**TÀI LIỆU HƯỚNG DẪN THỰC HÀNH**

**Tên bài thực hành:**

**Cài đặt và triển khai Spark MLlib**

**10/2023**

**MỤC LỤC**

[1. CƠ SỞ LÝ THUYẾT 2](#_Toc147408149)

[1.1. Giới thiệu chung 2](#_Toc147408150)

[1.2. Nguyên lý hoạt động 9](#_Toc147408151)

[1.3. Nguyên lý dịch & gỡ rối 10](#_Toc147408152)

[2. BÀI THÍ NGHIỆM THỰC HÀNH 12](#_Toc147408153)

[2.1. Mục đích - Yêu cầu 12](#_Toc147408154)

[**2.1.1. Mục đích 12**](#_Toc147408155)

[**2.1.2. Yêu cầu: 12**](#_Toc147408156)

[**2.1.3. Thời gian thực hiện 12**](#_Toc147408157)

[2.2. Chuẩn bị 12](#_Toc147408158)

[**2.2.1. Danh mục thiết bị thực hành 12**](#_Toc147408159)

[2.3. Nội dung 13](#_Toc147408160)

[**2.3.1. Các bước thực hiện 13**](#_Toc147408161)

[**2.3.2. Kết quả thực hiện 25**](#_Toc147408162)

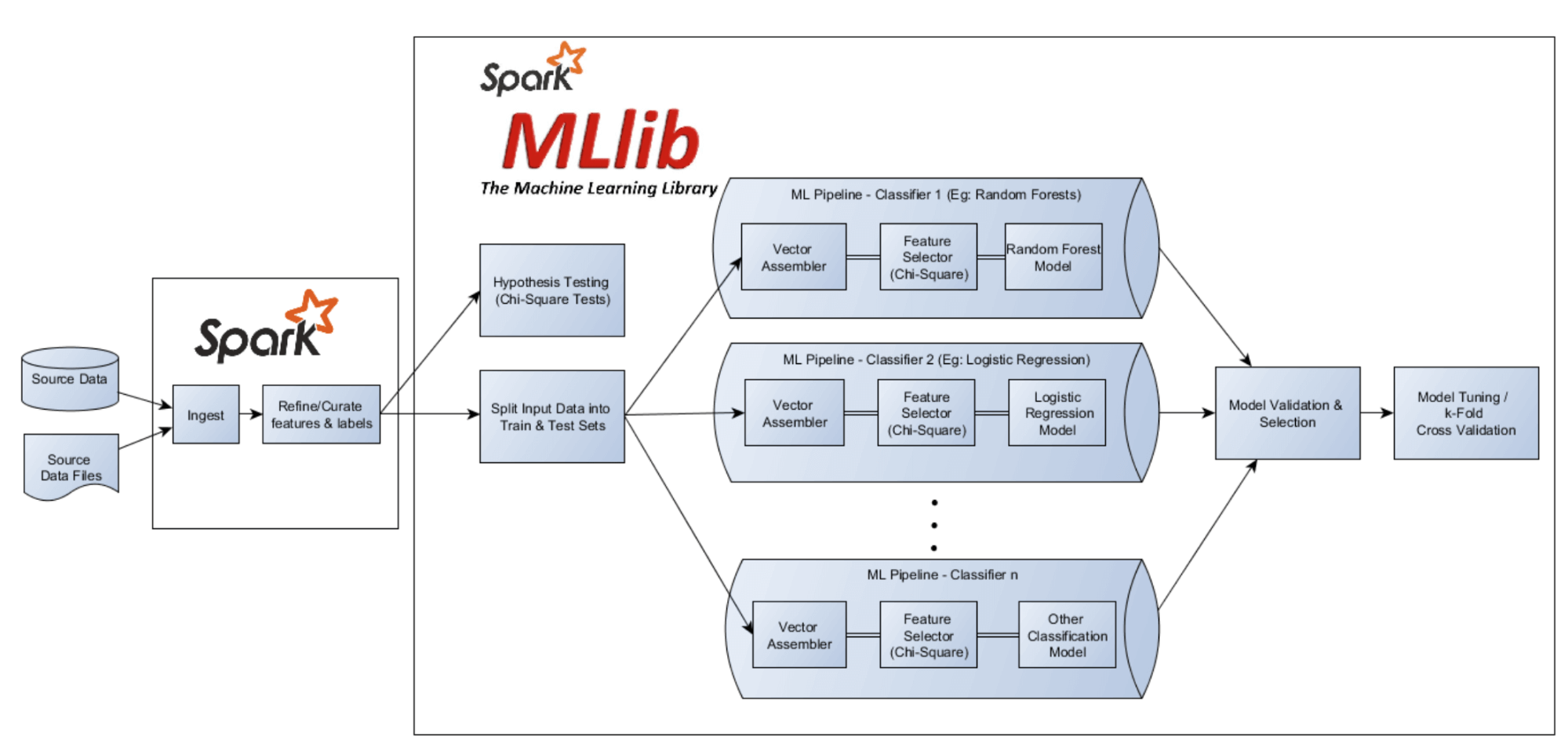
[TÀI LIỆU HƯỚNG DẪN THỰC HÀNH 25](#_Toc147408163)

