AI VIET NAM – COURSE 2023

Big Data Exercises

(Spark, PySpark and its Application)

Ngày 6 tháng 8 năm 2023

Câu 1: Dữ liệu có thể được xử lý, lưu trữ và truy xuất ở định dạng cố định được gọi là:

(A) Structured Data

(B) Unstructured Data

(C) Semi-structured Data

(D) không câu nào đúng

Câu 2: Việc xử lý dữ liệu lớn trong bộ nhớ làm cho PySpark trở nên lý tưởng cho việc tính toán _____.

(A) Virtual

(B) Real-time

(C) Static

(D) Dynamic

Câu 3: Trong bộ nhớ, PySpark xử lý dữ liệu nhanh hơn 100 lần và trên đĩa, tốc độ nhanh gấp __ lần:

(A) 100

(B) 200

(C) 10

(D) 50

Câu 4: Apache Spark framwork có thể thực hiện nhiều tác vụ khác nhau, chẳng hạn như _____, chạy thuật toán Machine Learning hoặc làm việc với biểu đồ.
All of the above

(A) Executing distributed SQL

(B) Creating data pipelines

(C) Inputting data into databases

(D) (A), (B), và (C)

Câu 5: Tính năng của PySpark SQL là gì?

(A) Consistence Data Access

(B) Incorporation with Spark

(C) Standard Connectivity

(D) (A), (B) và (C)

Câu 6: Apache Spark có khả năng chạy các chương trình xử lý hàng loạt khi được xử lý trong bộ nhớ nhanh hơn bao nhiêu so với MapReduce?

(A) 100

(B) 200

(C) 250

(D) 300

Câu 7: Thành phần nào sau đây không phải là một thành phần trong Hệ sinh thái của Spark?

(A) Sqoop

(B) MLlib

(C) GraphX

(D) BlinkDB

Câu 8: Điều nào sau đây là lý do khiến Spark nhanh hơn MapReduce?

(A) RDDs are immutable and fault-tolerant

(B) Support for different language APIs like Scala, Java, Python and R

(C) DAG execution engine and in-memory computation

(D) Không câu nào đúng

Câu 9: Những hạn chế của MapReduce trong Hadoop đã được khắc phục bởi Spark RDD bằng _____

(A) DAG

(B) Lazy-evaluation

(C) In-memory processing

(D) Tất cả các câu trên

Câu 10: Tổ chứ nào có cụm Hadoop lớn nhất thế giới.

(A) DAG

(B) Datamatics

(C) Facebook

crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	b	lstat	medv
0.00632	18	2.31	0	538	6.575	65.2	4.09	1	296	15.3	396.9	4.98	24
0.02731	0	7.07	0	469	6.421	78.9	4.9671	2	242	17.8	396.9	9.14	21.6
0.02729	0	7.07	0	469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
0.03237	0	2.18	0	458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
0.06905	0	2.18	0	458	7.147	54.2	6.0622	3	222	18.7	396.9	5.33	36.2
0.02985	0	2.18	0	458	6.43	58.7	6.0622	3	222	18.7	394.12	5.21	28.7
0.08829	12.5	7.87	0	524	6.012	66.6	5.5605	5	311	15.2	395.6	12.43	22.9
0.14455	12.5	7.87	0	524	6.172	96.1	5.9505	5	311	15.2	396.9	19.15	27.1
0.21124	12.5	7.87	0	524	5.631	100	6.0821	5	311	15.2	386.63	29.93	16.5
0.17004	12.5	7.87	0	524	6.004	85.9	6.5921	5	311	15.2	386.71	17.1	18.9
0.22489	12.5	7.87	0	524	6.377	94.3	6.3467	5	311	15.2	392.52	20.45	15
0.11747	12.5	7.87	0	524	6.009	82.9	6.2267	5	311	15.2	396.9	13.27	18.9

CRIM - per capita crime rate by town
ZN - proportion of residential land zoned for lots over 25,000 sq.ft.
INDUS - proportion of non-retail business acres per town.
CHAS - Charles River dummy variable (1 if tract bounds river; 0 otherwise)
NOX - nitric oxides concentration (parts per 10 million)
RM - average number of rooms per dwelling
AGE - proportion of owner-occupied units built prior to 1940
DIS - weighted distances to five Boston employment centres
RAD - index of accessibility to radial highways
TAX - full-value property-tax rate per $10,000
PTRATIO - pupil-teacher ratio by town
B - 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
LSTAT - % lower status of the population
MEDV - Median value of owner-occupied homes in $1000's

Hình 1: Dữ liệu thông tin về nhà ở tại Boston

(D) Không câu nào đúng

Bài thực hành 1: Cho trước thông tin dữ liệu về nhà ở Boston trong file **"BostonHousing.csv"** (hình 1). Yêu cầu của bài tập này là các bạn có thể upload file để lữu trữ trên online.

