

**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**  
**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**  
**KHOA HỆ THÔNG THÔNG TIN**



**BÁO CÁO ĐỒ ÁN**

**Môn học: Dữ liệu lớn**

**ĐỀ TÀI: DỰ ĐOÁN NĂNG LỰC HỌC TẬP VÀ  
ĐIỂM THI CUỐI NĂM CỦA HỌC SINH**

**Giảng viên hướng dẫn: Th.S Nguyễn Hồ Duy Tri**

**Sinh Viên Thực Hiện:**

Đỗ Huỳnh Mỹ Tâm	20520746
Đinh Thị Tú Uyên	20522139
Nguyễn Công Thành	20521918
Nguyễn Phạm Thanh Phong	21522458

**TP HỒ CHÍ MINH, tháng 6/2024**

## LỜI CẢM ƠN

Chúng em xin gửi lời cảm ơn chân thành và sâu sắc tới thầy Nguyễn Hồ Duy Tri – giảng viên hướng dẫn môn học “Dữ liệu lớn” tại trường Đại học Công nghệ Thông tin. Thầy đã cung cấp cho chúng em cơ hội để có được những kỹ năng và kiến thức cần thiết để hoàn thành chủ đề đồ án này.

Trong suốt quá trình học tập và trong quá trình thực hiện đề tài, nhóm chúng em đã vận dụng và tích lũy những kiến thức quý báu từ các bài giảng đồng thời kết hợp những kiến thức mới thông qua nghiên cứu. Tuy nhiên, do hiểu biết về lý luận và kinh nghiệm thực tế còn hạn chế nên trong đồ án của chúng em không tránh khỏi những sai sót và thiếu sót. Chúng em rất mong nhận được những phản hồi và góp ý từ thầy, điều này sẽ giúp chúng em có thêm kinh nghiệm và hoàn thiện hơn trong tương lai, không chỉ trong giai đoạn tới mà còn trong các dự án sau này.

Xin chân thành cảm ơn !

Nhóm sinh viên thực hiện:

Đỗ Huỳnh Mỹ Tâm

Đinh Thị Tú Uyên

Nguyễn Công Thành

Nguyễn Phạm Thanh Phong

## NHẬN XÉT CỦA GIẢNG VIÊN



## MỤC LỤC

LỜI CẢM ƠN	2
NHẬN XÉT CỦA GIÁNG VIÊN	3
MỤC LỤC	5
DANH MỤC BẢNG, HÌNH ẢNH	7
CHƯƠNG 1: LÝ DO CHỌN ĐỀ TÀI	8
CHƯƠNG 2: DỮ LIỆU	9
2.1 Giới thiệu tập dữ liệu	9
2.2 Thông tin chi tiết	9
2.3 Danh sách các thuộc tính	9
CHƯƠNG 3: MÔ TẢ BÀI TOÁN	13
CHƯƠNG 4: KỸ THUẬT TIỀN XỬ LÝ DỮ LIỆU	15
4.1 Thiết lập cụm Spark Standalone với 3 Workers	15
4.2 Import thư viện và dữ liệu	22
4.3 Phân tích dữ liệu thăm dò (EDA)	24
4.4 Tiền xử lý dữ liệu	29
4.4.1 Chọn lọc và loại thuộc tính không cần thiết và ít ảnh hưởng đến mô hình dự đoán	29
4.4.2 Xử lý các thuộc tính ở định dạng non-numeric	32
4.4.3 Xử lý các thuộc tính numeric	40
4.4.4 Lựa chọn thuộc tính (Feature Selection)	41
CHƯƠNG 5: THUẬT TOÁN KHAI THÁC DỮ LIỆU	44
5.1 Giới thiệu	44
5.1.1 Linear Regression	44
5.1.2 Linear Regression multi-features: Hồi quy tuyến tính cho nhiều biến	48
5.1.3 KNN (K - nearest neighbor)	49
5.2 Mô hình song song hóa giải thuật	52
5.3 Triển khai thuật toán	53
5.3.1 K-Nearest Neighbors Regression	53
5.3.2 K-Nearest Neighbors Classification	60
5.3.3 Linear Regression	66
CHƯƠNG 6: KẾT QUẢ ĐẠT ĐƯỢC	75
6.1 Phát biểu kết quả	75
6.2 So sánh, đánh giá	76

6.2.1 K-Nearest Neighbors Regression	76
6.2.2 K-Nearest Neighbors Classification	78
6.2.3 Linear Regression	79
6.3 Đánh giá chung	79
<b>CHƯƠNG 7: KẾT LUẬN</b>	<b>80</b>
7.1 Ưu điểm	80
7.2 Hạn chế	80
7.3 Hướng phát triển	81
<b>BẢNG PHÂN CÔNG CÔNG VIỆC CÁC THÀNH VIÊN</b>	<b>82</b>
<b>TÀI LIỆU THAM KHẢO</b>	<b>83</b>

## **DANH MỤC BẢNG, HÌNH ẢNH**

Bảng 1: Thông tin thuộc tính tmdb_5000_credits.csv	9
Bảng 2: Thông tin thuộc tính tmdb_5000_movies.csv	11
Hình 1:Mô hình cụm Spark	13
Hình 2: KNN - Classification	53
Hình 3: KNN - Regression	54
Hình 4:Mô hình song song hóa giải thuật KNN	56

## CHƯƠNG 1: LÝ DO CHỌN ĐỀ TÀI

Dữ liệu về thành tích học tập của học sinh cũng ngày càng lớn và đa dạng, bao gồm điểm số, đặc điểm nhân khẩu học, và các yếu tố xã hội, gia đình...

Việc nghiên cứu ứng dụng các phương pháp tiên tiến như K-Nearest Neighbors để dự đoán và phân tích điểm số học sinh có thể mang lại nhiều lợi ích cho hệ thống giáo dục, chẳng hạn như hỗ trợ các quyết định về chương trình học, phương pháp giảng dạy, và cung cấp gợi ý cá nhân hóa cho học sinh. Việc dự đoán điểm số học tập có thể giúp các nhà trường và giáo viên hiểu rõ hơn về đặc điểm và xu hướng của học sinh, từ đó đưa ra các can thiệp và hỗ trợ kịp thời. Việc ứng dụng các phương pháp tiên tiến trong máy học và trí tuệ nhân tạo vào dự đoán thành tích học tập của học sinh thể hiện sự kết hợp sáng tạo giữa khoa học dữ liệu và lĩnh vực giáo dục.

Kết quả của nghiên cứu này có thể góp phần nâng cao trải nghiệm học tập và cung cấp cơ hội phát triển mới cho học sinh.

## CHƯƠNG 2: DỮ LIỆU

### 2.1 Giới thiệu tập dữ liệu

Tập dữ liệu này chứa dữ liệu về nhân khẩu học, thành tích học tập của học sinh bậc Secondary Education (Highschool) ở Bồ Đào Nha, tương ứng với giáo dục cấp 3 ở Việt Nam.

Dữ liệu được thu thập bằng cách sử dụng các báo cáo và bảng câu hỏi của trường, bao gồm điểm thi của học sinh, nhân khẩu học. Hai bộ dữ liệu được cung cấp về kết quả học tập ở hai môn học riêng biệt: Toán và tiếng Bồ Đào Nha

### 2.2 Thông tin chi tiết

- DataSource: <https://www.kaggle.com/datasets/dillonmyrick/high-school-student-performance-and-demographics>
- Chủ sở hữu: DILLON MYRICK
- Last updated: 7 tháng trước
- File 1: student\_math\_clean.csv
- Size 1: 77.75 kB
- File 2: student\_portuguese\_clean.csv
- Size 2: 128.25 kB

### 2.3 Danh sách các thuộc tính

- **student\_math\_clean.csv** và **student\_portuguese\_clean.csv**: 34 thuộc tính

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
-----	----------------	--------------	-------

1	student_id	int	Là một số nguyên để định danh cho mỗi học sinh
2	school	String	Tên của trường học mà học sinh theo học
3	sex	String	Giới tính của học sinh, có thể là nam (M) hoặc nữ (F)
4	age	int	Tuổi của học sinh
5	address_type	String	Loại vùng học sinh sinh sống (Nông thôn hay Thành thị)
6	family_size	int	Số lượng thành viên trong gia đình của học sinh: có thể lớn hơn 3, ít hơn hoặc bằng 3
7	parent_status	String	Tình trạng hôn nhân của bố mẹ học sinh, có thể là sống cùng nhau hoặc ly hôn
8	mother_education	String	Trình độ giáo dục của mẹ học sinh
9	father_education	String	Trình độ giáo dục của cha học sinh
10	mother_job	String	Nghề nghiệp của mẹ
11	father_job	String	Nghề nghiệp của cha
12	school_choice_reason	String	Lý do chọn trường
13	guardian	String	Người giám hộ (cha hoặc mẹ hoặc khác)
14	travel_time	String	Khoảng thời gian di chuyển từ nhà đến trường
15	study_time	String	Khoảng thời gian học mỗi tuần

16	class_failures	int	Số lần học sinh không đạt yêu cầu trong lớp học
17	school_support	String	Học sinh này có được dạy học thêm từ nhà trường hay không
18	family_support	String	Học sinh này có được dạy học thêm từ gia đình hay không
19	extra_paid_classes	String	Học sinh này có học thêm hay không
20	activities	String	Có tham gia hoạt động ngoại khóa hay không
21	nursery_school	String	Có từng đi học mẫu giáo hay không
22	higher_ed	String	Có dự định theo học cao đẳng/đại học không
23	internet_access	String	Có được sử dụng internet khi ở nhà không
24	romantic_relationship	String	Học sinh có đang hẹn hò không
25	family_relationship	int	Chất lượng mối quan hệ gia đình( từ 1 - rất tệ đến 5 - xuất sắc)
26	free_time	int	Thời gian rảnh sau giờ học (từ 1 - rất thấp đến 5 - rất cao)
27	social	int	Mức độ thường xuyên đi chơi với bạn bè (từ 1 - rất thấp đến 5 - rất cao)
28	weekday_alcohol	int	Mức độ sử dụng rượu bia các ngày thứ (từ 1 - rất thấp đến 5 - rất cao)

29	weekend_alcohol	int	Mức độ sử dụng rượu bia cuối tuần (từ 1 - rất thấp đến 5 - rất cao)
30	health	int	Đánh giá sức khỏe (từ 1 - rất tệ đến 5 - rất tốt)
31	absences	int	Số ngày vắng trong năm học
32	grade_1	int	Điểm thi học kì 1 (từ 0 đến 20)
33	grade_1	int	Điểm thi học kì 2 (từ 0 đến 20)
34	final_grade	int	Điểm thi cuối kì (từ 0 đến 20)

## CHƯƠNG 3: MÔ TẢ BÀI TOÁN

Mô hình KNN Regression và Linear Regression được áp dụng để dự đoán điểm thi cuối năm của học sinh, trong khi mô hình KNN Classifier được sử dụng để xác định học sinh có học lực tốt không, dựa theo điều kiện nhân khẩu học. Qua đó, giúp nhà trường và gia đình có thể hỗ trợ đánh giá, hỗ trợ, định hướng và giúp đỡ kịp thời, nhằm hiểu các em hơn và cải thiện thành tích học tập của học sinh.

❖ Bài toán dự đoán điểm thi cuối năm của học sinh:

- Input: Điểm thi 2 kỳ trước đó (grade\_1, grade\_2), các thông tin về nhân khẩu học và học tập của học sinh
- Output: Điểm thi học kì 3 hay Điểm thi cuối năm của học sinh (final\_grade)

❖ Bài toán dự đoán năng lực học tập trong năm của học sinh:

- Input: Các thông tin về nhân khẩu học và học tập của học sinh

- Output: Năng lực học tập (tốt, is\_good)

➢ Điều kiện phân loại:

- Nhóm tham khảo theo cách phân loại học sinh tại Bồ Đào Nha [1]
- Điểm thi cả 3 kì đều từ khá trở lên (14), hoặc điểm thi qua từng kỳ không được giảm và điểm cuối kỳ phải từ khá trở lên.
- Điều kiện thứ 2 để ghi nhận những học sinh có sự duy trì và tiến bộ trong học tập suốt năm học để đạt được kết quả cuối năm tốt

$(\text{grade\_1} + \text{grade\_2} + \text{final\_grade}) / 3 \geq 14$

hoặc

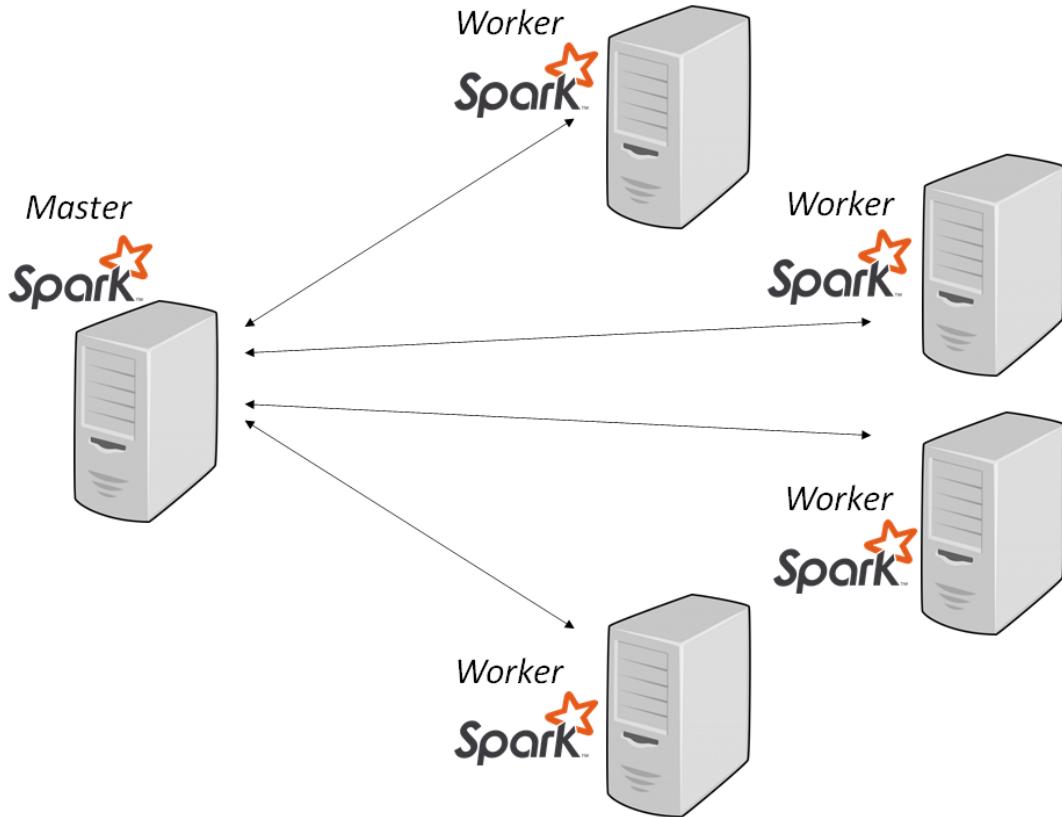
$\text{final\_grade} \geq 14$  và  $\text{final\_grade} \geq \text{grade\_2}$  và  $\text{grade\_2} \geq \text{grade\_1}$

Quy trình triển khai bắt đầu với việc chuẩn bị dữ liệu, bao gồm loại bỏ thông tin không cần thiết và kiểm tra dữ liệu, kiểu dữ liệu, dữ liệu bị thiếu, ... và sau đó thực hiện định dạng dữ liệu và chia dữ liệu thành tập huấn luyện, tập kiểm tra để đào tạo và đánh giá mô hình.

Để đáp ứng yêu cầu xử lý dữ liệu lớn, ta cần song song hóa thuật toán KNN và Linear Regression trên các nền tảng hỗ trợ xử lý dữ liệu lớn Apache Spark. Bằng cách này có thể tận dụng khả năng xử lý đồng thời trên nhiều node và cụm máy chủ, giúp cải thiện hiệu suất và thời gian xử lý trên tập dữ liệu lớn.

## CHƯƠNG 4: KỸ THUẬT TIỀN XỬ LÝ DỮ LIỆU

### 4.1 Thiết lập cụm Spark Standalone với 3 Workers



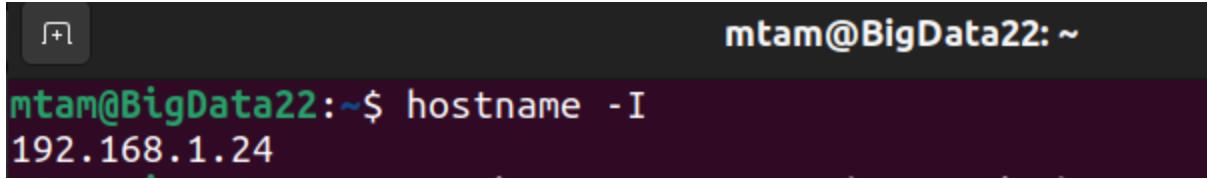
Hình 1: Mô hình cụm Spark

Chuẩn bị:

- Master và Worker phải cùng kết nối trong 1 mạng
- Biết được địa IP của từng Master và Worker

Xem địa chỉ IP:

```
hostname -I
```

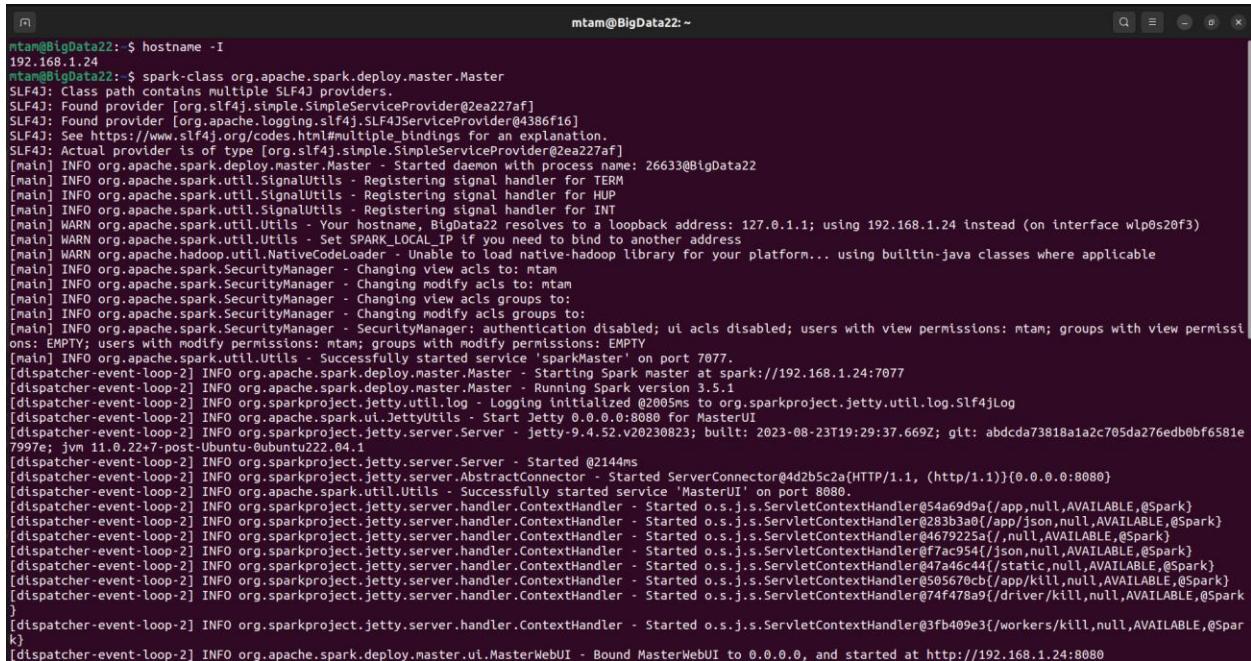


```
mtam@BigData22:~$ hostname -I
192.168.1.24
```