[PHỤ LỤC 26](#_Toc147408164)

# 1. CƠ SỞ LÝ THUYẾT

## 1.1. Giới thiệu chung

***Làm quen với kiến trúc***



Hình 1 Giới thiệu Spark MLlib

**+** Khái niệm***:***

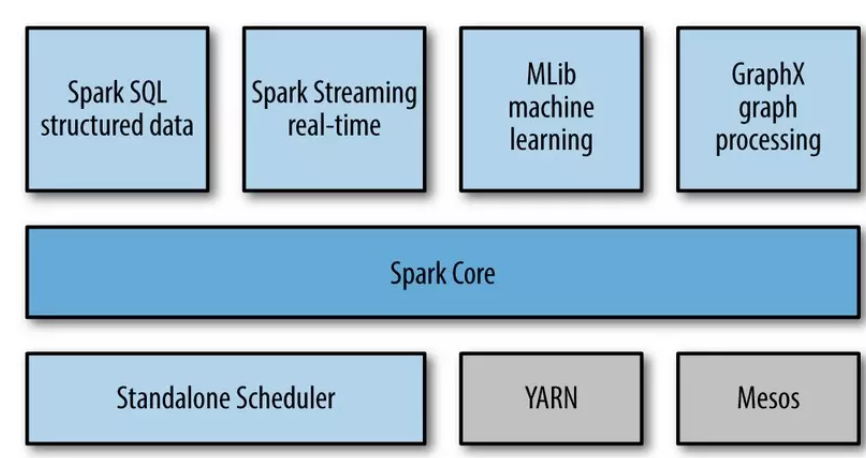
* Apache Spark: Apache Spark là một framework xử lý dữ liệu phân tán mã nguồn mở được phát triển bởi Apache Software Foundation. Nó cung cấp khả năng xử lý và phân tích dữ liệu lớn một cách hiệu quả, hỗ trợ xử lý batch, xử lý dự đoán thời gian thực, và xử lý dữ liệu trực tiếp từ nhiều nguồn khác nhau. Spark sử dụng cơ sở dữ liệu phân tán in-memory để tăng tốc quá trình xử lý dữ liệu.
* Python - Truyền phát có cấu trúc Spark: Đây là một cách sử dụng Python để xây dựng ứng dụng dựa trên Apache Spark để xử lý và phân tích dữ liệu có cấu trúc. Python là một ngôn ngữ lập trình phổ biến trong cộng đồng khoa học dữ liệu và Spark cung cấp API cho Python để xử lý dữ liệu một cách dễ dàng.

**+** Đặc điểm

* Python:
* Python là một ngôn ngữ lập trình mã nguồn mở, dễ đọc và dễ hiểu.
* Đa năng: Python được sử dụng cho nhiều mục đích, bao gồm phát triển ứng dụng web, khoa học dữ liệu, máy học, và nhiều lĩnh vực khác.
* Cộng đồng lớn: Python có cộng đồng phát triển và hỗ trợ lớn, có nhiều thư viện và framework sẵn sàng.
* API (Application Programming Interface):
* Giao diện lập trình ứng dụng: API là tập hợp các quy tắc và hướng dẫn cho phép các ứng dụng giao tiếp với nhau.
* Hỗ trợ tích hợp: API cho phép các ứng dụng và dịch vụ khác nhau tương tác và chia sẻ dữ liệu một cách dễ dàng.
* Apache Spark:
* Xử lý dữ liệu phân tán: Spark là một framework xử lý dữ liệu phân tán, có khả năng xử lý dữ liệu nhanh chóng và hiệu quả.
* Hỗ trợ đa ngôn ngữ: Spark hỗ trợ nhiều ngôn ngữ lập trình và cung cấp thư viện mạnh mẽ cho xử lý dữ liệu.

