TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÀI TẬP LỚN MÔN XỬ LÝ DỮ LIỆU LỚN**

**Tìm hiểu PySpark-Machine Learning-Dataframe**

*Người hướng dẫn*: **TS LÊ ANH CƯỜNG**

*Người thực hiện*: **LÊ QUANG PHỤC – 51703160**

Lớp **: BIGDATA**

Khoá  **: 21**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2021**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÀI TẬP LỚN MÔN XỬ LÝ DỮ LIỆU LỚN**

**Tìm hiểu PySpark-Machine Learning-Dataframe**

*Người hướng dẫn*: **TS LÊ ANH CƯỜNG**

*Người thực hiện*: **LÊ QUANG PHỤC – 51703160**

Lớp **: BIGDATA**

Khoá  **: 21**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2021**

LỜI CẢM ƠN

Nhóm em xin gửi lời cảm ơn chân thành đến các thầy cô Khoa Công nghệ thông tin trường Đại học Tôn Đức Thắng.

Trong thời gian học tập và rèn luyện tại trường Đại học Tôn Đức Thắng, nhóm em đã nhận được rất nhiều sự giúp đỡ tận tình của quý thầy cô. Thầy cô là người đã hướng dẫn và truyền cảm hứng cho nhóm em trong học tập từ những ngày đầu tiên bước chân vào môi trường đại học. Bên cạnh đó, quý thầy cô còn truyền đạt cho nhóm em rất nhiều kiến thức hay và bổ ích, rèn luyện những kĩ năng hữu ích cho công việc trong tương lai. Với lòng biết ơn sâu sắc từ tận đáy lòng, em xin chân thành gửi lời cảm ơn đến quý thầy cô, đặc biệt là Thầy Lê Anh Cường – người đã trực tiếp dẫn dắt nhóm em trong môn Xử Lý Dữ Liệu Lớn

Trong quá trình làm báo cáo, nhóm em khó có thể tránh khỏi những sai sót, rất mong quý thầy cô có thể đóng góp ý kiến để nhóm em có được thêm nhiều bài học và rút kinh nghiệm cho lần sau hoàn thành tốt hơn.

Lời cuối cùng em xin chân thành cảm ơn và gửi lời chúc tốt đẹp nhất đến quý thầy cô đã tạo cơ hội cho em có thể trau dồi bản thân trong môn học này.

**ĐỒ ÁN ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là sản phẩm bài tập lớn của chúng tôi và được sự hướng dẫn của TS LÊ ANH CƯỜNG. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong bài tập lớn còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

**Nếu phát hiện có bất kỳ sự gian lận nào nhóm xin hoàn toàn chịu trách nhiệm về nội dung bài tập lớn của mình.** Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do nhóm gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày tháng năm*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

*Lê Quang Phục*

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

**Phần xác nhận của GV hướng dẫn**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

**Phần đánh giá của GV chấm bài**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

TÓM TẮT

Trình bày tóm tắt vấn đề nghiên cứu, các hướng tiếp cận, cách giải quyết vấn đề và một số kết quả đạt được, những phát hiện cơ bản trong vòng 1 -2 trang.

MỤC LỤC

[LỜI CẢM ƠN i](#_Toc70881059)

[PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN iv](#_Toc70881060)

[TÓM TẮT v](#_Toc70881061)

[MỤC LỤC 1](#_Toc70881062)

[DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ 3](#_Toc70881063)

[CHƯƠNG 1 – TÌM HIỂU PYSPARK 4](#_Toc70881064)

[1.1 Apache Spark 4](#_Toc70881065)

[1.2 Tính năng Spark 4](#_Toc70881066)

[1.3 Thiết lập Spark với Python (PySpark) 6](#_Toc70881067)

[1.4 PySpark SparkContext và luồng dữ liệu 6](#_Toc70881068)

[1.5 Loại Spark 7](#_Toc70881069)

[1.5.1 Spark properties 7](#_Toc70881070)

[1.5.2 Spark RDD 10](#_Toc70881071)

[CHƯƠNG 2 – MACHINE LEARNING 15](#_Toc70881072)

[1.1 Sự ra đời của Machine Learning 15](#_Toc70881073)

[1.2 Machine Learning là gì? 15](#_Toc70881074)

[1.3 Thuật toán trong machine learning? 16](#_Toc70881075)

[1.4 Ứng dụng Machine Learning 17](#_Toc70881076)

[1.4.1 Tự động phân loại 17](#_Toc70881077)

[1.4.2 Ứng dụng xã hội 18](#_Toc70881078)

[1.4.3 Nhận diện hình ảnh 18](#_Toc70881079)

[CHƯƠNG 3 – DATAFRAME 19](#_Toc70881080)

[3.1 Spark Dataframe là gì? 19](#_Toc70881081)

[3.2 Cách tạo Dataframe 19](#_Toc70881082)

[3.2.1 Tạo từ RDD 19](#_Toc70881083)

[3.2.2 Tạo trực tiếp từ file 21](#_Toc70881084)

[3.3 Thao tác với dataframe 23](#_Toc70881085)

[3.3.1 Thử query bằng SQL 23](#_Toc70881086)

