

PySpark_Regression บน Google Colaboratory

Source: <https://towardsdatascience.com/building-a-linear-regression-with-pyspark-and-mllib-d065c3ba246a>

✓ 1. ติดตั้งโปรแกรมที่เกี่ยวข้อง

1.1 Download and install java JDK 8

1.2 Download and install Apache Spark 3.5.3

1.3 Install findspark library

```
!apt-get install openjdk-8-jdk-headless -qq > /dev/null
!wget -q https://dlcdn.apache.org/spark/spark-3.5.4/spark-3.5.4-bin-hadoop3.tgz
!tar xf spark-3.5.4-bin-hadoop3.tgz
!pip install -q findspark
```


✓ 2. Import Python Libraries และ กำหนด path ของโปรแกรม Java และ Apache Spark ติดตั้งไว้ใน Notebook

```
import os
import sys
import time
import shutil
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
os.environ["SPARK_HOME"] = "/content/spark-3.5.4-bin-hadoop3"
```

✓ 3. Map Google Drive เข้าไปใน Google Colab notebook

Google Drive จะแสดงใน drive->My Drive

```
from google.colab import drive
drive.mount('/content/drive')
```

 Mounted at /content/drive

✓ 4. Import findspark และ pyspark.sql

สร้าง SparkContext ใหม่ที่มีชื่อว่า "BostonHousing" จาก SparkSession

```
import findspark
findspark.init()

from pyspark.sql import SparkSession
from pyspark.sql import SQLContext

sc = SparkSession.builder.appName("BostonHousing").getOrCreate()
```

5. Read bank marketing file

สร้างตัวแปร data จากไฟล์ csv ที่มีชื่อว่า "Boston.csv" ที่เก็บไว้ในโฟลเดอร์ /content (root path ของ Google Colab) เป็นไฟล์ราคาม้านในเขตพื้นที่ Boston ชุดข้อมูลสามารถดาวน์โหลดได้ที่ Kaggle

1. CRIM — per capita crime rate by town.
2. ZN — proportion of residential land zoned for lots over 25,000 sq.ft.
3. INDUS — proportion of non-retail business acres per town.
4. CHAS — Charles River dummy variable (= 1 if tract bounds river; 0 otherwise).
5. NOX — nitrogen oxides concentration (parts per 10 million).
6. RM — average number of rooms per dwelling.
7. AGE — proportion of owner-occupied units built prior to 1940.
8. DIS — weighted mean of distances to five Boston employment centres.
9. RAD — index of accessibility to radial highways.
10. TAX — full-value property-tax rate per \$10,000.
11. PTRATIO — pupil-teacher ratio by town.
12. BLACK — $1000(B_k - 0.63)^2$ where B_k is the proportion of blacks by town.
13. LSTAT — lower status of the population (percent).
14. MV — median value of owner-occupied homes in \$1000s. This is the target variable.

แสดงจำนวนแถวข้อมูลทั้งหมด (data.count())

แสดงรายการข้อมูลตัวอย่าง 20 แถวแรก (data.show())

แสดงรายการข้อมูลคอลัมน์และประเภทข้อมูล (data.printSchema())

```
data_file = '/content/drive/MyDrive/ColabNotebooks/Boston.csv'
data = sc.read.csv(data_file, header = True, inferSchema = True)
print('Total Records = {}'.format(data.count()))
data.show()
```

```
data.printSchema()
```