+ Cấu trúc

* Apache Spark



Hình 2 kiến trúc của Apache Spark

* Thành phần chung của Spark là Spark Core: cung cấp những chức năng cơ bản nhất của Spark như lập lịch cho các tác vụ, quản lý bộ nhớ, fault recovery, tương tác với các hệ thống lưu trữ…Đặc biệt, Spark Core cung cấp API để định nghĩa RDD (Resilient Distributed DataSet) là tập hợp của các item được phân tán trên các node của cluster và có thể được xử lý song song.
* Spark có thể chạy trên nhiều loại Cluster Managers như Hadoop YARN, Apache Mesos hoặc trên chính cluster manager được cung cấp bởi Spark được gọi là Standalone Scheduler.
* Spark SQL cho phép truy vấn dữ liệu cấu trúc qua các câu lệnh SQL. Spark SQL có thể thao tác với nhiều nguồn dữ liệu như Hive tables, Parquet, và JSON.
* MLlib Cung cấp nhiều thuật toán của học máy như: classification, regression, clustering, collaborative filtering…
* GraphX là thư viện để xử lý đồ thị.

• Spark MLlib là mô-đun Spark để xử lý các bài toán học máy. MLlib là thư viện máy học (ML) của Spark. Mục tiêu của nó là làm cho các công cụ học máy có thể mở rộng dễ dàng trên thực tế.

## *1.2. Nguyên lý hoạt động*

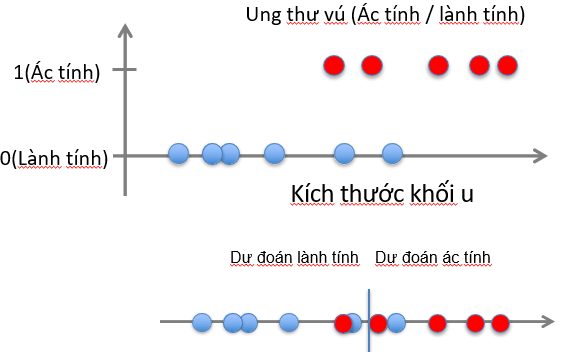
***Nguyên lý hoạt động của quy trình này có thể được mô tả như sau:***

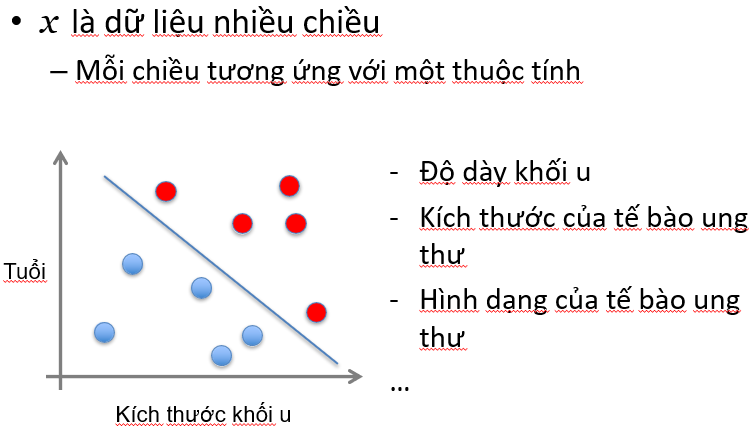
* Khi thực thi, MLlib hỗ trợ một số ngôn ngữ và cung cấp API cấp cao nhằm tận dụng hệ sinh thái phong phú của Spark để đơn giản hóa việc phát triển các quy trình máy học từ đầu đến cuối.
* Thuật toán ML: các thuật toán học tập phổ biến như phân loại, hồi quy, phân cụm và lọc cộng tác
* Đặc trưng hóa: trích xuất đặc trưng, chuyển đổi dữ liệu, giảm chiều kích thước và lựa chọn đặc trưng
* Đường ống: công cụ để xây dựng, đánh giá và điều chỉnh Đường ống ML
* Lưu trữ: lưu và tải các thuật toán, mô hình và đường ống
* Các tiện ích: đại số tuyến tính, thống kê, xử lý dữ liệu, v.v.
* SparkMLlib có thể chạy trong spark-shell, pysparkshell hoặc sparkRshell.
* Thuật toán phân loại:

Cho trước (x1, y1), (x2, y2), ..., (xn, yn)

Học 1 hàm số f(x) để dự đoán y khi cho dữ liệu x

y có giá trị rời rạc == phân loại



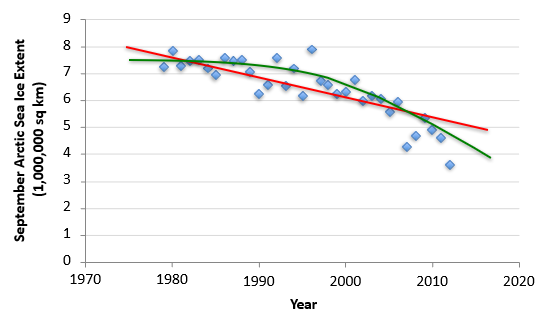


* Thuật toán hồi quy:

