Trường ĐH Khoa Học Tự Nhiên Tp. Hồ Chí Minh
TRUNG TÂM TIN HỌC

BIG DATA IN MACHINE LEARNING

Bài 6: Spark MLlib

Phòng LT & Mạng

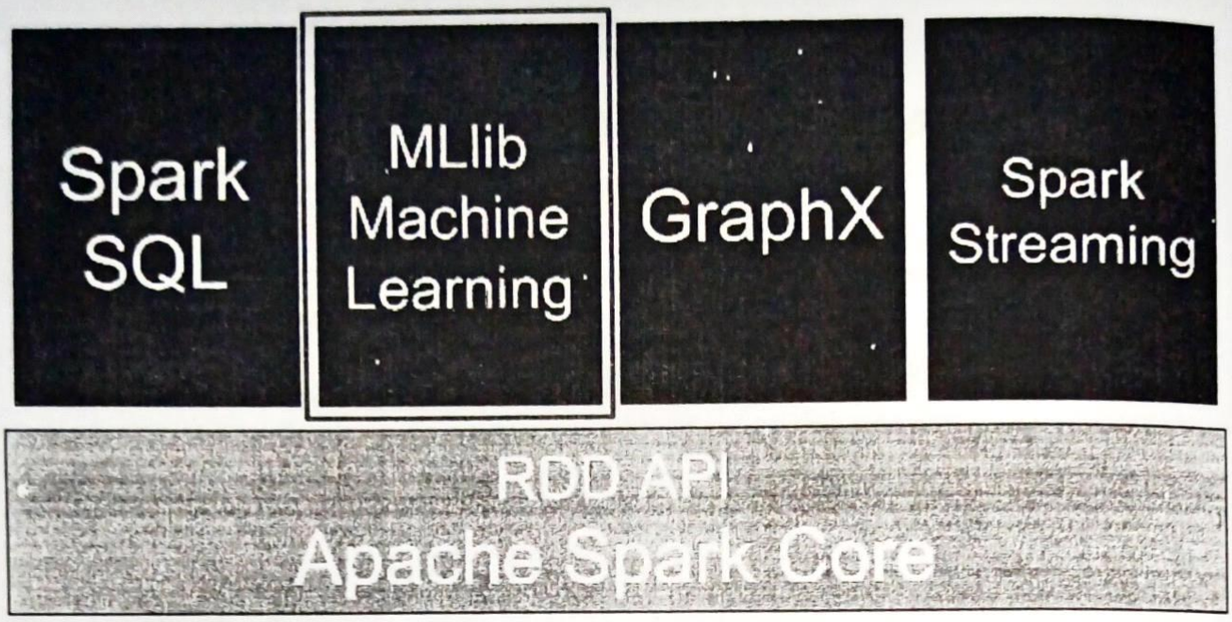
https://csc.edu.vn/Lap-Trinh-Va-Csdl/Big-Data-in-Machine-Learning_198

2020

Nội dung

1. Giới thiệu Spark MLlib
2. Spark MLlib algorithms
3. Xây dựng model
4. Đánh giá và đo lường
5. Cross-validation, Grid Search

Giới thiệu Spark MLlib



Giới thiệu Spark MLlib

❑ Spark MLlib

- Spark MLlib là một thư viện Machine Learning. Nó là một component phía trên Spark Core để phân tích dữ liệu bằng các thuật toán Machine Learning. Nó hoạt động trên các hệ thống phân tán (distributed system) và có thể mở rộng.
- Chúng ta có thể thực hiện các công việc classification, clustering, linear regression, và các thuật toán machine-learning khác với Spark MLlib.

