

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH



HỆ ĐIỀU HÀNH (CO2017) - Học phần mở rộng

**Thu thập và phân tích
hoạt động học tập của người học
trên các Learning Management System**

Giảng viên hướng dẫn: TS.Nguyễn Quang Hùng
Sinh viên: Nguyễn Duy Khang - 2011364
Phan Phước Minh - 2010418
Nguyễn Đức An - 2010102
Đoàn Trần Cao Trí - 2010733

Thành phố Hồ Chí Minh, tháng 4 năm 2022



PHÂN CÔNG CÔNG VIỆC

STT	Họ và tên	MSSV	Problems	% BTL
1	Nguyễn Duy Khang	2011364	Streaming Data Moodle to Kafka	100%
2	Nguyễn Đức An	2010102	Streaming Data Kafka to MongoDB	100%
3	Phan Phước Minh	2010418	Streaming Data MongoDB to Spark	100%
4	Đoàn Trần Cao Trí	2010733	Spark Analytic HCMUT Student Data	100%



Mục lục

1	Giới thiệu	4
1.1	Nội dung đề tài	4
1.2	Ý nghĩa đề tài	4
1.3	Yêu cầu đề tài	4
1.4	Kết quả đầu ra của đề tài	5
2	Learning Management Systems (LMS)	6
2.1	Tổng quan	6
2.2	Learning Record Store	6
2.3	xAPI	6
3	Kafka	7
3.1	Giới thiệu về Kafka	7
3.1.1	Sơ lược nguyên nhân ra đời	7
3.1.2	Định nghĩa, ưu điểm, tính năng nổi bật của Kafka	7
3.1.3	Ứng dụng Kafka	8
3.1.4	Sơ lược về Zookeeper	9
3.2	Kafka Connect	9
4	Apache Spark	11
4.1	Tìm hiểu lý thuyết Apache Spark	11
4.1.1	Lịch sử và truyền thống	11
4.1.2	Đặc điểm nổi bật của Spark	11
4.2	Sử dụng Spark bằng ngôn ngữ Python với PySpark	12
4.2.1	Spark Core	12
4.2.2	Spark Streaming	13
4.2.3	Spark SQL	13
4.2.4	Spark DataFrame	13
4.2.5	MLlib	13
4.3	Code	13
4.3.1	Thêm thư viện và module	13
4.3.2	Chia dữ liệu train và test	13
4.3.3	Xử lý dữ liệu	14
4.3.4	Xây dựng cây quyết định	14
4.3.5	Thử dự đoán một số dữ liệu từ tập dữ liệu test	14
5	Kiến trúc của hệ thống	16
6	Phân tích dữ liệu lớn	17
6.1	Giới thiệu về tập dữ liệu	17
6.2	Mục tiêu	17
6.3	Khái quát ý tưởng	17
6.4	Xử lý dữ liệu trực tiếp từ file	17
6.5	Cài đặt thư viện, thêm thư viện và module	18
6.6	Kết nối với tập dữ liệu từ google drive	18
6.7	Đọc dữ liệu và xây dựng dataframe	18
6.8	Trích xuất dữ liệu cần thiết	18
6.9	Xây dựng tập dữ liệu chuẩn bị cho việc huấn luyện	19



6.10	Xây dựng model và lưu vào gdrive	21
6.11	Quá trình dự đoán	21
6.11.1	Đọc dữ liệu cần dự đoán	21
6.11.2	Xây dựng hàm dự đoán	22
6.12	Đánh giá model	22
6.13	Tổng kết	22
6.14	Hướng dẫn chạy code	24

1 Giới thiệu

1.1 Nội dung đề tài

Giáo dục 4.0 và chuyển đổi số giáo dục đang là một mô hình giáo dục thông minh và có những thay đổi rất lớn trong công tác giáo dục ở Việt Nam nói riêng và cả thế giới nói chung. Mô hình giáo dục 4.0 được coi là động lực cho các em học sinh, sinh viên có điều kiện công bằng để tham gia học tập với các bạn cùng trang lứa. Càng thiết thực hơn nữa, trong giai đoạn giãn cách xã hội tại Việt Nam do dịch Covid chuyển biến phức tạp, càng nhiều tầng lớp xã hội tham gia vào giáo dục 4.0: từ các bạn học sinh, sinh viên đến những người công nhân, viên chức,...

Chính vì lẽ đó là các nền tảng hệ thống quản lý học tập như Moodle, Google Classroom, Canvas,... càng phải phát triển để đáp ứng lượng truy cập khổng lồ này, phục vụ phân tích, đánh giá chất lượng của người học trên hệ thống.

Với những thách thức đó, nhóm em chọn đề tài phát triển kiến trúc phần lõi thu gom dữ liệu của người học, các hoạt động trên hệ thống Learning Management System (LMS). Đồng thời, xây dựng các đánh giá hiệu quả của người học trên các khóa học và dự đoán kết quả của người học (dựa trên lịch sử). Cung cấp cho nhà quản lý nhà trường và khoa góc nhìn về toàn cục.

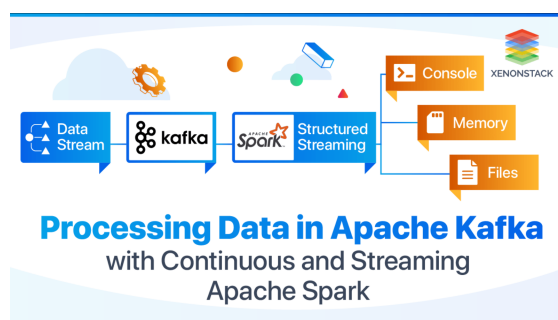
1.2 Ý nghĩa đề tài

Hệ thống được thực hiện nghiên cứu độc lập với các hệ thống Learning Management System (LMS). Nghĩa là hệ thống được xây dựng có thể phân tích và đưa ra kết quả từ các bộ dữ liệu thu thập được từ các LMS khác nhau.

Hỗ trợ đáp ứng nhu cầu về xử lý lượng lớn dữ liệu trong thời đại Big Data. Tốc độ phản hồi từ sever đến người dùng nhanh như đề xuất liên tục, thống kê; nâng cao trải nghiệm người dùng.

1.3 Yêu cầu đề tài

- Có sử dụng kiến thức về dữ liệu lớn, trí tuệ nhân tạo, Data Lake.
- Tìm hiểu về Learning Management System như Moodle
- Tìm hiểu về Kafka, Spark



Hình 1: Trình tự thực hiện



1.4 Kết quả đầu ra của đề tài

Khi hoàn thành xong đề tài này, nền tảng của nhóm có thể giúp giáo viên so sánh sự tham gia và kết quả với các khóa học khác.