Cho trước (x1, y1), (x2, y2), ..., (xn, yn)

Học 1 hàm số f(x) để dự đoán y khi cho dữ liệu x

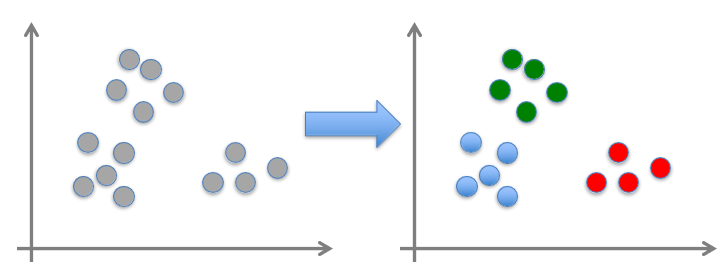
y có giá trị là số thực == hồi quy (y: tiêu dùng khách hàng, giá nhà, giá chứng khoán, doanh thu, chi phí, nhiệt độ,…)



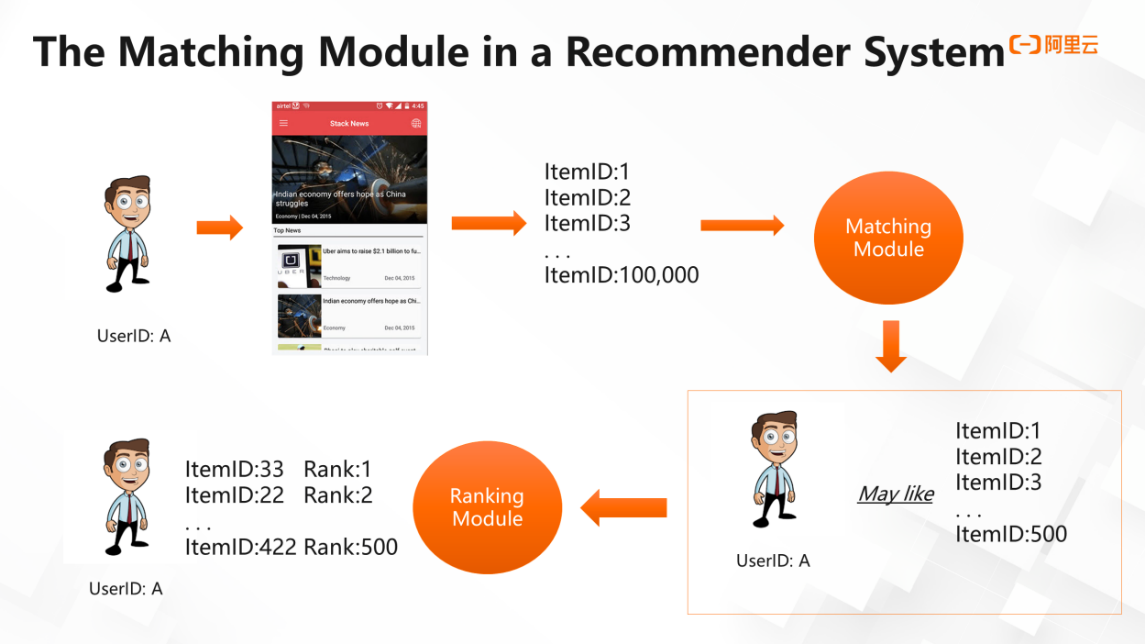
* Thuật toán phân cụm:

Cho x1, x2, ..., xn (không có nhãn)

Đầu ra: cấu trúc ẩn đằng sau dữ liệu x



* Hệ thống gợi ý:



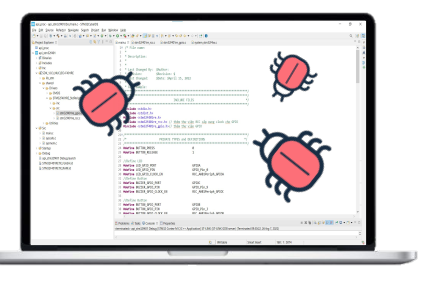
Là một dạng của hệ hỗ trợ ra quyết định, cung cấp giải pháp mang tính cá nhân hóa. Hệ gợi ý học từ người dùng và gợi ý các sản phẩm tốt nhất trong số các sản phẩm phù hợp.