Để hoàn thành bài tập này, bạn có 2 lựa chọn:

1. Bạn có thể tạo account trên https://cloud.google.com/ để sử dụng dịch vụ lưu trữ **GOOGLE CLOUD STORAGE** miễn phí trong vòng 3 tháng. Sau đó upload file lên cloud. Sau đó hoàn thiện đoạn code bên đưới để đọc dữ liệu về:

```
1
2  !pip install --upgrade google-cloud-storage
3  !pip install pyspark
4  import os
5  from google.cloud import storage
6  from pyspark.sql import SparkSession
7
8  os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = '/content/angular-yeti-389702-
       d5b06204e451.json'
9
10
11 from pyspark.sql import SparkSession
12
13 spark = SparkSession \
14         .builder \
15         .config("spark.jars", "gcs-connector-hadoop3-latest.jar") \
16         .getOrCreate()
17
18 spark._jsc.hadoopConfiguration().set('fs.gs.impl', 'com.google.cloud.hadoop.fs.gcs.
       GoogleHadoopFileSystem')
19 # This is required if you are using service account and set true,
20 spark._jsc.hadoopConfiguration().set('fs.gs.auth.service.account.enable', 'true')
21 spark._jsc.hadoopConfiguration().set('google.cloud.auth.service.account.json.keyfile',
       "angular-yeti-389702-d5b06204e451.json")
22
23 # Read data from google cloud storage
24 def read_file_from_cloud():
25
26   # ************* Your code here ********************
27
28   return df
29
30   #Question 11
31 df_cloud =  read_file_from_cloud()
```

```
print(df_cloud.count())
```

2. Bạn có thể sử dụng google drive để lưu trữ file và đọc file như sau:

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.master("local[*]").getOrCreate()
# Read data from google drive
def read_file_from_drive():
  ********** your code here ***********
  return data
```

Câu 11: Hãy cho biết kết quả của đoạn chương trình bên dưới:

```
  #Question 11
df =  read_file_from_cloud() # or df =  read_file_from_drive()

print(df.count())
```

(A) 506

(B) 806

(C) 606

(D) 706

Bài thực hành 2: Sử dụng kết quả từ bài thực hành 1, hãy tiếp tục viết chương trình truy vấn data sử dùng spark SQL.

Câu 12: Hãy cho biết kết quả của đoạn chương trình bên dưới:

```
#Question 12
## Execute this code for access data from google drive
df =  read_file_from_drive()
result = df.select("crim","zn","indus","medv").orderBy("medv").toPandas()
print(round(result.iloc[0,-1],2))

## Execute this code for access data from gooole cloud storage
df =  read_file_from_cloud()
df.createOrReplaceTempView("df")
query = """
          SELECT crim, zn, indus, medv
          FROM df
          order by medv
        """
result = spark.sql(query).toPandas()
print(round(result.iloc[0,-1],2))
```

(A) 4.0

(B) 5.0

(C) 6.0

(D) 7.0

Bài thực hành 3: Dựa vào kết quả của bài thực hành 1 2, hãy phát triển chương trình dự đoán thông tin **medv** dựa trên dựa trên 13 đặc trưng đầu vào gồm: **"crim", "zn", "indus", "chas", "nox", "rm", "age", "dis", "rad", "tax", "ptratio", "b", "lstat"** sử dụng giải thuật linear regression từ pyspark

a. Hãy hoàn thiện function **prepare_data_using_pyspark()** bên dưới đề chuẩn bị dữ liệu cho training giải thuật Linear Regression sử dụng VectorAssembler trong pyspark :