➡ Total Records = 506

_c0	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
1	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.09	1	296	15.3	396.9	4.98	24.0
2	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.9	9.14	21.6
3	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
4	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
5	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.9	5.33	36.2
6	0.02985	0.0	2.18	0	0.458	6.43	58.7	6.0622	3	222	18.7	394.12	5.21	28.7
7	0.08829	12.5	7.87	0	0.524	6.012	66.6	5.5605	5	311	15.2	395.6	12.43	22.9
8	0.14455	12.5	7.87	0	0.524	6.172	96.1	5.9505	5	311	15.2	396.9	19.15	27.1
9	0.21124	12.5	7.87	0	0.524	5.631	100.0	6.0821	5	311	15.2	386.63	29.93	16.5
10	0.17004	12.5	7.87	0	0.524	6.004	85.9	6.5921	5	311	15.2	386.71	17.1	18.9
11	0.22489	12.5	7.87	0	0.524	6.377	94.3	6.3467	5	311	15.2	392.52	20.45	15.0
12	0.11747	12.5	7.87	0	0.524	6.009	82.9	6.2267	5	311	15.2	396.9	13.27	18.9
13	0.09378	12.5	7.87	0	0.524	5.889	39.0	5.4509	5	311	15.2	390.5	15.71	21.7
14	0.62976	0.0	8.14	0	0.538	5.949	61.8	4.7075	4	307	21.0	396.9	8.26	20.4
15	0.63796	0.0	8.14	0	0.538	6.096	84.5	4.4619	4	307	21.0	380.02	10.26	18.2
16	0.62739	0.0	8.14	0	0.538	5.834	56.5	4.4986	4	307	21.0	395.62	8.47	19.9
17	1.05393	0.0	8.14	0	0.538	5.935	29.3	4.4986	4	307	21.0	386.85	6.58	23.1
18	0.7842	0.0	8.14	0	0.538	5.99	81.7	4.2579	4	307	21.0	386.75	14.67	17.5
19	0.80271	0.0	8.14	0	0.538	5.456	36.6	3.7965	4	307	21.0	288.99	11.69	20.2
20	0.7258	0.0	8.14	0	0.538	5.727	69.5	3.7965	4	307	21.0	390.95	11.28	18.2

only showing top 20 rows

root

```
-- _c0: integer (nullable = true)
-- crim: double (nullable = true)
-- zn: double (nullable = true)
-- indus: double (nullable = true)
-- chas: integer (nullable = true)
-- nox: double (nullable = true)
-- rm: double (nullable = true)
-- age: double (nullable = true)
-- dis: double (nullable = true)
-- rad: integer (nullable = true)
-- tax: integer (nullable = true)
-- ptratio: double (nullable = true)
-- black: double (nullable = true)
-- lstat: double (nullable = true)
-- medv: double (nullable = true)
```

✓ 6. สร้าง VectorAssembler

ใช้ VectorAssembler รวมคอลัมน์ feature ทั้งหมดไว้ในคอลัมน์เวกเตอร์เดียวกัน

แปลงรายการข้อมูลทุกคอลัมน์มารวมกันเป็นคอลัมน์ features และค่าคำตอบ medv

```
from pyspark.ml.feature import VectorAssembler
vectorAssembler = VectorAssembler(inputCols = ['crim', 'zn', 'indus', 'chas', 'nox', 'rm', 'age', 'dis', 'rad', 'tax', 'ptratio', 'lstat'])
vhouse_df = vectorAssembler.transform(data)
vhouse_df = vhouse_df.select(['features', 'medv'])
vhouse_df.show(3)
```



features	medv

```
|[0.00632,18.0,2.3...|24.0|
|[0.02731,0.0,7.07...|21.6|
|[0.02729,0.0,7.07...|34.7|
+-----+-----+
only showing top 3 rows
```

✓ 7. แบ่งตารางข้อมูลเป็น training data และ test data

แบ่งข้อมูลในตารางออกเป็น 70% เป็นข้อมูลฝึก และ 30% เป็นข้อมูลทดสอบ

```
splits = vhouse_df.randomSplit([0.7, 0.3])
train_df = splits[0]
test_df = splits[1]
```

✓ 8. สร้างตัวแบบจำแนกผลคำตอบแบบ Linear Regression

สร้างตัวแบบ Linear Regression โดยใช้ Feature Vector และ Label เป็นคอลัมน์ medv

```
from pyspark.ml.regression import LinearRegression
lr = LinearRegression(featuresCol='features', labelCol='medv', maxIter=10, regParam=0.3, elasticNetParam=0.8)
lr_model = lr.fit(train_df)
print("Coefficients: " + str(lr_model.coefficients))
print("Intercept: " + str(lr_model.intercept))
```

```
trainingSummary = lr_model.summary
print("RMSE: %f" % trainingSummary.rootMeanSquaredError)
print("r2: %f" % trainingSummary.r2)
```

```
➦ Coefficients: [-0.01423283842424336,0.0,0.0,3.0734504011631802,-7.952108128446414,3.6149182589071023,0
Intercept: 28.822987761787203
RMSE: 5.041310
r2: 0.720289
```