Xem địa chỉ IP để chuẩn bị thiết lập cụm Spark

### Bước 1: Deploy Spark Master

```
spark-class org.apache.spark.deploy.master.Master
```



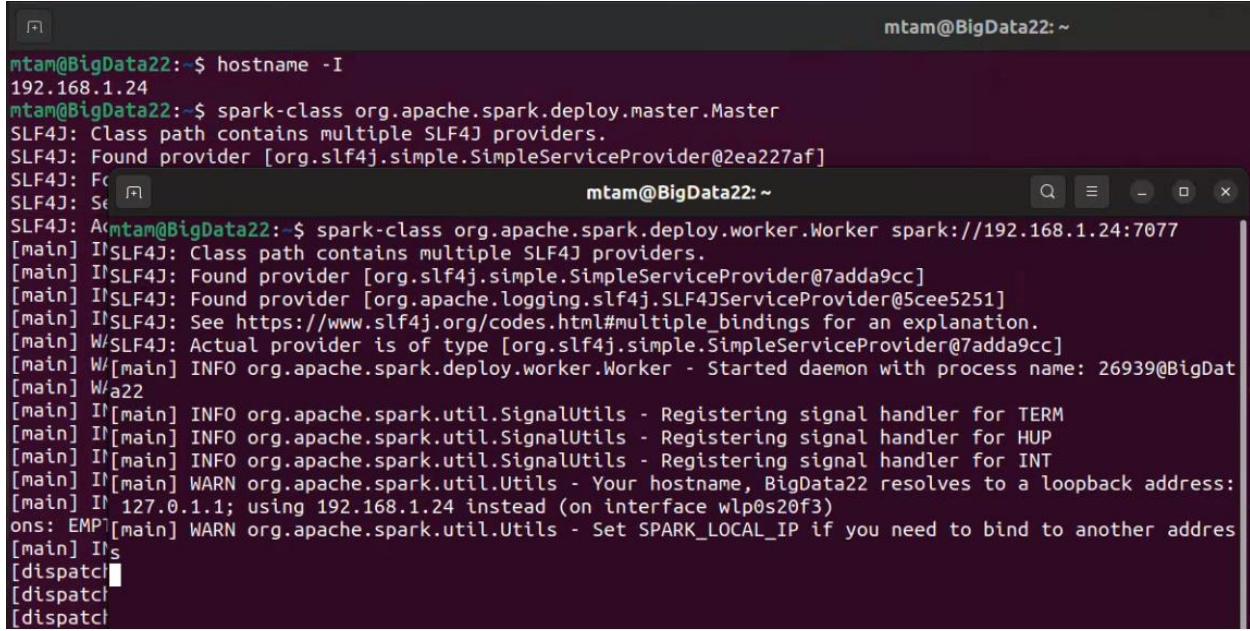
```
mtam@BigData22:~$ hostname -I
192.168.1.24
mtam@BigData22:~$ spark-class org.apache.spark.deploy.master.Master
SLF4J: Class path contains multiple SLF4J providers.
SLF4J: Found provider [org.slf4j.simple.SimpleServiceProvider@2ea227af]
SLF4J: Found provider [org.apache.logging.slf4j.SLF4JServiceProvider@4386f16]
SLF4J: See https://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual provider is of type [org.slf4j.simple.SimpleServiceProvider@2ea227af]
[main] INFO org.apache.spark.deploy.master.Master - Started daemon with process name: 26633@BigData22
[main] INFO org.apache.spark.util.SignalUtils - Registering signal handler for TERM
[main] INFO org.apache.spark.util.SignalUtils - Registering signal handler for HUP
[main] INFO org.apache.spark.util.SignalUtils - Registering signal handler for INT
[main] WARN org.apache.spark.util.Utils - Your hostname, BigData22 resolves to a loopback address: 127.0.1.1; using 192.168.1.24 instead (on interface wlp0s20f3)
[main] WARN org.apache.hadoop.util.NativeCodeLoader - Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
[main] INFO org.apache.spark.SecurityManager - Changing view acls to: mtam
[main] INFO org.apache.spark.SecurityManager - Changing modify acls to: mtam
[main] INFO org.apache.spark.SecurityManager - Changing view acls groups to:
[main] INFO org.apache.spark.SecurityManager - Changing modify acls groups to:
[main] INFO org.apache.spark.SecurityManager - SecurityManager: authentication disabled; ui acls disabled; users with view permissions: mtam; groups with view permissions: EMPTY; users with modify permissions: mtam; groups with modify permissions: EMPTY
[main] INFO org.apache.spark.util.Utils - Successfully started service 'sparkMaster' on port 7077.
[dispatcher-event-loop-2] INFO org.apache.spark.deploy.master.Master - Starting Spark master at spark://192.168.1.24:7077
[dispatcher-event-loop-2] INFO org.apache.spark.deploy.master.Master - Running Spark version 3.5.1
[dispatcher-event-loop-2] INFO org.apache.spark.util.log - Logging initialized @2005ms to org.sparkproject.jetty.util.log.Slf4jLog
[dispatcher-event-loop-2] INFO org.apache.spark.ui.JettyUtils - Start Jetty 0.0.0.0:8080 for MasterUI
[dispatcher-event-loop-2] INFO org.sparkproject.jetty.server.Server - jetty-9.4.52.v20230823; built: 2023-08-23T19:29:37.669Z; git: abcdca73818a1a2c705da276edb0bf6581e
7997e; jvm 11.0.22+7-post-Ubuntu-0ubuntu22.04.1
[dispatcher-event-loop-2] INFO org.sparkproject.jetty.server.Server - Started @214ms
[dispatcher-event-loop-2] INFO org.sparkproject.jetty.server.AbstractConnector - Started ServerConnector@4d2b5c2a[HTTP/1.1, (http/1.1)]{0.0.0.0:8080}
[dispatcher-event-loop-2] INFO org.apache.spark.util.Utils - Successfully started service 'MasterUI' on port 8080.
[dispatcher-event-loop-2] INFO org.sparkproject.jetty.server.handler.ContextHandler - Started o.s.j.s.ServletContextHandler@54a69d9e{/app,null,AVAILABLE,@Spark}
[dispatcher-event-loop-2] INFO org.sparkproject.jetty.server.handler.ContextHandler - Started o.s.j.s.ServletContextHandler@285b3a0{/app/json,null,AVAILABLE,@Spark}
[dispatcher-event-loop-2] INFO org.sparkproject.jetty.server.handler.ContextHandler - Started o.s.j.s.ServletContextHandler@4679225a{/null,AVAILABLE,@Spark}
[dispatcher-event-loop-2] INFO org.sparkproject.jetty.server.handler.ContextHandler - Started o.s.j.s.ServletContextHandler@f7ac954{/json,null,AVAILABLE,@Spark}
[dispatcher-event-loop-2] INFO org.sparkproject.jetty.server.handler.ContextHandler - Started o.s.j.s.ServletContextHandler@47a46c44{/static,null,AVAILABLE,@Spark}
[dispatcher-event-loop-2] INFO org.sparkproject.jetty.server.handler.ContextHandler - Started o.s.j.s.ServletContextHandler@565678cb{/app/kill,null,AVAILABLE,@Spark}
[dispatcher-event-loop-2] INFO org.sparkproject.jetty.server.handler.ContextHandler - Started o.s.j.s.ServletContextHandler@74f478a9{/driver/kill,null,AVAILABLE,@Spark}
}
[dispatcher-event-loop-2] INFO org.sparkproject.jetty.server.handler.ContextHandler - Started o.s.j.s.ServletContextHandler@3fb409e3{/workers/kill,null,AVAILABLE,@Spark}
}
[dispatcher-event-loop-2] INFO org.apache.spark.deploy.master.ui.MasterWebUI - Bound MasterWebUI to 0.0.0.0, and started at http://192.168.1.24:8080
```

Deploy Spark Master

### Bước 2: Deploy 3 Spark Workers

```
spark-class org.apache.spark.deploy.worker.Worker
spark://<master_ip>:<port> --host <IP_ADDR>
```

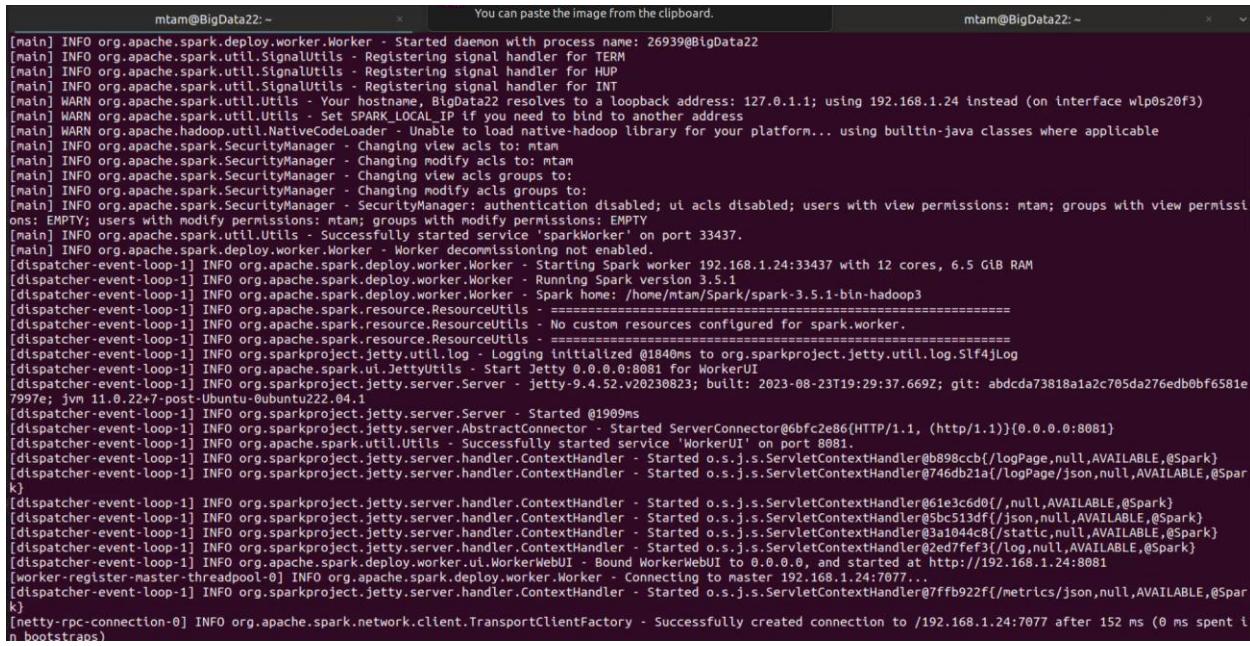
Trong đó <master-ip> và <master-port> bằng địa chỉ IP thực và cổng của Spark Master.



```
mtam@BigData22:~$ hostname -I
192.168.1.24
mtam@BigData22:~$ spark-class org.apache.spark.deploy.master.Master
SLF4J: Class path contains multiple SLF4J providers.
SLF4J: Found provider [org.slf4j.simple.SimpleServiceProvider@2ea227af]
SLF4J: Found provider [org.slf4j.simple.SimpleServiceProvider@7adda9cc]
SLF4J: See https://www.slf4j.org/codes.html#multiple_bindings for an explanation.
W/SLF4J: Actual provider is of type [org.slf4j.simple.SimpleServiceProvider@7adda9cc]
[main] W/[main] INFO org.apache.spark.deploy.worker.Worker - Started daemon with process name: 26939@BigData22
[main] W/a22
[main] I/[main] INFO org.apache.spark.util.SignalUtils - Registering signal handler for TERM
[main] I/[main] INFO org.apache.spark.util.SignalUtils - Registering signal handler for HUP
[main] I/[main] INFO org.apache.spark.util.SignalUtils - Registering signal handler for INT
[main] I/[main] WARN org.apache.spark.util.Utils - Your hostname, BigData22 resolves to a loopback address: 127.0.1.1; using 192.168.1.24 instead (on interface wlp0s20f3)
[main] W/SPARK_LOCAL_IP=[main] WARN org.apache.spark.util.Utils - Set SPARK_LOCAL_IP if you need to bind to another address
[main] I/s
[dispatch]
[dispatch]
[dispatch]
```

## Minh họa Deploy một Spark Workers

- Worker 1



```
mtam@BigData22:~$ spark-class org.apache.spark.deploy.worker.Worker
You can paste the image from the clipboard.
[main] INFO org.apache.spark.deploy.worker.Worker - Started daemon with process name: 26939@BigData22
[main] INFO org.apache.spark.util.SignalUtils - Registering signal handler for TERM
[main] INFO org.apache.spark.util.SignalUtils - Registering signal handler for HUP
[main] INFO org.apache.spark.util.SignalUtils - Registering signal handler for INT
[main] WARN org.apache.spark.util.Utils - Your hostname, BigData22 resolves to a loopback address: 127.0.1.1; using 192.168.1.24 instead (on interface wlp0s20f3)
[main] WARN org.apache.spark.util.Utils - Set SPARK_LOCAL_IP if you need to bind to another address
[main] WARN org.apache.hadoop.util.NativeCodeLoader - Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
[main] INFO org.apache.spark.SecurityManager - Changing view acls to: mtam
[main] INFO org.apache.spark.SecurityManager - Changing modify acls to: mtam
[main] INFO org.apache.spark.SecurityManager - Changing view acls groups to:
[main] INFO org.apache.spark.SecurityManager - Changing modify acls groups to:
[main] INFO org.apache.spark.SecurityManager - SecurityManager: authentication disabled; ui acls disabled; users with view permissions: mtam; groups with view permissions: EMPTY; users with modify permissions: mtam; groups with modify permissions: EMPTY
[main] INFO org.apache.spark.util.Utils - Successfully started service 'sparkworker' on port 33437.
[main] INFO org.apache.spark.deploy.worker.Worker - Worker decommissioning not enabled.
[dispatcher-event-loop-1] INFO org.apache.spark.deploy.worker.Worker - Starting Spark worker 192.168.1.24:33437 with 12 cores, 6.5 GiB RAM
[dispatcher-event-loop-1] INFO org.apache.spark.deploy.worker.Worker - Running Spark version 3.5.1
[dispatcher-event-loop-1] INFO org.apache.spark.deploy.worker.Worker - Spark home: /home/mtam/Spark/spark-3.5.1-bin-hadoop3
[dispatcher-event-loop-1] INFO org.apache.spark.resource.ResourceUtils - =====
[dispatcher-event-loop-1] INFO org.apache.spark.resource.ResourceUtils - No custom resources configured for spark.worker.
[dispatcher-event-loop-1] INFO org.apache.spark.resource.ResourceUtils - =====
[dispatcher-event-loop-1] INFO org.sparkproject.jetty.util.log - Logging initialized @1840Ms to org.sparkproject.jetty.util.log.Slf4jLog
[dispatcher-event-loop-1] INFO org.apache.spark.util.JettyUtils - Start Jetty 0.0.0.0:8081 for WorkerUI
[dispatcher-event-loop-1] INFO org.sparkproject.jetty.server.Server - Jetty-9.4.52.w20230823; built: 2023-08-23T19:29:37.669Z; git: abcdca73818a1a2c705da276edb0bf6581e7997e; jvm 11.0.22+7-post-Ubuntu-0ubuntu22.04.1
[dispatcher-event-loop-1] INFO org.sparkproject.jetty.server.Server - Started @1909ms
[dispatcher-event-loop-1] INFO org.sparkproject.jetty.server.AbstractConnector - Started ServerConnector@6bfc2e86([HTTP/1.1, (http/1.1)]{0.0.0.0:8081})
[dispatcher-event-loop-1] INFO org.apache.spark.util.Utils - Successfully started service 'WorkerUI' on port 8081.
[dispatcher-event-loop-1] INFO org.sparkproject.jetty.server.handler.ContextHandler - Started o.s.j.s.ServletContextHandler@b898ccb{/logPage,null,AVAILABLE,@Spark}
[dispatcher-event-loop-1] INFO org.sparkproject.jetty.server.handler.ContextHandler - Started o.s.j.s.ServletContextHandler@746db21a{/logPage/json,null,AVAILABLE,@Spark}
[dispatcher-event-loop-1] INFO org.sparkproject.jetty.server.handler.ContextHandler - Started o.s.j.s.ServletContextHandler@61e3c6d0{/null,AVAILABLE,@Spark}
[dispatcher-event-loop-1] INFO org.sparkproject.jetty.server.handler.ContextHandler - Started o.s.j.s.ServletContextHandler@5bc513df{/json,null,AVAILABLE,@Spark}
[dispatcher-event-loop-1] INFO org.sparkproject.jetty.server.handler.ContextHandler - Started o.s.j.s.ServletContextHandler@3a1044c8{/static,null,AVAILABLE,@Spark}
[dispatcher-event-loop-1] INFO org.sparkproject.jetty.server.handler.ContextHandler - Started o.s.j.s.ServletContextHandler@2ed7fef3{/log,null,AVAILABLE,@Spark}
[dispatcher-event-loop-1] INFO org.apache.spark.deploy.worker.ui.WorkerWebUI - Bound WorkerWebUI to 0.0.0.0, and started at http://192.168.1.24:8081
[worker-register-master-threadpool-0] INFO org.apache.spark.deploy.worker.Worker - Connecting to Master 192.168.1.24:7077...
[dispatcher-event-loop-1] INFO org.sparkproject.jetty.server.handler.ContextHandler - Started o.s.j.s.ServletContextHandler@7ffb922f{/metrics/json,null,AVAILABLE,@Spark}
[netty-rpc-connection-0] INFO org.apache.spark.network.client.TransportClientFactory - Successfully created connection to /192.168.1.24:7077 after 152 ms (0 ms spent in bootstraps)
```

## Deploy thành công Spark Workers 1

- Worker 2

```
[main] INFO org.apache.spark.util.SignalUtils - Registering signal handler for TERM
[main] INFO org.apache.spark.util.SignalUtils - Registering signal handler for HUP
[main] INFO org.apache.spark.util.SignalUtils - Registering signal handler for INT
[main] WARN org.apache.spark.util.Utils - Your hostname, BigData22 resolves to a loopback address: 127.0.1.1; using 192.168.1.24 instead (on interface wlp0s20f3)
[main] WARN org.apache.spark.util.Utils - Set SPARK_LOCAL_IP if you need to bind to another address
[main] WARN org.apache.hadoop.util.NativeCodeLoader - Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
[main] INFO org.apache.spark.SecurityManager - Changing view acls to: mtam
[main] INFO org.apache.spark.SecurityManager - Changing modify acls to: mtam
[main] INFO org.apache.spark.SecurityManager - Changing view acls groups to:
[main] INFO org.apache.spark.SecurityManager - Changing modify acls groups to:
[main] INFO org.apache.spark.SecurityManager - SecurityManager: authentication disabled; ui acls disabled; users with view permissions: mtam; groups with view permissions: EMPTY; users with modify permissions: mtam; groups with modify permissions: EMPTY
[main] INFO org.apache.spark.util.Utils - Successfully started service 'sparkWorker' on port 44571.
[main] INFO org.apache.spark.deploy.worker.Worker - Worker decommissioning not enabled.
[dispatcher-event-loop-1] INFO org.apache.spark.deploy.worker.Worker - Starting Spark worker 192.168.1.24:44571 with 12 cores, 6.5 GiB RAM
[dispatcher-event-loop-1] INFO org.apache.spark.deploy.worker.Worker - Running Spark version 3.5.1
[dispatcher-event-loop-1] INFO org.apache.spark.deploy.worker.Worker - Spark home: /home/mtam/Spark/spark-3.5.1-bin-hadoop3
[dispatcher-event-loop-1] INFO org.apache.spark.resource.ResourceUtils - =====
[dispatcher-event-loop-1] INFO org.apache.spark.resource.ResourceUtils - No custom resources configured for spark.worker.
[dispatcher-event-loop-1] INFO org.apache.sparkproject.jetty.util.log - Logging initialized @1934ms to org.sparkproject.jetty.util.log.Slf4jLog
[dispatcher-event-loop-1] INFO org.apache.spark.ui.JettyUtils - Start Jetty 0.0.0.0:8081 for WorkerUI
[dispatcher-event-loop-1] INFO org.sparkproject.jetty.server.Server - jetty-9.4.52.v20230823; built: 2023-08-23T19:29:37.669Z; git: abcdca73818a1a2c705da276edb0bf6581e7997e; jvm 11.0.22+7-post-Ubuntu-0ubuntu22.04.1
[dispatcher-event-loop-1] INFO org.sparkproject.jetty.server.Server - Started @2031ms
[dispatcher-event-loop-1] WARN org.apache.spark.util.Utils - Service 'WorkerUI' could not bind on port 8081. Attempting port 8082.
[dispatcher-event-loop-1] INFO org.sparkproject.jetty.server.AbstractConnector - Started ServerConnector@6dcfc818[HTTP/1.1, (http/1.1)]{0.0.0.0:8082}
[dispatcher-event-loop-1] INFO org.apache.spark.util.Utils - Successfully started service 'WorkerUI' on port 8082.
[dispatcher-event-loop-1] INFO org.sparkproject.jetty.server.handler.ContextHandler - Started o.s.j.s.ServletContextHandler@60d3f5f41{/logPage,null,AVAILABLE,@Spark}
[dispatcher-event-loop-1] INFO org.sparkproject.jetty.server.handler.ContextHandler - Started o.s.j.s.ServletContextHandler@70324e8c{/logPage/json,null,AVAILABLE,@Spark}
[dispatcher-event-loop-1] INFO org.sparkproject.jetty.server.handler.ContextHandler - Started o.s.j.s.ServletContextHandler@908cf66{/null,AVAILABLE,@Spark}
[dispatcher-event-loop-1] INFO org.sparkproject.jetty.server.handler.ContextHandler - Started o.s.j.s.ServletContextHandler@418e1dc1{/json,null,AVAILABLE,@Spark}
[dispatcher-event-loop-1] INFO org.sparkproject.jetty.server.handler.ContextHandler - Started o.s.j.s.ServletContextHandler@424bde9{/static,null,AVAILABLE,@Spark}
[dispatcher-event-loop-1] INFO org.apache.spark.deploy.worker.ui.WorkerWebUI - Bound WorkerWebUI to 0.0.0.0, and started at http://192.168.1.24:8082
[worker-register-master-threadpool-0] INFO org.apache.spark.deploy.worker.Worker - Connecting to master 192.168.1.24:7077...
[dispatcher-event-loop-1] INFO org.sparkproject.jetty.server.handler.ContextHandler - Started o.s.j.s.ServletContextHandler@5c47646b{/metrics/json,null,AVAILABLE,@Spark}
[netty-rpc-connection-0] INFO org.apache.spark.network.client.TransportClientFactory - Successfully created connection to /192.168.1.24:7077 after 150 ms (0 ms spent in handshakes)
```