https://www.edureka.co/blog/pyspark-mllib-tutorial/#what_is_mllib

Giới thiệu Spark MLlib

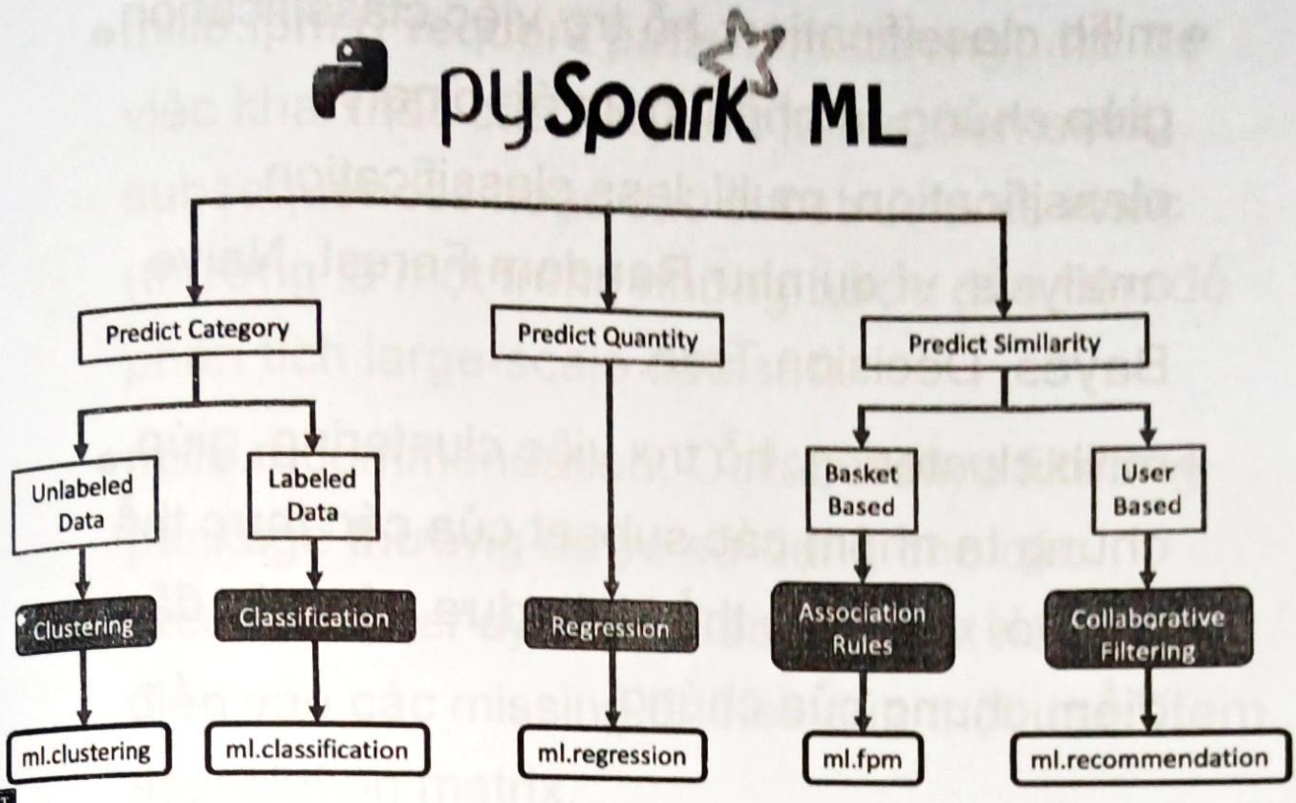
❑Tại sao chọn Spark MLlib?

- Các thuật toán Spark's MLlib được thiết kế dành cho parallel processing làm việc trên cluster
- Hỗ trợ nhiều ngôn ngữ lập trình như Scala, Java, Python và R
- Cung cấp high-level API để xây dựng machine learning pipelines

Nội dung

1. Giới thiệu Spark MLlib
2. **Spark MLlib algorithms**
3. Xây dựng model

Spark MLlib algorithms



Spark MLlib algorithms

- **3 "C" của Machine Learning trong Spark MLlib**
  - Collaborative filtering (recommender engines): Cung cấp các recommendation
  - Classification: Xác định loại của một new observation
  - Clustering: Nhóm dữ liệu dựa trên các đặc điểm tương tự

Spark MLlib algorithms

- mllib.classification: hỗ trợ việc classification, giúp chúng ta phân loại với: binary classification, multiclass classification analysis, ví dụ như Random Forest, Naive Bayes, Decision Tree...
- mllib.clustering: hỗ trợ việc clustering, giúp chúng ta nhóm các subset của các thực thể này với các thực thể khác dựa trên các đặc điểm chung của chúng.