[3.3.2 Tìm kiếm sử dụng filter, select 25](#_Toc70881087)

[3.3.3 Sử dụng group by 26](#_Toc70881088)

**DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT**

**CÁC KÝ HIỆU**

**CÁC CHỮ VIẾT TẮT**

NLP Xử lý ngôn ngữ tự nhiên

DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

**DANH MỤC HÌNH**

[Hình 2.1: Kiến trúc FTP 1](#_Toc387689394)

**DANH MỤC BẢNG**

[Bảng 3.1 Ví dụ cho chèn bảng 1](#_Toc387689363)

CHƯƠNG 1 – TÌM HIỂU PYSPARK

1.1 Apache Spark

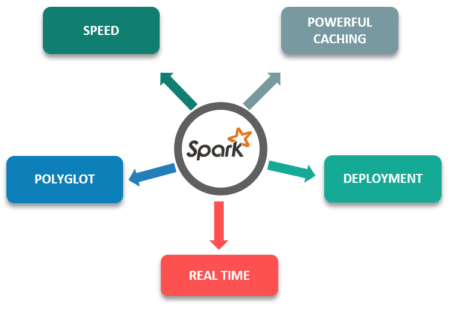
Apache Spark là một trong những khung được sử dụng rộng rãi nhất khi xử lý và làm việc với Big Data và Python là một trong những ngôn ngữ lập trình được sử dụng rộng rãi nhất cho Phân tích dữ liệu, Học máy và hơn thế nữa. Vì vậy, tại sao không sử dụng chúng cùng nhau? Đây là nơi Spark với Python còn được gọi là PySpark đi vào hình ảnh.

Với mức lương trung bình 110.000 đô la mỗi năm cho Nhà phát triển Apache Spark, không còn nghi ngờ gì nữa, Spark được sử dụng rất nhiều trong ngành. Do bộ thư viện phong phú của nó, Python được sử dụng bởi phần lớn các nhà khoa học dữ liệu và chuyên gia phân tích ngày nay. Tích hợp Python với Spark là một món quà lớn cho cộng đồng. Spark được phát triển bằng ngôn ngữ Scala, rất giống với Java. Nó biên dịch mã chương trình thành mã byte cho JVM để xử lý dữ liệu lớn Spark. Để hỗ trợ Spark với Python, cộng đồng Apache Spark đã phát hành.

1.2 Tính năng Spark

Apache Spark là một khung công tác điện toán cụm nguồn mở để xử lý thời gian thực được phát triển bởi Quỹ phần mềm Apache. Spark cung cấp một giao diện để lập trình toàn bộ các cụm với sự song song dữ liệu ngầm và khả năng chịu lỗi.

Dưới đây là một số tính năng của Apache Spark mang lại lợi thế cho các khung công tác khác:



* Tốc độ: Nó nhanh hơn 100 lần so với các khung xử lý dữ liệu quy mô lớn truyền thống.
* Bộ nhớ đệm mạnh mẽ: Lớp lập trình đơn giản cung cấp khả năng lưu trữ bộ nhớ cache và ổ đĩa mạnh mẽ.
* Triển khai: Có thể được triển khai thông qua Mesos, Hadoop thông qua Sợi hoặc trình quản lý cụm riêng của Spark.
* Thời gian thực: Tính toán thời gian thực và độ trễ thấp vì tính toán trong bộ nhớ.
* Polyglot: Đây là một trong những tính năng quan trọng nhất của khung này vì nó có thể được lập trình bằng Scala, Java, Python và R.

Mặc dù Spark được thiết kế bằng Scala, giúp nó nhanh hơn gần 10 lần so với Python, nhưng Scala chỉ nhanh hơn khi số lượng lõi được sử dụng ít hơn . Vì hầu hết các phân tích và quy trình hiện nay đều yêu cầu số lượng lớn lõi, lợi thế về hiệu năng của Scala không nhiều.

Đối với các lập trình viên, Python tương đối dễ học hơn vì các thư viện chuẩn và cú pháp của nó. Hơn nữa, đây là ngôn ngữ được gõ động, có nghĩa là RDD có thể chứa các đối tượng thuộc nhiều loại.

Mặc dù Scala có SparkMLlib nhưng nó không có đủ thư viện và công cụ cho mục đích Machine Learning và NLP . Hơn nữa, Scala thiếu trực quan hóa dữ liệu.

1.3 Thiết lập Spark với Python (PySpark)

Yahoo! sử dụng Apache Spark cho các khả năng Machine Learning của mình để cá nhân hóa tin tức và trang web của mình và cũng cho quảng cáo mục tiêu. Họ sử dụng Spark với Python để tìm hiểu loại tin tức nào mà người dùng quan tâm khi đọc và phân loại các câu chuyện tin tức để tìm ra loại người dùng nào sẽ quan tâm khi đọc từng loại tin tức.

