

Máster Universitario en Ingeniería Informática

CLOUD COMPUTING: SERVICIOS Y APLICACIONES

## PRÁCTICA 4

Procesamiento y minería de datos en Big Data  
con Spark sobre plataformas cloud



**UNIVERSIDAD  
DE GRANADA**



Carlos Santiago Sánchez Muñoz

*Email:* carlossamu7@correo.ugr.es

*DNI:* 75931715K

*20 de julio de 2021*

# Índice

<b>1. Introducción</b>	<b>2</b>
<b>2. Resolución</b>	<b>2</b>
2.1. Extracción de datos . . . . .	2
2.2. Procesamiento . . . . .	4
2.3. Regresión Logística . . . . .	5
2.4. Naive-Bayes . . . . .	6
2.5. Support Vector Machine . . . . .	7
2.6. Resultados . . . . .	8
<b>3. Conclusiones</b>	<b>9</b>
<b>4. Bibliografía</b>	<b>9</b>

## 1. Introducción

**Apache Spark** [1] es un *framework* de programación para procesamiento de datos distribuidos diseñado para ser rápido y de propósito general. Como su propio nombre indica, ha sido desarrollado en el marco del proyecto Apache, lo que garantiza su licencia *Open Source*.

Además, podremos contar con que su mantenimiento y evolución se llevarán a cabo por grupos de trabajo de gran prestigio, y existirá una gran flexibilidad e interconexión con otros módulos de Apache. Dichas etapas pueden incluir desde soporte para análisis interactivo de datos con **SQL** a la creación de complejos pipelines de machine learning y procesamiento en streaming, todo usando el mismo motor de procesamiento y las mismas APIs.

En esta práctica se va a usar un conjunto de datos de un volumen muy grande proporcionado como material de la asignatura. Las tareas a seguir son diversas. En primer lugar, se descargarán los datos del servidor escogiendo sólo aquellas columnas indicadas para mi estudio. A partir de ahí se podrá, en local, comenzar la tarea de aprendizaje.

El siguiente paso es preprocesar los datos, cuidando aspectos como el balanceo de los mismos y su división en conjunto de entrenamiento y test. Para todo esto voy a usar **Python** y **Spark**.

A continuación ya estaremos en condiciones de construir los modelos de aprendizaje automático usando **MLlib** y obtener resultados. Finalmente se expondrán los resultados, una discusión de los mismos y las conclusiones extraídas de este análisis.

## 2. Resolución

En esta sección se va a tratar toda la resolución del problema. Se comenzará explicando la extracción de datos, el preprocesado de los mismos y finalmente los modelos de aprendizaje que actuarán sobre ellos. Estos modelos tendrán al menos dos variantes en donde se usan parametrizaciones distintas.

### 2.1. Extracción de datos

El primer paso es la extracción de datos. Para ello se ha desarrollado en **Python** **este script** el cual accede al dataset y descarga todos los datos de aquellas columnas que en mi caso se le han indicado.

Dichas columnas son para mí:

```
class, PSSM_r2_2_W, PredCN_freq_central_2, PSSM_r2_2_R,  
      PSSM_r1_4_A, PSSM_r1_4_Q, PredCN_r2_1
```

El script desarrollado almacena un **csv** con esta información dentro de la carpeta **p4\_columns**.

Llevamos dicho script al servidor y allí lo ejecutamos. Veamos:

```
ccsa75931715@hadoop-master:~$ python3 p4_1_columns.py
21/07/19 21:07:29 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
21/07/19 21:07:35 WARN window.WindowExec: No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.

+-----+
|_c0|row_num|
+-----+
|@relation PSSM+Pr...|-1|
|@attribute separa...|0|
|@attribute propen...|1|
|@attribute length...|2|
|@attribute PredSS...|3|
|@attribute PredSS...|4|
|@attribute PredSS...|5|
|@attribute PredSS...|6|
|@attribute PredSS...|7|
|@attribute PredSS...|8|
|@attribute PredSS...|9|
|@attribute PredSS...|10|
|@attribute PredSS...|11|
|@attribute PredSS...|12|
|@attribute PredSS...|13|
|@attribute PredSS...|14|
|@attribute PredSS...|15|
|@attribute PredSS...|16|
|@attribute PredSS...|17|
|@attribute PredSS...|18|
+-----+
only showing top 20 rows
```

Imagen 1: Extracción de datos (parte 1)

Se muestra en el proceso las primeras 20 filas de la tabla extraída.