Spark MLlib algorithms

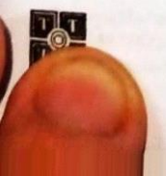
- mllib.regression: hỗ trợ việc tìm ra các mối quan hệ và sự phụ thuộc giữa các variables.
- mllib.linalg: MLlib utilities dùng cho linear algebra.

Spark MLlib algorithms

- mllib.fpm (Frequent pattern matching): hỗ trợ việc khai thác các frequent items, itemsets, subsequences hoặc các substructures khác (thường là một trong những bước đầu tiên để phân tích large-scale dataset).
- mllib.recommendation: Collaborative filtering package thường được sử dụng cho các recommender system. Các kỹ thuật này giúp điền vào các missing entries của một user item association matrix.

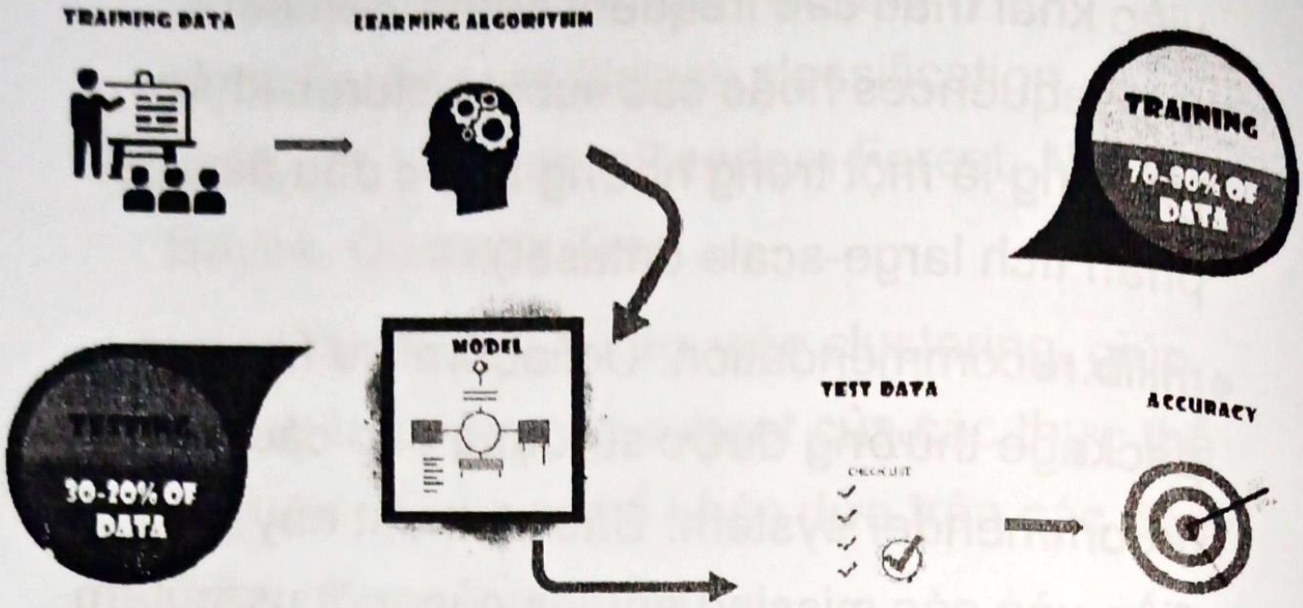
Nội dung

1. Giới thiệu Spark MLlib
2. Spark MLlib algorithms
3. Xây dựng model



Xây dựng model

❑Machine Learning Lifecycle



Xây dựng model

❑Các bước thực hiện

- Xác định vấn đề
- Chuẩn bị & chuẩn hóa dữ liệu, xác định inputs, output
- Chuẩn bị train/test dataset
- Xây dựng model với train dataset
- Đánh giá model với test dataset
- Lưu trữ & tải model