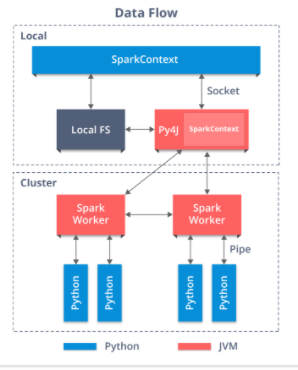
TripAdvisor sử dụng Apache Spark để cung cấp lời khuyên cho hàng triệu khách du lịch bằng cách so sánh hàng trăm trang web để tìm giá khách sạn tốt nhất cho khách hàng của mình. Thời gian để đọc và xử lý các đánh giá của các khách sạn theo định dạng có thể đọc được được thực hiện với sự trợ giúp của Apache Spark.

Một trong những nền tảng thương mại điện tử lớn nhất thế giới, Alibaba , điều hành một số công việc Apache Spark lớn nhất trên thế giới để phân tích hàng trăm petabyte dữ liệu trên nền tảng thương mại điện tử của mình.

1.4 PySpark SparkContext và luồng dữ liệu

Nói về Spark với Python, thư viện Py4j có thể làm việc với RDDs. PySpark Shell liên kết API Python với Spark Core và khởi chạy Bối cảnh Spark. Spark Context là trung tâm của bất kỳ ứng dụng Spark nào.

* Spark Context thiết lập các dịch vụ nội bộ và thiết lập kết nối với môi trường thực thi Spark.
* Đối tượng Spark Context trong chương trình trình điều khiển phối hợp tất cả các quy trình phân tán và cho phép phân bổ tài nguyên.
* Các trình quản lý cụm cung cấp các Executor, đó là các quy trình JVM có logic.
* Các đối tượng Spark Context gửi ứng dụng cho người thi hành.
* Spark Context thực thi các nhiệm vụ trong mỗi người thực thi.



1.5 Loại Spark

1.5.1 Spark properties

Thuộc tính Spark kiểm soát hầu hết các cài đặt ứng dụng và được cấu hình riêng cho từng ứng dụng. Các thuộc tính này có thể được đặt trực tiếp trên SparkConf được chuyển tới SparkContext. SparkConf cho phép định cấu hình một số thuộc tính phổ biến (ví dụ: URL chính và tên ứng dụng), cũng như các cặp khóa-giá trị tùy ý thông qua phương thức set(). Ví dụ: chúng ta có thể khởi tạo một ứng dụng với hai luồng như sau:

val conf = new SparkConf()

.setMaster("local[2]")

.setAppName("CountingSheep")

val sc = new SparkContext(conf)

Lưu ý rằng có thể có nhiều hơn 1 luồng ở chế độ cục bộ và trong các trường hợp như Spark Streaming, có thể yêu cầu nhiều hơn 1 luồng để ngăn chặn bất kỳ loại vấn đề starvation.

Các thuộc tính chỉ định một số khoảng thời gian nên được cấu hình với một đơn vị thời gian. Định dạng sau được chấp nhận:

25ms (milliseconds)

5s (seconds)

10m or 10min (minutes)

3h (hours)

5d (days)

1y (years)

Các thuộc tính chỉ định kích thước byte nên được cấu hình với đơn vị kích thước. Định dạng sau được chấp nhận:

1b (bytes)

1k or 1kb (kibibytes = 1024 bytes)

1m or 1mb (mebibytes = 1024 kibibytes)

1g or 1gb (gibibytes = 1024 mebibytes)

1t or 1tb (tebibytes = 1024 gibibytes)

1p or 1pb (pebibytes = 1024 tebibytes)

Trong khi các số không có đơn vị thường được hiểu là byte, một số ít được hiểu là KiB hoặc MiB. Xem tài liệu về các thuộc tính cấu hình riêng lẻ. Việc chỉ định đơn vị là mong muốn nếu có thể.

Dynamically Loading Spark Properties

Trong một số trường hợp, bạn có thể muốn tránh mã hóa cứng các cấu hình nhất định trong SparkConf. Ví dụ: nếu bạn muốn chạy cùng một ứng dụng với các bản chính khác nhau hoặc số lượng bộ nhớ khác nhau. Spark cho phép bạn chỉ cần tạo một conf trống:

val sc = new SparkContext(new SparkConf())

Sau đó, bạn có thể cung cấp các giá trị cấu hình trong thời gian chạy:

./bin/spark-submit --name "My app" --master local[4] --conf spark.eventLog.enabled=false

--conf "spark.executor.extraJavaOptions=-XX:+PrintGCDetails -XX:+PrintGCTimeStamps" myApp.jar

Spark shell và công cụ spark-submit hỗ trợ hai cách để tải cấu hình động. Đầu tiên là các tùy chọn dòng lệnh, chẳng hạn như --master, như hình trên. spark-submit có thể chấp nhận bất kỳ thuộc tính Spark nào sử dụng --conf / -c flag , nhưng sử dụng flag đặc biệt cho các thuộc tính đóng một vai trò trong việc khởi chạy ứng dụng Spark. Chạy ./bin/spark-submit --help sẽ hiển thị toàn bộ danh sách các tùy chọn này.

bin / spark-submit cũng sẽ đọc các tùy chọn cấu hình từ conf / spark-defaults.conf, trong đó mỗi dòng bao gồm một khóa và một giá trị được phân tách bằng khoảng trắng. Ví dụ:

spark.master spark://5.6.7.8:7077

spark.executor.memory 4g

spark.eventLog.enabled true

spark.serializer org.apache.spark.serializer.KryoSerializer

Mọi giá trị được chỉ định dưới dạng flag hoặc trong tệp thuộc tính sẽ được chuyển đến ứng dụng và được hợp nhất với những giá trị được chỉ định thông qua SparkConf. Các thuộc tính được đặt trực tiếp trên SparkConf được ưu tiên cao nhất, sau đó các flag được chuyển đến spark-submit hoặc spark-shell, sau đó là các tùy chọn trong tệp spark-defaults.conf. Một vài khóa cấu hình đã được đổi tên kể từ các phiên bản Spark trước đó; trong những trường hợp như vậy, các tên khóa cũ hơn vẫn được chấp nhận, nhưng được ưu tiên thấp hơn bất kỳ trường hợp nào của khóa mới hơn.