***=>Toàn bộ quy trình này cho phép truy cập, xử lý dữ liệu và xây dựng mô hình học máy một cách hiệu quả và có khả năng mở rộng, phù hợp cho các ứng dụng đòi hỏi xử lý lượng dữ liệu lớn và đáng tin cậy.***

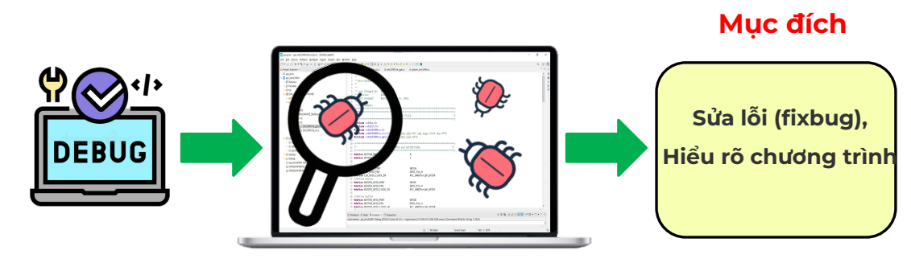
## *1.3. Nguyên lý dịch & gỡ rối*

***+*** Bug được hiểu là thuật ngữ dùng để chỉ các lỗi kỹ thuật có thể xảy ra trong quá trình thiết kế và vận hành các chương trình lập trình.

+ Bug khiến cho phần mềm, ứng dụng không thực thi được hoặc thực thi sai.

**

+ Debug là quá trình tìm kiếm và phát hiện nguyên nhân gây ra lỗi

****

***+ Các tùy chọn với chế độ debug***

* *Debugging API Data Ingestion (Gỡ lỗi Nhận dữ liệu từ API):*
* In-Memory Logging: Dữ liệu nhận được từ API có thể được ghi lại trong bộ nhớ để dễ dàng kiểm tra và gỡ lỗi.
* Data Validation: Kiểm tra tính chính xác của dữ liệu đầu vào và gửi thông báo lỗi nếu có sự cố.
* Debugging Spark Data Processing (Gỡ lỗi Xử lý dữ liệu bằng Spark):

Logging và Tracing: Sử dụng các hệ thống log và truy vết để theo dõi dữ liệu và tiến trình xử lý trong Spark.

* Unit Testing: Phát triển các bài kiểm tra đơn vị để xác định xem các phần xử lý dữ liệu hoạt động đúng cách.
* Spark UI: Sử dụng giao diện người dùng Spark để xem xét thông tin về quá trình xử lý và đánh giá hiệu suất.

# 2. BÀI THÍ NGHIỆM THỰC HÀNH

## 2.1. Mục đích - Yêu cầu

### 2.1.1. Mục đích

+ Giúp sinh viên làm quen với các công nghệ hiện đại, nắm rõ cách thức sử dụng Spark MLlib để thực thi các truy vấn SQL.

+ Biết cách đọc thư viện được dựng sẵn.

+ Biết các khởi tạo SparkSession

+ Biết cách đọc, kiểm tra, thống kê mô tả dữ liệu.

+ Biết cách xử lý và phân tích dữ liệu.

+ Biết cách chuẩn bị dữ liệu để xây dựng mô hình học máy.

+ Biết cách huấn luyện mô hình học máy: classification, regression, clustering, collaborative filtering, …

+ Biết cách đánh giá mô hình máy học và đưa ra kết quả dự đoán cho tương lai.

### 2.1.2. Yêu cầu:

* Nắm vững kiến thức được giới thiệu, tự tìm hiểu thêm nhiều chức năng khác.
* Hoàn thành bài code sử dụng Spark MLlib để xây dựng các mô hình học máy tiêu biểu theo yêu cầu của giảng viên.

### 2.1.3. Thời gian thực hiện

+ Thời gian mỗi buổi thực hành là từ 3-4 giờ, chia làm 10-12 nhóm nhỏ, mỗi nhóm 2 sinh viên/1 máy tính. Sinh viên tìm hiểu cơ sở lý thuyết ngắn gọn và các bước thực hiện có thể thao tác dễ dàng.