Xây dựng model

❑Ví dụ

- Vấn đề: Kiểm tra dữ liệu "Ecommerce Customer Data" từ website và app của công ty. Sau đó, xem xét liệu có thể xây dựng một regression model để dự đoán chi tiêu hàng năm của khách hàng dành cho sản phẩm của công ty hay không.

Xây dựng model

- Chuẩn bị & chuẩn hóa dữ liệu, xác định inputs, output

```
# Use Spark to read in the Ecommerce Customers csv file.
data = spark.read.csv("Ecommerce_Customers.csv",inferSchema=True,header=True)

# Print the Schema of the DataFrame
data.printSchema()
```

```
root
 |-- Email: string (nullable = true)
 |-- Address: string (nullable = true)
 |-- Avatar: string (nullable = true)
 |-- Avg Session Length: double (nullable = true)
 |-- Time on App: double (nullable = true)
 |-- Time on Website: double (nullable = true)
 |-- Length of Membership: double (nullable = true)
 |-- Yearly Amount Spent: double (nullable = true)
```

```
data.head()
```

```
Row(Email='mstephenson@fernandez.com', Address='835 Frank TunnelWrightmouth, MI 82180-9605', Avatar='Violet', Avg Session Length=34.49726772511229, Time on App=12.65565114916675, Time on Website=39.57766801952616, Length of Membership=4.082620632952961 5, Yearly Amount Spent=587.9510539684005)
```

Xây dựng model

```
# Import VectorAssembler and Vectors
from pyspark.ml.linalg import Vectors
from pyspark.ml.feature import VectorAssembler

data.columns

['Email',
 'Address',
 'Avatar',
 'Avg Session Length',
 'Time on App',
 'Time on Website',
 'Length of Membership',
 'Yearly Amount Spent']

assembler = VectorAssembler(
    inputCols=["Avg Session Length", "Time on App",
               "Time on Website",'Length of Membership'],
    outputCol="features") # inputs

data_pre = assembler.transform(data)
```

Xây dựng model

```
data_pre.select("features").show(2, False)

+-----------------------------------------------------------------------+
|features                                                               |
+-----------------------------------------------------------------------+
|[34.49726772511229,12.65565114916675,39.57766801952616,4.0826206329529615]|
|[31.92627202636016,11.109460728682564,37.268958868297744,2.664034182132 62]|
+-----------------------------------------------------------------------+
only showing top 2 rows

final_data = data_pre.select("features",'Yearly Amount Spent')
```

inputs

output

Xây dựng model

- Chuẩn bị train/test dataset

```
train_data,test_data = final_data.randomSplit([0.7,0.3])

train_data.describe().show()
```

```
+-------+------------------+
|summary|Yearly Amount Spent|
+-------+------------------+
|  count|               362|
|   mean| 498.6790796762861|
| stddev| 79.75887925186575|
|    min| 282.4712457199145|
|    max| 765.5184619388373|
+-------+------------------+
```

```
test_data.describe().show()
```

```
+-------+------------------+
|summary|Yearly Amount Spent|
+-------+------------------+
|  count|               138|
|   mean| 500.9772933802891|
| stddev|  78.4012371946312|
|    min|256.67058229005585|
|    max| 689.7876041747194|
+-------+------------------+
```

