Módulo 1. Utilización, procesamiento y visualización de grandes volúmenes de datos

Keyuan Zhao A01366831

Introducción

El objetivo del presente documento es detallar los pasos realizados para creación de un modelo regresión lineal con PysPark.

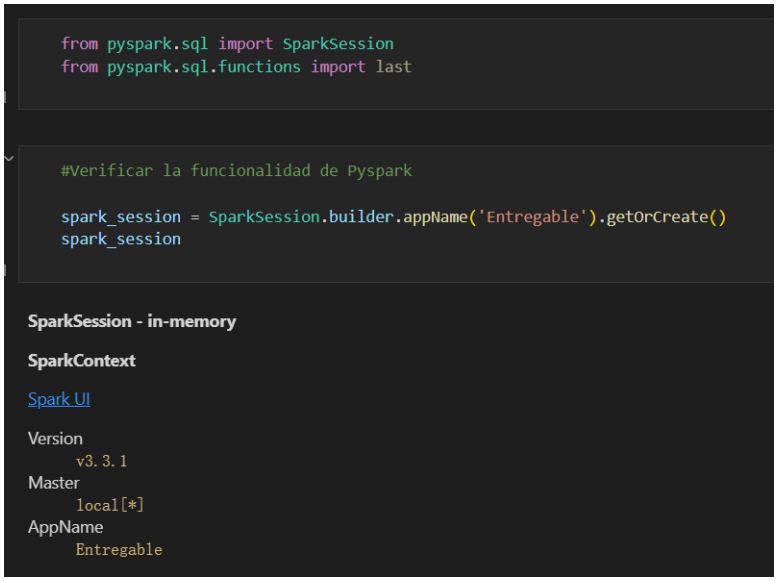
Contexto del dataset

El dataset fue descargado en la página de Kaggle y pesa 1.8 GB aproximadamente, lo cual contiene registros de Tickers de un conjunto de empresas estadounidenses desde enero de 1962 hasta marzo de 2018. Lo cual se registraron datos como:

- Ticker (nombre de Ticker de cada empresa)
- Date (fecha de registro)
- Open (costo del Ticker inicial)
- High (máximo costo del Ticker)
- Low (mínimo costo del Ticker)
- Close (costo del Ticker final)
- Volumen (cantidad de Tickers)
- Ex – dividend (dividiendo)
- Split- radio (rango de relación)

Crear el ambiente de trabajo

Debido a la rúbrica de esta actividad, se usó PySpark como framework de Big Data para procesar los datos. Aunque dichos datos no son considerados como Big Data ya que no presenta mayor complejidad al momento de analizarlo y no es necesario hacer otras modificaciones.



```
from pyspark.sql import SparkSession
from pyspark.sql.functions import last
```

```
#Verificar la funcionalidad de Pyspark

spark_session = SparkSession.builder.appName('Entregable').getOrCreate()
spark_session
```