## 2.2. Chuẩn bị

### 2.2.1. Danh mục thiết bị thực hành

+ Phần cứng: 01 máy tính. Tất cả được được đồng bộ theo số thứ tự từ 1-20

+ Phần mềm: Apache Spark, PySpark

**Danh mục thiết bị thực hành phòng lab**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Stt | Tên thiết bị | Số lượng | Thông số  kỹ thuật | Vai trò |
| 1 | Máy tính để bàn | 01 | - Intel Core i3-2100 3.1 Ghz/3M Cache  - 4GB DDR3 (nâng cấp lên ram 8GB thêm 300K)  -SSD 120G Gb chuẩn SATA 3 - 6Gb/s Ổ SSD | -Tính toán, viết chương trình, chạy chương trình |

## 2.3. Nội dung

### 2.3.1. Các bước thực hiện

***Nội dung 1:*** cài đặt Apache Spark, cài đặt pip, cài đặt Pyspark.

**Bước 1: Tạo một project mới “PROJECT1”**

**Bước 2: Cài đặt Apache Spark**

|  |
| --- |
| """  *Cài đặt các gói cần thiết cho Spark*  """  sudo apt install default-jdk scala git -y  """  *Tải xuống và thiết lập Spark cho Ubuntu*  """  <https://spark.apache.org/downloads.html>  tar xvzf <ten\_file\_nen\_spark>.tgz  """  *Cấu hình môi trường Spark*  """  export SPARK\_HOME=<duong\_dan\_toi\_thu\_muc\_ban\_vua\_dat\_spark>  export PATH=$PATH:$SPARK\_HOME/bin:$SPARK\_HOME/sbin  export PYSPARK\_PYTHON=/usr/bin/python3  """  *Cài đọc pip, pyspark, pandas*  """  sudo apt install pip  sudo apt install python-pip  pip install pyspark  pip install pandas |

***Nội dung 2: thực hành phân tích, xử lý, xây dựng mô hình học máy với Spark***

***Mô hình phân loại (Classification)***

|  |
| --- |
| # *Khai báo các thư viện*  from pyspark.sql import SparkSession  from pyspark.ml.feature import StringIndexer, OneHotEncoder  from pyspark.ml.feature import VectorAssembler  from pyspark.ml.classification import LogisticRegression, RandomForestClassifier, GBTClassifier  from pyspark.ml.evaluation import MulticlassClassificationEvaluator  # *Mở SparkSession*  spark = SparkSession.builder.appName('Classification Model').getOrCreate()  # *Đọc dữ liệu*  df = spark.read.csv('drive/My Drive/PTIT/BigDataCourse/Data/titanic\_train.csv', header=True, inferSchema=True)  # *Kiểm tra dữ liệu*  df.show(4)  df.printSchema()  print('Number of rows: \t', df.count())  print('Number of columns: \t', len(df.columns))  # *Phân tích dữ liệu*  df.groupBy('Survived').mean('Fare', 'Age').show()  df.groupBy('Survived').pivot('Sex').count().show()  df.groupBy('Survived').pivot('Pclass').count().show()  for col in df.columns:  print(col, '\t\t', df.filter(df[col].isNull()).count())  # *Xử lý dữ liệu*  df = df.fillna({'Embarked': 'S'})  df = df.withColumn('FamilySize', df['Parch'] + df['SibSp']).\  drop('Parch', 'SibSp')  df = df.drop('PassengerID', 'Age', 'Cabin', 'Name', 'Ticket')  for col in df.columns:  print(col, '\t\t', df.filter(df[col].isNull()).count())  # *Chuẩn bị dữ liệu để huấn luyện mô hình*  stringIndex = StringIndexer(inputCols=['Sex', 'Embarked'],  outputCols=['SexNum', 'EmbNum'])  stringIndex\_model = stringIndex.fit(df)  df = stringIndex\_model.transform(df).drop('Sex', 'Embarked')  df.show(4)  vec\_asmbl = VectorAssembler(inputCols=df.columns[1:],  outputCol='features')  df = vec\_asmbl.transform(df).select('features', 'Survived')  df.show(4, truncate=False)  # *Chia dữ liệu huấn luyện và kiểm tra*  train\_df, test\_df = df.randomSplit([0.7, 0.3])  train\_df.show(4, truncate=False)  # *Huấn luyện mô hình*  ridge = LogisticRegression(labelCol='Survived',  maxIter=100,  elasticNetParam=0, # Ridge regression is choosen  regParam=0.3)  model = ridge.fit(train\_df)  lasso = LogisticRegression(labelCol='Survived',  maxIter=100,  elasticNetParam=1, # Lasso  regParam=0.0003)  model = lasso.fit(train\_df)  # *Dự đoán kết quả*  pred = model.evaluate(test\_df)  # *Đánh giá mô hình*  pred.accuracy  # *Mô hình Random Forest*  evaluator = MulticlassClassificationEvaluator(labelCol='Survived',  metricName='accuracy')  rf = RandomForestClassifier(labelCol='Survived',  numTrees=100, maxDepth=3)  model = rf.fit(train\_df)  pred = model.transform(test\_df)  evaluator.evaluate(pred)  # *Mô hình GBT*  gb = GBTClassifier(labelCol='Survived', maxIter=100, maxDepth=4)  model = gb.fit(train\_df)  pred = model.transform(test\_df)  evaluator.evaluate(pred) |