Xây dựng model

- Xây dựng model với train dataset

```
# Create a Linear Regression Model object
lr = LinearRegression(featuresCol="features",
                      labelCol='Yearly Amount Spent',
                      predictionCol='Predict_Yearly Amount Spent')

# Fit the model to the data and call this model lrModel
lrModel = lr.fit(train_data,)

# Print the coefficients and intercept for linear regression
print("Coefficients: {} Intercept: {}".format(lrModel.coefficients,lrModel.intercept))
```

```
Coefficients: [26.161383802828247,38.78912330841802,0.3185072139092366 4,62.02924777306868] Intercept: -1063.492944068675
```

Xây dựng model

- Đánh giá model với test dataset

```
test_results = lrModel.evaluate(test_data)

# Interesting results....
test_results.residuals.show(5)
```

```
+-------------------+
|          residuals|
+-------------------+
|-10.357682038778307|
|-3.8258631089308324|
| -3.436071927198725|
| -7.547407786212261|
| 11.107052784368989|
+-------------------+
only showing top 5 rows
```

```
print("RMSE: {}".format(test_results.rootMeanSquaredError))
print("MSE: {}".format(test_results.meanSquaredError))
print("r2: {}".format(test_results.r2))
```

```
RMSE: 9.340690236364381
MSE: 87.24849409171289
r2: 0.9858039997864344
```

Xây dựng model

- Đánh giá model với test dataset

```
# Check test dataset
test_model = lrModel.transform(test_data)

# Inspect results
test_model.select("Predict_Yearly Amount Spent", "Yearly Amount Spent").show(5)
```

```
+--------------------------+------------------+
|Predict_Yearly Amount Spent|Yearly Amount Spent|
+--------------------------+------------------+
|          330.2865518419719|  319.9288698031936|
|         471.32776353592044|  467.5019004269896|
|          493.6426719120534|  490.2065999848547|
|          564.8000945332669|  557.2526867470547|
|          416.2494780179238|  427.3565308022928|
+--------------------------+------------------+
only showing top 5 rows
```

Xây dựng model

- Lưu trữ & tải model

```
# Save model
lrModel.save('lrModel_Ecommerce_Customers')

from pyspark.ml.regression import LinearRegressionModel
# Load model from
lrModel2 = LinearRegressionModel.load('lrModel_Ecommerce_Customers')
```

Xây dựng model

- Dự đoán dữ liệu mới

```
# Predict new values (Assuming select test_data)
unlabeled_data = test_data.select('features')

predictions = lrModel2.transform(unlabeled_data)

predictions.show(5)
```

```
+-----------------+--------------------------+
|         features|Predict_Yearly Amount Spent|
+-----------------+--------------------------+
|[30.3931845423455...|         330.2865518419719|
|[30.8364326747734...|        471.32776353592044|
|[30.8794843441274...|         493.6426719120534|
|[31.1280900496166...|         564.8000945332669|
|[31.1695067987115...|         416.2494780179238|
+-----------------+--------------------------+
only showing top 5 rows
```

Chapter 6: Linear Regression

Demo

Basically what we do here is examine a dataset with Ecommerce Customer Data for a company's website and mobile app. Then we want to see if we can build a regression model that will predict the customer's yearly spend on the company's product.

First thing to do is start a Spark Session

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

In [2]:
```
import pyspark
```

In [3]:
```
from pyspark import SparkContext
from pyspark.conf import SparkConf
from pyspark.sql import SparkSession
```

In [4]:
```
spark = SparkSession.builder.appName('lr_example').getOrCreate()
```

In [5]:
```
from pyspark.ml.regression import LinearRegression
```

In [6]:
```
# Use Spark to read in the Ecommerce Customers csv file.
data = spark.read.csv("Ecommerce_Customers.csv",inferSchema=True,header=True)
```

In [7]:
```
# Print the Schema of the DataFrame
data.printSchema()
```

```
root
 |-- Email: string (nullable = true)
 |-- Address: string (nullable = true)
 |-- Avatar: string (nullable = true)
 |-- Avg Session Length: double (nullable = true)
 |-- Time on App: double (nullable = true)
 |-- Time on Website: double (nullable = true)
 |-- Length of Membership: double (nullable = true)
 |-- Yearly Amount Spent: double (nullable = true)
```

