamı gereği raporda yalnızca temel ve açıklayıcı kod bloklarına yer verdim.
Tüm kodların tam hâline README dosyasından ve ekteki proje klasöründen erişilebilir.
(Scala, PySpark ve Hive SQL dosyalarının tamamı, çalıştırılabilir şekilde proje klasöründe bulunmaktadır.)

Sonuç:

Bu çalışmada aynı veri seti üzerinde Hadoop, Hive ve Scala ,PySpark kullanarak karşılaştırmalı bir analiz yaptım. Hive, MapReduce tabanlı çalışma yapısı nedeniyle daha yavaş çalışırken, PySpark bellek içi (in-memory) mimarisi sayesinde çok daha hızlı ve esnek bir işlem performansı sağladı. Ayrıca MapReduce aşamalarına bağlı hatalar PySpark işlem tarzında görülmedi.