***Mô hình hồi quy (Regression)***

|  |
| --- |
| # Khai báo các thư viện  from pyspark.sql import SparkSession  from pyspark.ml.regression import LinearRegression  from pyspark.ml.linalg import Vectors  from pyspark.ml.feature import VectorAssembler  # Mở spark session  spark = SparkSession.builder.appName("Linear Regression Model").getOrCreate()  # Đọc dữ liệu  data = spark.read.csv('drive/My Drive/PTIT/BigDataCourse/Data/Ecommerce\_Customers.csv',header = True, inferSchema = True)  # Kiểm tra dữ liệu  data.show(4)  data.printSchema()  for item in data.head(1)[0]:  print(item)  data.columns  # Xử lý và chuẩn bị dữ liệu  assembler = VectorAssembler(inputCols =['Avg Session Length','Time on App','Time on Website','Length of Membership'],  outputCol='features')  output = assembler.transform(data)  output.printSchema()  output.head(1)  final\_data = output.select('features','Yearly Amount Spent')  # Chia dữ liệu huấn luyện và kiểm tra  train\_data,test\_data = final\_data.randomSplit([0.7,0.3])  train\_data.describe().show()  test\_data.describe().show()  # Huấn luyện mô hình  regressor = LinearRegression(labelCol='Yearly Amount Spent')  model = regressor.fit(train\_data)  # Đánh giá kết quả mô hình  pred\_data = model.evaluate(test\_data)  pred\_data.residuals.show()  pred\_data.rootMeanSquaredError  pred\_data.r2  # Dự đoán cho dữ liệu mới  unlabeled\_data = test\_data.select('features')  test\_predictions = model.transform(unlabeled\_data)  test\_predictions.show() |

***Mô hình phân cụm (Clustering)***

|  |
| --- |
| # Khai báo các thư viện cần thiết  from pyspark.sql import SparkSession  from pyspark.ml.feature import VectorAssembler  from pyspark.ml.feature import StandardScaler  from pyspark.ml.clustering import KMeans  # Mở spark session  spark = SparkSession.builder.appName('Kmeans').getOrCreate()  # Đọc dữ liệu  data = spark.read.csv('drive/My Drive/PTIT/BigDataCourse/Data/hack\_data.csv',header=True,inferSchema=True)  # Kiểm tra dữ liệu  data.printSchema()  data.columns  # Xử lý dữ liệu tạo vector cho các đặc trưng  assembler = VectorAssembler(inputCols=['Session\_Connection\_Time',  'Bytes Transferred',  'Kali\_Trace\_Used',  'Servers\_Corrupted',  'Pages\_Corrupted',  'WPM\_Typing\_Speed'], outputCol='features')  final\_data = assembler.transform(data)  final\_data.printSchema()  # Xử lý scale dữ liệu  scaler = StandardScaler(inputCol='features',outputCol='scaledFeat')  final\_data = scaler.fit(final\_data).transform(final\_data)  final\_data.printSchema()  # Thử với k = 2  kmeans = KMeans(featuresCol='scaledFeat', k=2)  # Xây dựng mô hình  model = kmeans.fit(final\_data)  results = model.transform(final\_data)  # In ra điểm center  centers = model.clusterCenters()  print(centers)  # In ra kết quả  results.printSchema()  results.show(10)  results.describe().show()  results.groupBy('prediction').count().show() |