In [8]:
```
data.show(5)
```

```
+--------------------+--------------------+--------------+------------------+
|               Email|             Address|        Avatar|Avg Session Length|
Time on App|   Time on Website|Length of Membership|Yearly Amount Spent|
+--------------------+--------------------+--------------+------------------+
------------------+-----------------+--------------------+------------------+
|mstephenson@ferna...|835 Frank TunnelW...|        Violet| 34.49726772511229|
12.65565114916675| 39.57766801952616|  4.0826206329529615|  587.9510539684005|
|   hduke@hotmail.com|4547 Archer Commo...|     DarkGreen| 31.926272026360016|
11.109460728682564|37.268958868297744|    2.66403418213262|  392.2049334443264|
|    pallen@yahoo.com|24645 Valerie Uni...|        Bisque|33.000914755642675|
11.330278057777512|37.110597442120856|   4.104543202376424| 487.54750486747207|
|riverarebecca@gma...|1414 David Throug...|    SaddleBrown| 34.305556629755554|
13.717513665142507| 36.72128267790313|   3.120178782748092|  581.8523440352177|
|mstephens@davidso...|14023 Rodriguez P...|MediumAquaMarine| 33.330672523646394|
12.795188551078114| 37.53665330059473|   4.446308318351434|  599.4060920457634|
+--------------------+--------------------+--------------+------------------+
------------------+-----------------+--------------------+------------------+
only showing top 5 rows
```

In [9]:
```
data.head()
```

Out[9]:
```
Row(Email='mstephenson@fernandez.com', Address='835 Frank TunnelWrightmouth, MI
82180-9605', Avatar='Violet', Avg Session Length=34.49726772511229, Time on App
=12.65565114916675, Time on Website=39.57766801952616, Length of Membership=4.0
826206329529615, Yearly Amount Spent=587.9510539684005)
```

In [10]:
```
for item in data.head():
    print(item)
```

```
mstephenson@fernandez.com
835 Frank TunnelWrightmouth, MI 82180-9605
Violet
34.49726772511229
12.65565114916675
39.57766801952616
4.0826206329529615
587.9510539684005
```

Setting Up DataFrame for Machine Learning

In [11]:
```
# It needs to be in the form of two columns
# ("label","features")

# Import VectorAssembler and Vectors
from pyspark.ml.linalg import Vectors
from pyspark.ml.feature import VectorAssembler
```

In [12]:
```
data.columns
```

Out[12]:
```
['Email',
 'Address',
 'Avatar',
 'Avg Session Length',
 'Time on App',
 'Time on Website',
 'Length of Membership',
 'Yearly Amount Spent']
```

In [13]:
```
assembler = VectorAssembler(
    inputCols=["Avg Session Length", "Time on App",
               "Time on Website",'Length of Membership'],
    outputCol="features") # inputs
```

In [14]:
```
data_pre = assembler.transform(data)
```

In [15]:
```
data_pre.select("features").show(2, False)
```

```
+-----------------------------------------------------------------------------+
|features                                                                     |
+-----------------------------------------------------------------------------+
|[34.49726772511229,12.65565114916675,39.57766801952616,4.0826206329529615]|
|[31.92627202636016,11.109460728682564,37.268958868297744,2.66403418213262]|
+-----------------------------------------------------------------------------+
only showing top 2 rows
```

In [16]:
```
data_pre.show(2)
```

```
+--------------------+--------------------+--------+------------------+------
----------+------------------+--------------------+-------
------------+
|               Email|             Address|  Avatar|Avg Session Length|
Time on App|   Time on Website|Length of Membership|Yearly Amount Spent|
features|
+--------------------+--------------------+--------+------------------+------
----------+------------------+--------------------+-------
------------+
|mstephenson@ferna...|835 Frank TunnelW...|  Violet| 34.49726772511229| 12.655
65114916675| 39.57766801952616|  4.0826206329529615|  587.9510539684005|[34.497
2677251122...|
|   hduke@hotmail.com|4547 Archer Commo...|DarkGreen| 31.92627202636016|11.1094
60728682564|37.268958868297744|    2.66403418213262|  392.2049334443264|[31.926
2720263601...|
+--------------------+--------------------+--------+------------------+------
----------+------------------+--------------------+-------
------------+
only showing top 2 rows
```