Ngoài ra, cung cấp thông tin để giáo viên có thể thực hiện như liên hệ trực tiếp với học sinh, chỉ định nội dung đặc biệt, khuyến khích tham gia hoặc cung cấp dịch vụ dạy kèm đặc biệt.

Về phía học sinh, sinh viên, có thể tìm hiểu về hiệu suất học tập cá nhân, hoặc so sánh với nhóm.

2 Learning Management Systems (LMS)

2.1 Tổng quan

LMS, theo nghĩa tiếng Việt là “*hệ thống quản lý học tập*”, là những hệ thống hỗ trợ việc quản lý, giám sát người học thông qua việc cung cấp các file tài liệu, tổ chức các bài kiểm tra hay theo dõi hành vi của người học.

Quản trị viên của hệ thống có thể thu thập một số thông tin về hành vi của học viên:

- Những thông tin liên quan đến submission (nộp bài).
- Những thông tin về tương tác với khóa học (xem khóa học, xem các module).

Trong phạm vi nghiên cứu của nhóm, nhóm tập trung nghiên cứu hệ thống LMS là Moodle.

Moodle [3] là LMS phổ biến nhất tại các trường thuộc hệ thống Đại học Quốc gia TP HCM ở Việt Nam. Moodle là một dự án mã nguồn mở, có thể dễ dàng mở rộng thêm các tính năng có sẵn thông qua các plugins.

2.2 Learning Record Store

Có thể hiểu đây là một cơ sở dữ liệu để lưu trữ những thông tin cần thiết cho việc phân tích các dữ liệu thu thập được nêu ở trên.

Một LRS có thể được sử dụng cho nhiều LMS khác nhau.

2.3 xAPI

Ở đây, để thống nhất giữa các LMS khác nhau, người ta đưa ra một chuẩn có tên là xAPI.

Khi đó, những dữ liệu thu thập được mô tả ở phần trên sẽ được biểu diễn như sau:

```
{
  "actor": {
    "name": "Sally Glider",
    "mbox": "mailto:sally@example.com"
  },
  "verb": {
    "id": "http://adlnet.gov/expapi/verbs/completed",
    "display": {"en-US": "completed"}
  },
  "object": {
    "id": "http://example.com/activities/solo-hang-gliding",
    "definition": {
      "name": { "en-US": "Solo Hang Gliding" }
    }
  },
  "result": {
    "completion": true,
    "success": true,
    "score": {
      "scaled": .95
    }
  }
}
```

3 Kafka

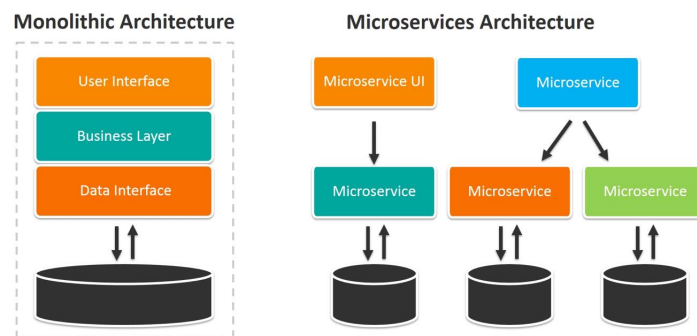
3.1 Giới thiệu về Kafka

3.1.1 Sơ lược nguyên nhân ra đời

Kiến trúc nguyên khối (monolithic architecture): là kiến trúc sơ khai khi người lập trình hiện thực một dự án. Khi sử dụng kiến trúc này, người dùng hiện thực các thành phần khép kín và kết nối với nhau. Điểm lợi của kiến trúc này là dễ sử dụng, dễ hiện thực cho người lập trình. Tuy nhiên nó lại có một số nhược điểm như sau:

- Với các dự án lớn, số lượng code lớn, sẽ gây ra mất kiểm soát, khó quản lí code.
- Khi muốn chỉnh sửa, cần phải đồng bộ giữa các thành phần, làm mất thời gian.
- Khi có một thành phần bị lỗi, toàn bộ hệ thống sẽ phải dừng lại.

Vì vậy, kiến trúc microservices được ra đời để giúp quản lí được các dự án lớn. Để xử lý được tốt các tương tác giữa các thành phần, Kafka được LinkedIn tạo ra như một môi trường để giao tiếp, tương tác chung.



Hình 2: Monolithic and Microservices Architecture

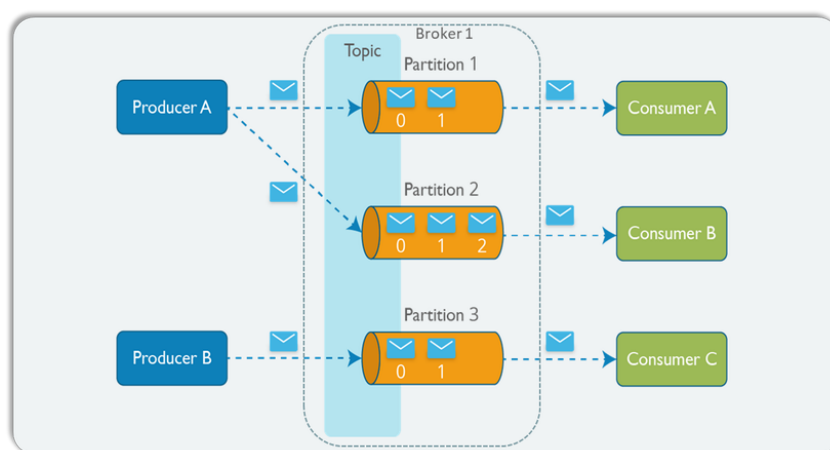
3.1.2 Định nghĩa, ưu điểm, tính năng nổi bật của Kafka

Định nghĩa: Kafka [1] là một hệ thống message theo cơ chế Pub-Sub. Nó cho phép các nhà sản xuất (gọi là producer) viết các message vào Kafka mà một, hoặc nhiều người tiêu dùng (gọi là consumer) có thể đọc, xử lý được những message đó.

Một số thành phần chính của Kafka:

- **Message:** Thông tin được gửi đi, có thể là text, binary, Json hoặc một định dạng format nào đó.
- **Broker:** Một host có thể chạy nhiều server kafka, mỗi server như vậy gọi là một broker. Các broker này cùng trở tới chung 1 zookeeper, gọi là cụm broker(hay là Clusters). Broker là nơi chứa các partition. Một broker có thể chứa nhiều partition.
- **Topic:** Là nơi chứa các message được publish từ Producer tới Kafka, nhìn về mặt database thì topic giống như một table trong cơ sở dữ liệu quan hệ, và mỗi message như một bản ghi của table đó.