✓ 9. ทดสอบประสิทธิภาพของตัวแบบ Linear Regression กับ test data

ใช้ข้อมูลทดสอบ (test data) มาทำนายผลโดยตัวแบบ
ทำนายราคาบ้าน

```
lr_predictions = lr_model.transform(test_df)
lr_predictions.select("prediction","medv","features").show(5)
from pyspark.ml.evaluation import RegressionEvaluator
lr_evaluator = RegressionEvaluator(predictionCol="prediction", \
```

```
labelCol="medv",metricName="r2")
print("R Squared (R2) on test data = %g" % lr_evaluator.evaluate(lr_predictions))

test_result = lr_model.evaluate(test_df)
print("Root Mean Squared Error (RMSE) on test data = %g" % test_result.rootMeanSquaredError)
```

```
↗ +-----+-----+
| prediction|medv|      features|
+-----+-----+
| 31.2485368053113|24.0|[0.00632,18.0,2.3...|
|31.933627544563205|32.7|[0.01301,35.0,1.5...|
| 37.83991291585025|50.0|[0.01381,80.0,0.4...|
|31.053231398344177|31.6|[0.01432,100.0,1....|
| 41.74581699069377|50.0|[0.01501,90.0,1.2...|
+-----+-----+
only showing top 5 rows
```

```
R Squared (R2) on test data = 0.700003
Root Mean Squared Error (RMSE) on test data = 4.58258
```

✓ 10. Save and load Linear Regression model

ตรวจสอบว่ามีโฟลเดอร์ LogisticRegression Model หรือไม่ ถ้ามีให้ลบโฟลเดอร์

บันทึก LinearRegressionModel ลงในโฟลเดอร์

คัดลอกโฟลเดอร์ของ LinearRegressionModel ไปเก็บไว้ใน Google Drive

```
import os.path
from os import path
# Save and load model
sourcedirname = "LinearRegressionModel"
targetdirname = "content/drive/MyDrive/ColabNotebooks/LinearRegressionModel"

try:
    # remove existing LinearRegressionModel folder
    !rm -rf LinearRegressionModel

    # save model into LinearRegressionModel folder
    lr_model.save(sourcedirname)

    if path.isdir(targetdirname):
        # remove existing LinearRegressionModel folder in google drive
        !rm -rf drive/MyDrive/ColabNotebooks/LinearRegressionModel

    if path.isdir(sourcedirname):
        # copy LinearRegressionModel folder to google drive
        !cp -avr LinearRegressionModel drive/MyDrive/ColabNotebooks/LinearRegressionModel/

    # try to read model for prediction
    sameModel = lr_model.load(sourcedirname)
except Exception as e:
    print ("Error create and load prediction model", e)
```

```
↗ 'LinearRegressionModel/metadata/_SUCCESS' -> 'drive/MyDrive/ColabNotebooks/LinearRegressionModel/LinearRe
'LinearRegressionModel/metadata/_SUCCESS.crc' -> 'drive/MyDrive/ColabNotebooks/LinearRegressionModel/Line
```

```
'LinearRegressionModel/metadata/part-00000' -> 'drive/MyDrive/ColabNotebooks/LinearRegressionModel/LinearR  
'LinearRegressionModel/metadata/.part-00000.crc' -> 'drive/MyDrive/ColabNotebooks/LinearRegressionModel/Line  
'LinearRegressionModel/data/_SUCCESS' -> 'drive/MyDrive/ColabNotebooks/LinearRegressionModel/LinearRegres  
'LinearRegressionModel/data/.SUCCESS.crc' -> 'drive/MyDrive/ColabNotebooks/LinearRegressionModel/LinearReg  
'LinearRegressionModel/data/part-00000-51c3cf19-b9ab-492e-bcca-b3badadf1a6b-c000.snappy.parquet' -> 'drive,  
'LinearRegressionModel/data/.part-00000-51c3cf19-b9ab-492e-bcca-b3badadf1a6b-c000.snappy.parquet.crc' -> 'c
```