_c631	_c488	_c26	_c472	_c331	_c176	_c71
0	0	0.059	4	5	-1	3
1	-4	0.083	-5	-3	-3	4
0	-5	0.141	-6	-1	-1	2
0	-2	0.074	-7	-2	-1	4
0	-3	0.000	-1	4	0	3
1	-7	0.111	-4	0	-2	4
1	-4	0.103	-4	0	0	4
0	-6	0.060	5	2	-4	4
0	1	0.082	0	1	1	1
1	-5	0.111	-3	1	-5	2
1	-5	0.214	1	4	3	3
0	-2	0.000	-2	0	0	0
0	-5	0.107	-7	-1	0	2
0	-2	0.047	-3	-3	-2	4
0	-7	0.100	-5	-3	0	3
0	0	0.235	-4	0	1	1
0	-3	0.079	-4	-1	2	4
1	-4	0.024	-4	-4	-2	4
0	-6	0.081	1	-1	-2	1
1	-5	0.074	0	-3	2	1

only showing top 20 rows

Imagen 2: Extracción de datos (parte 2)

Se puede comprobar mediante `hdfs dfs -ls` si se han descargado los datos.

```
ccsa75931715@hadoop-master:~$ hdfs dfs -ls
21/07/19 21:08:38 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 1 items
drwxr-xr-x - ccsa75931715 ccsa75931715 0 2021-07-19 21:08 p4_columns
```

Imagen 3: Resultados extracción de datos

Y dentro de dicha carpeta:

```
ccsa75931715@hadoop-master:~$ hdfs dfs -ls ./p4_columns/
21/07/19 21:09:47 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 2 items
-rw-r--r--  2 ccsa75931715 ccsa75931715      0 2021-07-19 21:08 p4_columns/_SUCCESS
-rw-r--r--  2 ccsa75931715 ccsa75931715 42620107 2021-07-19 21:08 p4_columns/part-00000-0e099e95-6fd8-4854-9dfc-f8bdb3a5fbbc-c000.csv
```

Imagen 4: Resultados extracción de datos

Ya sólo faltaría llevar esos datos de hadoop al remoto local y finalmente a mi máquina local. Siguiendo la referencia [2] lo que hice fue usar `-copyToLocal` desde hadoop.

```
ccsa75931715@hadoop-master:~$ hdfs dfs -copyToLocal ./p4_columns /home/ccsa75931715
21/07/20 00:12:14 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
```

Imagen 5: Copiando al remoto los datos

Para llevarlo al Escritorio de mi máquina local he usado `scp`. Prefiero trabajar en local para evitar problemas de uso si otras personas están trabajando en paralelo.

```
carlos@carlos-Aspire-VN7-571G:~/Escritorio$ scp ccsa75931715@hadoop.ugr.es:/home/ccsa75931715/p4_columns/part-00000-0e099e95-6fd8-4854-9dfc-f8bdb3a5fbbc-c000.csv /home/carlos/Escritorio/
ccsa75931715@hadoop.ugr.es's password:
part-00000-0e099e95-6fd8-4854-9dfc-f8bdb3a5fbbc-c000.csv                                100% 41MB 10.8MB/s  00:03
carlos@carlos-Aspire-VN7-571G:~/Escritorio$ scp ccsa75931715@hadoop.ugr.es:/home/ccsa75931715/p4_columns/_SUCCESS /home/carlos/Escritorio/
ccsa75931715@hadoop.ugr.es's password:
_SUCCESS                                     100%  0    0.0KB/s  00:00
```

Imagen 6: Copiando los datos del remoto a mi local

## 2.2. Procesamiento

Una vez realizada la extracción de datos se puede comenzar el preprocesamiento de los mismos. Este procesamiento de los datos tendrá dos finalidades fundamentalmente:

- Por un lado equiparar las dos clases del problema evitando que exista desbalanceo.
- Por el otro dividir los datos una vez hecho esto en entrenamiento y test (al 80–20 %).