- **Partition:** Nơi lưu trữ message của topic. Một topic có thể có nhiều partition. Khi khởi tạo topic cần set số partition cho topic đó. Partition càng nhiều thì khả năng làm việc song song cho đọc và ghi được thực hiện nhanh hơn. Các message trong partition được lưu theo thứ tự bất biến(offset). Một partition sẽ có tối thiểu 1 replica để đề phòng trường hợp bị lỗi. Số lượng replica luôn nhỏ hơn số lượng broker.
- **Producer:** Chương trình/service tạo ra message, đẩy message publish vào Topic.
- **Consumer:** Chương trình/service có chức năng subscribe vào một Topic để tiêu thụ, xử lý các message đó.



Hình 3: Mô hình cấu trúc Kafka chi tiết

Với thiết kế riêng của mình, Kafka có những ưu điểm sau:

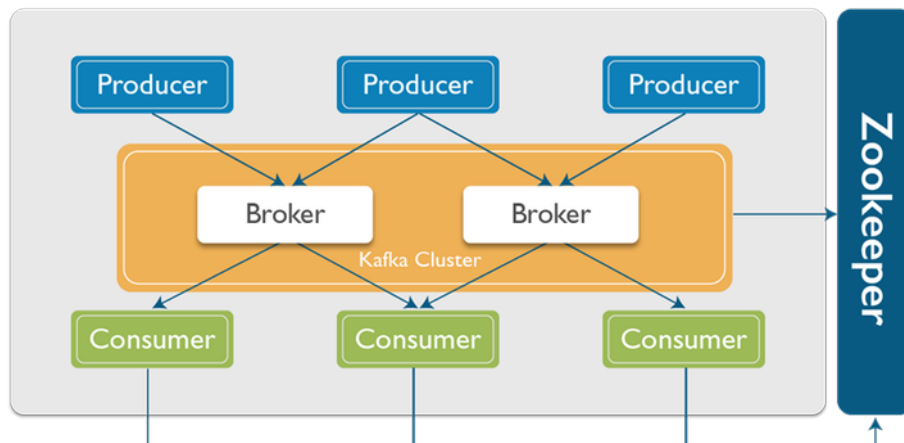
- Xử lý dữ liệu thời gian thực (real-time)
- Dữ liệu phân tán (distributed): dữ liệu được phân tán ra nhiều cụm khác nhau trong cluster.
- Khả năng phục hồi (resilient): Khi hệ thống gặp lỗi, có thể phục hồi được dữ liệu ngay trước khi lỗi.
- Khả năng chịu lỗi (fault tolerant): Khi có một broker bị lỗi thì các broker khác đều có thể tiếp tục làm việc.
- Khả năng mở rộng (scalable).

3.1.3 Ứng dụng Kafka

Một số ứng dụng của Kafka là:

- Vận hành hệ thống: lưu trữ bộ số dữ liệu, theo dõi hoạt động, tổng hợp các hoạt động truy cập,...
- Phân tích dữ liệu: Xử lý dữ liệu dòng, thu thập dữ liệu,...
- Công nghệ phần mềm: Hệ thống tin nhắn, triển khai các thay đổi trên hệ thống thông qua các event,...

3.1.4 Sơ lược về Zookeeper



Hình 4: Mô hình Kafka cơ bản

Định nghĩa: Zookeeper là một dịch vụ tập trung để duy trì và cấu hình dữ liệu, cung cấp đồng bộ hóa linh hoạt và mạnh mẽ trong hệ thống phân tán.

Liên hệ giữa Zookeeper và Kafka: Zookeeper theo dõi, lưu trữ trạng thái của Kafka brokers, topics, partitions,... và thông báo mọi thay đổi của Kafka. Kafka **không thể sử dụng** nếu không có Zookeeper.

3.2 Kafka Connect

Khi nói tới việc ứng dụng Kafka để xây dựng Data pipelines, ta có thể hình dung ra 2 trường hợp:

- Thứ nhất, data pipeline với Apache Kafka là một trong hai endpoint. Ví dụ: lấy dữ liệu từ Kafka đẩy vào S3(Amazon Simple Storage Service) , hoặc lấy dữ liệu từ Database đẩy vào Kafka
- Thứ hai, Sử dụng kafka để xây dựng một pipeline giữa 2 hệ thống riêng biệt. Ví dụ: lấy data từ Twitter đưa vào hệ thống Elasticsearch bao gồm việc lấy data từ Twitter đẩy vào Kafka sau đó từ kafka đưa vào Elasticsearch.

Apache Kafka có thể đóng vai trò là một buffer cực lớn và đáng tin cậy: sự độc lập giữa việc đọc và ghi, giữa chính các producer và các consumer kết hợp với khả năng bảo mật hiệu quả giúp Kafka đáp ứng yêu cầu của phần lớn data pipelines.

- **Timeliness:** Kafka - một streaming data platform với khả năng lưu trữ đáng tin cậy và khả năng mở rộng tốt có thể đáp ứng tốt từ data pipeline gần thời gian thực cho tới việc đồng bộ dữ liệu theo lô. Producer có thể ghi dữ liệu vào Kafka định kỳ hoặc bất cứ khi nào cần. Consumer có thể đọc dữ liệu mới nhất ngay khi nó được ghi vào, cũng có thể định kỳ đọc dữ liệu hàng giờ hoặc vài giờ một lần.
- **Reliability:** Đối với các hệ thống lớn, việc mất mát dữ liệu, trùng lặp dữ liệu có thể gây ra hậu quả lớn. Kafka đảm bảo đảm bảo yêu cầu "exactly-once" cho mỗi sự kiện được đưa vào hệ thống.

- **High and Varying Throughput:** Kafka rất linh hoạt trong việc thích ứng mỗi khi yêu cầu mở rộng hệ thống, các consumer và producer được thêm vào hệ thống một cách độc lập tùy thuộc vào yêu cầu đặt ra. Hoạt động như một bộ nhớ đệm cực lớn giữa producers và consumer, Kafka còn cung cấp một cơ chế backpressure giúp cho dữ liệu được giữ ở trong cụm cho đến khi consumer kịp xử lý.
- **Data Formats:** Việc lựa chọn định dữ liệu tùy thuộc vào yêu cầu đặt ra của mỗi hệ thống, nghiệp vụ khác nhau. Kafka hỗ trợ rất nhiều: XML, text, JSON, Avro, CSV, ... Producer và Consumer có thể sử dụng bất kì bộ serializer nào để biểu diễn dữ liệu dưới bất kì định dạng nào bạn muốn.
- **Security:** Bảo mật luôn là vấn đề quan trọng. Kafka có thể đáp ứng tốt yêu cầu về bảo mật. Bạn có thể chắc chắn rằng dữ liệu của bạn khi đi qua Kafka có thể được mã hóa nếu bạn muốn vì bạn chủ động trong ghi dữ liệu. Kafka cũng hỗ trợ cơ chế xác thực (SASL), do đó bạn có thể yên tâm rằng những người có quyền đọc mới được đọc topic của bạn. Bạn cũng có thể custom thêm để ghi log lại các sự kiện mà bạn muốn theo dõi.