## Deploy thành công Spark Workers 2

### ● Worker 3

```
mtam@BigData22:~ mtam@BigData22:~ mtam@BigData22:~
[main] INFO org.apache.spark.util.SignalUtils - Registering signal handler for HUP
[main] INFO org.apache.spark.util.SignalUtils - Registering signal handler for INT
[main] WARN org.apache.spark.util.Utils - Your hostname, BigData22 resolves to a loopback address: 127.0.1.1; using 192.168.1.24 instead (on interface wlp0s20f3)
[main] WARN org.apache.spark.util.Utils - Set SPARK_LOCAL_IP if you need to bind to another address
[main] WARN org.apache.hadoop.util.NativeCodeLoader - Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
[main] INFO org.apache.spark.SecurityManager - Changing view acls to: mtam
[main] INFO org.apache.spark.SecurityManager - Changing modify acls to: mtam
[main] INFO org.apache.spark.SecurityManager - Changing view acls groups to:
[main] INFO org.apache.spark.SecurityManager - Changing modify acls groups to:
[main] INFO org.apache.spark.SecurityManager - SecurityManager: authentication disabled; ui acls disabled; users with view permissions: mtam; groups with view permissions: EMPTY; users with modify permissions: mtam; groups with modify permissions: EMPTY
[main] INFO org.apache.spark.util.Utils - Successfully started service 'sparkWorker' on port 46333.
[main] INFO org.apache.spark.deploy.worker.Worker - Worker decommissioning not enabled.
[dispatcher-event-loop-1] INFO org.apache.spark.deploy.worker.Worker - Starting Spark worker 192.168.1.24:46333 with 12 cores, 6.5 GiB RAM
[dispatcher-event-loop-1] INFO org.apache.spark.deploy.worker.Worker - Running Spark version 3.5.1
[dispatcher-event-loop-1] INFO org.apache.spark.resource.ResourceUtils - =====
[dispatcher-event-loop-1] INFO org.apache.spark.resource.ResourceUtils - No custom resources configured for spark.worker.
[dispatcher-event-loop-1] INFO org.apache.sparkproject.jetty.util.log - Logging initialized @1827ms to org.sparkproject.jetty.util.log.Slf4jLog
[dispatcher-event-loop-1] INFO org.apache.spark.ui.JettyUtils - Start Jetty 0.0.0.0:8081 for WorkerUI
[dispatcher-event-loop-1] INFO org.sparkproject.jetty.server.Server - jetty-9.4.52.v20230823; built: 2023-08-23T19:29:37.669Z; git: abcdca73818a1a2c705da276edb0bf6581e7997e; jvm 11.0.22+7-post-Ubuntu-0ubuntu22.04.1
[dispatcher-event-loop-1] INFO org.sparkproject.jetty.server.Server - Started @1904ms
[dispatcher-event-loop-1] WARN org.apache.spark.util.Utils - Service 'WorkerUI' could not bind on port 8081. Attempting port 8082.
[dispatcher-event-loop-1] WARN org.apache.spark.util.Utils - Service 'WorkerUI' could not bind on port 8082. Attempting port 8083.
[dispatcher-event-loop-1] INFO org.sparkproject.jetty.server.AbstractConnector - Started ServerConnector@65fa2c429[HTTP/1.1, (http/1.1)]{0.0.0.0:8083}
[dispatcher-event-loop-1] INFO org.apache.spark.util.Utils - Successfully started service 'WorkerUI' on port 8083.
[dispatcher-event-loop-1] INFO org.sparkproject.jetty.server.handler.ContextHandler - Started o.s.j.s.ServletContextHandler@7d8fe7ef{/logPage,null,AVAILABLE,@Spark}
[dispatcher-event-loop-1] INFO org.sparkproject.jetty.server.handler.ContextHandler - Started o.s.j.s.ServletContextHandler@b3fc26e{/logPage/json,null,AVAILABLE,@Spark}
[dispatcher-event-loop-1] INFO org.sparkproject.jetty.server.handler.ContextHandler - Started o.s.j.s.ServletContextHandler@23087b1d{/null,AVAILABLE,@Spark}
[dispatcher-event-loop-1] INFO org.sparkproject.jetty.server.handler.ContextHandler - Started o.s.j.s.ServletContextHandler@671339ba{/json,null,AVAILABLE,@Spark}
[dispatcher-event-loop-1] INFO org.sparkproject.jetty.server.handler.ContextHandler - Started o.s.j.s.ServletContextHandler@700ec94{/static,null,AVAILABLE,@Spark}
[dispatcher-event-loop-1] INFO org.sparkproject.jetty.server.handler.ContextHandler - Started o.s.j.s.ServletContextHandler@65af045de{/log,null,AVAILABLE,@Spark}
[dispatcher-event-loop-1] INFO org.apache.spark.deploy.worker.ui.WorkerWebUI - Bound WorkerWebUI to 0.0.0.0, and started at http://192.168.1.24:8083
[worker-register-master-threadpool-0] INFO org.apache.spark.deploy.worker.Worker - Connecting to master 192.168.1.24:7077...
[dispatcher-event-loop-1] INFO org.sparkproject.jetty.server.handler.ContextHandler - Started o.s.j.s.ServletContextHandler@1976e557{/metrics/json,null,AVAILABLE,@Spark}
[netty-rpc-connection-0] INFO org.apache.spark.network.client.TransportClientFactory - Successfully created connection to /192.168.1.24:7077 after 69 ms (0 ms spent in handshakes)
```

## Deploy thành công Spark Workers 3

### Bước 3: Truy cập Web UI

## IS405.O11.HTCL – Dữ liệu lớn

The screenshot shows the Spark Master UI at `spark://192.168.1.24:7077`. It displays the following information:

- URL: `spark://192.168.1.24:7077`
- Alive Workers: 3
- Cores in use: 36 Total, 0 Used
- Memory in use: 19.6 GiB Total, 0.0 B Used
- Resources in use:
- Applications: 0 Running, 0 Completed
- Drivers: 0 Running, 0 Completed
- Status: ALIVE

**Workers (3)**

Worker Id	Address	State	Cores	Memory	Resources
worker-20240603012551-192.168.1.24-33437	192.168.1.24:33437	ALIVE	12 (0 Used)	6.5 GiB (0.0 B Used)	
worker-20240603012604-192.168.1.24-44571	192.168.1.24:44571	ALIVE	12 (0 Used)	6.5 GiB (0.0 B Used)	
worker-20240603012616-192.168.1.24-46333	192.168.1.24:46333	ALIVE	12 (0 Used)	6.5 GiB (0.0 B Used)	

**Running Applications (0)**

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration

**Completed Applications (0)**

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration

Web UI sau khi Deploy thành công 3 Worker

Có thể thấy Address gồm địa chỉ IP và Port của từng Worker trong terminal trùng với danh sách Worker được liệt kê bằng Web UI

### Bước 4: Run application in Jupyter Notebook

```
In [1]: import findspark
findspark.init()

In [2]:
import pyspark
from pyspark.sql import SparkSession

try:
    spark.stop()
except NameError:
    pass

# Tạo Spark session
spark = SparkSession.builder \
    .appName("PROJECT") \
    .master('spark://192.168.1.24:7077') \
    .getOrCreate()
```

Tạo Spark Session mới với tùy chọn thiết lập Master

```
In [3]: spark
Out[3]: SparkSession - in-memory
          SparkContext
          Spark UI
          Version
          v3.5.1
          Master
          spark://192.168.1.24:7077
         AppName
          PROJECT
```

Tạo thành công Application với thiết lập Master và AppName

#### Bước 5: Kiểm tra trên Web UI

The screenshot shows a web browser window with the title "Spark Master at spark://192.168.1.24:7077". The page content includes:

- System statistics:
  - URL: spark://192.168.1.24:7077
  - Alive Workers: 3
  - Cores in use: 36 Total, 36 Used
  - Memory in use: 19.6 GiB Total, 3.0 GiB Used
  - Resources in use:
  - Applications: 1 Running, 0 Completed
  - Drivers: 0 Running, 0 Completed
  - Status: ALIVE
- A table titled "Workers (3)" showing three worker details:

Worker Id	Address	State	Cores	Memory	Resources
worker-20240603012551-192.168.1.24-33437	192.168.1.24:33437	ALIVE	12 (12 Used)	6.5 GiB (1024.0 MiB Used)	
worker-20240603012604-192.168.1.24-44571	192.168.1.24:44571	ALIVE	12 (12 Used)	6.5 GiB (1024.0 MiB Used)	
worker-20240603012616-192.168.1.24-46333	192.168.1.24:46333	ALIVE	12 (12 Used)	6.5 GiB (1024.0 MiB Used)	

- A table titled "Running Applications (1)" showing one application entry:

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20240603012652-0000	(kill) PROJECT	36	1024.0 MiB		2024/06/03 01:26:52	mtam	RUNNING	23 s

- A table titled "Completed Applications (0)" showing zero entries.

Khi kết nối với Worker là máy khác, ta sẽ thấy có IP khác. Ví dụ với máy của thành viên khác trong nhóm:

## IS405.O11.HTCL – Dữ liệu lớn

The screenshot shows the Apache Spark 3.5.1 master interface at <http://192.168.100.119:8080>. The top navigation bar includes back, forward, search, and refresh icons, along with a note about being 'Not secure'.

**Spark Master at spark://192.168.100.119:7077**

URL: [spark://192.168.100.119:7077](http://192.168.100.119:7077)

Alive Workers: 3

Cores in use: 36 Total, 36 Used

Memory in use: 25.6 GiB Total, 3.0 GiB Used

Resources in use:

Applications: 2 Running, 2 Completed

Drivers: 0 Running, 0 Completed

Status: ALIVE

**Workers (3)**

Worker Id	Address	State	Cores	Memory	Resources
worker-20240602175343-192.168.100.167-36675	192.168.100.167-36675	ALIVE	12 (12 Used)	12.5 GiB (1024.0 MiB Used)	
worker-20240602181058-192.168.100.119-43963	192.168.100.119-43963	ALIVE	12 (12 Used)	6.5 GiB (1024.0 MiB Used)	
worker-20240602181155-192.168.100.119-46101	192.168.100.119-46101	ALIVE	12 (12 Used)	6.5 GiB (1024.0 MiB Used)	

**Running Applications (2)**

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20240602182347-0003	(kill) PROJECT	0	1024.0 MiB		2024/06/02 18:23:47	phong	WAITING	2.2 min
app-20240602181725-0001	(kill) PROJECT	36	1024.0 MiB		2024/06/02 18:17:25	mtam	RUNNING	8.6 min

**Completed Applications (2)**

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20240602182319-0002	PROJECT	0	1024.0 MiB		2024/06/02 18:23:19	phong	FINISHED	28 s
app-20240602181704-0000	PROJECT	36	1024.0 MiB		2024/06/02 18:17:04	mtam	FINISHED	21 s

The screenshot shows the Apache Spark 3.5.1 master interface at <http://192.168.100.119:8080>. The top navigation bar includes back, forward, search, and refresh icons, along with a note about being 'Not secure'.

**Spark Master at spark://192.168.100.119:7077**

URL: [spark://192.168.100.119:7077](http://192.168.100.119:7077)

Alive Workers: 3

Cores in use: 36 Total, 36 Used

Memory in use: 25.6 GiB Total, 3.0 GiB Used

Resources in use:

Applications: 1 Running, 3 Completed

Drivers: 0 Running, 0 Completed

Status: ALIVE

**Workers (3)**

Worker Id	Address	State	Cores	Memory	Resources
worker-20240602175343-192.168.100.167-36675	192.168.100.167-36675	ALIVE	12 (12 Used)	12.5 GiB (1024.0 MiB Used)	
worker-20240602181058-192.168.100.119-43963	192.168.100.119-43963	ALIVE	12 (12 Used)	6.5 GiB (1024.0 MiB Used)	
worker-20240602181155-192.168.100.119-46101	192.168.100.119-46101	ALIVE	12 (12 Used)	6.5 GiB (1024.0 MiB Used)	

**Running Applications (1)**

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20240602182347-0003	(kill) PROJECT	36	1024.0 MiB		2024/06/02 18:23:47	phong	RUNNING	5.4 min

**Completed Applications (3)**

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20240602181725-0001	PROJECT	36	1024.0 MiB		2024/06/02 18:17:25	mtam	FINISHED	11 min
app-20240602182319-0002	PROJECT	0	1024.0 MiB		2024/06/02 18:23:19	phong	FINISHED	28 s
app-20240602181704-0000	PROJECT	36	1024.0 MiB		2024/06/02 18:17:04	mtam	FINISHED	21 s

## 4.2 Import thư viện và dữ liệu

**Bước 1:** Import các thư viện cần thiết

```
In [5]: # Data visualization
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from plotnine import *

# Pyspark
from pyspark.sql.functions import *
from pyspark.sql import functions as f
from pyspark.sql.types import *
from pyspark.sql.window import Window

from operator import add, getitem
from collections import Counter

from numpy.linalg import norm
from math import sqrt
```

Import các thư viện cần thiết

**Bước 2:** Đọc dữ liệu từ 2 file csv là student\_math\_clean.csv và student\_portuguese\_clean.csv

### Đọc dữ liệu

```
In [6]: df_math = spark.read.format("csv")\
    .option("header", "true")\
    .option("inferSchema", "true")\
    .option("escape", "'')\
    .option("na_values", ["", " ", "N/A", "NA"])\
    .load("Data_project/student_math_clean.csv")

df_math.cache()
[dag-scheduler-event-loop] INFO org.apache.spark.scheduler - Job 1 is finished. Cancelling potential speculative or zombie tasks for this job
[task-result-getter-1] INFO org.apache.spark.scheduler.TaskSchedulerImpl - Removed TaskSet 1.0, whose tasks have all completed, from pool
[dag-scheduler-event-loop] INFO org.apache.spark.scheduler.TaskSchedulerImpl - Killing all running tasks in stage 1: Stage finished
[Thread-3] INFO org.apache.spark.scheduler.DAGScheduler - Job 1 finished: load at NativeMethodAccessorImpl.java: 0, took 0.544571 s
[Thread-3] INFO org.apache.spark.sql.execution.datasources.FileSourceStrategy - Pushed Filters:
[Thread-3] INFO org.apache.spark.sql.execution.datasources.FileSourceStrategy - Post-Scan Filters:
[Thread-3] WARN org.apache.spark.sql.catalyst.util.SparkStringUtils - Truncated the string representation of a plan since it was too large. This behavior can be adjusted by setting 'spark.sql.debug.maxToStringFields'.
```

```
Out[6]: DataFrame[student_id: int, school: string, sex: string, age: int, address_type: string, family_size: string, parent_status: string, mother_education: string, father_education: string, mother_job: string, father_job: string, school_choice_reason: string, guardian: string, travel_time: string, study_time: string, class_failures: int, school_support: string, family_support: string, extra_paid_classes: string, activities: string, nursery_school: string, higher_ed: string, internet_access: string, romantic_relationship: string, family_relationship: int, free_time: int, social: int, weekday_alcohol: int, weekend_alcohol: int, health: int, absences: int, grade_1: int, grade_2: int, final_grade: int]
```

Đọc dữ liệu từ student\_math\_clean.csv vào spark dataframe

```
In [7]: df_portuguese = spark.read.format("csv")\
    .option("header", "true")\
    .option("inferSchema", "true")\
    .option("escape", "")\
    .option("na_values", ["", " ", "N/A", "NA"])\
    .load("Data_project/student_portuguese_clean.csv")

df_portuguese.cache()
[task-result-getter-3] INFO org.apache.spark.scheduler.TaskSchedulerImpl - Removed TaskSet 3.0, whose tasks have all completed, from pool
[dag-scheduler-event-loop] INFO org.apache.spark.scheduler.DAGScheduler - ResultStage 3 (load at NativeMethodAccess$Impl.java:0) finished in 0.115 s
[dag-scheduler-event-loop] INFO org.apache.spark.scheduler.DAGScheduler - Job 3 is finished. Cancelling potential speculative or zombie tasks for this job
[dag-scheduler-event-loop] INFO org.apache.spark.scheduler.TaskSchedulerImpl - Killing all running tasks in stage 3: Stage finished
[Thread-3] INFO org.apache.spark.scheduler.DAGScheduler - Job 3 finished: load at NativeMethodAccess$Impl.java:0, took 0.121290 s
[Thread-3] INFO org.apache.spark.sql.execution.datasources.FileSourceStrategy - Pushed Filters:
[Thread-3] INFO org.apache.spark.sql.execution.datasources.FileSourceStrategy - Post-Scan Filters:

Out[7]: DataFrame@student_id: int, school: string, sex: string, age: int, address_type: string, family_size: string, parent_status: string, mother_education: string, father_education: string, mother_job: string, father_job: string, school_choice_reason: string, guardian: string, travel_time: string, study_time: string, class_failures: int, school_support: string, family_support: string, extra_paid_classes: string, activities: string, nursery_school: string, higher_ed: string, internet_access: string, romantic_relationship: string, family_relationship: int, free_time: int, social: int, weekday_alcohol: int, weekend_alcohol: int, health: int, absences: int, grade_1: int, grade_2: int, final_grade: int]
```

Đọc dữ liệu từ student\_portuguese\_clean.csv vào spark dataframe

### 4.3 Phân tích dữ liệu thăm dò (EDA)

**Bước 1:** In schema cùng với số dòng và số cột của df\_portuguese bao gồm 34 thuộc tính và 649 dòng

#### EDA

```
In [8]: # Hiển thị thông tin dữ liệu
print('Portuguese')
print(f"Số dòng: {df_portuguese.count()}")
print(f"Số cột: {len(df_portuguese.columns)}")

df_portuguese.printSchema()
```

Số dòng: 649  
Số cột: 34

số dòng và cột của dataframe df\_portuguese khi chưa được xử lý

```
root
|-- student_id: integer (nullable = true)
|-- school: string (nullable = true)
|-- sex: string (nullable = true)
|-- age: integer (nullable = true)
|-- address_type: string (nullable = true)
|-- family_size: string (nullable = true)
|-- parent_status: string (nullable = true)
|-- mother_education: string (nullable = true)
|-- father_education: string (nullable = true)
|-- mother_job: string (nullable = true)
|-- father_job: string (nullable = true)
|-- school_choice_reason: string (nullable = true)
|-- guardian: string (nullable = true)
|-- travel_time: string (nullable = true)
|-- study_time: string (nullable = true)
|-- class_failures: integer (nullable = true)
|-- school_support: string (nullable = true)
|-- family_support: string (nullable = true)
|-- extra_paid_classes: string (nullable = true)
|-- activities: string (nullable = true)
|-- nursery_school: string (nullable = true)
|-- higher_ed: string (nullable = true)
|-- internet_access: string (nullable = true)
|-- romantic_relationship: string (nullable = true)
|-- family_relationship: integer (nullable = true)
|-- free_time: integer (nullable = true)
|-- social: integer (nullable = true)
|-- weekday_alcohol: integer (nullable = true)
|-- weekend_alcohol: integer (nullable = true)
|-- health: integer (nullable = true)
|-- absences: integer (nullable = true)
|-- grade_1: integer (nullable = true)
|-- grade_2: integer (nullable = true)
|-- final_grade: integer (nullable = true)
```

Schema của dataframe df\_portuguese khi chưa được xử lý

**Bước 2:** In schema cùng với số dòng và số cột của df\_math bao gồm 34 thuộc tính và 395 dòng

```
In [9]: # Hiển thị thông tin dữ liệu
print('Math')
print(f"Số dòng: {df_math.count()}")
print(f"Số cột: {len(df_math.columns)}")
df_math.printSchema()

Số dòng: 395
Số cột: 34

DataFrame[student_id: int, school: string, sex: string, age: int, address_type: string, family_size: string, parent_status: string, mother_education: string, father_education: string, mother_job: string, father_job: string, school_choice_reason: string, guardian: string, travel_time: string, study_time: string, class_failures: int, school_support: string, family_support: string, extra_paid_classes: string, activities: string, nursery_school: string, higher_ed: string, internet_access: string, romantic_relationship: string, family_relationship: int, free_time: int, social: int, weekday_alcohol: int, weekend_alcohol: int, health: int, absences: int, grade_1: int, grade_2: int, final_grade: int]

root
|-- student_id: integer (nullable = true)
|-- school: string (nullable = true)
|-- sex: string (nullable = true)
|-- age: integer (nullable = true)
|-- address_type: string (nullable = true)
|-- family_size: string (nullable = true)
|-- parent_status: string (nullable = true)
|-- mother_education: string (nullable = true)
```

số dòng, cột và schema của dataframe df\_math khi chưa được xử lý

### **Bước 3:** Tiến hành merge 2 file csv thành một

Vì xem mỗi dòng dữ liệu là thành tích học tập trong 1 năm của học sinh. Và vì 2 dataframe có cùng schema nên nhóm sẽ merge thành 1 dataframe df bằng hàm unionByName(), nối liền dưới dựa theo tên cột.