El código asociado está en el [repositorio](#). La ejecución la he realizado usando **Spark** en local. Levanto la composición de contenedores con `docker-compose up` y posteriormente en otro terminal `docker exec -it imagen_contenedor /bin/bash`. Desde dentro puedo usar **spark** y tengo los ficheros de Python disponibles ya que la carpeta `/data` de mi local la he mapeado a `/tmp/data` del contenedor levantado.

Para ejecutar el fichero de procesamiento lanzo la orden:

```
bin/spark-submit -master spark://master:7077 -total-executor-cores 1
-executor-memory 1g /tmp/data/P4_2_preprocessing.py
```

El ratio de desabanceo es 2. En la carpeta mapeada están los resultados de este proceso. Existen dos csv que he renombrado a `test.csv` y `train.csv`.

```

carlos@carlos-Aspire-V17:~$ docker exec -it docker-spark-master_master_1 /bin/bash
root@master:/usr/spark-2.4.1# bin/spark-submit --master spark://master:7077 --total-executor-cores 1 --executor-memory 1g /tmp/data/P4_2_preprocessing.py
2021-07-20 23:15:47,053 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
2021-07-20 23:15:48,156 INFO spark.SparkContext: Running Spark version 2.4.1
2021-07-20 23:15:48,183 INFO spark.SparkContext: Submitted application: Cloud Computing P4 - Preprocessing
2021-07-20 23:15:48,238 INFO spark.SecurityManager: Changing view acls to: root
2021-07-20 23:15:48,238 INFO spark.SecurityManager: Changing modify acls to: root
2021-07-20 23:15:48,238 INFO spark.SecurityManager: Changing view acls groups to:
2021-07-20 23:15:48,239 INFO spark.SecurityManager: Changing modify acls groups to:
2021-07-20 23:15:48,239 INFO spark.SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view permissions: Set(root); groups
et(); users with modify permissions: Set(root); groups with modify permissions: Set()
2021-07-20 23:15:48,501 INFO util.Utils: Successfully started service 'sparkDriver' on port 7001.
2021-07-20 23:15:48,528 INFO spark.SparkEnv: Registering MapOutputTracker
2021-07-20 23:15:48,552 INFO spark.SparkEnv: Registering BlockManagerMaster
2021-07-20 23:15:48,555 INFO storage.BlockManagerMasterEndpoint: Using org.apache.spark.storage.DefaultTopologyMapper for getting topology information
2021-07-20 23:15:48,555 INFO storage.BlockManagerMasterEndpoint: BlockManagerMasterEndpoint up
2021-07-20 23:15:48,566 INFO storage.DiskBlockManager: Created local directory at /tmp/blockmgr-0caecaec-acbd-435c-9eef-359ca1717288
2021-07-20 23:15:48,587 INFO memory.MemoryStore: MemoryStore started with capacity 366.3 MB
2021-07-20 23:15:48,603 INFO spark.SparkEnv: Registering OutputCommitCoordinator
2021-07-20 23:15:48,701 INFO util.log: Logging initialized @2747ms
2021-07-20 23:15:48,770 INFO server.Server: jetty-9.3.z-SNAPSHOT, build timestamp: unknown, git hash: unknown
2021-07-20 23:15:48,791 INFO server.Server: Started @2839ms
2021-07-20 23:15:48,813 INFO server.AbstractConnector: Started ServerConnector@37d19e5[HTTP/1.1,[http/1.1]]{0.0.0.0:4040}
2021-07-20 23:15:48,813 INFO util.Utils: Successfully started service 'SparkUI' on port 4040.

```

Imagen 7: Preprocesamiento de los datos

Hemos alcanzado un punto de desarrollo del problema en el que se está en condiciones de aprender de los datos con los modelos de inteligencia artificial presentes en MLLib.