4 Apache Spark

4.1 Tìm hiểu lý thuyết Apache Spark

4.1.1 Lịch sử và truyền thống

Để tìm hiểu về Apache Spark [2], trước tiên ta nhìn lại về Hadoop, một framework có truyền thống lâu đời trong việc phân tích và xử lý dữ liệu lớn. Ưu điểm lớn nhất của Hadoop được dựa trên một mô hình lập trình song song với xử lý dữ liệu lớn là MapReduce, mô hình này cho phép khả năng tính toán có thể mở rộng, linh hoạt, khả năng chịu lỗi và chi phí rẻ.



Hình 5: Hadoop và Spark

Dù có rất nhiều điểm mạnh về khả năng tính toán song song và khả năng chịu lỗi cao nhưng Apache Hadoop có một nhược điểm là tất cả các thao tác đều phải thực hiện trên ổ đĩa cứng và điều này đã làm giảm tốc độ tính toán đi gấp nhiều lần.

Để khắc phục được nhược điểm này thì một hệ thống khác mang tên Apache Spark được ra đời bởi Matei Zaharia tại UC Berkeley's AMPLab vào năm 2009 và mở nguồn vào năm sau 2010. Apache Spark có thể chạy nhanh hơn 10 lần so với Hadoop ở trên đĩa cứng và 100 lần khi chạy trên bộ nhớ RAM.

4.1.2 Đặc điểm nổi bật của Spark

Sở dĩ Spark nhanh là vì nó xử lý mọi thứ ở RAM. Nhờ xử lý ở bộ nhớ nên Spark cung cấp các phân tích dữ liệu thời gian thực cho các chiến dịch quảng cáo, machine learning (học máy), hay các trang web mạng xã hội.

Spark nhận được nhiều sự hưởng ứng từ cộng đồng Big data trên thế giới do cung cấp khả năng tính toán nhanh và nhiều thư viện đi kèm hữu ích như Spark SQL (với kiểu dữ liệu DataFrames), Spark Streaming, MLlib (machine learning: classification, regression, clustering, collaborative filtering, và dimensionality reduction) và GraphX (biểu diễn đồ thị nhờ kết quả tính toán song song).

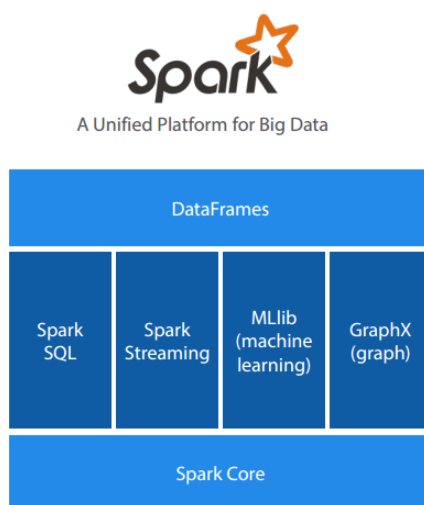
- Xử lý dữ liệu: Spark xử lý dữ liệu theo lô và thời gian thực.
- Tính tương thích: Có thể tích hợp với tất cả các nguồn dữ liệu và định dạng tệp được hỗ trợ bởi cụm Hadoop.
- Hỗ trợ ngôn ngữ: hỗ trợ Java, Scala, Python và R.
- Xử lý thời gian thực: Apache Spark có thể xử lý các luồng sự kiện dữ liệu thời gian thực với tốc độ hàng triệu sự kiện mỗi giây.

Hiện nay, có rất nhiều hãng lớn đã dùng Spark cho các sản phẩm của mình như Yahoo, ebay, IBM, Cisco...



Hình 6: Những doanh nghiệp sử dụng Apache Spark

4.2 Sử dụng Spark bằng ngôn ngữ Python với PySpark



Hình 7: Các thành phần của PySpark [?]

Apache Spark gồm có 5 thành phần chính : Spark Core, Spark Streaming, Spark SQL, MLlib và GraphX, trong đó:

4.2.1 Spark Core

Là thành phần cốt lõi của spark, thành phần này hỗ trợ sử dụng spark trên một số ngôn ngữ như python, scala, java,...

4.2.2 Spark Streaming

Spark Streaming giúp cho kỹ sư và các nhà phân tích dữ liệu xử lý thông tin trong thời gian thực, khi mà các dữ liệu được truyền tải và cập nhật liên tục từ các nguồn như Kafka, Flume, và Amazon Kinesis,...

4.2.3 Spark SQL

Spark SQL là một module Spark dùng cho truy vấn cấu trúc dữ liệu. Module cung cấp nhiều hàm và phương thức hỗ trợ truy vấn dữ liệu dựa trên công cụ truy vấn SQL phân tán (distributed SQL query engine).

4.2.4 Spark DataFrame

Spark DataFrame hỗ trợ xây dựng dataframe từ tập dữ liệu nhưng với khả năng tối ưu hóa phong phú hơn.

4.2.5 MLlib

MLlib là một nền tảng học máy phân tán bên trên Spark do kiến trúc phân tán dựa trên bộ nhớ. Giúp thực hiện xây dựng model hiệu quả và đơn giản.

4.3 Code

Code được thực hiện theo các phần sau

4.3.1 Thêm thư viện và module

Cài đặt thư viện

```
1 !pip install pyspark

import pandas
import pyspark
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
```

4.3.2 Chia dữ liệu train và test

80% dữ liệu sẽ dùng để huấn luyện model và 20% còn lại để thử lại

```
1 df = pandas.read_csv(r'Final Dataset.csv').copy()
2 df_train = df[0: int(len(df) * 0.8)]
3 df_test = df[int(len(df) * 0.8) + 1:]
4 # print(df_train)
5 # print(df_test)
```

4.3.3 Xử lý dữ liệu

Để xây dựng cây quyết định (Decision Tree) thì các giá trị phải ở dạng số. Nghĩa là đối với các ô giá trị kiểu chuỗi thì phải được hash về dạng số nào đó.