Nhưng trước khi merge cần đánh dấu lại thành tích học tập của môn nào. Tạo cột “subject\_code” và nếu là điểm math sẽ mang giá trị 1, portuguese là 0

#### **Merging the two csv files**

Đánh dấu môn học trước khi gộp 2 dataset lại

- portuguese: 0
- math: 1

```
In [10]: # Thêm cột portuguese cho math với giá trị 0
df_portuguese = df_portuguese.withColumn("subject_code", lit(0))

# Thêm cột subject_code cho math với giá trị 1
df_math = df_math.withColumn("subject_code", lit(1))
```

Đánh dấu môn học trước khi gộp 2 dataset lại

```
In [11]: # Kiểm tra trước khi kết hợp
# Lấy danh sách các cột
cols_math = df_math.columns
cols_portuguese = df_portuguese.columns

# Kiểm tra xem các cột có giống nhau không
if cols_math != cols_portuguese:
    print("ERROR: Hai DataFrame không có cùng cấu trúc!")
else:
    # Kết hợp hai DataFrame
    df = df_math.unionByName(df_portuguese)
    print("OK")
```

OK

Kiểm tra schema trước khi merge bằng hàm unionByName()

### **Bước 4:**

```
In [12]: # Tạo một cột mới với giá trị ID tăng dần bắt đầu từ 1
df = df.withColumn("id_temp", monotonically_increasing_id())

# Sử dụng Window function để sắp xếp lại các ID từ 1 đến N
from pyspark.sql.window import Window
from pyspark.sql.functions import row_number

windowSpec = Window.orderBy("id_temp")
df = df.withColumn("id", row_number().over(windowSpec))

# Loại bỏ cột student_id và id_temp, giữ lại cột id với giá trị từ 1 đến N
df = df.drop("student_id", "id_temp")

df.show()
```

Thêm row id cho dataframe mới, và loại bỏ student\_id

	s romantic_relationship	f family_relationship	t free_time	social	w weekday_alcohol	w weekend_alcohol	h health	a absences	g grade_1	g grade_2	f final_grade	su subject_code	i id	
o	no	4	3	4	1	1	3	6	5	6	6	1	1	
s	no	5	3	3	1	1	3	4	5	5	6	1	2	
s	no	4	3	2	2	3	3	10	7	8	10	1	3	
s	yes	3	2	2	1	1	5	2	15	14	15	1	4	
o	no	4	3	2	1	2	5	4	6	10	10	1	5	
s	no	5	4	2	1	2	5	10	15	15	15	1	6	
s	no	4	4	4	1	1	3	0	12	12	11	1	7	
o	no	4	1	4	1	1	1	6	6	5	6	1	8	
s	no	4	2	2	1	1	1	0	16	18	19	1	9	
s	no	5	5	1	1	1	5	0	14	15	15	1	10	
s	no	3	3	3	1	2	2	0	10	8	9	1	11	
s	no	5	2	2	1	1	4	4	10	12	12	1	12	
s	no	4	3	3	1	3	5	2	14	14	14	1	13	
s	no	5	4	3	1	2	3	2	10	10	11	1	14	
s	yes	4	5	2	1	1	3	0	14	16	16	1	15	
s	no	4	4	4	1	2	2	4	14	14	14	1	16	
s	no	3	2	3	1	2	2	6	13	14	14	1	17	
o	no	5	3	2	1	1	4	4	8	10	10	1	18	
s	no	5	5	5	2	4	5	16	6	5	5	1	19	
s	no	3	1	3	1	3	5	4	8	10	10	1	20	

Row id cho dataframe mới đã được thêm

Loại student\_id vì sau khi merge dataframe, sẽ có hiện tượng trùng student\_id (trong phần miêu tả dataset tại Additional Note tác giả có đề cập đến điều này [2]). Student\_id ở 2 dataset trùng nhau nhưng không của cùng của 1 học sinh, thế nên student\_id không còn giá trị khai thác.

**Bước 5:** Thống kê các thuộc tính định lượng, định tính như count, max, min, stdev, tứ phân vị, ... của mỗi thuộc tính trong dataframe

**Thống kê** thống kê các thuộc tính định lượng như: count, max, min, mean, stddev, tóm phán vị...

```
In [13]: df.summary().show()
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|summary|school| sex|      age|address_type|      family_size| parent_status| mother_education|
father_education|mother_job|father_job|school_choice_reason|guardian| travel_time| study_time| class_failures|
school_support|family_support|extra_paid_classes|activities|nursery_school|higher_ed|internet_access|romantic_relationship|
family_relationship|      free_time|      social| weekday_alcohol| weekend_alcohol|
health|      absences|      grade_1|      grade_2|      final_grade|      subject_code|
id|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| count| 1044| 1044|      1044|      1044|      1044|      1044|      1044|      1044|
1044|      1044|      1044|      1044|      1044|      1044|      1044|      1044|      1044|
1044|          1044|          1044|          1044|          1044|          1044|          1044|          1044|          1044|
1044|          1044|          1044|          1044|          1044|          1044|          1044|          1044|          1044|
| mean|    NULL|    NULL| 16.726053639846743|    NULL|    NULL|    NULL|    NULL|    NULL|
NULL|    NULL|    NULL|    NULL|    NULL|    NULL|    NULL| 0.26436781609195403|
```

Thông các thuộc tính định lượng, định tính của mỗi thuộc tính trong dataframe

	summary	school	sex	age	address_type	family_size	parent_status	mother_education	father_education	mother_job	father_job
count	1044	1044		1044		1044		1044		1044	
mean	NULL	NULL		16.726053639846743		NULL		NULL		NULL	
stddev	NULL	NULL		1.2399746931649538		NULL		NULL		NULL	
min	GP	F		15	Rural	Greater than 3	Apart	5th to 9th grade	5th to 9th grade	at_home	at home
25%	NULL	NULL		16	NULL	NULL	NULL	NULL	NULL	NULL	NULL
50%	NULL	NULL		17	NULL	NULL	NULL	NULL	NULL	NULL	NULL
75%	NULL	NULL		18	NULL	NULL	NULL	NULL	NULL	NULL	NULL
max	MS	M		22	Urban	Less than or equal to 6	Living together	secondary education	secondary education	teacher	teacher

[Thread-3] INFO org.apache.spark.sql.catalyst.expressions.codegen.CodeGenerator - Code generated in 10.927981 ms

	weekend_alcohol	health	absences	grade_1	grade_2	final_grade	subject_code	
	1044	1044	1044	1044	1044	1044	1044	1044
2.2844827586206895	3.543103448275862	4.434865900383142	[11.21360153256705]	11.246168582375478	[11.341954022988507]	0.3783524904214559	522	
1.2851048062618582	[1.4247034122490199]	6.210016564763475	[2.983393934015448]	3.28507106798263	[3.864795804383607]	0.48520863015415704	301.5211435372319	
1	1	1	0	0	0	0	0	1
1	3	3	0	9	9	10	0	261
2	4	4	2	11	11	11	0	522
3	5	5	6	13	13	14	1	783
5	5	5	75	19	19	20	1	1044

	higher_ed	internet_access	romantic_relationship	family_relationship	free_time	social	weekday_alcohol	weekend_alcohol
1	1044	1044	1044	1044	1044	1044	1044	1044
..	NULL	NULL	NULL	3.935823754789272	3.2011494252873565	3.1561302681992336	1.4942528735632183	2.2844827586206895
.	NULL	NULL	NULL	0.9334007716663543	1.0315068335871396	1.1525746602053926	0.9117142815596276	1.2851048062618582
	no	no	no	1	1	1	1	1
-	NULL	NULL	NULL	4	3	2	1	1
.	NULL	NULL	NULL	4	3	3	1	2
.	NULL	NULL	NULL	5	4	4	2	3
:	yes	yes	yes	5	5	5	5	5

## Bước 6: Thống kê số dòng null trong mỗi thuộc tính

Thống kê số dòng null trong tập dữ liệu

```
In [14]: # Thống kê Lượng giá trị null cho từng cột
null_counts = df.agg(*[count(when(col(c).isNull(), c)).alias(c) for c in df.columns])
total_null_counts_percent = null_counts.agg(*[sum(col(c)).alias(c) for c in df.columns])

# Phân tích missing data
total_rows = df.count()
null_percentages = null_counts.select(*[(col(c) / total_rows).alias(c + '_percent') for c in null_counts.columns])

# Hiển thị
missing_data = null_counts.union(null_percentages).toPandas().transpose()
missing_data.columns = ["Number missing", "Percent %"]
missing_data = missing_data.sort_values(by="Percent %", ascending=False)
display(missing_data)
```

	Number missing	Percent %
school	0.0	0.0
weekday_alcohol	0.0	0.0
higher_ed	0.0	0.0
internet_access	0.0	0.0
romantic_relationship	0.0	0.0
family_relationship	0.0	0.0
free_time	0.0	0.0
social	0.0	0.0
weekend_alcohol	0.0	0.0
activities	0.0	0.0
health	0.0	0.0
absences	0.0	0.0

Nhận thấy không có cột nào bị mất dữ liệu. Ngoài ra, nhận thấy tên cột không ký tự đặc biệt, cũng không có khoảng cách. Kiểu dữ liệu cũng phù hợp.

	Number missing	Percent %
school	0.0	0.0
weekday_alcohol	0.0	0.0
higher_ed	0.0	0.0
internet_access	0.0	0.0
romantic_relationship	0.0	0.0
family_relationship	0.0	0.0
free_time	0.0	0.0
social	0.0	0.0
weekend_alcohol	0.0	0.0
activities	0.0	0.0
health	0.0	0.0
absences	0.0	0.0
grade_1	0.0	0.0
grade_2	0.0	0.0
final_grade	0.0	0.0
subject_code	0.0	0.0
nursery_school	0.0	0.0
extra_paid_classes	0.0	0.0
sex	0.0	0.0
mother_job	0.0	0.0
age	0.0	0.0
address_type	0.0	0.0
family_size	0.0	0.0
parent_status	0.0	0.0
mother_education	0.0	0.0
father_education	0.0	0.0
father_job	0.0	0.0
family_support	0.0	0.0
school_choice_reason	0.0	0.0
guardian	0.0	0.0
travel_time	0.0	0.0
study_time	0.0	0.0
class_failures	0.0	0.0
school_support	0.0	0.0
id	0.0	0.0

## 4.4 Tiết xử lý dữ liệu

### 4.4.1 Chọn lọc và loại bỏ thuộc tính không cần thiết và ít ảnh hưởng đến mô hình dự đoán

Trong tập dữ liệu, nhóm sẽ ưu tiên giữ các thuộc tính liên quan đến việc học (nhóm 1) và về học sinh (nhóm 2), hơn là về gia đình học sinh (nhóm 3):

- Nhóm 1:

- class\_failures, study\_time, absences, grade\_1, grade\_2
- school\_choice\_reason, travel\_time, school\_support, family\_support, extra\_paid\_classes, higher\_ed, internet\_access, school
- Nhóm 2:
  - sex, age, address\_type, activities, romantic\_relationship, family\_relationship, free\_time, social, weekday\_alcohol, weekend\_alcohol, health, nursery\_school
- Nhóm 3:
  - family\_size, parent\_status, mother\_education, father\_education, mother\_job, father\_job, guardian

```
In [15]: # Bỏ đi tính cá nhân trong tập dữ liệu
drop_cols_1 = ['school']

# Bỏ các thuộc tính ít cần
drop_cols_2 = ['family_size', 'parent_status', 'mother_job', 'father_job', 'nursery_school', 'travel_time']

# Kết hợp cả hai danh sách
drop_cols = drop_cols_1 + drop_cols_2
df = df.drop(*drop_cols)
```

```
In [16]: df.printSchema()

root
 |-- sex: string (nullable = true)
 |-- age: integer (nullable = true)
 |-- address_type: string (nullable = true)
 |-- mother_education: string (nullable = true)
 |-- father_education: string (nullable = true)
 |-- school_choice_reason: string (nullable = true)
 |-- guardian: string (nullable = true)
 |-- study_time: string (nullable = true)
 |-- class_failures: integer (nullable = true)
 |-- school_support: string (nullable = true)
 |-- family_support: string (nullable = true)
 |-- extra_paid_classes: string (nullable = true)
 |-- activities: string (nullable = true)
 |-- higher_ed: string (nullable = true)
 |-- internet_access: string (nullable = true)
 |-- romantic_relationship: string (nullable = true)
 |-- family_relationship: integer (nullable = true)
 |-- free_time: integer (nullable = true)
 |-- social: integer (nullable = true)
 |-- weekday_alcohol: integer (nullable = true)
 |-- weekend_alcohol: integer (nullable = true)
 |-- health: integer (nullable = true)
 |-- absences: integer (nullable = true)
 |-- grade_1: integer (nullable = true)
 |-- grade_2: integer (nullable = true)
 |-- final_grade: integer (nullable = true)
 |-- subject_code: integer (nullable = false)
 |-- id: integer (nullable = false)
```

Kiểm tra lại số dòng, số cột và xem lại dữ liệu:

```
In [17]: # Sau khi chọn lọc thuộc tính
print(f"Số dòng: {df.count()}")
print(f"Số cột: {len(df.columns)}")
df.show(1)

Số dòng: 1044
Số cột: 28

[dag-scheduler-event-loop] INFO org.apache.spark.storage.memory.MemoryStore - Block broadcast_25 stored as val
```

```
In [17]: # Sau khi chọn lọc thuộc tính
print(f"Số dòng: {df.count()}")
print(f"Số cột: {len(df.columns)}")

df.show(1)
+-----+-----+-----+-----+-----+-----+-----+
|sex|age|address_type|mother_education|school_choice_reason|guardian| study_time|class_failures
|school_support|family_support|extra_paid_classes|activities|higher_ed|internet_access|romantic_relationship|fami
ly_relationship|free_time|social|weekday_alcohol|weekend_alcohol|health|absences|grade_1|grade_2|final_grade|subj
ect_code| id|
+-----+-----+-----+-----+-----+-----+-----+
| F| 18| Urban|higher education|higher education| course| mother|2 to 5 hours| 0
| yes| no| 1| 1| 3| 6| 5| 6| 6| 1| 1|
+-----+-----+-----+-----+-----+-----+-----+
only showing top 1 row
```

Kiểm tra giá trị Distinct của từng thuộc tính để phục vụ các bước tiền xử lý sau:

```
In [18]: # Loại bỏ cột student_id, grade_1, grade_2, final_grade không cần xem số giá trị duy nhất
df_distinct = df.drop("id", "grade_1", "grade_2", "final_grade")

# Tính số giá trị duy nhất của từng cột hiện tại
distinct_counts = df_distinct.agg(*([countDistinct(col(c)).alias(c) for c in df_distinct.columns]))

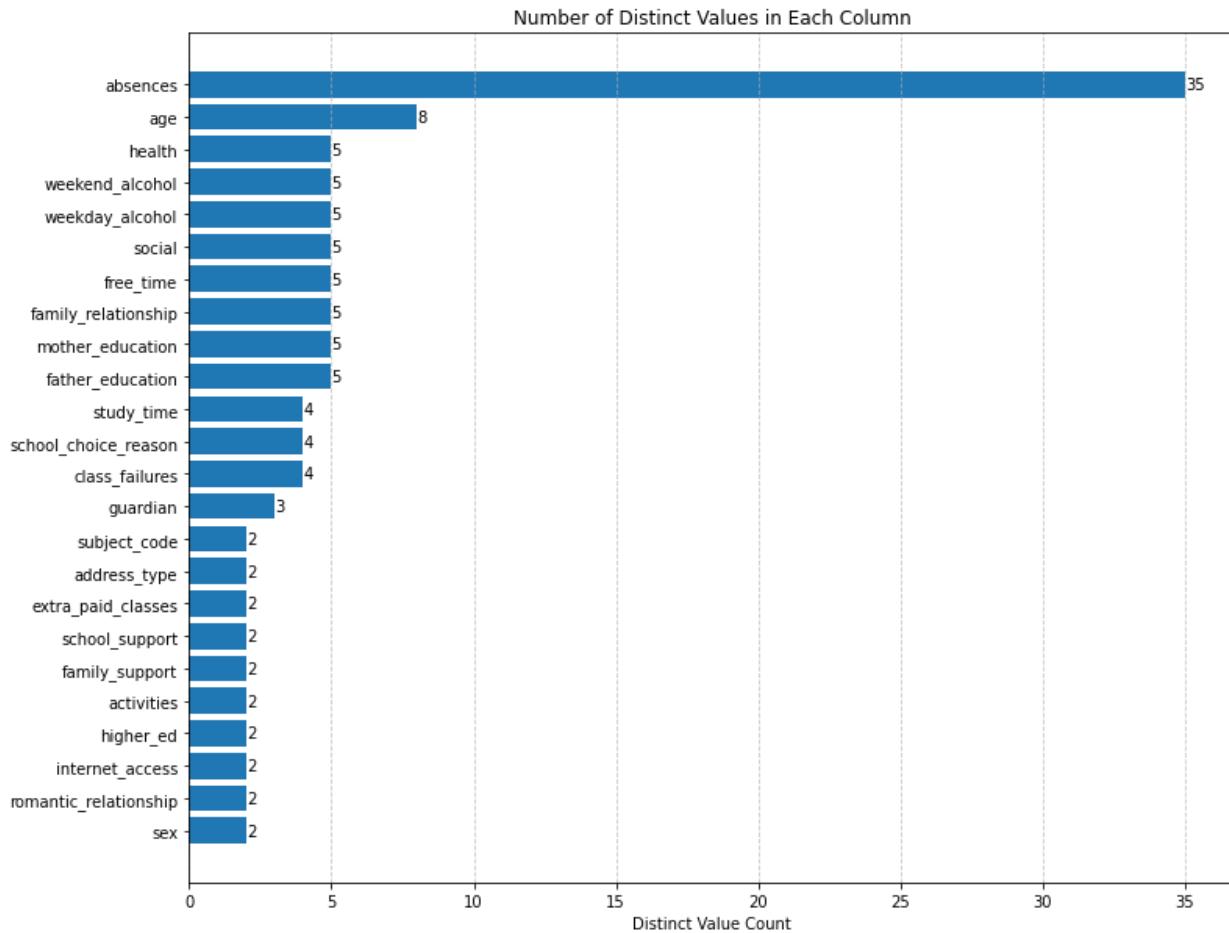
# Chuyển kết quả sang Pandas DataFrame (dữ liệu sau khi count nhỏ)
distinct_counts_pd = distinct_counts.toPandas().T.reset_index()
distinct_counts_pd.columns = ['column', 'distinct_count']

# Sắp xếp kết quả từ ít giá trị nhất đến nhiều giá trị nhất
sorted_distinct_counts_pd = distinct_counts_pd.sort_values(by='distinct_count')

# Vẽ biểu đồ
plt.figure(figsize=(12, 10))
bars = plt.barh(sorted_distinct_counts_pd['column'], sorted_distinct_counts_pd['distinct_count'])
plt.xlabel('Distinct Value Count')
plt.title('Number of Distinct Values in Each Column')
plt.grid(axis='x', linestyle='--', alpha=0.7)

# Thêm giá trị label cho mỗi bar
for bar in bars:
    plt.text(bar.get_width(), bar.get_y() + bar.get_height()/2, f'{int(bar.get_width())}', va='center', ha='left')

plt.show()
```



#### 4.4.2 Xử lý các thuộc tính ở định dạng non-numeric

##### 4.4.2.1 Dạng thuộc tính có nhiều hơn 2 giá trị distinct

4.4.2.1.1 mother\_education, father\_education, school\_choice\_reason, guardian

**Bước 1:** Xác định giá trị distinct:

```
In [20]: # Lấy tất cả các thẻ 'loại duy nhất'
unique_mother_edu_df = df.select("mother_education").distinct()
unique_father_edu_df = df.select("father_education").distinct()

# Loại bỏ các dòng có giá trị rỗng trong cột "mother_education"
unique_mother_edu_df.show(truncate=False)
unique_father_edu_df.show(truncate=False)
```

```
+-----+
|mother_education |
+-----+
|5th to 9th grade
|none
|secondary education
|primary education (4th grade)
|higher education
+-----+
```

```
+-----+
|father_education |
+-----+
|5th to 9th grade
|none
|secondary education
|primary education (4th grade)
|higher education
+-----+
```

```
In [25]: # Lấy tất cả các giá trị duy nhất
unique_school_choice_reason_df = df.select("school_choice_reason").distinct()
unique_guardian_df = df.select("guardian").distinct()

unique_school_choice_reason_df.show(truncate=False)
unique_guardian_df.show(truncate=False)
```

```
+-----+
|guardian|
+-----+
|father  |
|mother  |
|other   |
+-----+
```

**Bước 2:** Thêm cột cho mỗi giá trị distinct và gán giá trị 1 nếu giá trị distinct đó xuất hiện, 0 nếu ngược lại

```
In [21]: # Chuyển danh sách học vấn thành một danh sách Python
list_mother_edu = [row["mother_education"] for row in unique_mother_edu_df.toLocalIterator()]
list_father_edu = [row["father_education"] for row in unique_father_edu_df.toLocalIterator()]

# Thêm cột cho mỗi giá trị distinct và gán giá trị 1 nếu giá trị distinct đó xuất hiện, ngược lại gán giá trị 0
for mother_edu in list_mother_edu:
    mother_edu_column_name = f"mother_education_is_{mother_edu.replace(' ', '_')}"
    df = df.withColumn(mother_edu_column_name, col("mother_education").contains(mother_edu).cast("int"))

for father_edu in list_father_edu:
    father_edu_column_name = f"father_education_is_{father_edu.replace(' ', '_')}"
    df = df.withColumn(father_edu_column_name, col("father_education").contains(father_edu).cast("int"))
```

```
In [26]: # Chuyển thành một danh sách Python
list_school_choice = [row["school_choice_reason"] for row in unique_school_choice_reason_df.toLocalIterator()]
list_guardian = [row["guardian"] for row in unique_guardian_df.toLocalIterator()]

# Thêm cột cho mỗi giá trị distinct và gán giá trị 1 nếu giá trị distinct đó xuất hiện, ngược lại gán giá trị 0
for school_choice in list_school_choice:
    school_choice_column_name = f"{school_choice.replace(' ', '_)}_school_choice"
    df = df.withColumn(school_choice_column_name, col("school_choice_reason").contains(school_choice).cast("int"))

for guardian in list_guardian:
    guardian_column_name = f"{guardian.replace(' ', '_)}_guardian"
    df = df.withColumn(guardian_column_name, col("guardian").contains(guardian).cast("int"))|
```

### Bước 3: In ra kết quả

```
In [22]: # Chọn cột "mother_education" và các cột bắt đầu từ "5th to 9th grade" trả về và hiển thị
df.select(["mother_education", "father_education"]
          + [col(column) for column in df.columns[df.columns.index("mother_education_is_5th_to_9th_grade"):]])
.show(truncate=False)
```