SparkSession - in-memory

SparkContext

Spark UI

Version
v3.3.1
Master
local[*]
AppName
Entregable

Cargar los datos y visualización con PySpark

Una vez iniciado la sesión en PySpark, cargamos los datos y usamos la función de show() para mostrar el contenido del dataset mediante una tabla. Así como la siguiente imagen podemos visualizar que por el tamaño del dataset, sólo se mostró las primeras filas con todas las columnas del dataset.

```
df = spark_session.read.csv('WIKI_PRICES.csv', header = True, inferSchema = True)
```

```
df.show()
```

```
+------+-------------------+-----+-----+-----+-----+---------+-----------+-----------+-------------+-------------+-------------+-------------+----------+
|ticker|               date| open| high|  low|close|   volume|ex-dividend|split_ratio|     adj_open|     adj_high|      adj_low|    adj_close|adj_volume|
+------+-------------------+-----+-----+-----+-----+---------+-----------+-----------+-------------+-------------+-------------+-------------+----------+
|     A|1999-11-18 00:00:00| 45.5| 50.0| 40.0| 44.0|4.47399E7|        0.0|        1.0|31.041951216877|34.112034304261|27.289627443409|30.018590187749| 4.47399E7|
|     A|1999-11-19 00:00:00|42.94| 43.0|39.81|40.38|1.08971E7|        0.0|        1.0|29.295415060499|29.336349501664|27.160001713052|27.548878904121| 1.08971E7|
|     A|1999-11-22 00:00:00|41.31| 44.0|40.06| 44.0|4705200.0|        0.0|        1.0| 28.18336274218|30.018590187749|27.330561884574|30.018590187749| 4705200.0|
|     A|1999-11-23 00:00:00| 42.5|43.63|40.25|40.25|4274400.0|        0.0|        1.0|28.995229158622|29.766161133898| 27.46018761493| 27.46018761493| 4274400.0|
|     A|1999-11-24 00:00:00|40.13|41.94| 40.0|41.06|3464400.0|        0.0|        1.0|  27.3783187326|28.613174374414|27.289627443409|28.012802570659| 3464400.0|
|     A|1999-11-26 00:00:00|40.88| 41.5|40.75|41.19|1237100.0|        0.0|        1.0|27.889999247164|28.312988472536|27.801307957973| 28.10149385985| 1237100.0|
|     A|1999-11-29 00:00:00| 41.0|42.44|40.56|42.13|2914700.0|        0.0|        1.0|27.971868129494|28.954294717457|27.671682227616| 28.74280010477| 2914700.0|
|     A|1999-11-30 00:00:00| 42.0|42.94|40.94|42.19|3083000.0|        0.0|        1.0|28.654108815579|29.295415060499|27.930933688329|28.783734545935| 3083000.0|
|     A|1999-12-01 00:00:00|42.19|43.44|41.88|42.94|2115400.0|        0.0|        1.0|28.783734545935|29.636535403542|28.572239933249|29.295415060499| 2115400.0|
|     A|1999-12-02 00:00:00|43.75| 45.0|43.19|44.13|2195900.0|        0.0|        1.0|29.848030016228|30.700830873835| 29.46597523202|30.107281476941| 2195900.0|
|     A|1999-12-03 00:00:00|44.94|45.69|44.31| 44.5|2175700.0|        0.0|        1.0| 30.65989643267|31.171576947233|30.230084800436|30.359710530792| 2175700.0|
|     A|1999-12-06 00:00:00|45.25|46.44|45.19|45.75|1610000.0|        0.0|        1.0|30.871391045356|31.683257461797|30.830456604191|31.212511388399| 1610000.0|
|     A|1999-12-07 00:00:00|45.75| 46.0|44.31|45.25|1585100.0|        0.0|        1.0|31.212511388399| 31.38307155992|30.230084800436|30.871391045356| 1585100.0|
|     A|1999-12-08 00:00:00|45.25|45.63|44.81|45.19|1350400.0|        0.0|        1.0|30.871391045356|31.130642506068|30.571205143478|30.830456604191| 1350400.0|
|     A|1999-12-09 00:00:00|45.25|45.94|45.25|45.81|1451400.0|        0.0|        1.0|30.871391045356|31.342137118755|30.871391045356|31.253445829564| 1451400.0|
|     A|1999-12-10 00:00:00|45.69|45.94|44.75|44.75|1190800.0|        0.0|        1.0|31.171576947233|31.342137118755|30.530270702313|30.530270702313| 1190800.0|
|     A|1999-12-13 00:00:00| 45.5|46.25|44.38| 45.5|2875900.0|        0.0|        1.0|31.041951216877|31.553631731441|30.277841648462|31.041951216877| 2875900.0|
|     A|1999-12-14 00:00:00|45.38|45.38|42.06| 43.0|1665900.0|        0.0|        1.0|30.960082334547|30.960082334547|28.695043256744|29.336349501664| 1665900.0|
|     A|1999-12-15 00:00:00| 42.0|42.31| 41.0|41.69|2087100.0|        0.0|        1.0|28.654108815579|28.865603428265|27.971868129494|28.442614202893| 2087100.0|
|     A|1999-12-16 00:00:00| 42.0| 48.0| 42.0|47.25|1848300.0|        0.0|        1.0|28.654108815579| 32.74755293209|28.654108815579|32.235872417526| 1848300.0|
+------+-------------------+-----+-----+-----+-----+---------+-----------+-----------+-------------+-------------+-------------+-------------+----------+
```

Con la función de printSchema() logramos visualizar nombre y el tipo de datos para cada columnas del dataset, teniendo en cuenta esto se facilita el manejo de los datos.

```
df.printSchema()
```

```
root
 |-- ticker: string (nullable = true)
 |-- date: timestamp (nullable = true)
 |-- open: double (nullable = true)
 |-- high: double (nullable = true)
 |-- low: double (nullable = true)
 |-- close: double (nullable = true)
 |-- volume: double (nullable = true)
 |-- ex-dividend: double (nullable = true)
 |-- split_ratio: double (nullable = true)
 |-- adj_open: double (nullable = true)
 |-- adj_high: double (nullable = true)
 |-- adj_low: double (nullable = true)
 |-- adj_close: double (nullable = true)
 |-- adj_volume: double (nullable = true)
```