Mỗi giá trị kiểu chuỗi mới sẽ được chuyển đổi thành một số. Quy ước dãy số được chuyển đổi bắt đầu từ 0 và tăng dần thêm 1. Dữ liệu ở cột tên học sinh **ApplicantName** đang học không

```
defaultdict(<function <lambda> at 0x7f44a26f57a0>, {'Poor': 0, 'Fair': 1, 'Good': 2, 'Excellent': 3, 'Very Good': 4, 'Adequate': 5})
defaultdict(<function <lambda> at 0x7f44a26f20e0>, {'Medium': 0, 'Low': 1, 'High': 2})
defaultdict(<function <lambda> at 0x7f44a26f2ef0>, {'No': 0})
defaultdict(<function <lambda> at 0x7f44a26f2cb0>, {'Yes': 0, 'No': 1})
defaultdict(<function <lambda> at 0x7f44a26f2440>, {'Yes': 0, 'No': 1})
defaultdict(<function <lambda> at 0x7f44a26f48c0>, {'Yes': 0, 'No': 1})
defaultdict(<function <lambda> at 0x7f44a26f4cb0>, {'Yes': 0, 'No': 1})
defaultdict(<function <lambda> at 0x7f44a26f4ef0>, {'Yes': 0, 'No': 1})
defaultdict(<function <lambda> at 0x7f44a26f4dd0>, {'Low': 0, 'High': 1, 'Medium': 2})
defaultdict(<function <lambda> at 0x7f44a26f4d40>, {'Medium': 0, 'Low': 1})
defaultdict(<function <lambda> at 0x7f44a26f4950>, {'Fail': 0, 'Fair': 1, 'Good': 2, 'Excellent': 3, 'Very Good': 4, 'Adequate': 5})
defaultdict(<function <lambda> at 0x7f44a26f4e60>, {'Fail': 0, 'Fair': 1, 'Good': 2, 'Excellent': 3, 'Very Good': 4, 'Adequate': 5})
defaultdict(<function <lambda> at 0x7f44a26f4680>, {'Fail': 0, 'Fair': 1, 'Good': 2, 'Excellent': 3, 'Very Good': 4, 'Adequate': 5})
defaultdict(<function <lambda> at 0x7f44a26f4c20>, {'Poor': 0, 'Fair': 1, 'Good': 2, 'Excellent': 3, 'Very Good': 4, 'Adequate': 5})
defaultdict(<function <lambda> at 0x7f44a26f4b00>, {'Poor': 0, 'Fair': 1, 'Good': 2, 'Excellent': 3, 'Very Good': 4, 'Adequate': 5})
defaultdict(<function <lambda> at 0x7f44a26f4b90>, {0: 0, 1: 1, 2: 2, 3: 3, 4: 4, 5: 5, 6: 6, 8: 7})
defaultdict(<function <lambda> at 0x7f44a26f4a70>, {0: 0, 1: 1, 2: 2, 3: 3, 4: 4, 5: 5, 6: 6, 7: 7, 13: 8})
```

Hình 8: Xử lý dữ liệu kiểu chuỗi

mang giá trị dự đoán ở cuối kết quả khóa học, nên trước khi đưa vào train model, ta phải bỏ qua cột giá trị này.

4.3.4 Xây dựng cây quyết định

```
1 X = df_train[features[:-1]] #remove Results
2 y = df_train[features[-1]]
3 dtree = DecisionTreeClassifier()
4 dtree = dtree.fit(X.values, y.values)
```

4.3.5 Thử dự đoán một số dữ liệu từ tập dữ liệu test

Dữ liệu dự đoán một số học sinh sau: Nhận xét thấy kết quả dự đoán tương đối giống với kết quả hiện có của tập dữ liệu **test**. Kết quả model dự đoán:

```
1 Frequency = 92.3076923076923%
```

Prediction Pass		Actually Pass
Student 305		
Prediction Pass		Actually Pass
Student 306		
Prediction Fail		Actually Pass
Student 307		
Prediction Fail		Actually Fail
Student 308		
Prediction Pass		Actually Pass
Student 309		
Prediction Pass		Actually Pass
Student 310		
Prediction Pass		Actually Pass
Student 311		
Prediction Fail		Actually Fail
Student 312		
Prediction Pass		Actually Pass
Student 313		
Prediction Fail		Actually Fail
Student 314		
Prediction Fail		Actually Fail
Student 315		
Prediction Fail		Actually Pass
Student 316		
Prediction Fail		Actually Pass
Student 317		
Prediction Fail		Actually Fail

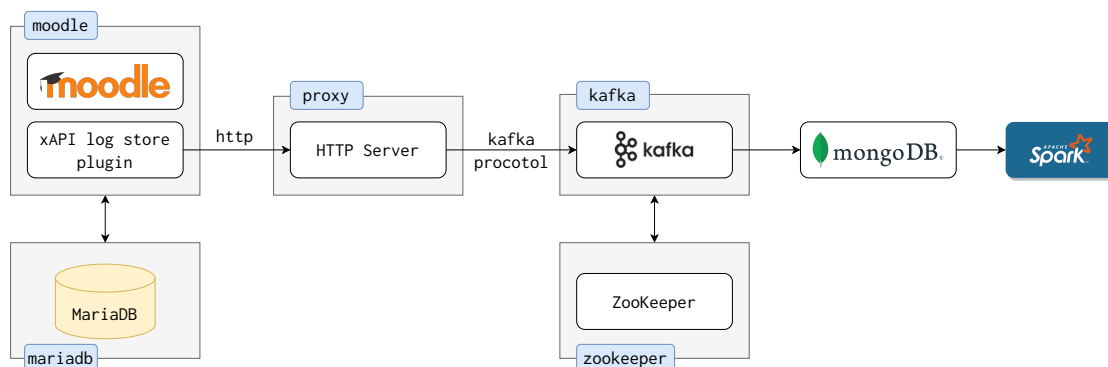
Hình 9: Kết quả dự đoán một số kết quả học tập của học sinh

5 Kiến trúc của hệ thống

Hình 10 mô tả kiến trúc mà nhóm nghiên cứu đề xuất cho đề tài này.

LƯỢC ĐỒ KIẾN TRÚC CỦA NHÓM

18/05/2022



Hình 10: Kiến trúc của hệ thống

Đầu tiên, ở các hệ thống LMS, dữ liệu về hành vi của học viên sẽ được thu thập và gửi về Kafka. Dữ liệu này sẽ được gửi liên tục thành dòng. Ở đây, nhóm nghiên cứu chọn Moodle LMS để có thể dễ dàng mở rộng bằng các plugin.