Các thuộc tính của Spark chủ yếu có thể được chia thành hai loại: một là liên quan đến triển khai, như “spark.driver.memory”, “spark.executor.instances”, loại thuộc tính này có thể không bị ảnh hưởng khi thiết lập lập trình thông qua SparkConf trong thời gian chạy, hoặc hành vi tùy thuộc vào trình quản lý cụm và chế độ triển khai bạn chọn, vì vậy bạn nên đặt thông qua tệp cấu hình hoặc tùy chọn dòng lệnh spark-submit; một loại khác chủ yếu liên quan đến kiểm soát thời gian chạy Spark, như “spark.task.maxFailures”, loại thuộc tính này có thể được đặt theo một trong hai cách.

Viewing Spark Properties

Giao diện người dùng web ứng dụng tại http: // <driver>: 4040 liệt kê các thuộc tính Spark trong tab "Môi trường". Đây là một nơi hữu ích để kiểm tra để đảm bảo rằng các thuộc tính của bạn đã được đặt chính xác. Lưu ý rằng chỉ các giá trị được chỉ định rõ ràng thông qua spark-defaults.conf, SparkConf hoặc dòng lệnh mới xuất hiện. Đối với tất cả các thuộc tính cấu hình khác, bạn có thể giả sử giá trị mặc định được sử dụng.

1.5.2 Spark RDD

Resilient Distributed Datasets (RDD) là một cấu trúc dữ liệu cơ bản của Spark. Nó là một tập hợp bất biến phân tán của một đối tượng. Mỗi dataset trong RDD được chia ra thành nhiều phần vùng logical. Có thể được tính toán trên các node khác nhau của một cụm máy chủ (cluster).

RDDs có thể chứa bất kỳ kiểu dữ liệu nào của Python, Java, hoặc đối tượng Scala, bao gồm các kiểu dữ liệu do người dùng định nghĩa. Thông thường, RDD chỉ cho phép đọc, phân mục tập hợp của các bản ghi. RDDs có thể được tạo ra qua điều khiển xác định trên dữ liệu trong bộ nhớ hoặc RDDs, RDD là một tập hợp có khả năng chịu lỗi mỗi thành phần có thể được tính toán song song.

Có hai cách để tạo RDDs:

•Tạo từ một tập hợp dữ liệu có sẵn trong ngôn ngữ sử dụng như Java, Python, Scala.

•Lấy từ dataset hệ thống lưu trữ bên ngoài như HDFS, Hbase hoặc các cơ sở dữ liệu quan hệ.

Thực thi trên Spark RDD

- Iterative Operation trên Spark RDD:



- Interactive Operations trên Spark RDD:



Các loại RDD



• Các RDD biểu diễn một tập hợp cố định, đã được phân vùng các record để có thể xử lý song song.

• Các record trong RDD có thể là đối tượng Java, Scale hay Python tùy lập trình viên chọn. Không giống như DataFrame, mỗi record của DataFrame phải là một dòng có cấu trúc chứa các field đã được định nghĩa sẵn.

• RDD đã từng là API chính được sử dụng trong series Spark 1.x và vẫn có thể sử dụng trong version 2.X nhưng không còn được dùng thường xuyên nữa.

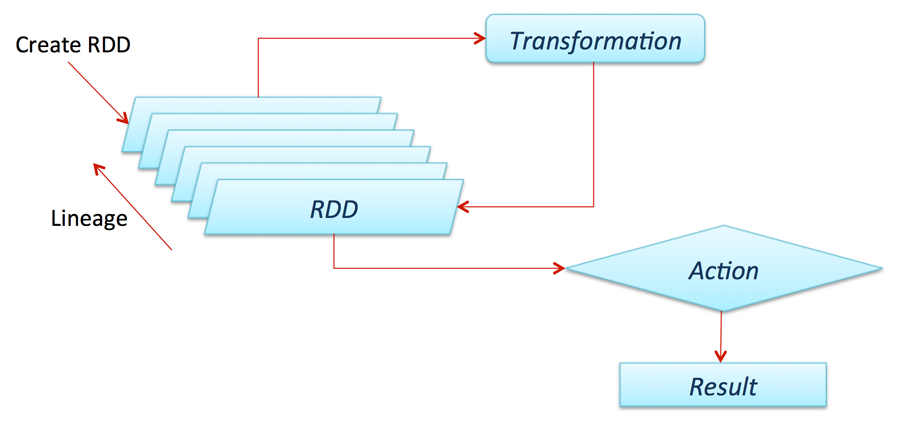
• RDD API có thể được sử dụng trong Python, Scala hay Java:

o Scala và Java: Perfomance tương đương trên hầu hết mọi phần. (Chi phí lớn nhất là khi xử lý các raw object)

o Python: Mất một lượng performance, chủ yếu là cho việc serialization giữa tiến trình Python và JVM

Các transformation và action với RDD

RDD cung cấp các transformation và action hoạt động giống như DataFrame lẫn DataSets. Transformation xử lý các thao tác lazily và Action xử lý thao tác cần xử lý tức thời.