Ví dụ với "mother\_education" và "father\_education"

mother_education	father_education	mother_education_is_5th_to_9th_grade	mother_education_is_none	mother_education_is_secondary_education
higher education	higher education	0	0	0
primary education (4th grade)	primary education (4th grade)	0	0	0
primary education (4th grade)	primary education (4th grade)	0	0	0
higher education	5th to 9th grade	0	0	0
secondary education	secondary education	0	0	1
higher education	secondary education	0	0	0
5th to 9th grade	5th to 9th grade	1	0	0
higher education	higher education	0	0	0
secondary education	5th to 9th grade	0	0	1
secondary education	higher education	0	0	1
higher education	higher education	0	0	0
5th to 9th grade	primary education (4th grade) 1	0	0	0
higher education	higher education	0	0	0
higher education	secondary education	0	0	0
5th to 9th grade	5th to 9th grade	1	0	0
higher education	higher education	0	0	0
higher education	higher education	0	0	0
secondary education	secondary education	0	0	1
secondary education	5th to 9th grade	0	0	1
higher education	secondary education	0	0	0

only showing top 20 rows

Code áp dụng cho "school\_choice\_reason" và "guardian" cũng tương tự

```
In [27]: df.select(["school_choice_reason", "guardian"]
+ [col(column) for column in df.columns[df.columns.index("reputation_school_choice"):]])
.show(truncate=False)
```

#### **Bước 4:** Xóa các cột cũ

```
In [23]: # Tiến hành xóa sau khi xử lý xong
df = df.drop("mother_education", "father_education", "school_choice_reason", "guardian")
```

#### 4.4.2.1.2 study\_time:

**Bước 1:** Xem giá trị distinct

```
In [30]: # Lấy tất cả các giá trị duy nhất
unique_study_time_df = df.select("study_time").distinct()

unique_study_time_df.show(truncate=False)

+-----+
|study_time|
+-----+
|5 to 10 hours|
|>10 hours   |
|<2 hours    |
|2 to 5 hours|
+-----+
```

**Bước 2:** Phân nhóm theo số, thời gian càng nhiều số càng lớn. Và kiểm tra kết quả.

```
In [31]: # Chuyển đổi cột travel_time thành giá trị số

df = df.withColumn("study_time_encoded",
                    when(col("study_time") == "<2 hours", 0)
                     .when(col("study_time") == "2 to 5 hours", 1)
                     .when(col("study_time") == "5 to 10 hours", 2)
                     .otherwise(3))

df.select("study_time", "study_time_encoded").show(truncate=False)

+-----+-----+
|study_time |study_time_encoded|
+-----+-----+
|2 to 5 hours|1
|2 to 5 hours|1
|2 to 5 hours|1
|5 to 10 hours|2
|2 to 5 hours|1
|5 to 10 hours|2
|<2 hours|0
|2 to 5 hours|1
|5 to 10 hours|2
|<2 hours|0
|5 to 10 hours|2
|2 to 5 hours|1
|<2 hours|0
|<2 hours|0
+-----+-----+
only showing top 20 rows
```

### **Bước 3:** Xóa các cột cũ

```
In [32]: df = df.drop("study time")
```

#### 4.4.2.2 Dạng thuộc tính có 2 giá trị distinct

#### **4.4.2.2.1 Dạng Yes No (binary)**

Gồm các thuộc tính: "school\_support", "family\_support", "extra\_paid\_classes", "activities", "higher\_ed", "internet\_access", "romantic\_relationship"

### Bước 1: Xem các giá trị của thuộc tính

```
In [39]: columns = ["school_support", "family_support",
                   "extra_paid_classes", "activities",
                   "higher_ed", "internet_access",
                   "romantic_relationship"]

df.select(columns).show(truncate=False)
```

school_support	family_support	extra_paid_classes	activities	higher_ed	internet_access	romantic_relationship
yes	no	no	no	yes	no	no
no	yes	no	no	yes	yes	no
yes	no	yes	no	yes	yes	no
no	yes	yes	no	yes	yes	yes
no	yes	yes	yes	yes	no	no
no	yes	yes	yes	yes	yes	no
no	no	no	no	yes	yes	no
yes	yes	no	no	yes	no	no
no	yes	yes	no	yes	yes	no
no	yes	yes	yes	yes	yes	no
no	yes	yes	no	yes	yes	no
no	yes	yes	yes	yes	yes	no
no	yes	yes	no	yes	yes	no
no	yes	no	no	yes	yes	yes
no	yes	no	no	yes	yes	no
no	yes	yes	yes	yes	yes	no
yes	yes	no	yes	yes	no	no
no	yes	no	yes	yes	yes	no
no	no	yes	yes	yes	yes	no

only showing top 20 rows

### Bước 2: Thay thế Yes bằng 1, và No bằng 0

```
In [41]: from pyspark.sql.functions import when, col

# Lặp qua các cột và thay thế giá trị "yes" bằng 1, ngược lại bằng 0
for col_name in columns:
    df = df.withColumn(col_name, when(col(col_name) == "yes", 1).otherwise(0))
```

↓

### Bước 3: Kiểm tra kết quả

```
In [42]: # Hiển thị DataFrame kết quả  
df.select(columns).show()
```

school_support	family_support	extra_paid_classes	activities	higher_ed	internet_access	romantic_relationship
1	0	0	0	1	0	0
0	1	0	0	1	1	0
1	0	1	0	1	1	0
0	1	1	1	1	1	1
0	1	1	0	1	0	0
0	1	1	1	1	1	0
0	0	0	0	1	1	0
1	1	0	0	1	0	0
0	1	1	0	1	1	0
0	1	1	1	1	1	0
0	1	1	0	1	1	0
0	1	0	1	1	1	0
0	1	1	1	1	1	0
0	1	1	0	1	1	0
0	1	0	0	1	1	1
0	1	0	0	1	1	0

#### 4.4.2.2.2 Còn lại

Gồm các thuộc tính: "sex", "address\_type"

#### Bước 1: Xem các giá trị distinct

```
In [34]: for col in columns:  
    df.select(col).distinct().show(truncate=False)
```

+---+
sex
+---+
F
M
+---+
+---+
address_type
+---+
Urban
Rural
+---+

#### Bước 2: Thu thập giá trị distinct của từng thuộc tính thành mảng 2 chiều để xử lý sau

```
In [35]: distinct_values = []

# Lặp qua các cột và lưu các giá trị distinct vào mảng 2 chiều
for column_name in columns:
    distinct_set = set(df.select(column_name).distinct().rdd.flatMap(lambda x: x).toLocalIterator())
    distinct_set.discard(None)
    distinct_values.append(list(distinct_set))

# Hiển thị kết quả
for i, col in enumerate(columns):
    print(f"Distinct values for column '{col}': {distinct_values[i]}")

# In kết quả mảng 2 chiều
print("2D Array of distinct values:")
for row in distinct_values:
    print(row)

Distinct values for column 'sex': ['M', 'F']
Distinct values for column 'address_type': ['Urban', 'Rural']
2D Array of distinct values:
['M', 'F']
['Urban', 'Rural']
```

**Bước 3:** Tạo cột mới mang giá trị 1 hoặc 0 để ghi nhận một dòng ma giá trị đầu của distinct theo sex và address\_type

```
In [36]: for i, col_name in enumerate(columns):
    first_value = distinct_values[i][0]
    first_value = first_value.replace(' ', '_')
    new_col_name = col_name + "_is_" + first_value
    df = df.withColumn(new_col_name, when(df[col_name] == first_value, 1).otherwise(0))
    ...
```

**Bước 4:** Kiểm tra kết quả

```
In [37]: df.select("sex", "address_type", "sex_is_M", "address_type_is_Urban").show()

+---+-----+-----+
|sex|address_type|sex_is_M|address_type_is_Urban|
+---+-----+-----+
| F|    Urban|     0|          1|
| M|    Urban|     1|          1|
| M|    Urban|     1|          1|
| F|    Urban|     0|          1|
| M|    Urban|     1|          1|
| M|    Urban|     1|          1|
| F|    Urban|     0|          1|
| F|    Urban|     0|          1|
| M|    Urban|     1|          1|
| M|    Urban|     1|          1|
| M|    Urban|     1|          1|
| F|    Urban|     0|          1|
| F|    Urban|     0|          1|
| F|    Urban|     0|          1|
| M|    Urban|     1|          1|
| M|    Urban|     1|          1|
+---+-----+-----+
only showing top 20 rows
```

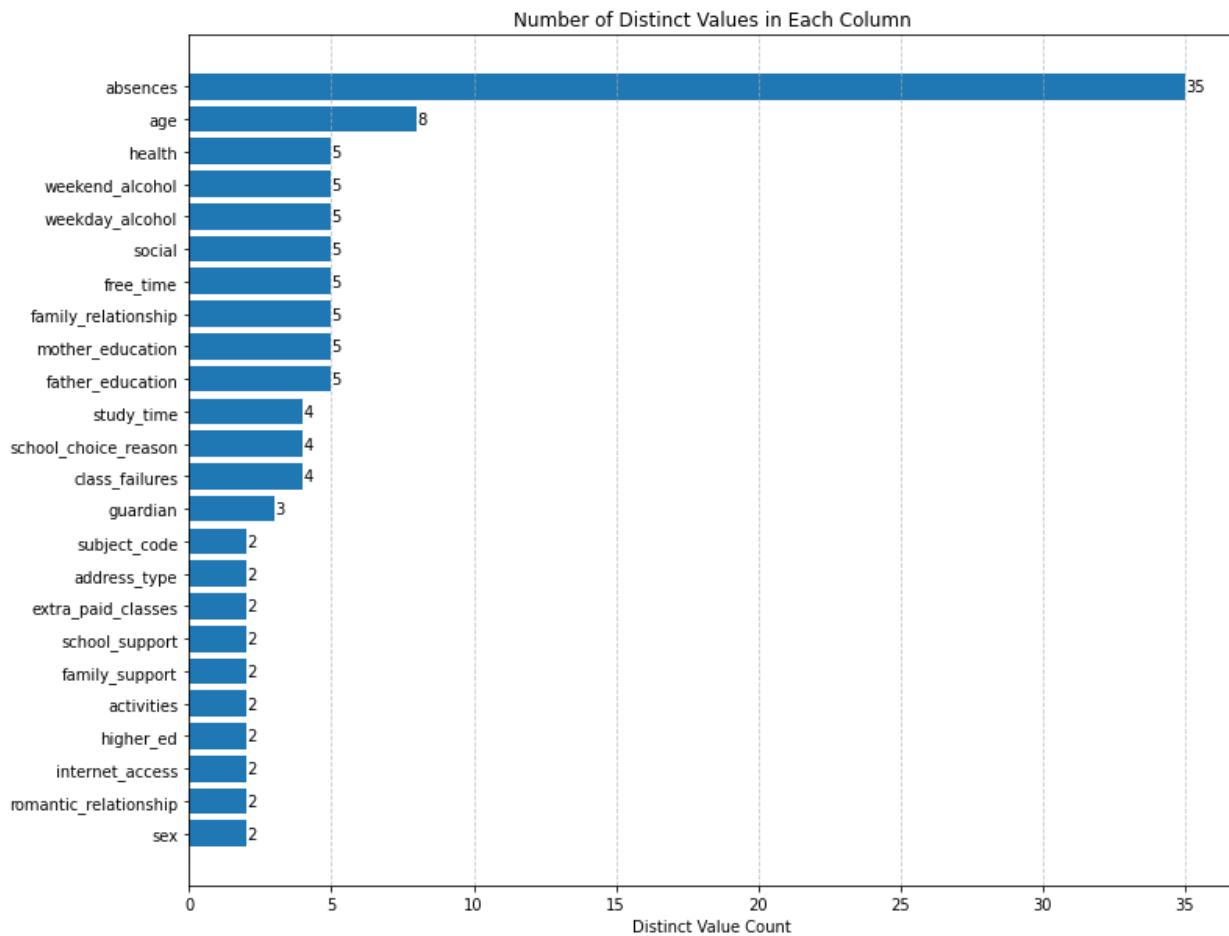
**Bước 5:** Xóa 2 thuộc tính "sex" và "address\_type" sau khi xử lý xong

```
[38]: # Xóa cột release_date  
df = df.drop("sex", "address_type")
```

#### 4.4.3 Xử lý các thuộc tính numeric

Gồm: age, class\_failures, family\_relationship, free\_time, social, weekday\_alcohol, weekend\_alcohol, health, absences, subject\_code

Vì số lượng giá trị distinct không nhiều, và range max cũng chỉ có đến 35. Nên nhóm quyết định sẽ giữ nguyên và không xử lý thêm.



#### 4.4.4 Lựa chọn thuộc tính (Feature Selection)

**Bước 1:** Chọn các cặp thuộc tính có tương quan cao để loại bỏ bằng ma trận Correlation

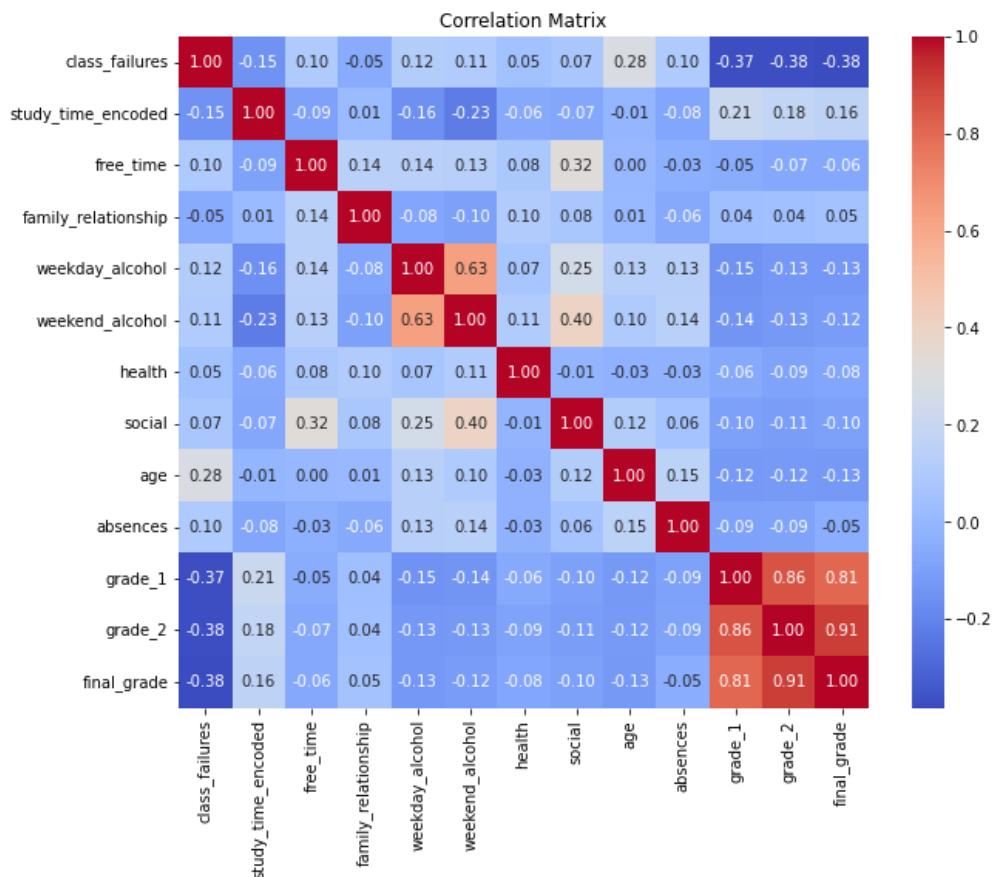
Dùng ma trận tương quan để kiểm tra độ tương quan giữa các thuộc tính sau:

"class\_failures", "study\_time\_encoded", "free\_time", "family\_relationship",  
 "weekday\_alcohol", "weekend\_alcohol", "health", "social", "age", "absences", "grade\_1",  
 "grade\_2", "final\_grade"

```
In [45]: # Chuyển đổi DataFrame PySpark thành DataFrame pandas
pandas_df = df.select("class_failures", "study_time_encoded", "free_time",
                      "family_relationship", "weekday_alcohol", "weekend_alcohol",
                      "health", "social", "age", "absences",
                      "grade_1", "grade_2", "final_grade").toPandas()

# Vẽ biểu đồ ma trận tương quan bằng seaborn
correlation_matrix = pandas_df.corr()
plt.figure(figsize=(10, 8))

# Vẽ heatmap
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix')
plt.show()
```



Các thuộc tính đa phần không có mối tương quan cao. Nhưng còn 1 cặp có độ tương quan là 0.63:

- weekday\_alcohol - weekend\_alcohol

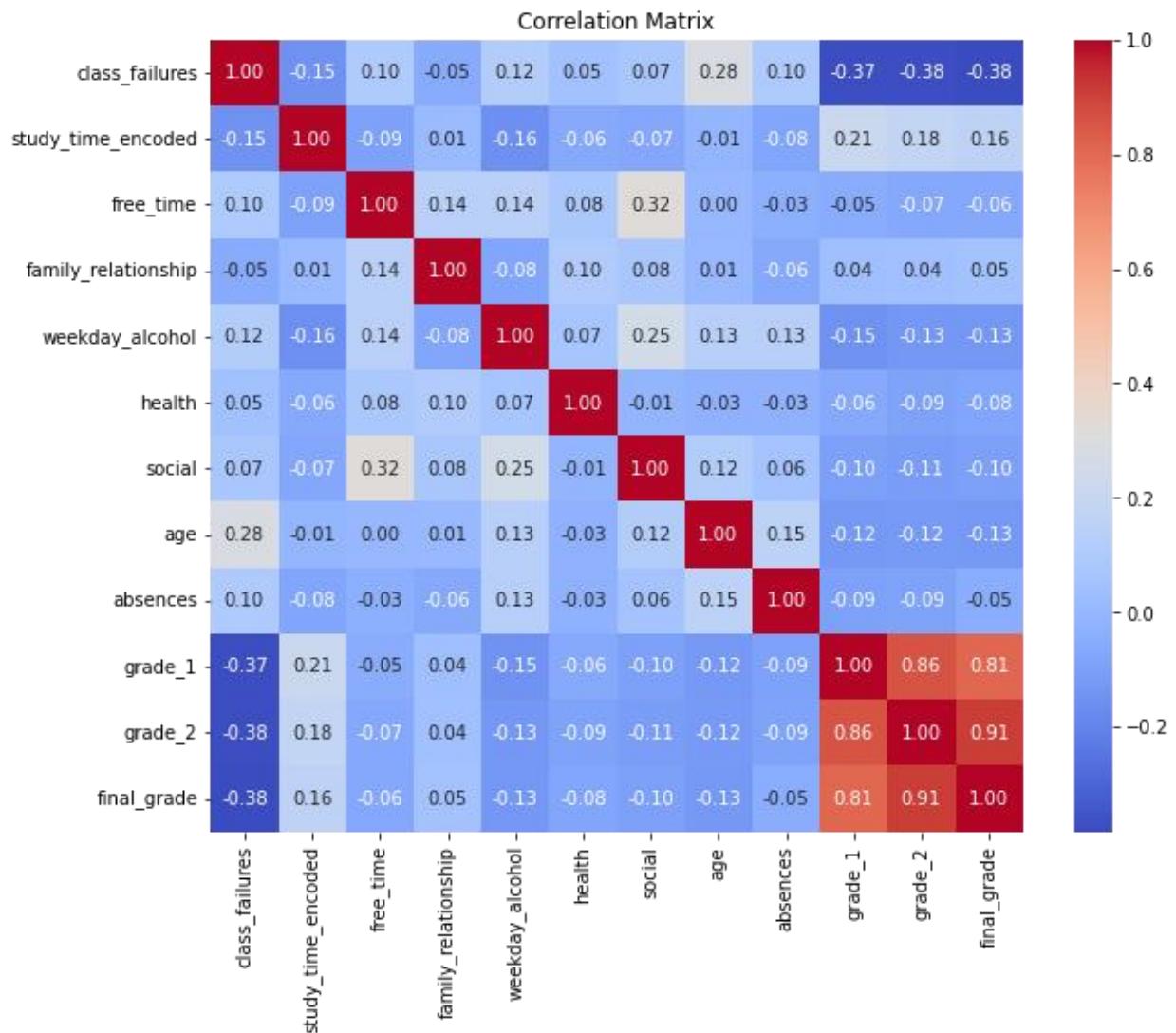
Nếu cân nhắc loại bỏ 1 trong 2. Nhưng nhóm chọn bỏ weekend\_alcohol vì có sự tương quan hơn với social - học sinh có mức xã giao cao.

=> Loại weekend\_alcohol

- grade\_1 - grade\_2: cần giữ lại cho bước Phân loại

## **Bước 2:** Loại thuộc tính "weekend alcohol"

```
In [46]: # Loại bỏ thuộc tính weekend alcohol  
df = df.drop("weekend_alcohol")
```

**Bước 3:** Kiểm tra lại ma trận Correlation

## CHƯƠNG 5: THUẬT TOÁN KHAI THÁC DỮ LIỆU

### 5.1 Giới thiệu

#### 5.1.1 Linear Regression

Linear Regression (hồi quy tuyến tính) là một kỹ thuật phân tích dữ liệu dự đoán những gì muốn dự đoán được gọi là biến phụ thuộc (y) và biến đang sử dụng để dự đoán giá trị của biến kia được gọi là biến độc lập (x). Thuật toán được dùng để ước tính các hệ số của phương trình tuyến tính, Linear Regression còn giúp tìm kiếm đường tối ưu giảm thiểu tổng chênh lệch bình phương giữa các giá trị dự đoán và thực tế. Được áp dụng trong nhiều lĩnh vực khác nhau. Trong bài toán dự đoán, Linear Regression có hai dạng chính là đơn biến và đa biến[3],[4].