Plugin được chọn ở đây là Moodle Logstore xAPI [4]. Dữ liệu được gửi vào một endpoint qua HTTP nên nhóm thiết lập một container **proxy** để một đầu nhận HTTP và đầu còn lại kết nối với Kafka.

Sau khi nhận dữ liệu từ Kafka Consumer, chúng ta sẽ tiến hành lưu các log vào file txt rồi tiến hành xử lý để phân các log ra đúng với định dạng file json.

Sau khi nhận được các file json, ta sẽ nhập chúng vào MongoDB (ở đây chúng ta sử dụng như một Data Warehouse). Khi cần lấy dữ liệu từ MongoDB để sử dụng Spark phân tích, chúng ta sẽ sử dụng *MongoDB-Spark-connector*.

Sau đó, nhóm sẽ sử dụng Spark để phân tích các dữ liệu.

6 Phân tích dữ liệu lớn

6.1 Giới thiệu về tập dữ liệu

Tập dữ liệu chứa thông tin về các điểm thành phần và điểm tổng kết của hơn 3 triệu sinh viên trong học kỳ nhất định. Mỗi điểm dữ liệu chứa các thông tin cơ bản bao gồm: năm học, học kỳ, môn học trong học kỳ gồm tên tiếng Việt và mã môn học, mã số sinh viên, các điểm thành phần bao gồm điểm kiểm tra (KT), điểm bài tập (BT), điểm bài tập lớn (BTLDA), thí nghiệm (TN) và điểm thi (THI); mỗi điểm thành phần, tùy vào môn học sẽ có tỉ lệ quyết định đến điểm tổng kết (TKET). Ngoài ra, đối với một số môn học, còn một cách thức khác để tính điểm tổng kết là trung bình cộng các cột điểm F_DIEM1, F_DIEM2, F_DIEM10

Tạo đường dẫn tại google drive để chạy code: Tạo giao diện chính của google drive, tạo thư mục **OSmrData**, bên trong là các file .xlsx và folder **Models** cùng cấp.

6.2 Mục tiêu

Mục tiêu là sử dụng tập dữ liệu này để xây dựng model có khả năng **dự đoán điểm tổng kết** của một môn học của một sinh viên tại một học kỳ nào đó khi biết được một số thông tin điểm thành phần.

6.3 Khái quát ý tưởng

Điểm tổng kết sẽ được tính bằng hai cách

- Trung bình cộng các cột điểm F_DIEM1, F_DIEM2, F_DIEM10.
- Các điểm thành phần nhân với tỉ lệ tương ứng

Model được chọn là **Cây quyết định** - decision tree. Với cách xây dựng và dự đoán **regression**.

Ý tưởng trực tiếp là nếu xây dựng chỉ 1 model cho tất cả thuộc tính bao gồm mã môn học (đặc trưng cho môn học đó) và các điểm thành phần thì model sẽ trở nên phức tạp, khó chỉnh sửa và tốn bộ nhớ.

Ý tưởng là chia nhỏ model trên thành các model con, mỗi model chỉ đảm nhận dự đoán của 1 mã môn học. Nếu một tập dữ liệu bị thay đổi hoặc bổ sung đáng kể, model sẽ dễ dàng được build lại.

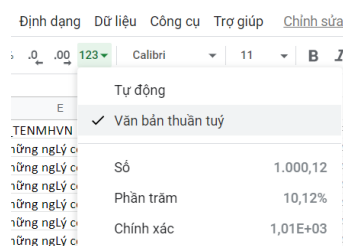
6.4 Xử lý dữ liệu trực tiếp từ file

Lỗi tự động chuyển đổi định dạng

Lỗi nếu không xử lý thì giá trị khi đọc vào sẽ là **java.util.GregorianCalendar**, đây là kiểu định dạng ngày tháng, do dữ liệu được đưa lên google drive bị thay đổi định dạng. Ví dụ điểm có định dạng số thực, có giá trị là 7.5 nhưng tự động bị thay đổi định dạng ngày tháng là 07/05/2022.

Hướng giải quyết là định dạng về *Văn bản thuần túy* trực tiếp trên excel đối với các cột dữ liệu điểm thành phần, tỉ lệ và điểm tổng kết.

Sắp xếp dữ liệu theo mã môn học Sử dụng công cụ sort hiện đại và có sẵn trên excel để sắp xếp dữ liệu theo mã môn học để dễ phân nhóm tập dữ liệu.



6.5 Cài đặt thư viện, thêm thư viện và module

```
1 from pyspark.mllib.regression import LabeledPoint
2 from pyspark.mllib.tree import DecisionTree, DecisionTreeModel
3 from pyspark.mllib.util import MLUtils
4 from pyspark import SparkContext
5 from numpy import array
6 from pyspark.sql import SparkSession
7 from pyspark.mllib.regression import LabeledPoint
8 from pyspark.mllib.tree import DecisionTree
9 from pyspark.sql.types import *
10 import pandas as pd
11 import copy
12 import numpy
13 import math
14 import glob
15 from pyspark import SparkContext, SparkConf
16 from google.colab import drive
17 from collections import defaultdict
18 import copy
19 import numpy
```

6.6 Kết nối với tập dữ liệu từ google drive

Đường dẫn dữ liệu: /content/drive/MyDrive/OSmrData/

Trong đó sẽ bao gồm tập dữ liệu và thư mục chứa các models đã được huấn luyện

6.7 Đọc dữ liệu và xây dựng dataframe











Việc xây dựng cây quyết định cần một tập dữ liệu có cùng mã môn học F_MAMH, nên nhóm sẽ thực hiện sort dữ liệu để phân nhóm tập dữ liệu theo mã môn học.

Sau đó định nghĩa kiểu dữ liệu của từng trường dữ liệu (cột) - **schema** thì mới được phép tiến hành gọi hàm createDataFrame của pyspark để chuyển đổi từ file đọc pandas sang dataframe trong hệ sinh thái pyspark.

6.8 Trích xuất dữ liệu cần thiết

Để thu nhỏ dataframe, nhóm sẽ lược bớt một số trường (cột) dữ liệu không ảnh hưởng đến điểm tổng kết như tên môn học bằng tiếng việt, năm học, học kỳ, mã số sinh viên, ... Các trường mà nhóm chọn bao gồm các điểm thành phần và điểm tổng kết ở cuối cùng.