Selección de los datos y del modelo

Para facilitar el análisis del resultado usamos el modelo regresión lineal con ayuda de la librería MLlib. Y los datos que usamos para el modelo son: Open, High, Low, Close; para ello, debemos agrupar estos datos con el promedio sin importar el año, ya que así evitamos tener valor extremadamente grande que rompe la correlación para interpretación de los datos.

```
df = df.groupBy("Ticker").avg("open","close","high","low")
```

```
df.show()
```

```
+------+------------------+------------------+------------------+------------------+
|Ticker|         avg(open)|        avg(close)|         avg(high)|          avg(low)|
+------+------------------+------------------+------------------+------------------+
|  ACFN| 5.159595073641425| 5.167712239715598|  5.31348743863215|   5.0101387675639|
|  ALXN|61.846555155659544| 61.80857145942056| 62.94784869533896| 60.62664041749151|
|   AAT| 33.56090773053562| 33.56303699613469|33.872553837658714| 33.23921960242961|
|  ABMD|24.849846952245322|24.871593335058844|25.358382593503315|24.334952478322702|
|   AIV| 33.79793227104989| 33.80576316672244| 34.12653463602813|33.449152063066016|
|  ARAY| 7.613426659528919|7.6105299785867215|7.7798464668094205| 7.438157316202708|
|  ACCO|11.012743428931586|11.017502363693659| 11.20868480932873|10.824503907973531|
|   ALE|31.721052625816604|31.733457472187798|31.977249296309214|31.460719789863937|
|  ALSN|28.975527290705397|28.974222148978274| 29.27270527356622|28.666580553724447|
|  AGIO| 59.62478334749366|  59.6699762107052|61.308424808836136|57.885247918436825|
|   ALG|27.153825964132697|27.172286938581088| 27.49453145532461| 26.81968312966189|
|  ALKS| 23.22345341956584|23.221512637526104|23.793371141837614| 22.62824689265541|
|    AA| 39.07909116809114|39.124330484330464|39.752142165242155| 38.46961339031338|
|  ACHC|13.282660036315608|13.285934087157466|13.510581660614044|13.036880496863652|
|   APP| 3.723830087028593|3.7142337339411506| 3.806755822627426| 3.617135267302122|
|  ARUN|16.325645538761567| 16.32278376401757|16.670201072647497|15.964038761579744|
|  AAPL|101.23343664893625|101.19447202127645|102.46695825531889| 99.89625025531926|
|   ACW|4.9963550236406515| 4.986302482269509|5.0908800827423075| 4.885528309692673|
|  ANDE|29.525136688836977|29.540100647132913|30.008799910120427| 29.04074948768647|
|   ALL| 47.13675176828282| 47.15096415426471| 47.60703083693396|46.686172907665345|
+------+------------------+------------------+------------------+------------------+
```

Posteriormente, seleccionamos las columnas independientes: Open, High, Low; y la columna dependiente: Close. Ya que antes de entrenar nuestro modelo es necesario convertir estas columnas como vector.

```
# Usar la librería de VectorAssembler a las variables indenpendientes
from pyspark.ml.feature import VectorAssembler

featassembler = VectorAssembler(inputCols=['avg(open)','avg(high)','avg(low)'], outputCol = "Independent")
featassembler
```

Entrenamiento del modelo

Los datos se dividieron en dos partes: train (75%) y test (25%).

```
train, test = final_data.randomSplit([0.75, 0.25])
```

Después de usar la parte train para entrenar el modelo, obtuvimos los coeficientes betas y el coeficiente de determinación (R2). Lo cual el resultado es muy favorable y es precisa si hacemos alguna predicción con este modelo, ya que la coeficiente de determinación es de 99%.

```
#Regresión lineal
lr = LinearRegression(featuresCol = 'Independent', labelCol="avg(close)", maxIter=10, regParam=0.3, elasticNetParam=0.8)
lr_model = lr.fit(train)
print("Coefficients: " + str(lr_model.coefficients))
print("Intercept: " + str(lr_model.intercept))
```

```
Coefficients: [0.3305940591048401,0.329540860063884,0.33408348998765974]
Intercept: 0.19172031774823473
```

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

```
RMSE: 0.278268
r2: 0.999977
```

Fuente del dataset: https://www.kaggle.com/datasets/marketneutral/quandl-wiki-prices-us-equites