**Cách thức hoạt động của Linear Regression:** là một mô hình tuyến tính, ví dụ: mô hình trong đó giả định mối quan hệ tuyến tính giữa các biến đầu vào (x) và biến đầu ra (y). Sử dụng thuật toán để tìm ra hệ số góc và điểm giao trực tung sao cho hàm dự đoán tuyến tính đạt sai số nhỏ nhất. Quá trình bao gồm các bước chính sau:

- **Phương trình hồi quy tuyến tính:**

$$y = mx + b$$

Trong đó:

- + y là biến phụ thuộc (response)
- + x là biến độc lập (predictor)
- + m là độ dốc của đường thẳng
- + b là giao điểm y của đường thẳng
- **Công thức tính m và b:**

$$m = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}$$

$$b = \bar{y} - m \cdot \bar{x}$$

- **Công thức tính giá trị trung bình:**

+ Giá trị trung bình của x:

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$$

+ Giá trị trung bình của y:

$$\bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$$

- **Công thức tính Hiệu phương sai và phương sai:**

+ Hiệu phương sai:

$$cov(x, y) = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})$$

+ Phương sai:

$$var(x) = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2$$

- **Công thức tính độ dốc (m)**

+ Tổng dư bình phương:

$$\boxed{\frac{\partial S}{\partial m} = -2 \sum_{i=1}^n x_i (y_i - (mx_i + b)) = 0}$$

- + Đạo hàm tùng phần với m và b khi đặt về 0:

$$\frac{\partial S}{\partial m} = -2 \sum_{i=1}^n x_i(y_i - (mx_i + b)) = 0$$

$$\frac{\partial S}{\partial b} = -2 \sum_{i=1}^n (y_i - (mx_i + b)) = 0$$

- **Cách giải quyết độ dốc m:**

- + Phương trình ban đầu m:

$$\sum_{i=1}^n x_i y_i = m \sum_{i=1}^n x_i^2 + b \sum_{i=1}^n x_i$$

- + Thay phương trình thứ hai vào phương trình thứ nhất:

$$\sum_{i=1}^n y_i = m \sum_{i=1}^n x_i + nb$$

- + Sau một vài thay thế ta được phương trình sau:

$$m = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}$$

- + Tương đương công thức tổng:

$$m = \frac{cov(x,y)}{var(x)}$$

→ *Mục đích của việc sử dụng Linear Regression:* là tìm các hệ số hồi quy sao cho tổng sai số bình phương SSE giữa các dự đoán và giá trị quan sát là nhỏ nhất

### Ưu điểm:

- + *Xử lý lớn lượng dữ liệu lớn:* Spark được thiết kế để xử lý lượng dữ liệu lớn trên nhiều nút máy tính, cho phép thực hiện linear regression trên tập dữ liệu lớn mà không gặp vấn đề hiệu suất.
- + *Phân tán tính toán:* Spark sử dụng cấu trúc dữ liệu phân tán (Resilient Distributed Dataset - RDD) để phân tán tính toán cho các máy chủ khác nhau, tận dụng sức mạnh tính toán song song, giúp tăng hiệu suất tính toán của linear regression.
- + *Tích hợp với các công cụ phân tích dữ liệu khác:* Spark có thể tích hợp dễ dàng với các thư viện và công cụ phân tích dữ liệu khác như Spark SQL, MLlib, và Spark DataFrames để thực hiện các công việc phức tạp hơn liên quan đến dữ liệu

### Nhược điểm:

- + *Khó khăn trong việc xử lý dữ liệu thô và tiền xử lý:* Đối với dữ liệu lớn, việc tiền xử lý và chuẩn bị dữ liệu có thể trở nên phức tạp và tốn thời gian. Linear regression trong Spark yêu cầu dữ liệu được định dạng và tối ưu hóa trước khi thực hiện tính toán.
- + *Yêu cầu kiến thức về Spark và lập trình song song:* Việc triển khai và tối ưu hóa linear regression trong Spark đòi hỏi kiến thức về Spark cũng như lập trình song song, điều này có thể là một thách thức đối với người mới bắt đầu.
- + *Có thể phức tạp về hiệu suất:* Mặc dù Spark được thiết kế để xử lý dữ liệu lớn, việc tối ưu hóa hiệu suất vẫn có thể là một thách thức đối với các ứng dụng cụ thể, đặc biệt là khi đối mặt với các vấn đề như kích thước dữ liệu rất lớn hoặc phân tán không đồng đều

5.1.2 Linear Regression multi-features: Hồi quy tuyến tính cho nhiều biến

Mô hình hồi quy tuyến tính cho nhiều biến có dạng:

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n$$

Trong đó:

- $\hat{y}$  là giá trị dự đoán.
- $\theta_0, \theta_1, \theta_2, \dots, \theta_n$  là các tham số (weights) của mô hình.
- $x_1, x_2, \dots, x_n$  là các feature (đặc trưng) đầu vào.

Hàm mất mát (Loss function)

Chúng ta thường sử dụng hàm mất mát bình phương trung bình (Mean Squared Error - MSE) để đo lường sự khác biệt giữa giá trị dự đoán và giá trị thực:

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^m (\hat{y}^{(i)} - y^{(i)})^2$$

Trong đó:

- m là số lượng mẫu.
- $y^{(i)}$  là giá trị thực của mẫu thứ i.
- $\hat{y}^{(i)}$  là giá trị dự đoán của mẫu thứ i.

Gradient Descent

Gradient descent là một thuật toán tối ưu hóa được sử dụng để giảm thiểu hàm mất mát bằng cách cập nhật các tham số của mô hình. Công thức cập nhật cho mỗi tham số theta là:

$$\theta_j := \theta_j - \alpha \frac{\partial J(\theta)}{\partial \theta_j}$$

Trong đó:

- $\alpha$  là tốc độ học (learning rate).
- $\frac{\partial J(\theta)}{\partial \theta_j}$  là đạo hàm riêng của hàm mất mát theo tham số theta(i).

Công thức đạo hàm riêng của hàm mất mát theo tham số  $\theta_j$  là:

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m} \sum_{i=1}^m (\hat{y}^{(i)} - y^{(i)}) x_j^{(i)}$$

Quy trình thực hiện Gradient Descent cho Hồi quy Tuyến tính

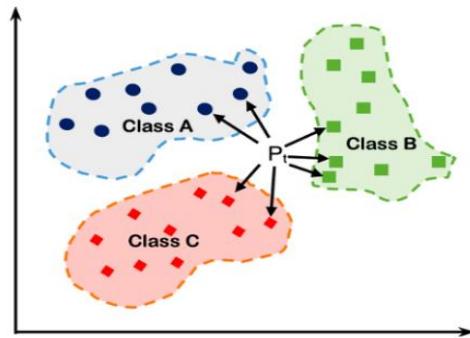
1. **Khởi tạo các tham số:** Đặt các giá trị ban đầu cho  $\theta_0, \theta_1, \dots, \theta_n$  (thường là 0 hoặc các giá trị ngẫu nhiên nhỏ).
2. **Lặp lại cho đến khi hội tụ:**
  - Tính giá trị dự đoán  $\hat{y}^{(i)}$  cho mỗi mẫu.
  - Tính các đạo hàm riêng  $\frac{\partial J(\theta)}{\partial \theta_i}$  cho mỗi tham số  $\theta_j$
  - Cập nhật các tham số  $\theta_j$  theo công thức gradient descent.
3. **Dừng lại khi hàm mất mát không thay đổi đáng kể hoặc sau một số lượng bước lặp nhất định.**

### 5.1.3 KNN (K - nearest neighbor)

K-nearest neighbor là một trong những thuật toán supervised-learning đơn giản nhất trong Machine Learning. Khi training, thuật toán này không học một điều gì từ dữ liệu training (hay còn gọi là lazy learning), mọi tính toán được thực hiện khi nó cần dự đoán kết quả của dữ liệu mới. K-nearest neighbor có thể áp dụng được vào cả hai loại của bài toán Supervised learning là **Classification** và **Regression**. KNN còn được gọi là một thuật toán Instance-based hay Memory-based learning.

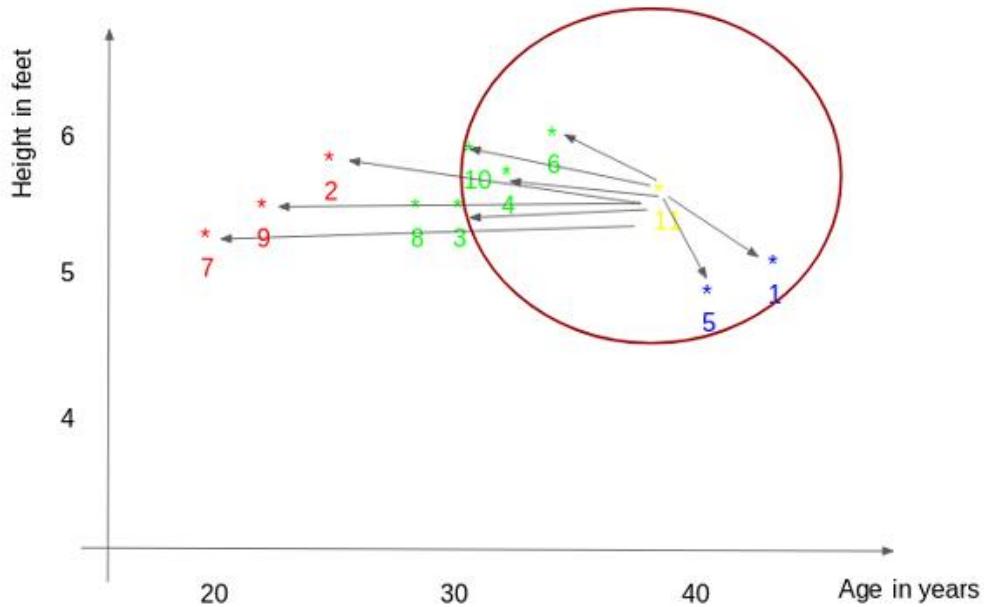
Trong bài toán Classification, label của một điểm dữ liệu mới được suy ra trực tiếp từ K điểm dữ liệu gần nhất trong training set. Label của một test data có thể được quyết định bằng major voting (bầu chọn theo số phiếu) giữa các điểm gần nhất, hoặc nó có thể được suy ra bằng cách đánh trọng số khác nhau cho mỗi trong các điểm gần nhất đó rồi suy ra label.

K Nearest Neighbors



Hình 2: KNN - Classification

Trong bài toán Regression, đầu ra của một điểm dữ liệu sẽ bằng chính đầu ra của điểm dữ liệu đã biết gần nhất (trong trường hợp K=1), hoặc là trung bình có trọng số của đầu ra của những điểm gần nhất, hoặc bằng một mối quan hệ dựa trên khoảng cách tới các điểm gần nhất đó



Hình 3: KNN - Regression

Cách hoạt động của KNN:

Thuật toán K-Nearest Neighbors (KNN) hoạt động dựa trên việc dự đoán nhãn hoặc giá trị của điểm dữ liệu mới bằng cách xem xét K điểm dữ liệu gần nhất trong tập dữ liệu huấn luyện. Quá trình này bao gồm các bước chính như sau [5]:

- Xác định Khoảng Cách: tính toán khoảng cách giữa điểm dữ liệu mới và mỗi điểm dữ liệu trong tập huấn luyện(Euclidean, Manhattan) [5].

Euclidean

$$\sqrt{\sum_{i=1}^k (x_i - y_i)^2}$$

Manhattan

$$\sum_{i=1}^k |x_i - y_i|$$

- Chọn K Lân Cận Gần Nhất: Sắp xếp các khoảng cách theo thứ tự tăng dần và chọn K điểm dữ liệu có khoảng cách ngắn nhất.
- Phân Loại hoặc Dự Đoán: Đối với nhiệm vụ phân loại, gán nhãn cho điểm dữ liệu mới dựa trên lớp đa số trong số K điểm gần nhất. Đối với nhiệm vụ hồi quy, tính giá trị trung bình hoặc trung bình có trọng số của giá trị của K điểm gần nhất và gán nó làm giá trị dự đoán.
- Áp Dụng Feature Scaling (tùy chọn): Đôi khi, quá trình chuẩn hóa các đặc trưng có thể được áp dụng để đảm bảo rằng tất cả các đặc trưng đều có ảnh hưởng tương đồng đến kết quả[5].

**Ưu điểm:**

- Dễ thực hiện vì độ phức tạp của thuật toán không cao.
- Thích ứng dễ dàng – Theo hoạt động của thuật toán KNN, nó lưu trữ tất cả dữ liệu trong bộ nhớ lưu trữ và do đó, bất cứ khi nào một ví dụ hoặc điểm dữ liệu mới

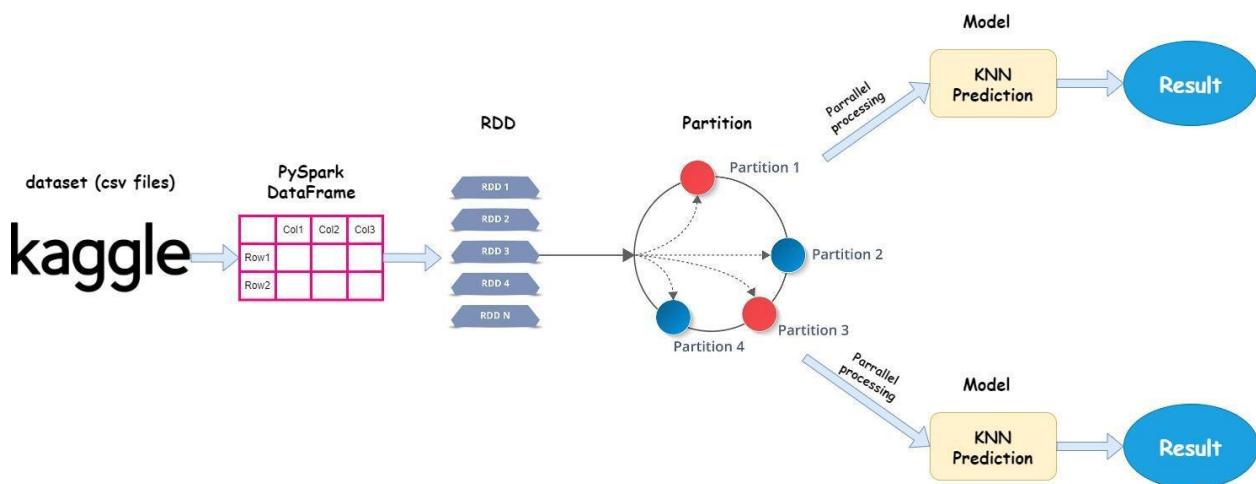
được thêm vào thì thuật toán sẽ tự điều chỉnh theo ví dụ mới đó và cũng có đóng góp cho các dự đoán trong tương lai .

- Ít tham số – Tham số duy nhất được yêu cầu trong quá trình đào tạo thuật toán KNN là giá trị của k và lựa chọn thước đo khoảng cách mà chúng tôi muốn chọn từ thước đo đánh giá của mình.

### Nhược điểm:

- Không Thích Hợp Cho Quy Mô Lớn – Như chúng ta đã nghe nói, thuật toán KNN cũng được coi là một thuật toán "Lazy". Ý nghĩa chính của thuật ngữ này là nó đòi hỏi rất nhiều công suất tính toán cũng như lưu trữ dữ liệu. Điều này khiến thuật toán trở nên tốn thời gian và tài nguyên.
- Lựa chọn k và thước đo khoảng cách: một thách thức khác khi sử dụng KNN là nó phụ thuộc vào việc lựa chọn k và thước đo khoảng cách. Cả hai tham số này đều có thể có tác động đáng kể đến hiệu suất và độ chính xác của KNN.

## 5.2 Mô hình song song hóa giải thuật



Hình 4: Mô hình song song hóa giải thuật KNN

## 5.3 Triển khai thuật toán

### 5.3.1 K-Nearest Neighbors Regression

Mô hình KNN Regression được áp dụng để dự đoán điểm thi cuối năm (final\_grade) của học sinh, dựa theo điều kiện nhân khẩu học và điểm thi của các kì thi trước đó. Qua đó, hỗ trợ đánh giá, hỗ trợ, định hướng và giúp đỡ cải thiện thành tích học tập của học sinh.

Bài toán dự đoán điểm thi cuối năm của học sinh:

- Input: Điểm thi 2 kỳ trước đó (grade\_1, grade\_2), các thông tin về nhân khẩu học và học tập của học sinh
- Output: Điểm thi học kì 3 hay Điểm thi cuối năm của học sinh (final\_grade)

#### Bước 1: Chuẩn bị dữ liệu

Lựa chọn các biến độc lập

```
In [49]: # Chọn feature, tất cả của df ngoài "id" và target  
selected_columns = [col for col in df.columns if col not in ["id", "final_grade"]]
```

Tạo dataframe mới “df\_regression” bao gồm các cột id: id của mỗi kết quả học tập, final\_grade (biến mục tiêu), và cột feature: danh sách mảng các giá trị double của các biến độc lập

IS405.O11.HTCL – Dữ liệu lớn

In ra cấu trúc (schema) của DataFrame df\_regression

```
In [51]: df_regression.printSchema()
```

```
root
 |-- id: integer (nullable = false)
 |-- final_grade: integer (nullable = true)
 |-- feature: array (nullable = false)
 |   |-- element: integer (containsNull = true)
```

Chia dữ liệu thành 2 tập huấn luyện với kiểm thử với tỉ lệ train:test -7:3

```
In [52]: train, test = df_regression.randomSplit([0.7, 0.3])
```

Tập train sẽ gồm 2 cột: feature, final grade (biến mục tiêu và biến độc lập)

Tập kiểm thử gồm 2 cột: test\_feature, test\_id. Cột “test\_id” giúp xác định mã id kết quả học tập sau khi thực hiện dự đoán từ mô hình

```
In [53]: # Train data
train = train.select(col("feature"), col("final_grade"))
train.take(1)

# Test data
test = test.select(col("feature").alias("test_feature"), col("id").alias("test_id")).dropDuplicates()
test.take(1)
[spark://192.168.1.24:4040] INFO org.apache.spark.scheduler.TaskSetManager - Created task 0.0 in stage 137.0 (TID 137)
[Executor task launch worker for task 0.0 in stage 137.0 (TID 137)] INFO org.apache.spark.sql.catalyst.expressions.codegen.CodeGenerator - Code generated in 12.785702 ms
[Executor task launch worker for task 0.0 in stage 137.0 (TID 137)] INFO org.apache.spark.executor.Executor - Finished task 0.0 in stage 137.0 (TID 137). 5215 bytes result sent to driver
[task-result-getter-1] INFO org.apache.spark.scheduler.TaskSetManager - Finished task 0.0 in stage 137.0 (TID 137) in 34 ms on 192.168.1.24 (executor driver) (1/1)
[task-result-getter-1] INFO org.apache.spark.scheduler.TaskSchedulerImpl - Removed TaskSet 137.0, whose tasks have all completed, from pool
[dag-scheduler-event-loop] INFO org.apache.spark.scheduler.DAGScheduler - ResultStage 137 (take at /tmp/ipykernel_32880/2715825711.py:7) finished in 0.039 s
[dag-scheduler-event-loop] INFO org.apache.spark.scheduler.DAGScheduler - Job 91 is finished. Cancelling potential speculative or zombie tasks for this job
[dag-scheduler-event-loop] INFO org.apache.spark.scheduler.TaskSchedulerImpl - Killing all running tasks in stage 137: Stage finished
[Thread-3] INFO org.apache.spark.scheduler.DAGScheduler - Job 91 finished: take at /tmp/ipykernel_32880/2715825711.py:7, took 0.040338 s

Out[53]: [Row(test_feature=[17, 0, 0, 1, 0, 0, 1, 1, 0, 5, 3, 3, 1, 3, 4, 5, 5, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1], test_id=2)]
```

### **Bước 2:** Xây dựng mô hình

Hàm này tính độ tương đồng cosine giữa hai vectơ x và y. Nó sử dụng một vòng lặp để duyệt qua từng phần tử tương ứng của hai vectơ và tính tích vô hướng. Sau đó, nó chia tích vô hướng cho tích độ dài của hai vectơ

```
In [54]: def cos(x,y):
    s = 0
    for i,j in zip(x,y):
        s = s + i*j
    return s/(norm(x)*norm(y))
```