Drive của tôi > OSmrData ▾

Tên ↑	Chủ sở hữu	Sửa đổi lần cuối	Kích cỡ tệp
 Models	tôi	21:51	—
 diem_f0_md_predict.xlsx	tôi	09:02	22 KB
 diem_f0_md.xlsx	tôi	08:58	1,1 MB
 diem_f1_md.xlsx	tôi	24 thg 10, 2018	51,1 MB
 diem_f2_md.xlsx	tôi	11 thg 5, 2022	43,9 MB
 diem_f3_md.xlsx	tôi	24 thg 10, 2018	41,4 MB
 diem_f4_md.xlsx	tôi	24 thg 10, 2018	41,2 MB
 diem_f5_md.xlsx	tôi	24 thg 10, 2018	41,6 MB
 diem_f6_md.xlsx	tôi	24 thg 10, 2018	38,7 MB
 diem_f7_md.xlsx	tôi	25 thg 10, 2018	10,4 MB

Hình 11: Đường dẫn vào thư mục chứa tập dữ liệu và Models

```
1 dfPoint = df.select('KT', 'BT', 'BTLDA', 'TN', 'THI', 'F_DIEM1', 'F_DIEM2', 'F_DIEM10',
    'TKET')
```

Để truy xuất nhóm dữ liệu có cùng mã môn học F_MAMH, ta sẽ trích xuất nhanh cột dữ liệu F_MAMH dưới dạng list. Sau đó lưu trữ hàng đầu và hàng cuối của nhóm có cùng mã môn học F_MAMH.

```
1 dfMAMH = df.select('F_MAMH').rdd.flatMap(lambda x: x).collect()
```

6.9 Xây dựng tập dữ liệu chuẩn bị cho việc huấn luyện

Trong pyspark, dùng phân tán RDD để map dữ liệu về dạng LabeledPoint phục vụ cho việc xây dựng cây quyết định.

Tập dữ liệu phân tán phục hồi hoặc RDD trong PySpark là cấu trúc dữ liệu cốt lõi của PySpark. PySpark RDD là một đối tượng cấp thấp và có hiệu quả cao trong việc thực hiện các tác vụ phân tán.

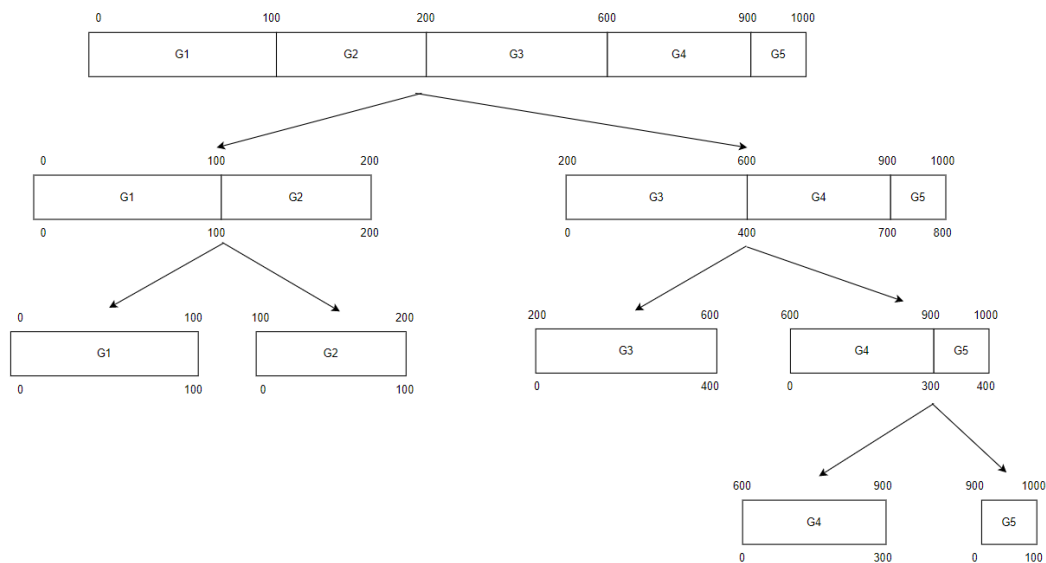
LabeledPoint là kiểu dữ liệu phục vụ cho việc xây dựng cây quyết định gồm hai tham số đầu vào là giá trị dự đoán \mathbf{Y} và danh sách các biến phụ thuộc (features) \mathbf{X} .

Xây dựng LabeledPoint tuyến tính - tức là mỗi dòng dữ liệu đọc được sẽ được chuyển đổi về dạng LabeledPoint theo từng nhóm, thì chương trình sẽ xử lý với tốc độ chậm hơn, do mỗi lần truy xuất, chương trình phải lấy từ biến dataframe(df) chứa 500.000 dữ liệu để trích xuất tại vị trí đầu và cuối của nhóm đó. Vì số lượng nhóm là rất lớn nên mỗi lần truy xuất như vậy sẽ tốn rất nhiều thời gian.

```
1 # old method with more than 1 hour
2 for i in range(1, len(MAMH_index)):
```

```
3 # print(MAMH_index[i - 1], end = ' ')
4 subdf = dfPoint.collect()[MAMH_index[i - 1]:MAMH_index[i]]
5 subdf = spark.createDataFrame(subdf)
6 dataTrain[MAMH_name[i - 1]] = subdf.rdd.map(lambda row: LabeledPoint(row[-1],
    row[:-1]))
```

Do đó, phương pháp đề xuất sẽ là **cắt tỉa nhị phân** từ dataframe ban đầu bằng đệ quy, cụ thể được minh họa như sau:



Hình 12: Mô hình cắt tủa nhị phân

Mỗi dataframe sau khi cắt tỉa sẽ được đánh số index lại bắt đầu từ 0 thay vì index như trong dataframe gốc. Ví dụ ở tầng gốc đi xuống 2 tầng, dataframe G3 ban đầu có index tại [200, 600]. Tuy nhiên tại tầng này sẽ là [0, 400].

```

1 def recur(_subdf, _lefti, _righti, t):
2     if (_lefti >= _righti):
3         return
4     if _lefti + 1 == _righti:
5         print(_lefti, ' ', _righti)
6         # print(_subdf)
7         _subdf = spark.createDataFrame(_subdf)
8         dataTrain[MAMH_name[_lefti]] = _subdf.rdd.map(lambda row: LabeledPoint(row[-1],
9             row[:-1]))
10    else:
11        middle = int((_lefti + _righti) / 2)
12        # print('left: ', _lefti - t, ' ', middle - t, ' ', _righti - t, ' ', t)
13        print('left cut: ', MAMH_index[_lefti - t], ' ', MAMH_index[middle - t])
14        recur(_subdf[MAMH_index[_lefti] - MAMH_index[t]: MAMH_index[middle] -
15            MAMH_index[t]], _lefti, middle, _lefti)
16        # print('right: ', _lefti - t, ' ', middle - t, ' ', _righti - t, ' ', t)
17        print('right cut: ', MAMH_index[middle - t], ' ', MAMH_index[_righti - t])

```

```
16     recur(_subdf[MAMH_index[middle] - MAMH_index[t]: MAMH_index[_righti] -  
17           MAMH_index[t]], middle, _righti, middle)  
18 recur(dfPoint.collect()[MAMH_index[0]:MAMH_index[len(MAMH_index) - 1]], 0,  
       len(MAMH_index) - 1, 0)
```

Kết quả là ngoài sự mong đợi, thời gian xây dựng bằng linear là hơn **1 giờ** đồng hồ, trong khi với phương pháp cắt tỉa nhị phân thì thời gian hoàn thành chỉ **3 phút**.