-Một số transformation:

Nhiều phiên bản transformation của RDD có thể hoạt động trên các Structured API, transformation xử lý lazily, tức là chỉ giúp dựng execution plans, dữ liệu chỉ được truy xuất thực sự khi thực hiện action

• distinct: loại bỏ trùng lắp trong RDD

• filter: tương đương với việc sử dụng where trong SQL – tìm các record trong RDD xem những phần tử nào thỏa điều kiện. Có thể cung cấp một hàm phức tạp sử dụng để filter các record cần thiết – Như trong Python, ta có thể sử dụng hàm lambda để truyền vào filter

• map: thực hiện một công việc nào đó trên toàn bộ RDD. Trong Python sử dụng lambda với từng phần tử để truyền vào map

• flatMap: cung cấp một hàm đơn giản hơn hàm map. Yêu cầu output của map phải là một structure có thể lặp và mở rộng được.

• sortBy: mô tả một hàm để trích xuất dữ liệu từ các object của RDD và thực hiện sort được từ đó.

• randomSplit: nhận một mảng trọng số và tạo một random seed, tách các RDD thành một mảng các RDD có số lượng chia theo trọng số.

- Một số action:

Action thực thi ngay các transformation đã được thiết lập để thu thập dữ liệu về driver để xử lý hoặc ghi dữ liệu xuống các công cụ lưu trữ.

• reduce: thực hiện hàm reduce trên RDD để thu về 1 giá trị duy nhất

• count: đếm số dòng trong RDD

• countApprox: phiên bản đếm xấp xỉ của count, nhưng phải cung cấp timeout vì có thể không nhận được kết quả.

• countByValue: đếm số giá trị của RDD

chỉ sử dụng nếu map kết quả nhỏ vì tất cả dữ liệu sẽ được load lên memory của driver để tính toán

chỉ nên sử dụng trong tình huống số dòng nhỏ và số lượng item khác nhau cũng nhỏ.

• countApproxDistinct: đếm xấp xỉ các giá trị khác nhau

• countByValueApprox: đếm xấp xỉ các giá trị

• first: lấy giá trị đầu tiên của dataset

• max và min: lần lượt lấy giá trị lớn nhất và nhỏ nhất của dataset

• take và các method tương tự: lấy một lượng giá trị từ trong RDD. take sẽ trước hết scan qua một partition và sử dụng kết quả để dự đoán số lượng partition cần phải lấy thêm để thỏa mãn số lượng lấy.

• top và takeOrdered: top sẽ hiệu quả hơn takeOrdered vì top lấy các giá trị đầu tiên được sắp xếp ngầm trong RDD.

• takeSamples: lấy một lượng giá trị ngẫu nhiên trong RDD

CHƯƠNG 2 – MACHINE LEARNING

*Tổng quan*: Machine learning còn gọi là máy học là xu hướng của tương lai, các ứng dụng của nó đã phục vụ cho cuộc sống ngày càng tân tiến của nhân loại.

1.1 Sự ra đời của Machine Learning

Một trong những khác biệt chính giữa con người và máy tính là con người học hỏi từ những kinh nghiệm trong quá khứ, nhưng với máy tính hoặc máy móc cần được phải được thực hiện theo một quy trình có sẵn. Máy tính là những máy logic nghiêm ngặt với ý nghĩa thông thường.

Điều đó có nghĩa là nếu chúng ta muốn máy làm điều gì đó, chúng ta phải cung cấp cho nó những quy trình và các hướng dẫn chi tiết, từng bước về chính xác những việc cần làm.

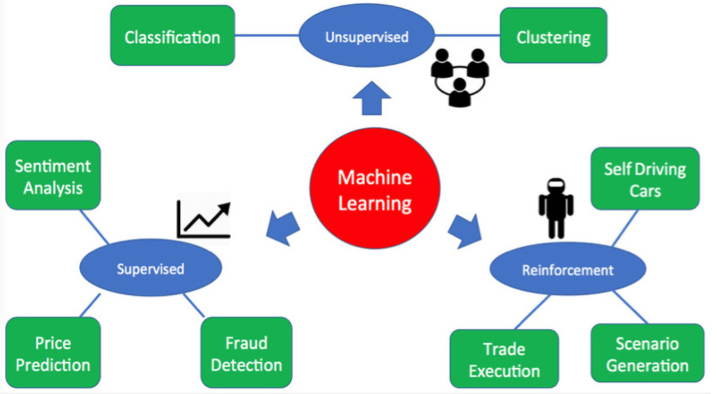
Vì vậy, con người đã viết nên các kịch bản và lập trình để máy tính làm theo các hướng dẫn và có khả năng tự học hỏi. Đó là cái cách mà Machine Learning ra đời. Khái niệm máy học chính xác là việc máy tính học hỏi từ dữ liệu trong quá khứ và rút kinh nghiệm qua thời gian.