Code triển khai class :

```
def cos(x,y):  
    s = 0  
    for i,j in zip(x,y):  
        s = s + i*j  
    return s/(norm(x)*norm(y))  
  
class KNearestNeighborsRegression:
```

```
#khởi tạo và nhận đầu vào
def __init__(self, train, test, k):
    self.train = train
    self.test = test
    self.k = k

def get_predict(self):
    k = self.k
    tr = self.train
    td = self.test
    join_df = tr.crossJoin(td)
    cos_map_df = join_df\
        .rdd.map(lambda x:
(float(cos(x.feature,x.test_feature)), x.final_grade, x.test_id))\
        .toDF(["cos", "final_grade", "test_id"])
    windowDept =
Window.partitionBy("test_id").orderBy(col("cos").desc(),
col("final_grade").desc())

    top_k_nearest_items_df = cos_map_df\
        .withColumn("row", row_number().over(windowDept)).filter(col("row") <= k)

    final_grade_predict_df = top_k_nearest_items_df.rdd.map(lambda x:
(x.test_id, x.cos*x.final_grade))\
        .reduceByKey(add)\

        .map(lambda x: (x[0],
(x[1]/k)))\
        .toDF(["id",
"final_grade_predict"])\
        .orderBy(col("final_grade_predict").desc())\

    return final_grade_predict_df

def evaluate(self):
    k = self.k
```

```
w = self.train.orderBy(f.rand())
test_size = int(w.count()/4)
test_df = w.limit(test_size)
train_df = w.subtract(test_df)
test_df = test_df.withColumnRenamed("feature", "test_feature") \
    .withColumn("test_id",
monotonically_increasing_id())

actual_final_grade = test_df.select(col("test_id"),
col("final_grade").alias("actual_final_grade"))

join_df = train_df.crossJoin(test_df)

cos_map_df = join_df \
    .rdd.map(lambda x:
(float(cos(x.feature,x.test_feature)), x.final_grade, x.test_id)) \
    .toDF(["cosine_score", "final_grade", "test_id"])

windowDept =
Window.partitionBy("test_id").orderBy(col("cosine_score").desc(),
col("final_grade").desc())

top_k_nearest_items_df = cos_map_df \
    .withColumn("row", row_number().over(windowDept)).filter(col("row") <= k)

final_grade_predict_df = top_k_nearest_items_df.rdd.map(lambda x:
(x.test_id, x.cosine_score*x.final_grade)) \
    .reduceByKey(add) \
    .map(lambda x: (x[0], x[1]/k)) \
    .toDF(["test_id", "final_grade_predict"])

join_df = final_grade_predict_df.join(actual_final_grade,
["test_id"])

# Calculate RMSE, MSE, and MAE
result_rdd = join_df.rdd.map(lambda x: ((x.final_grade_predict -
x.actual_final_grade) ** 2))
rmse = (result_rdd.reduce(add) / result_rdd.count()) ** 0.5
```

```
mse = (result_rdd.reduce(add) / result_rdd.count())

result_rdd1 = join_df.select(abs(col("final_grade_predict") - col("actual_final_grade")).alias("mae"))

mae = result_rdd1.rdd.map(lambda x: x.mae).reduce(add) / result_rdd1.count()

return {"RMSE": rmse, "MAE": mae, "MSE": mse}

# Function to find the best k
def find_best_k(train, test, k_range):
    best_k = None
    best_metrics = {"RMSE": float("inf"), "MAE": float("inf"), "MSE": float("inf")}

    for k in k_range:
        knn = KNearestNeighborsRegression(train, test, k)
        metrics = knn.evaluate()
        print(f"Evaluating k={k}: RMSE={metrics['RMSE']}, MAE={metrics['MAE']}, MSE={metrics['MSE']}")

        if metrics["RMSE"] < best_metrics["RMSE"]:
            best_metrics["RMSE"] = metrics["RMSE"]
            best_k_rmse = k
        if metrics["MAE"] < best_metrics["MAE"]:
            best_metrics["MAE"] = metrics["MAE"]
            best_k_mae = k
        if metrics["MSE"] < best_metrics["MSE"]:
            best_metrics["MSE"] = metrics["MSE"]
            best_k_mse = k

    return {
        "Best k for RMSE": best_k_rmse,
        "Best k for MAE": best_k_mae,
        "Best k for MSE": best_k_mse,
        "Best metrics": best_metrics}
```

Trong đó:

- **def \_\_init\_\_():** Hàm khởi tạo thiết lập mô hình KNN với đầu vào là tập huấn luyện (train), tập kiểm thử (test) và giá trị k cho k hàng xóm gần nhất
- **get\_predict()** : Tính độ tương đồng cosine giữa từng cặp mục trong tập huấn luyện và kiểm thử, chọn k láng giềng gần nhất, và dự đoán giá trị trung bình của từng mục trong tập kiểm thử dựa trên trung bình có trọng số của giá trị trung bình của k hàng xóm gần nhất.
- **evaluate()**: Đánh giá hiệu suất của mô hình .Chọn một tứ phân vị ngẫu nhiên của dữ liệu huấn luyện làm tập kiểm thử, áp dụng mô hình , và tính toán Sai số trung bình bình phương căn (RMSE), sai số trung bình tuyệt đối (MAE), và sai số trung bình bình phương (MSE) giữa giá trị dự đoán và giá trị thực tế.
- **find\_best\_k()**: Tìm giá trị k tốt nhất cho mô hình KNN dựa trên các tiêu chí đánh giá RMSE, MAE, và MSE. Lặp qua các giá trị k trong k\_range, đánh giá mô hình với từng k, và chọn k cho từng tiêu chí sao cho giá trị đánh giá là nhỏ nhất. Cuối cùng, hàm trả về các giá trị k tốt nhất và các giá trị đánh giá tương ứng.

Code triển khai thuật toán sử dụng các phép toán DataFrame của Spark để xử lý và phân tích dữ liệu, và sử dụng RDD (Resilient Distributed Dataset) cho tính toán song song phân tán

**Bước 3:** Chạy dự đoán với các giá trị k lần lượt là 3, 5, 7, 9, 11

```
In [56]:  
k_range = [3, 5, 7, 9, 11]  
  
# Find the best k  
best_k = find_best_k(train, test, k_range)  
print("Best k values:", best_k)  
  
Evaluating k=11: RMSE=2.277548389944328, MAE=1.4914522198243751, MSE=5.187226668538  
Best k values: {'Best k for RMSE': 7, 'Best k for MAE': 7, 'Best k for MSE': 7, 'Best metrics': {'RMSE': 1.606468  
0738347001, 'MAE': 1.1836253135831856, 'MSE': 2.580739672250172}}
```

Xác định k có số điểm nhỏ nhất là 7. Tuy nhiên vẫn còn sai số khác nhiều, mặc dù vẫn còn chấp nhận được.

Với k = 7, thì:

- RMSE: 1.6
- MAE: 1.18
- MSE: 2.58

### 5.3.2 K-Nearest Neighbors Classification

Đối với bài toán phân lớp ta sẽ dự đoán năng lực học tập trong năm của học sinh:

- Input: Các thông tin về nhân khẩu học và học tập của học sinh
- Output: Năng lực học tập (tốt, is\_good)
- Điều kiện phân loại:
  - Nhóm tham khảo theo cách phân loại học sinh tại Bồ Đào Nha [1]
  - Điểm thi cả 3 kì đều từ khá trở lên (14), hoặc điểm thi qua từng kỳ không được giảm và điểm cuối kỳ phải từ khá trở lên.
  - Điều kiện thứ 2 để ghi nhận những học sinh có sự duy trì và tiến bộ trong học tập suốt năm học để đạt được kết quả cuối năm tốt

$(grade\_1 + grade\_2 + final\_grade) / 3 \geq 14$

hoặc

$final\_grade \geq 14$  và  $final\_grade \geq grade\_2$  và  $grade\_2 \geq grade\_1$

**Bước 1:** Chuẩn bị và kiểm tra dữ liệu

Tạo thuộc tính quyết định “is\_good” từ 3 thuộc tính "grade\_1", "grade\_2" và "final\_grade"

```
In [58]: # Định nghĩa luật phân loại
condition1 = (col("grade_1") + col("grade_2") + col("final_grade")) / 3 >= 14
condition2 = (col("final_grade") >= col("grade_2")) & (col("grade_2") >= col("grade_1")) & (col("final_grade") >= 14)
df = df.withColumn("is_good", when(condition1 | condition2, 1).otherwise(0))
df.select("grade_1", "grade_2", "final_grade", "is_good").show()
```

grade_1	grade_2	final_grade	is_good
5	6	6	0
5	5	6	0
7	8	10	0
15	14	15	1
6	10	10	0
15	15	15	1
12	12	11	0
6	5	6	0
16	18	19	1
14	15	15	1
10	8	9	0
10	12	12	0
14	14	14	1
10	10	11	0
14	16	16	1
14	14	14	1

Nhận thấy nếu phân loại học sinh theo điều kiện này thì sẽ có khoảng 27% trong tổng tập dữ liệu được xếp nhóm “is\_good” = 1

```
In [59]: # lọc lại các dòng có is_good = 1
df_is_good = df.filter(df.is_good == 1)

# Đếm tổng số dòng và các dòng có is_good = 1
total_count = df.count()
is_good_count = df_is_good.count()

# Tính tỉ lệ (ratio)
ratio = is_good_count / total_count

print(f"Ratio of rows where is_good = 1: {ratio} ")

# Các dòng có is_good = 1
df_is_good.select("grade_1", "grade_2", "final_grade", "is_good").show(30)
```

grade_1	grade_2	final_grade	is_good
15	14	15	1
15	15	15	1
16	18	19	1
14	15	15	1
14	14	14	1
14	16	16	1
14	14	14	1
13	14	14	1
13	14	15	1
12	15	15	1
15	15	16	1
15	16	15	1
17	16	17	1
17	16	16	1
12	14	15	1

Loại bỏ id gốc của df, và các biến không cần nữa vì đã có target. Sau đó tạo thành dataframe cho việc phân loại.

```
In [60]: df_classifier = df.drop("grade_1", "grade_2", "final_grade", "id")
In [61]: df_classifier.show()
```

age	class_failures	school_support	family_support	extra_paid_classes	activities	higher_ed	internet_access	romantic_relationship	family_relationship	free_time	social	weekday_alcohol	health	absences	subject_code	mother_education_is_5th_to_9th_grade	mother_education_is_none	mother_education_is_secondary_education	mother_education_is_primary_education_(4th_grade)	mother_education_is_higher_education	father_education_is_5th_to_9th_grade	father_education_is_none	father_education_is_secondary_education	father_education_is_primary_education_(4th_grade)	father_education_on_is_higher_education	reputation_school_choice	course_school_choice	other_school_choice	home_school_choice	father_guardian	mother_guardian	other_guardian	study_time_encoded	sex_is_M	address_type_is_Urban	is_good
18	0	0	0	3	1	4	0	1	3	0	6	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

Chia dữ liệu thành 2 tập huấn luyện với kiểm thử với tỉ lệ train:test -7:3

```
In [62]: # Chia tập dữ liệu với tỉ lệ train:test là 7:3
train_data, test_data = df_classifier.randomSplit([0.7, 0.3], seed=42)
```

Lựa chọn các thuộc tính độc lập

```
In [63]: decision_attribute = 'is_good'
feature_columns = [column for column in df_classifier.columns if column != decision_attribute]
```

Sử dụng hàm broadcast để broadcast dữ liệu huấn luyện (train\_data.rdd.collect()) từ node chủ đến tất cả các node làm việc. Dữ liệu huấn luyện được thu thập thành một danh sách (list) để broadcast.

```
In [65]: # Broadcast train_data to all worker nodes
broadcast_train_data = sc.broadcast(train_data.rdd.collect())
# Convert the test_data DataFrame to an RDD for parallel processing
test_data_rdd = test_data.rdd
```

Chuyển đổi DataFrame test\_data thành một Resilient Distributed Dataset (RDD) để có thể thực hiện các phép biến đổi và hành động xử lý song song

```
In [64]: # Hàm tính khoảng cách
def euclidean_distance(row1, row2):
    distance = 0
    for column in feature_columns:
        distance += (row1[column] - row2[column]) ** 2
    return distance**0.5
```

## Bước 2: Xây dựng mô hình

```
In [66]: # Function to calculate Euclidean distance between two rows
def euclidean_distance(row1, row2):
    distance = 0
    for column in feature_columns:
        distance += (row1[column] - row2[column]) ** 2
    return distance**0.5

k = 10

# Function to find k neighbors for each test row
def find_neighbors(iterator, train_data_broadcast):
    distances = []
    for test_row in iterator:
        for train_row in train_data_broadcast.value:
            euc_dist = euclidean_distance(test_row, train_row)
            distances.append((euc_dist, train_row['is_good']))
    distances.sort(key=lambda x: x[0])
    yield distances[:k]

# Use mapPartitions transformation to find neighbors for each test row in parallel
test_neighbors_rdd = test_data_rdd.mapPartitions(lambda iterator: find_neighbors(iterator, broadcast_train_data))
```

### Giai đoạn 1: Tính khoảng cách Euclidean và Tìm k điểm Láng giềng Gần Nhất

#### 1. Tính khoảng cách Euclidean:

Duyệt qua mỗi hàng trong tập kiểm thử (test\_data\_rdd).

Với mỗi hàng kiểm thử, tính khoảng cách Euclidean đến mỗi hàng trong tập huấn luyện (train\_data\_broadcast.value).

#### 2. Lựa chọn k láng giềng gần nhất:

Đối với mỗi hàng kiểm thử, tạo một danh sách các khoảng cách Euclidean đến các hàng trong tập huấn luyện cùng với nhãn "is\_good" tương ứng.

Sắp xếp danh sách này theo thứ tự tăng dần của khoảng cách Euclidean.

Giữ lại nhãn "is\_good" của mỗi hàng trong tập huấn luyện tương ứng.

Lựa chọn k giá trị nhỏ nhất từ danh sách đã sắp xếp.

#### 3. Lưu trữ kết quả cho mỗi hàng kiểm thử:

Lưu trữ nhãn "is\_good" của k láng giềng gần nhất.

Tạo tuples gồm nhãn "is\_good" và khoảng cách Euclidean của k láng giềng gần nhất cho mỗi hàng kiểm thử.

#### 4. Kiểm thử với các giá trị k khác nhau:

Thực hiện các bước 1-3 với các giá trị k khác nhau để kiểm thử sự ảnh hưởng của giá trị k đối với dự đoán của mô hình (tinh chỉnh siêu tham số - hyperparameter tuning).

### Bước 3: Dự đoán

```
In [67]: # Use mapPartitions transformation to find neighbors for each test row in parallel
test_neighbors_rdd = test_data_rdd.mapPartitions(lambda iter: find_neighbors(iter, broadcast_train_data))

# Function to predict classes based on neighbors
def predict_class(neighbors):
    classes = [neighbor[1] for neighbor in neighbors]
    counter = Counter(classes)
    most_common_class = counter.most_common(1)[0][0]
    return most_common_class

# Use map transformation to predict classes for each test row in parallel
test_predictions_rdd = test_neighbors_rdd.map(predict_class)

test_predictions = test_predictions_rdd.collect()
```

### Giai đoạn 2: Dự đoán

Sử dụng mapPartitions để áp dụng hàm find\_neighbors và tìm ra k láng giềng gần nhất cho mỗi hàng trong tập kiểm thử (test\_data\_rdd) dựa trên tập huấn luyện (broadcast\_train\_data). Kết quả được lưu trữ trong test\_neighbors\_rdd.

Sau đó, sử dụng hàm predict\_class để dự đoán lớp cho mỗi hàng trong tập kiểm thử dựa trên nhãn của k láng giềng gần nhất.

Kết quả dự đoán được thu thập từ test\_predictions\_rdd bằng cách sử dụng collect() để có danh sách dự đoán cuối cùng cho tất cả các hàng trong tập kiểm thử, được lưu trữ trong biến test\_predictions.

### Bước 4: Đánh giá độ chính xác của mô hình

### Giai đoạn 3: Đánh giá độ chính xác của mô hình

Lấy nhãn của tập test để tiến hành đánh giá độ chính xác

```
In [68]: # Lấy nhãn của tập test để tiến hành đánh giá độ chính xác
test_labels = [row['is_good'] for row in test_data.select('is_good').rdd.collect()]
```

Hàm tính độ chính xác

```
In [69]: # Function to calculate accuracy
def accuracy_score(y_pred, y_test):
    correct_predictions = len([pred for pred, true_label in zip(y_pred, y_test) if pred == true_label])
    total_predictions = len(y_pred)
    accuracy = (correct_predictions / total_predictions) * 100
    return accuracy
```

Mô hình phân loại chính xác 63.81%, kết quả dự đoán của mô hình khá tốt

```
In [70]: # Calculate and print accuracy
accuracy_test = accuracy_score(test_predictions, test_labels)
print('The accuracy score for KNN test is:', accuracy_test)
```

The accuracy score for KNN test is: 63.81578947368421

**Bước 5:** In ra kết quả dự đoán

### Giai đoạn 4: In ra kết quả dự đoán

Tạo DataFrame để lưu trữ kết quả dự đoán từ mô hình

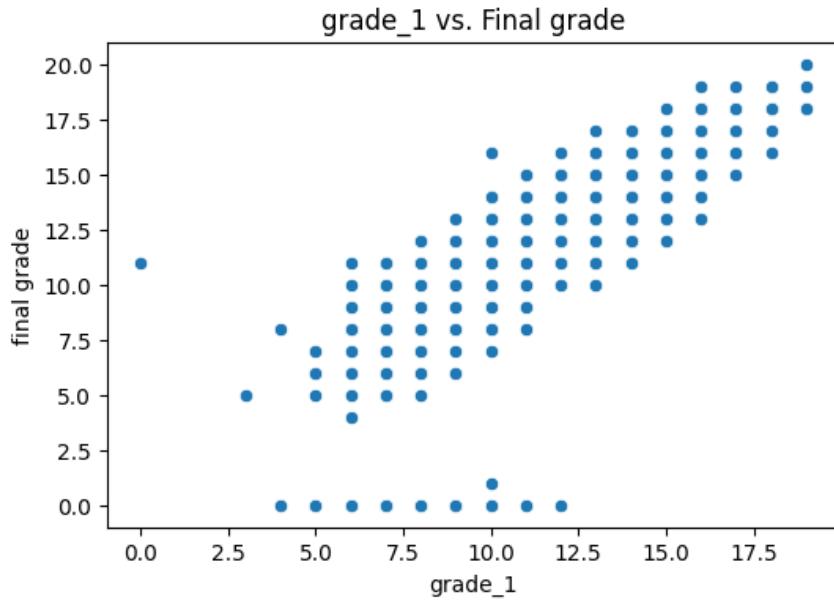
```
In [71]: # Tạo DataFrame từ list của tuples
results_data = [(predicted_label, actual_label) for predicted_label, actual_label in zip(test_predictions, test_labels)]
df_results = spark.createDataFrame(results_data, ["KNN Model Predicted Labels with k=10", "Actual Labels"])

# Hiển thị kết quả
df_results.show()
```

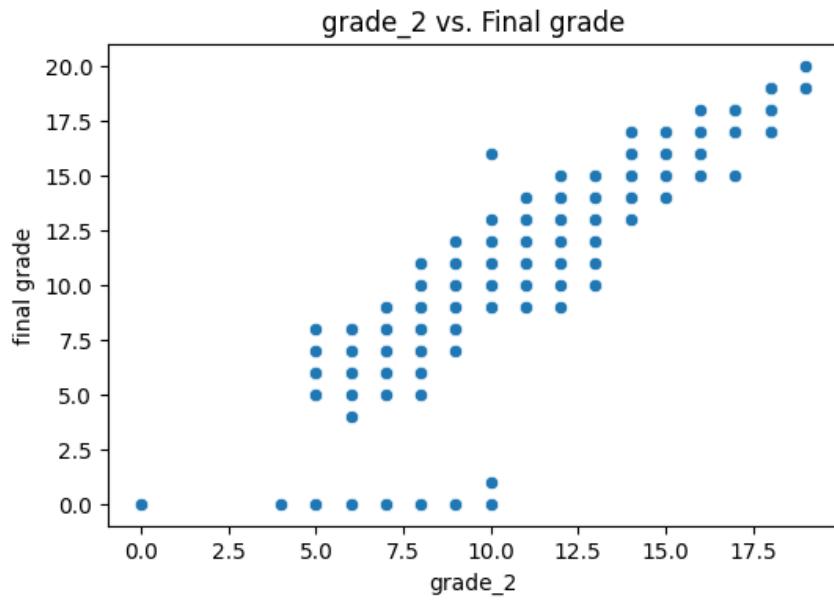
	KNN Model Predicted Labels with k=10	Actual Labels
0	0	0
0	0	1
0	1	1
0	1	0
0	1	1
0	0	0
0	1	1
0	0	0
0	1	1
0	0	0
0	1	1
1	0	0

### 5.3.3 Linear Regression

- **Bước 1:** Chuẩn bị dữ liệu



Biểu đồ thể hiện mối quan hệ giữa grade\_1 và Final grade



Biểu đồ thể hiện mối quan hệ giữa grade\_2 và Final grade

Lựa chọn chia dataset ngẫu nhiên 80%:20%

Dataset cho cả quá trình train và test chỉ bao gồm 3 cột: grade\_1, grade\_2 và final\_grade

#### ▼ 6. Lựa chọn thuộc tính

- nhận thấy Correlation của weekend-alcohol & weekend\_alcohol cao
  - correlation của grade\_1 & grade\_2 với final\_grade cao
- Do target ban đầu lựa chọn là final\_grade, cho nên sẽ bỏ cặp weekend-alcohol & weekend\_alcohol
  - Ta quyết định lấy 2 features là **grade1** & **grade2**

```
[ ] 1 selected_columns = ['grade_1', 'grade_2', 'final_grade']
2 df_selected = df_uni.select(*selected_columns)
3
4 # Splitting the data into training and testing sets (80% train, 20% test)
5 train_data, test_data = df_selected.randomSplit([0.8, 0.2], seed=42)
6 train_data.show(5)
7 test_data.show(5)

[grade_1|grade_2|final_grade]
+---+---+---+
| 3| 5| 5|
| 4| 0| 0|
| 5| 0| 0|
| 5| 5| 5|
| 5| 5| 6|
+---+---+---+
only showing top 5 rows

[grade_1|grade_2|final_grade]
+---+---+---+
| 5| 0| 0|
| 5| 6| 6|
| 5| 9| 7|
| 6| 5| 0|
| 6| 5| 6|
+---+---+---+
only showing top 5 rows
```

**Bước 2:** Chuẩn bị mô hình

Import thêm những hàm xử lý cần thiết

```
from pyspark.sql import SparkSession
from pyspark.sql import Row
from pyspark.sql.functions import lit, col, sqrt, mean, pow, abs
from pyspark.sql.types import StructType, StructField, FloatType, StringType
import math
import numpy as np
```

+ Định nghĩa cho các thông số của mô hình Linear Regression:

```
class LinearRegressionGD:
    def __init__(self, learning_rate=0.01, n_iterations=1000):
        self.learning_rate = learning_rate
        self.n_iterations = n_iterations
        self.coefficients = None
        self.cost_history = []
```