6.10 Xây dựng model và lưu vào gdrive

Từ tập dữ liệu **dataTrain** thuộc kiểu dictionary bên trên, ta sẽ xây dựng Models là tập hợp các model con, đặc trưng bởi mã môn học

```
1 Models[key] = DecisionTree.trainRegressor(value, {})
```

Code hoàn chỉnh

```
1 # from google.colab import drive  
2 sparkContext=spark.sparkContext  
3 Models = defaultdict(lambda: 'empty')  
4 mPath = "/content/drive/MyDrive/OSmrData/Models/"  
5 for key, value in dataTrain.items():  
6     # print(key, value)  
7     Models[key] = DecisionTree.trainRegressor(value, {})  
8     try:  
9         Models[key].save(spark.sparkContext, mPath + str(key))  
10    except:  
11        print("Model have already exist")  
12        # print(Models[key].toDebugString())  
13    # print(Models)
```

6.11 Quá trình dự đoán

6.11.1 Đọc dữ liệu cần dự đoán

Dữ liệu cần dự đoán sẽ đặt tại file **diem_f0_md_predict.xlsx**. Format của file tương đối trùng với các file trước, có thể thay đổi dữ liệu để thử nghiệm model

```
1 pdfPredict = pd.read_excel('/content/drive/MyDrive/OSmrData/diem_f0_md_predict.xlsx',  
                             sheet_name='Sheet1')  
2 dfPredict = pandas_to_spark(pdfPredict)  
3 print(dfPredict.collect()[0])  
4 predictMethod(dfPredict.collect()[0])
```

Kết quả dự đoán đối với kiểu dữ liệu Row bên dưới có điểm tổng kết 7.0 là

```
1 Row(NAMHOC=2016.0, TENHK=2.0, NHHK=162.0, F_MAMH='SP1009', F_TENMHVN='ngli CM ca  
    ngCSVN ', F_DVHT=3.0, F_PHTRAMKT=30.0, F_PHTRAMTH=70.0, F_MANH='A10', F_TO='B',  
    STT=1.0, F_MAKH='DC', F_TENLOP='DC1404', F_KHOI='DHDCDIA14', F_MANG='DIA',  
    F_TENNGVN='a cht-Du kh', MASV1=62316547.0, KT=6.7, TILEKT=30.0, BT=9.2,
```

```
2 TILEBT=20.0, BTLDA=0.0, TILEBTLD=0.0, TN=nan, TILETN=0.0, THI=6.5, TILETHI=50,  
TKET=7.0, F_DIEM1=nan, F_DIEM2=nan, F_DIEM10=nan, GHICHU=nan, T=nan)  
2 Predict total score is 7.021306818181818
```

Nhận xét, với kết quả là thực tế là 7.0 và kết quả được model dự đoán trên là khá chính xác.

6.11.2 Xây dựng hàm dự đoán

- Nếu tất cả điểm F_DIEM khác không:
 - Nếu không có mã môn học:
 - * Nếu các điểm thành phần và tỉ lệ đủ => Tính điểm tổng kết
 - * Ngược lại => Không thể dự đoán
 - Ngược lại,
 - * Thử load model[key]
 - * => Đưa ra kết quả dự đoán
- Ngược lại,
 - Nếu có ít nhất 1 điểm F_DIEM => Tính điểm tổng kết dựa vào trung bình cộng.
 - Ngược lại => Không thể dự đoán.

6.12 Đánh giá model

Do model sử dụng là decision tree với giải thuật regression nên để đánh giá độ tốt của model, nhóm sử dụng giá trị trung bình hàm mất mát average loss function

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

Trong đó:

- y là giá trị thực
- \bar{y} là giá trị dự đoán
- n là số điểm dữ liệu

```
1 Total query prediction 200  
2 Total error data 3  
3 Mean Square Error 0.27825477933204584
```

Tổng số truy vấn (điểm dữ liệu) cần dự đoán là 200, số điểm dữ liệu bị lỗi là 3.
Giá trị mất mát trung bình là 0.278. Nghĩa là giá trị dự đoán có sai số với giá trị thực trong khoảng ± 0.278


6.13 Tổng kết

Tổng kết phần làm code của nhóm, đạt được một số mục tiêu sau:

- Ý tưởng và phương pháp được thực hiện thành công 100%.
- Thư mục chứa các model con hoàn chỉnh.



Drive của tôi > OSmrData >

Tên 	Chủ sở hữu	Sửa đổi lần cuối	Kích cỡ tệp
 CI3021	tôi	15:48	—
 CI3023	tôi	15:48	—
 CI3025	tôi	15:48	—
 CI3027	tôi	15:48	—
 CI3029	tôi	15:48	—
 CI3031	tôi	15:47	—
 CI3033	tôi	15:47	—
 CI3035	tôi	15:47	—
 CI3037	tôi	15:47	—
 CI3039	tôi	15:47	—
 CI3041	tôi	15:47	—

Hình 13: Thư mục chứa các model con



6.14 Hướng dẫn chạy code

Tại google drive của cá nhân, tạo thư mục tên **OSmrData** và tải lên folder **Models** đính kèm theo đường link sau: [Models](#) như Hình 11

Xây dựng model và lưu model vào drive

Tải lên folder **Models** trên tập dữ liệu excel. Chạy tuần tự bằng Ctrl + F9 tất cả đoạn code của chương trình.

Dự đoán điểm tổng kết dựa vào tập dữ liệu hiện có

Chạy đoạn code tải và cài đặt **pyspark**, thêm thư viện và module. Sau đó chạy tất cả đoạn code trong phần **Bắt đầu dự đoán**



Tài liệu tham khảo

- [1] Apache kafka. <https://kafka.apache.org/>.
- [2] Apache spark™ - unified engine for large-scale data analytics. <https://spark.apache.org/>.
- [3] Moodle - open-source learning platform. <https://moodle.org/>.
- [4] Moodle logstore xapi. https://github.com/xAPI-vle/moodle-logstore_xapi, 2020.