1.2 Machine Learning là gì?

Học máy là một ứng dụng của trí tuệ nhân tạo (AI) cung cấp cho các hệ thống khả năng tự động học hỏi và cải thiện từ kinh nghiệm mà không cần lập trình rõ ràng. Học máy tập trung vào việc phát triển các chương trình máy tính có thể truy cập dữ liệu và sử dụng nó để tự học.

Quá trình học bắt đầu bằng các quan sát hoặc dữ liệu. Ví dụ, để tìm kiếm các mẫu trong dữ liệu và đưa ra quyết định tốt hơn trong tương lai dựa trên các ví dụ mà chúng tôi cung cấp. Mục đích chính là cho phép các máy tính tự động học mà không cần sự can thiệp hay trợ giúp của con người và điều chỉnh các hành động tương ứng.

1.3 Thuật toán trong machine learning?



Các thuật toán học máy thường được phân loại là giám sát hoặc không giám sát.

**Các thuật toán học máy được giám sát**

Nó có thể áp dụng những gì đã được học trong quá khứ vào dữ liệu mới bằng cách sử dụng các ví dụ được gắn nhãn để dự đoán các sự kiện trong tương lai. Bắt đầu từ việc phân tích một tập dữ liệu huấn luyện đã biết, thuật toán học tạo ra một hàm được suy ra để đưa ra dự đoán về các giá trị đầu ra.

**Các thuật toán học máy không giám sát**

Ngược lại, thuật toán học máy không giám sát được sử dụng khi thông tin được sử dụng để đào tạo không được phân loại cũng không được dán nhãn. Nghiên cứu học tập không giám sát làm thế nào các hệ thống có thể suy ra một chức năng để mô tả một cấu trúc ẩn từ dữ liệu không được gắn nhãn.

**Các thuật toán Machine Learning bán giám sát**

Các thuật toán học máy được giám sát bán nằm ở đâu đó giữa học tập có giám sát và không giám sát, vì chúng sử dụng cả dữ liệu được gắn nhãn và không nhãn cho đào tạo - thường là một lượng nhỏ dữ liệu được gắn nhãn và một lượng lớn dữ liệu không được gắn nhãn. Các hệ thống sử dụng phương pháp này có thể cải thiện đáng kể độ chính xác trong học tập.

**Thuật toán học máy gia cố**

Các thuật toán học máy gia cố là một phương pháp học tương tác với môi trường của nó bằng cách tạo ra các hành động và phát hiện ra các lỗi hoặc manh mối. Thử nghiệm và tìm kiếm lỗi và manh mối

Phương pháp này cho phép máy móc, máy tính với phần mềm tự động xác định hành vi lý tưởng trong một bối cảnh cụ thể để tối đa hóa hiệu suất của nó.

1.4 Ứng dụng Machine Learning

1.4.1 Tự động phân loại

Phân loại tin tức là một ứng dụng điểm chuẩn khác của phương pháp học máy.Vận dụng như thế nào? Như một vấn đề thực tế là bây giờ khối lượng thông tin đã tăng lên rất nhiều trên web. Tuy nhiên, mỗi người có sở thích hoặc lựa chọn cá nhân của mình. Vì vậy, để chọn hoặc thu thập một phần thông tin phù hợp trở thành một thách thức đối với người dùng từ vô số nội dung trên trang web.

Phân loại các danh mục một cách rõ ràng, dễ điều hướng giúp cho các khách hàng mục tiêu chắc chắn sẽ tăng khả năng truy cập các trang tin tức. Hơn nữa, độc giả hoặc người dùng có thể tìm kiếm tin tức cụ thể một cách hiệu quả và nhanh chóng.

Có một số phương pháp học máy trong mục đích này, tức là, máy vectơ hỗ trợ, **naive Bayes, k-nearest neighbor, v.v.**

1.4.2 Ứng dụng xã hội

Học máy đang được sử dụng trong một loạt các ứng dụng ngày nay. Một trong những ví dụ nổi tiếng nhất là Facebook News Feed. Nguồn cấp tin tức sử dụng học máy để cá nhân hóa từng nguồn cấp dữ liệu thành viên.

Nếu một thành viên thường xuyên dừng lại để đọc hoặc thích một bài đăng của một người bạn cụ thể, News Feed sẽ bắt đầu hiển thị nhiều hơn về hoạt động của người bạn đó trước đó trong nguồn cấp dữ liệu.

Đằng sau hệ thống ấy, phần mềm sử dụng phân tích thống kê và phân tích dự đoán để xác định các mẫu trong dữ liệu người dùng và sử dụng các mẫu đó để điền vào News Feed. Nếu thành viên không còn dừng lại để đọc, thích hoặc bình luận trên các bài đăng của bạn bè, dữ liệu mới đó sẽ được bao gồm trong tập dữ liệu và News Feed sẽ điều chỉnh tương ứng.