***Hệ thống gợi ý (Recommendation)***

|  |
| --- |
| # Khai báo thư viện  from pyspark.sql import SparkSession  from pyspark.ml import Pipeline  from pyspark.sql import SQLContext  from pyspark.sql.functions import mean,col,split, col, regexp\_extract, when, lit  from pyspark.ml.feature import StringIndexer, VectorAssembler, IndexToString  from pyspark.ml.evaluation import MulticlassClassificationEvaluator  from pyspark.ml.feature import QuantileDiscretizer  # Tạo SparkSession  spark = SparkSession.builder.appName('recommender\_system').getOrCreate()  # Đọc dữ liệu  df = spark.read.csv("drive/My Drive/PTIT/BigDataCourse/Data/movie\_ratings\_df.csv", inferSchema=True, header=True)  # Kiểm tra dữ liệu  df.limit(3).toPandas()  df.printSchema()  # Xử lý dữ liệu chuẩn bị huấn luyện mô hình  stringIndexer = StringIndexer(inputCol='title', outputCol='title\_new')  # Applying stringindexer object on dataframe movie title column  model = stringIndexer.fit(df)  # creating new dataframe with transformed values  indexed = model.transform(df)  # validate the numerical title values  indexed.limit(5).toPandas()  # split the data into training and test datatset  train, test = indexed.randomSplit([0.75,0.25])  from pyspark.ml.recommendation import ALS  #Training the recommender model using train datatset  rec=ALS( maxIter=10  ,regParam=0.01  ,userCol='userId'  ,itemCol='title\_new'  ,ratingCol='rating'  ,nonnegative=True  ,coldStartStrategy="drop")  #fit the model on train set  rec\_model=rec.fit(train)  #making predictions on test set  predicted\_ratings=rec\_model.transform(test)  predicted\_ratings.limit(5).toPandas()  # Importing Regression Evaluator to measure RMSE  from pyspark.ml.evaluation import RegressionEvaluator  # create Regressor evaluator object for measuring accuracy  evaluator=RegressionEvaluator(metricName='rmse',predictionCol='prediction',labelCol='rating')  # apply the RE on predictions dataframe to calculate RMSE  rmse=evaluator.evaluate(predicted\_ratings)  # print RMSE error  print(rmse)  # Đề xuất các bộ phim  # First we need to create dataset of all distinct movies  unique\_movies=indexed.select('title\_new').distinct()  #create function to recommend top 'n' movies to any particular user  def top\_movies(user\_id,n):  """  This function returns the top 'n' movies that user has not seen yet but might like  """  #assigning alias name 'a' to unique movies df  a = unique\_movies.alias('a')  #creating another dataframe which contains already watched movie by active user  watched\_movies=indexed.filter(indexed['userId'] == user\_id).select('title\_new')  #assigning alias name 'b' to watched movies df  b=watched\_movies.alias('b')  #joining both tables on left join  total\_movies = a.join(b, a.title\_new == b.title\_new,how='left')  #selecting movies which active user is yet to rate or watch  remaining\_movies=total\_movies.where(col("b.title\_new").isNull()).select(a.title\_new).distinct()  #adding new column of user\_Id of active useer to remaining movies df  remaining\_movies=remaining\_movies.withColumn("userId",lit(int(user\_id)))  #making recommendations using ALS recommender model and selecting only top 'n' movies  recommendations=rec\_model.transform(remaining\_movies).orderBy('prediction',ascending=False).limit(n)  #adding columns of movie titles in recommendations  movie\_title = IndexToString(inputCol="title\_new", outputCol="title",labels=model.labels)  final\_recommendations=movie\_title.transform(recommendations)  #return the recommendations to active user  return final\_recommendations.show(n,False)  # Test: recommend 5 movies for user of id=60  top\_movies(60,5) |

**Cấu hình và Chạy Spark**

|  |
| --- |
| ***spark-submit --master local[2] <Đường dẫn>/<file\_code>.py*** |

# TÀI LIỆU HƯỚNG DẪN THỰC HÀNH

[1] Tài liệu:

https://github.com/roshankoirala/pySpark\_tutorial/blob/master/Classification\_with\_pySpark.ipynb

https://github.com/Ansu-John/Linear-Regression-with-Spark/tree/main/resources

https://github.com/YMolla/K-means-clustering-using-pyspark/blob/main/K-means%20clustering\_pyspark.ipynb

https://www.kaggle.com/code/tientd95/pyspark-for-data-science

# PHỤ LỤC

**Phiếu báo cáo kết quả thực hành (Sinh viên)**

Tên bài: ……………………………………………………………………………...

Họ và tên sinh viên……………………………Mã sinh viên……………………...

………………………………………………………………..……………………...

Nhóm…………………..Lớp……………… ..Ngày…..tháng…..năm…………..

Giáo viên hướng dẫn…………………………Ca thực tập……………………….

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | Nội dung  thực hành | Mức độ  hoàn thành (%) | Thời gian  hoàn thành | Đánh giá kết quả (100) |
| 1 |  |  |  |  |
| 2 |  |  |  |  |
| 3 |  |  |  |  |
| Thảo luận sinh viên: | | | | |

**Phiếu đánh giá kết quả thực hành (Giảng viên)**

Tên bài: ……………………………………………………………………………...

Họ và tên sinh viên……………………………Mã sinh viên……………………...

Nhóm…………………..Lớp…………………Ngày…..tháng…..năm…………..

Giáo viên hướng dẫn………………………….Ca thực tập……………………….

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Thứ tự** | **Nội dung đánh giá** | **Điểm chuẩn** | **Yêu cầu** | **Ghi chú** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| Tổng điểm: | | | | |
| Nhận xét giảng viên: | | | | |