- ❖ learning\_rate: mức độ học
  - ❖ n\_iterations: số lần chạy tối đa cho các hệ số
  - ❖ coefficient: danh sách các hệ số dưới dạng python list
  - ❖ cost\_history: giá trị của cost qua các lần train, dùng để kiểm tra và điều chỉnh n\_iterations và learning\_rate.
- + Định nghĩa cho hàm fit (train) của model:

```

def fit(self, df, target_col):
    # Add intercept column
    df = df.withColumn("intercept", lit(1.0))

    # Assemble the feature matrix X and target vector y
    feature_cols = [col for col in df.columns if col != target_col]
    num_features = len(feature_cols)

    # Initialize coefficients
    self.coefficients = [0.0] * num_features

    for iteration in range(self.n_iterations):
        # Tính predictions Tại lần iteration
        predictions = df.rdd.map(lambda row: self.predict_row(row, feature_cols))
        # Tính ra bộ đạo hàm w1,w2
        gradients = df.rdd.zip(predictions) \
            .map(lambda row_pred: self.compute_gradients(row_pred, feature_cols, target_col)) \
            .reduce(lambda a, b: [x + y for x, y in zip(a, b)])
        #update
        #w1=w1- learning_rate * đạo_hàm(w1)
        #w2=w2- learning_rate * đạo_hàm(w2)
        self.coefficients = [w - self.learning_rate * g for w, g in zip(self.coefficients, gradients)]
        if iteration % 10 == 0:
            temp = df.rdd.zip(predictions) \
                .map(lambda my_row: self.cal_error(row_pred=my_row,target_col=target_col))
            df_temp = temp.toDF(['error'])
            mse = df_temp.select(mean(col("error"), 2)).first()[0]
            print(f"mse at {iteration}: {mse}")

```

Với mỗi bước chạy trong n\_iterations đã định nghĩa trước đó cho model

1. Ta tính giá trị dự đoán hiện tại thành 1 pipeline
2. Tiếp tục với pipeline của kết quả dự đoán. Với mỗi kết quả dự đoán, ta tiếp tục tính giá trị đạo hàm = (giá trị predict - giá trị thực) \* (giá trị hiện tại của dòng tại cột feature)
3. Sau kết quả tính đạo hàm, ta sẽ có được danh sách các hệ số được tính theo công thức:  $W = W - \text{learning\_rate} * (\text{gradient tương ứng})$
4. Với mỗi 10 bước chạy, ta sẽ log giá trị trung bình của error bình phương (Mean Square Error), error được tính bằng độ chênh giữa giá trị thực và giá trị predict
- + Các hàm hỗ trợ tính toán bao gồm:

```

def cal_error(self, row_pred, target_col):
    row, pred = row_pred
    error = pred - row[target_col]
    return Row(error=error)

def predict_row(self, row, feature_cols):
    return sum(row[feature_cols[i]] * self.coefficients[i] for i in range(len(feature_cols))) # 1 con so sum

def compute_absolute_error(self, row_pred, target_col):
    row, pred = row_pred
    error = pred - row[target_col]
    return float(error)

def compute_gradients(self, row_pred, feature_cols, target_col):
    row, pred = row_pred
    error = pred - row[target_col]
    return [error * row[feature_cols[i]] for i in range(len(feature_cols))]

def calculate_cost(self, df, target_col, feature_cols):
    squared_errors = df.rdd.map(lambda row: (self.predict_row(row, feature_cols) - row[target_col]) ** 2)
    mean_squared_error = squared_errors.mean()
    return mean_squared_error

```

- ❖ Hàm cal\_error: trả về dòng có field là error chính là độ chênh lệch giữa kết quả dự đoán & kết quả thực
- ❖ Hàm predict\_row: kết quả dự đoán, sử dụng công thức tuyến tính cho nhiều biến
- ❖ Hàm compute\_absolute\_error: hỗ trợ tính toán giá trị tuyệt đối của error
- ❖ hàm compute\_gradients: trả về danh sách giá trị đạo hàm tại vị trí của dòng và tương ứng với hệ số của feature, cụ thể ở đây là [**đạo hàm từng phần của hệ số grade\_1 tại một iteration, đạo hàm từng phần của hệ số grade\_2 tại một iteration]**]
- ❖ Hàm calcualate\_cost: trả về giá trị Mean Square Error
  
- + Định nghĩa hàm tính toán các thông số về độ lệch, các thông số bao gồm [6]:
- ❖ MSE: Mean Square Error
- ❖ RMSE: Round Mean Square Error
- ❖ MAE: Mean Average Error
- ❖ ME: Mean Error

```
# def predict(self,df):
def __calculate_metrics(self,df):
    df = df.withColumn("residual", col("final_grade") - col("predictions"))
    # Calculate MSE
    mse = df.select(mean(pow(col("residual"), 2))).first()[0]

    # Calculate RMSE
    rmse = math.sqrt(mse)
    # Calculate MAE
    mae = df.select(mean(abs(col("residual")))).first()[0]
    # Calculate ME
    me = df.select(mean(col("residual"))).first()[0]
    # Collect the results
    metrics = {
        "RMSE": rmse,
        "MAE": mae,
        "MSE": mse,
        "ME": me
    }
    return metrics
```

- + Chuẩn bị hàm Predict:

```
def predict(self, df, featureNames = []):
    # Start with the intercept
    prediction_expr = lit(self.coefficients[-1])

    # Iterate through the features and coefficients
    for i, feature in enumerate(featureNames):
        prediction_expr += df[feature] * lit(self.coefficients[i])

    # Add the predictions column to the DataFrame
    df_with_predictions = df.withColumn("predictions", prediction_expr)
    metrics = self.__calculate_metrics(df_with_predictions)
    return [df_with_predictions, metrics]
```

Các thông số của model đã định nghĩa và qua việc train/fit từ trước được sử dụng để dự đoán trên bộ data hoàn toàn mới: 20% data gốc dùng cho test data

### Bước 3: Train model và predict

- + Định nghĩa model với learning\_rate = 0.000008 và n\_iterations = 250 và bắt đầu train.

```
my_df = train_data.select('grade_1','grade_2','final_grade')
my_df.show()
# Initialize the model
model = LinearRegressionGD(learning_rate=0.000008, n_iterations=250)

# Fit the model
my_test = test_data.select('grade_1','grade_2','final_grade')
model.fit(my_df, "final_grade")

|
# my_pred_df = df.select('grade_1','grade_2')

# # Make predictions
predictions,metrics = model.predict(my_df,['grade_1','grade_2'])
predictions.show()
print(metrics['MAE'])
print(metrics['MSE'])
print(metrics['RMSE'])
print(metrics['ME'])
# print(model.cost_history)
```

- + Khi train model cần lưu ý:
  - ❖ Các bản ghi của giá trị cost qua từng đoạn của vòng lặp train được in ra. Nếu nhận thấy cost giảm đều, thì đã train đúng.
    - Nếu nhận thấy cost vừa giảm vừa tăng, có thể learning quá lớn.
    - Nếu nhận thấy cost giảm quá chậm, có thể learning rate quá thấp.
    - Nếu nhận thấy learning rate đã phù hợp, ta có thể tăng số lần vòng lặp **n\_iterations** để đặt được độ lệch mong muốn đối với mô hình
  - ❖ Các giá trị MAE, MSE, RMSE, ME sẽ giúp ta đánh giá mô hình, lần lượt là:
    - MSE: Mean Square Error
    - RMSE: Round Mean Square Error
    - MAE: Mean Average Error
    - ME: Mean Error
- + Kết quả sau khi train:

```
mse at 0: 102.4105115451554
mse at 10: 6.216337790470677
mse at 20: 3.145300830953708
mse at 30: 2.9863050620948544
mse at 40: 2.9288806167276404
mse at 50: 2.883641617012526
mse at 60: 2.8462368362374284
mse at 70: 2.8152349576969153
mse at 80: 2.789520967360437
mse at 90: 2.768175896447758
mse at 100: 2.75044068853561
mse at 110: 2.735688319531624
mse at 120: 2.7234008286475517
mse at 130: 2.713150346759358
mse at 140: 2.704583424958203
mse at 150: 2.697408089163997
mse at 160: 2.6913831465988536
mse at 170: 2.686309352407647
mse at 180: 2.682022112851456
mse at 190: 2.6783854577858364
mse at 200: 2.6752870616308715
mse at 210: 2.6726341304468266
mse at 220: 2.670350004455873
mse at 230: 2.6683713515572416
mse at 240: 2.66664584903217
```

Nhận thấy Mean square error giảm dần, không có giá trị tăng tại bất kỳ điểm nào. Do đó learning đã đạt ngưỡng phù hợp, do tốc độ giảm không quá chậm

Ta bắt đầu tăng n\_iterations lên càng cao để nhận được độ chênh lệch mong muốn

grade_1	grade_2	final_grade	predictions
3	5	5	4.805104817225811
4	0	0	0.33630690356947823
5	0	0	0.44418508883137897
5	5	5	5.020861187749611
5	5	6	5.020861187749611
5	8	7	7.766866847100551
6	0	0	0.5520632740932796
6	0	0	0.5520632740932796
6	5	0	5.128739373011513
6	5	0	5.128739373011513
6	5	0	5.128739373011513
6	5	0	5.128739373011513
6	5	0	5.128739373011513
6	5	5	5.128739373011513
6	5	5	5.128739373011513
6	5	6	5.128739373011513
6	5	6	5.128739373011513
6	6	4	6.044074592795159
6	6	8	6.044074592795159
6	7	0	6.959409812578805

only showing top 20 rows

0.8958242179896676  
2.6652734505285713  
1.632566522543131  
-0.06641231313989339

Kết quả được in có chứa giá trị dự đoán của 20 dòng đầu tiên.

Đồng thời, các hệ số chênh lệch là:

- MSE: ~0.89
- RMSE: ~2.66
- MAE: ~1.63
- ME: ~ - 0.06

Có thẻ đánh giá độ chênh lệch có thẻ chấp nhận được

## CHƯƠNG 6: KẾT QUẢ ĐẠT ĐƯỢC

### 6.1 Phát biểu kết quả

Thành công trong việc triển khai mô hình KNN cho cả hai bài toán: dự đoán điểm thi cuối kỳ của học sinh; và mô hình Linear Regression cho phân loại năng lực học tập trong năm của học sinh

#### + Dự đoán điểm thi cuối kỳ của học sinh:

- Mục tiêu là dự đoán điểm thi cuối kỳ của học sinh. Tuy nhiên, kết quả dự đoán vẫn chưa đạt tới mức tối ưu, có thể cần xem xét và tối ưu hóa thêm để cải thiện độ chính xác của mô hình.
- Linear Regression có kết quả tốt hơn KNN Regression vì
  - Linear Regression: có áp dụng các phương pháp nhằm tăng độ chính xác tốt như correlation cho chọn lọc feature, gradient descend, ... phù hợp hơn với Tập dữ liệu
  - KNN Regression: mô hình không tối ưu cho loại bài toán này và cũng không áp dụng các phương pháp nhằm tăng độ chính xác tốt.

#### + Phân loại năng lực học tập trong năm của học sinh:

- Sử dụng KNN để dự đoán năng lực học tập trong năm của học sinh thông qua các thông tin về nhân khẩu học và học tập của học sinh.
- Mặc dù mô hình đã thực hiện phân loại, nhưng kết quả vẫn có thể được cải thiện thông qua việc điều chỉnh siêu tham số và mở rộng tập dữ liệu.

#### + Thách thức trong triển khai:

- Triển khai thuật toán KNN trong Spark mà không sử dụng thư viện hỗ trợ là một thách thức. Việc này đặt ra yêu cầu cao về hiểu biết sâu rộng về cách Spark và

RDD hoạt động, và có thể cần thêm thời gian và nỗ lực để tối ưu hóa quy trình triển khai.

## 6.2 So sánh, đánh giá

### 6.2.1 K-Nearest Neighbors Regression

#### • RMSE

RMSE là viết tắt của “Root Mean Square Error” có nghĩa là “Sai số bình phương trung bình căn bậc hai”. Đây là một thước đo phổ biến được sử dụng để đánh giá chất lượng của mô hình dự đoán so với dữ liệu thực tế trong lĩnh vực thống kê và machine learning. [3]

RMSE đo lường độ lớn của sai số giữa các giá trị dự đoán và giá trị thực tế. Cụ thể, nó thực hiện các bước sau:

1. Tính toán sai số cho từng điểm dữ liệu bằng cách lấy hiệu số giữa giá trị dự đoán và giá trị thực tế.
2. Bình phương các sai số.
3. Tính trung bình của các bình phương sai số.
4. Lấy căn bậc hai của giá trị trung bình ở bước 3 để có RMSE.

Công thức:

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$$

Trong đó:

- $n$ : số lượng điểm dữ liệu
- $y_i$ : giá trị thực tế tại điểm dữ liệu thứ  $i$
- $\hat{y}_i$ : giá trị dự đoán tại điểm dữ liệu thứ  $i$

- **MAE**

MAE là viết tắt của “Mean Absolute Error” có nghĩa là “Sai số tuyệt đối trung bình”. MAE là một thước đo khác được sử dụng để đánh giá hiệu suất của mô hình dự đoán, giống như RMSE (Root Mean Square Error). [4]

MAE đo lường trung bình của giá trị tuyệt đối của sai số giữa dự đoán và giá trị thực tế. Cụ thể, MAE được tính như sau:

$$MAE = \sqrt{\frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|}$$

Trong đó:

- n: số lượng điểm dữ liệu
- $y_i$ : giá trị thực tế tại điểm dữ liệu thứ i
- $\hat{y}_i$ : giá trị dự đoán tại điểm dữ liệu thứ i

- **MSE**

MSE là viết tắt của “Mean Squared Error” có nghĩa là “Sai số bình phương trung bình”. MSE là một thước đo được sử dụng rộng rãi để đánh giá chất lượng của mô hình dự đoán, đặc biệt trong lĩnh vực thống kê và machine learning. [5]

MSE đo lường trung bình của bình phương sai số giữa giá trị dự đoán và giá trị thực tế. Cụ thể, MSE được tính như sau:

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

Trong đó:

- n: số lượng điểm dữ liệu

- $y_i$ : giá trị thực tế tại điểm dữ liệu thứ i
- $\hat{y}_i$ : giá trị dự đoán tại điểm dữ liệu thứ i

Sử dụng 3 độ đo để đánh giá mô hình K-Nearest Neighbors Regression (k=7)

Algorithm name	RMSE	MAE	MSE
K-Nearest Neighbors Regression (k=7)	1.6	1.18	2.58

### Đánh giá:

- Root Mean Square Error (RMSE): Giá trị RMSE là 1.6, đặc trưng cho độ lớn của sai số giữa giá trị dự đoán và giá trị thực tế. Sự thấp của giá trị này cho thấy mô hình có khả năng dự đoán khá chính xác (1.6 so với thang từ 0 đến 20)
- Mean Absolute Error (MAE): Với giá trị MAE là 1.18, mô hình thể hiện sự chính xác cao trong việc ước lượng giá trị thực tế. Giá trị này thấp, chứng tỏ khả năng dự đoán tốt và **còn** ảnh hưởng bởi giá trị ngoại lệ ở 1 mức độ nhất định. Lý do là chưa kiểm tra các ngoại lệ, outlier, ... nên còn gây nhiễu
- Mean Squared Error (MSE): MSE là 2.58, một giá trị tương đối, mặc dù MSE cần phải được xem xét cùng với các yếu tố khác.

## 6.2.2 K-Nearest Neighbors Classification

- **Accuracy**

Trong mô hình phân loại, "accuracy" (độ chính xác) là một độ đo thường được sử dụng để đánh giá khả năng của mô hình trong việc dự đoán đúng các nhãn hay lớp của các mẫu dữ liệu. Độ chính xác được tính bằng cách chia số lượng dự đoán đúng cho tổng số lượng mẫu dữ liệu [7].

Công thức của accuracy là:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

Với mô hình K-Nearest Neighbors Classification ( $k=10$ ) có độ chính xác như sau:

Algorithm name	Accuracy (%)
K-Nearest Neighbors Classification ( $k=10$ )	63.815

### Đánh giá:

- Mô hình đạt được độ chính xác là 57.85%. Đây là một giá trị trung bình, có thể chỉ ra khả năng dự đoán đúng khoảng một nửa số mẫu dữ liệu.
- Chưa thể tìm ra chỉ số  $k$  giúp cho mô hình đạt hiệu suất tốt nhất

### 6.2.3 Linear Regression

Algorithm name	RMSE	MAE	MSE
Linear Regression	1.632	0.846	2.665

## 6.3 Đánh giá chung

- Trong bài toán dự đoán: Mô hình Regression cho thấy khả năng dự đoán cao và ổn định, đặc biệt với giá trị thấp của RMSE và MAE.
- Trong bài toán phân lớp: Mô hình Classification đạt được độ chính xác trung bình, nhưng cần xem xét thêm về giá trị  $k$  để cải thiện hiệu suất.

## CHƯƠNG 7: KẾT LUẬN

### 7.1 Uu điểm

- Xây dựng được mô hình KNN và Linear Regression và áp dụng cho cả 2 bài toán là dự đoán và phân lớp
- Triển khai được thuật toán trong PySpark và xử lý song song thông qua RDD
- Hiểu được các yếu tố quan trọng đối với sự đánh giá học lực của học sinh trong vấn đề giáo dục
- Học được nhiều kiến thức về xử lý dữ liệu lớn và triển khai mô hình trên nền tảng Apache Spark

### 7.2 Hạn chế

- Chưa nắm vững kiến thức chuyên sâu đến việc xử lý dữ liệu lớn và cách triển khai thuật toán trên nền tảng Big Data nên gặp nhiều thiếu sót trong quá trình làm quen với và áp dụng
- Chưa áp dụng được nhiều thuật toán trong đồ án để giải quyết bài toán đặt ra
- Giới hạn về tập dữ liệu: Tập dữ liệu này có kích thước nhỏ cả về số lượng đặc trưng cũng như số lượng mẫu quan sát được, từ đó tạo ra những hạn chế trong quá trình phân tích. Có nhiều thuộc tính ở định dạng JSON nên gặp khó khăn trong việc xử lý. Chưa khai thác được ý nghĩa và tầm quan trọng của các thuộc tính
- Do tập dữ liệu khá phức tạp hoặc cần nhiều bước tiền xử lý quan trọng hơn nên khả năng dự đoán của các mô hình chưa thực sự cao.

### 7.3 Hướng phát triển

- Kiểm tra khả năng cải thiện mô hình thông qua việc chọn lọc, chuyển đổi hoặc tạo mới các đặc trưng.
- Thử nghiệm mô hình khác: Xem xét các mô hình khác ngoài KNN và Linear Regression, như Random Forest, Gradient Boosting, hoặc mô hình deep learning, để đảm bảo mô hình KNN là sự chọn lựa tốt nhất cho bài toán
- Đối với mô hình Regression: Tối ưu hóa giá trị k để đảm bảo hiệu suất tốt nhất của mô hình.
- Đối với mô hình Classification: Tiếp tục thử nghiệm với các giá trị k khác nhau và xem xét các độ đo khác như Precision, Recall để có cái nhìn toàn diện hơn về hiệu suất của mô hình.

## BẢNG PHÂN CÔNG CÔNG VIỆC CÁC THÀNH VIÊN

Thành viên Công việc	Đỗ Thị Mỹ Tâm	Đinh Thị Tú Uyên	Nguyễn Công Thành	Nguyễn Phạm Thanh Phong
Viết báo cáo	✓	✓	✓	✓
Tìm hiểu lý thuyết	✓	✓	✓	✓
Soạn slide trình chiếu		✓	✓	
EDA			✓	
Tiền xử lý dữ liệu	✓	✓		✓
Triển khai Linear Regression			✓	✓
Triển khai KNN- Regression	✓			
Triển khai KNN- Classification	✓	✓		
Evaluate Model	✓		✓	
<b>Hoàn thành (100%)</b>	100%	100%	100%	100%

## TÀI LIỆU THAM KHẢO

- [1] “Portugal Grading System.” Accessed: Jun. 02, 2024. [Online]. Available: <https://www.scholaro.com/db/Countries/Portugal/Grading-System>
- [2] “High School Student Performance & Demographics.” Accessed: Jun. 03, 2024. [Online]. Available: <https://www.kaggle.com/datasets/dillonmyrick/high-school-student-performance-and-demographics>
- [3] “Linear Regression in Machine learning,” GeeksforGeeks. Accessed: Jun. 01, 2024. [Online]. Available: <https://www.geeksforgeeks.org/ml-linear-regression/>
- [4] “About Linear Regression | IBM.” Accessed: Jun. 20, 2023. [Online]. Available: <https://www.ibm.com/topics/linear-regression>
- [5] “K-Nearest Neighbor(KNN) Algorithm,” GeeksforGeeks. Accessed: Jun. 02, 2024. [Online]. Available: <https://www.geeksforgeeks.org/k-nearest-neighbours/>
- [6] “MAE, MSE, RMSE, Coefficient of Determination, Adjusted R Squared — Which Metric is Better? | by Akshita Chugh | Analytics Vidhya | Medium.” Accessed: Jun. 03, 2024. [Online]. Available: <https://medium.com/analytics-vidhya/mae-mse-rmse-coefficient-of-determination-adjusted-r-squared-which-metric-is-better-cd0326a5697e>
- [7] “Classification: Accuracy | Machine Learning,” Google for Developers. Accessed: Jun. 03, 2024. [Online]. Available: <https://developers.google.com/machine-learning/crash-course/classification/accuracy>