Không chỉ riêng facebook, ta có thể bắt gặp những tính năng tương tự đó qua các mạng xã hội khác như google, instagram,....

1.4.3 Nhận diện hình ảnh

Nhận dạng hình ảnh là một trong những ví dụ về máy học và trí tuệ nhân tạo phổ biến nhất. Về cơ bản, nó là một cách tiếp cận để xác định và phát hiện các đặc trưng của một đối tượng trong hình ảnh kỹ thuật số. Hơn nữa, kỹ thuật này có thể được sử dụng để phân tích sâu hơn, chẳng hạn như nhận dạng mẫu, nhận diện hình khuôn, nhận dạng khuôn mặt, nhận dạng ký tự quang học và nhiều hơn nữa,...

CHƯƠNG 3 – DATAFRAME

3.1 Spark Dataframe là gì?

Là phiên bản Spark 1.3. Nó là một tập hợp phân phối dữ liệu được sắp xếp vào các cột được đặt tên. Khái niệm khôn ngoan, nó bằng với bảng trong cơ sở dữ liệu quan hệ hoặc khung dữ liệu trong R / Python. Chúng tôi có thể tạo DataFrame bằng cách sử dụng:

• Tệp dữ liệu có cấu trúc

• Bàn trong tổ ong

• Cơ sở dữ liệu bên ngoài

• Sử dụng RDD hiện có

• Giống như viết SQL, đầy đủ chức năng như select, where ... đặc biệt là join với các DataFrame khác.

• Sử dụng các method như filter, select để trích xuất dữ liệu theo cột, hàng.

• Xử gọn các loại data như Log ... với groupBy → agg

• Thêm 1 cột dễ dàng với UDF(User Defined Function)

• Giống như SQL, Spark DataFrame đã hỗ trợ Pivot (Spark 1.6 trở lên) rất hữu ích cho việc lập bảng biểu, báo cáo.

3.2 Cách tạo Dataframe

3.2.1 Tạo từ RDD

Nếu bạn đã có RDD với tên column và type tương ứng (TimestampType, IntegerType, StringType)thì bạn có thể dễ dàng tạo DataFrame bằng

**sqlContext.createDataFrame(my\_rdd, my\_schema)**

Với printSchema(), dtypes sẽ in thông tin của schema và count() trả về số record

Và nếu chỉ muốn in n record đầu tiên thì sử dụng show(n)

fields = [StructField("access\_time", TimestampType(), True), StructField("userID", IntegerType(), True), StructField("campaignID", StringType(), True)]

schema = StructType(fields)

whole\_log\_df = sqlContext.createDataFrame(whole\_log, schema)

print whole\_log\_df.count()

print whole\_log\_df.printSchema()

print whole\_log\_df.dtypes

print whole\_log\_df.show(5)

#327430

#root

# |-- access\_time: timestamp (nullable = true)

# |-- userID: integer (nullable = true)

# |-- campaignID: string (nullable = true)#

#[('access\_time', 'timestamp'), ('userID', 'int'), ('campaignID', 'string')]#

#+--------------------+------+-----------+

#| access\_time|userID| campaignID|

#+--------------------+------+-----------+

#|2015-04-27 20:40:...|144012|Campaign077|

#|2015-04-27 00:27:...| 24485|Campaign063|

#|2015-04-27 00:28:...| 24485|Campaign063|

#|2015-04-27 00:33:...| 24485|Campaign038|

#|2015-04-27 01:00:...| 24485|Campaign063|

3.2.2 Tạo trực tiếp từ file

from pyspark.shell import spark

from pyspark.sql.types import \*

# Định nghĩa Schema

struct = StructType([

StructField('a', StringType(), False),

StructField('b', StringType(), False),

StructField('c', StringType(), False)])

# Tạo DataFrame từ file CSV

df\_data = spark.read.csv('click\_data\_sample', struct)

Ngoài cách trên còn rất nhiều cách khác, có thể kể đến là gọi cổ 1 thằng thuộc họ nhà Spark Package tên là spark-csv ra xài, hỗ trợ nhiều method hữu ích, dễ chơi, dễ trúng thưởng hơn.

Lưu ý là nếu không định nghĩa schema thì tất cả các column sẽ có kiểu string. Nhưng nếu dùng thêm thằng em inferSchema thì mọi chuyện sẽ êm xuôi.Không tin đơn giản như vậy ư, bạn thử như code dưới xem.

whole\_log\_df\_2 = sqlContext.read.format("com.databricks.spark.csv").option("header", "true").load("/user/hadoop/click\_data\_sample.csv")

print whole\_log\_df\_2.printSchema()

print whole\_log\_df\_2.show(5)

#root

# |-- click.at: string (nullable = true)

# |-- user.id: string (nullable = true)

# |-- campaign.id: string (nullable = true)#

#+-------------------+-------+-----------+

#| click.at|user.id|campaign.id|

#+-------------------+-------+-----------+

#|2015-04-27 20:40:40| 144012|Campaign077|

#|2015-04-27 00:27:55| 24485|Campaign063|

#|2015-04-27 00:28:13| 24485|Campaign063|

#|2015-04-27 00:33:42| 24485|Campaign038|

#|2015-04-27 01:00:04| 24485|Campaign063|

whole\_log\_df\_3 = sqlContext.read.format("com.databricks.spark.csv").option("header", "true").option("inferSchema", "true").load("click\_data\_sample.csv")

print whole\_log\_df\_3.printSchema()

#root

# |-- click.at: timestamp (nullable = true)

# |-- user.id: integer (nullable = true)

# |-- campaign.id: string (nullable = true)

3.3 Thao tác với dataframe

3.3.1 Thử query bằng SQL

Bằng cách sử dụng registerTempTable, bạn sẽ có một table được tham chiếu đến Dataframe đó, bạn có thể sử dụng tên table này để viết query SQL. Nếu bạn sử dụng sqlContext.sql('query SQL') thì giá trị trả về cũng là Dataframe.