In [17]:
```
final_data = data_pre.select("features",'Yearly Amount Spent')
```

In [18]:
```
train_data,test_data = final_data.randomSplit([0.7,0.3])
```

In [19]:
```
train_data.describe().show()
```

```
+-------+------------------+
|summary|Yearly Amount Spent|
+-------+------------------+
|  count|               343|
|   mean|  500.9283755502173|
| stddev|  83.87024231143093|
|    min| 256.67058229005585|
|    max|  765.5184619388373|
+-------+------------------+
```

In [20]:
```
test_data.describe().show()
```

```
+-------+------------------+
|summary|Yearly Amount Spent|
+-------+------------------+
|  count|               157|
|   mean| 495.78717398452744|
| stddev|  68.43381964871429|
|    min| 302.18954780965197|
|    max|  725.5848140556806|
+-------+------------------+
```

In [21]:
```
# Create a Linear Regression Model object
lr = LinearRegression(featuresCol="features",
                      labelCol='Yearly Amount Spent',
                      predictionCol='Predict_Yearly Amount Spent')
```

In [22]:
```
# Fit the model to the data and call this model lrModel
lrModel = lr.fit(train_data,)
```

In [23]:
```
# Print the coefficients and intercept for linear regression
print("Coefficients: {} Intercept: {}".format(lrModel.coefficients,
                                              lrModel.intercept))
```

```
Coefficients: [25.919298522549543,39.05259649768098,0.7963446754170292,61.58826
630305803] Intercept: -1075.2099532383284
```

In [24]:
```
test_results = lrModel.evaluate(test_data)
```

In [25]:
```
# Interesting results....
test_results.residuals.show(5)
```

```
+-------------------+
|          residuals|
+-------------------+
|-10.826809207839347|
|  0.808945904875543|
| 4.5681283726872834|
|  9.952012731730065|
|-3.0124996202594048|
+-------------------+
only showing top 5 rows
```

In [26]:
```
# Check test dataset
test_model = lrModel.transform(test_data)
```

In [27]:
```
# Inspect results
test_model.select("Predict_Yearly Amount Spent",
                  "Yearly Amount Spent").show(5)
```

```
+--------------------------+------------------+
|Predict_Yearly Amount Spent|Yearly Amount Spent|
+--------------------------+------------------+
|         330.75567901103295|  319.9288698031936|
|          441.2554678531901| 442.06441375806565|
|          387.9292708163341|  392.4973991890214|
|         417.40451807056274|  427.3565308022928|
|         426.48303279408333|  423.4705331738239|
+--------------------------+------------------+
only showing top 5 rows
```

In [28]:
```
print("RMSE: {}".format(test_results.rootMeanSquaredError))
print("MSE: {}".format(test_results.meanSquaredError))
print("r2: {}".format(test_results.r2))
```

```
RMSE: 10.080953257096533
MSE: 101.6256185717652
r2: 0.9781608015705986
```

Excellent results!

In [30]:
```
# Save model
lrModel.save('lrModel_Ecommerce_Customers')
```

In [ ]:
```
from pyspark.ml.regression import LinearRegressionModel
# Load model from
lrModel2 = LinearRegressionModel.load('lrModel_Ecommerce_Customers')
```

In [ ]:
```
# Predict new values (Assuming select test_data)
unlabeled_data = test_data.select('features')
```

In [ ]:
```
predictions = lrModel2.transform(unlabeled_data)
```

In [ ]:
```
predictions.show(5)
```