```

from pyspark.ml.regression import LinearRegression
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.evaluation import RegressionEvaluator

def prepare_data_using_pyspark(df):
  assembler = VectorAssembler(
  inputCols=["crim", "zn", "indus", "chas", "nox", "rm", "age", "dis", "rad", "tax", "
    ptratio", "b", "lstat"],
  outputCol="features")

  data = assembler.transform(df)
  final_data = []

  # ********** Your code here *****************

  return final_data


```

Câu 13: Hãy cho biết kết quả của đoạn chương trình bên dưới:

```
#Question 13
df =  read_file_from_drive() # or df =  read_file_from_cloud()
data = prepare_data_using_pyspark(df)
print(data)

```

(A) DataFrame[features: vector, medv: int]

(B) DataFrame[features: vector, medv: float]

(C) DataFrame[features: vector, medv: string]

(D) DataFrame[features: vector, medv: double]

b. Dựa vào kết quả của bài thực hành 3(a), hãy tiếp tục hoàn thiện function **train(data)** để huấn luyện data theo mô hình Linear Regression sử dụng thư viện pyspark.

```
train_data, test_data = data.randomSplit([0.8, 0.2], seed=42)
def train(train_data):
  lr = LinearRegression(featuresCol="features", labelCol="medv", predictionCol="
    predicted_medv")
  lr_model = None

  # ************** your code here ************************
```

```
  return lr_model
```

Câu 14: Hãy cho biết kết quả của đoạn chương trình bên dưới:

```
# Question 14
df =  read_file_from_drive() # or df =  read_file_from_cloud()
data = prepare_data_using_pyspark(df)
train_data, test_data = data.randomSplit([0.8, 0.2], seed=42)
lr_model = train(train_data)
coefficients = lr_model.coefficients
intercept = lr_model.intercept

print("Coefficients: ", coefficients)
print("Intercept: {:.3f}".format(intercept))
```

(A) Intercept: 38.617

(B) Intercept: 28.617

(C) Intercept: 18.617

(D) Intercept: 48.617

Câu 15: chúng ta sẽ sử dụng mô hình được huấn luyện để đưa ra dự đoán cho **test_data** và đánh giá độ chính xác (R-squared (R2) và RMSE) bằng cách sử dụng thư viện RegressionEvaluator. Hãy cho biết kết quả của đoạn chương trình bên dưới:

```
#Question 15

from pyspark.ml.evaluation import RegressionEvaluator
train_data, test_data = data.randomSplit([0.8, 0.2], seed=42)
lr_model = train(train_data)
predictions = lr_model.transform(test_data)

evaluator = RegressionEvaluator(labelCol="medv", predictionCol="predicted_medv",
    metricName="rmse")
rmse = evaluator.evaluate(predictions)
print("Root Mean Squared Error (RMSE) on test data: {:.3f}".format(rmse))

evaluator_r2 = RegressionEvaluator(labelCol="medv", predictionCol="predicted_medv",
    metricName="r2")
r2 = evaluator_r2.evaluate(predictions)
print("R-squared (R2) on test data: {:.3f}".format(r2))


```

(A) RMSE on test data: 2.672; R2 on test data: 0.793

(B) RMSE on test data: 3.672; R2 on test data: 0.793

(C) RMSE on test data: 4.672; R2 on test data: 0.793

(D) RMSE on test data: 1.672; R2 on test data: 0.793

Câu 16: Để xác định đặc trưng đầu vào (input feature) nào đóng góp nhiều nhất cho mô hình dự đoán đã được huấn luyện, chúng ta có thể phân tích giá trị tuyệt đối của các hệ số (coefficients) của phương trình tuyến tính. Các đặc trưng có hệ số giá trị tuyệt đối cao hơn có ảnh hưởng cao hơn đến kết quả dự đoán. Hãy cho biết kết quả của đoạn chương trình bên dưới:

```

assembler = VectorAssembler(
inputCols=["crim", "zn", "indus", "chas", "nox", "rm", "age", "dis", "rad", "tax", "
    ptratio", "b", "lstat"],
outputCol="features")
data = assembler.transform(df)

feature_importance = sorted(list(zip(data.columns[:-1], map(abs, coefficients))), key=
    lambda x: x[1], reverse=True)

print("The most important feature:", feature_importance[0][0])


```

(A) ptratio

(B) chas

(C) rm

(D) nox