Có 1 lưu ý là: Bạn cũng có thể viết subquery nhưng subquery cần được gán Alias, nếu không toạch lô (Syntax error) đấy.

#SQL query

whole\_log\_df.registerTempTable("whole\_log\_table")

print sqlContext.sql(" SELECT \* FROM whole\_log\_table where campaignID == 'Campaign047' ").count()

#18081

print sqlContext.sql(" SELECT \* FROM whole\_log\_table where campaignID == 'Campaign047' ").show(5)

#+--------------------+------+-----------+

#| access\_time|userID| campaignID|

#+--------------------+------+-----------+

#|2015-04-27 05:26:...| 14151|Campaign047|

#|2015-04-27 05:26:...| 14151|Campaign047|

#|2015-04-27 05:26:...| 14151|Campaign047|

#|2015-04-27 05:27:...| 14151|Campaign047|

#|2015-04-27 05:28:...| 14151|Campaign047|

#+--------------------+------+-----------+

#Trường hợp thêm biến số vào trong câu SQL

for count in range(1, 3):

print "Campaign00" + str(count)

print sqlContext.sql("SELECT count(\*) as access\_num FROM whole\_log\_table where campaignID == 'Campaign00" + str(count) + "'").show()

#Campaign001

#+----------+

#|access\_num|

#+----------+

#| 2407|

#+----------+#

#Campaign002

#+----------+

#|access\_num|

#+----------+

#| 1674|

#+----------+

#Trường hợp Sub Query：

print sqlContext.sql("SELECT count(\*) as first\_count FROM (SELECT userID, min(access\_time) as first\_access\_date FROM whole\_log\_table GROUP BY userID) subquery\_alias WHERE first\_access\_date < '2015-04-28'").show(5)

#+------------+

#|first\_count |

#+------------+

#| 20480|

#+------------+

3.3.2 Tìm kiếm sử dụng filter, select

Đối với DataFrame , tìm kiếm kèm điều kiện rất đơn giản. Giống với câu query ở trên nhưng filter, select dễ dàng hơn rất nhiều. Vậy filter và select khác nhau thế nào?

Cùng là để tìm kiếm nhưng filter trả về những row thoả mãn điều kiện, trong đó select lấy dữ liệu theo column.

Ví dụ filter

print whole\_log\_df.filter(whole\_log\_df["access\_time"] < "2015-04-28").count()

#41434

print whole\_log\_df.filter(whole\_log\_df["access\_time"] > "2015-05-01").show(3)

#+--------------------+------+-----------+

#| access\_time|userID| campaignID|

#+--------------------+------+-----------+

#|2015-05-01 22:11:...|114157|Campaign002|

#|2015-05-01 23:36:...| 93708|Campaign055|

#|2015-05-01 22:51:...| 57798|Campaign046|

#+--------------------+------+-----------+

#Ví dụ select

print whole\_log\_df.select("access\_time", "userID").show(3)

#+--------------------+------+

#| access\_time|userID|

#+--------------------+------+

#|2015-04-27 20:40:...|144012|

#|2015-04-27 00:27:...| 24485|

#|2015-04-27 00:28:...| 24485|

#+--------------------+------+

3.3.3 Sử dụng group by

groupBy có chức năng giống với reduceByKey của RDD, nhưng nó còn cung cấp 1 rổ method. Ở đây mình sẽ run thử code count, agg.

groupBy→count

Ví dụ sau sẽ lấy key là campaignID và tiến hành groupBy. Sau đó dùng count() để lấy số record ứng với mỗi key.

print whole\_log\_df.groupBy("campaignID").count().sort("count", ascending=False).show(5)

#+-----------+-----+

#| campaignID|count|

#+-----------+-----+

#|Campaign116|22193|

#|Campaign027|19206|

#|Campaign047|18081|

#|Campaign107|13295|

#|Campaign131| 9068|

#+-----------+-----+

print whole\_log\_df.groupBy("campaignID", "userID").count().sort("count", ascending=False).show(5)

#+-----------+------+-----+

#| campaignID|userID|count|

#+-----------+------+-----+

#|Campaign047| 30292| 633|

#|Campaign086|107624| 623|

#|Campaign047|121150| 517|

#|Campaign086| 22975| 491|

#|Campaign122| 90714| 431|

#+-----------+------+-----+

**TÀI LIỆU THAM KHẢO**

**Tiếng Việt**

**Tiếng Anh**

**PHỤ LỤC**