Todos los modelos que se van a exponer aquí están aquí.

```

root@master:/usr/spark-2.4.1# bin/spark-submit --master spark://master:7077 --total-executor-cores 1 --executor-memory 1g /tmp/data/P4_3_models.py
2021-07-21 20:33:59,743 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
2021-07-21 20:34:01,432 INFO spark.SparkContext: Running Spark version 2.4.1
2021-07-21 20:34:01,467 INFO spark.SparkContext: Submitted application: Cloud Computing P4 - Machine Learning
2021-07-21 20:34:01,523 INFO spark.SecurityManager: Changing view acls to: root
2021-07-21 20:34:01,523 INFO spark.SecurityManager: Changing modify acls to: root
2021-07-21 20:34:01,523 INFO spark.SecurityManager: Changing view acls groups to:
2021-07-21 20:34:01,523 INFO spark.SecurityManager: Changing modify acls groups to:
2021-07-21 20:34:01,523 INFO spark.SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view permissions: Set(root); groups with view
et(); users with modify permissions: Set(root); groups with modify permissions: Set()
2021-07-21 20:34:01,935 INFO util.Utils: Successfully started service 'sparkDriver' on port 7001.
2021-07-21 20:34:01,978 INFO spark.SparkEnv: Registering MapOutputTracker
2021-07-21 20:34:02,016 INFO spark.SparkEnv: Registering BlockManagerMaster
2021-07-21 20:34:02,019 INFO storage.BlockManagerMasterEndpoint: Using org.apache.spark.storage.DefaultTopologyMapper for getting topology information
2021-07-21 20:34:02,020 INFO storage.BlockManagerMasterEndpoint: BlockManagerMasterEndpoint up
2021-07-21 20:34:02,033 INFO storage.DiskBlockManager: Created local directory at /tmp/blockmgr-81475004-bf63-4a7b-aea5-6b434d28692e
2021-07-21 20:34:02,058 INFO memory.MemoryStore: MemoryStore started with capacity 366.3 MB
2021-07-21 20:34:02,076 INFO spark.SparkEnv: Registering OutputCommitCoordinator
2021-07-21 20:34:02,161 INFO util.log: Logging initialized @7505ms
2021-07-21 20:34:02,248 INFO server.Server: jetty-9.3.z-SNAPSHOT, build timestamp: unknown, git hash: unknown
2021-07-21 20:34:02,273 INFO server.Server: Started @7618ms
2021-07-21 20:34:02,301 INFO server.AbstractConnector: Started ServerConnector@0e64f10[HTTP/1.1,[http/1.1]]{0.0.0.0:4040}
2021-07-21 20:34:02,301 INFO util.Utils: Successfully started service 'SparkUI' on port 4040.
2021-07-21 20:34:02,345 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@674895def[/jobs,null,AVAILABLE,@spark]
2021-07-21 20:34:02,346 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@76b7868f[/jobs/json,null,AVAILABLE,@spark]
2021-07-21 20:34:02,347 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@13bb65b7[/jobs/job,null,AVAILABLE,@spark]
2021-07-21 20:34:02,352 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@4d9ad2b3[/jobs/job/json,null,AVAILABLE,@spark]
2021-07-21 20:34:02,353 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@5c4906a1[/stages,null,AVAILABLE,@spark]
2021-07-21 20:34:02,355 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@7a813ab1[/stages/json,null,AVAILABLE,@spark]
2021-07-21 20:34:02,356 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@60b160a6[/stages/stage,null,AVAILABLE,@spark]
2021-07-21 20:34:02,359 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@29e45cdc[/stages/stage/json,null,AVAILABLE,@spark]
2021-07-21 20:34:02,361 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@42bbdb16[/stages/pool,null,AVAILABLE,@spark]
2021-07-21 20:34:02,362 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@1006ab69[/stages/pool/json,null,AVAILABLE,@spark]
2021-07-21 20:34:02,369 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@570e6f67[/storage,null,AVAILABLE,@spark]
2021-07-21 20:34:02,371 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@5c07b066[/storage/json,null,AVAILABLE,@spark]
2021-07-21 20:34:02,373 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@434b8e7d[/storage/rdd,null,AVAILABLE,@spark]

```

Imagen 8: Ejecución modelos de aprendizaje de los datos

## 2.3. Regresión Logística

El primer modelo es una regresión logística que sabemos que es bastante satisfactoria en problemas de clasificación binaria. El modelo a aplicar es **LogisticRegression**. Los parámetros que he ajustado son los siguientes:

- **maxIter**: número máximo de iteraciones.
- **regParam**: parámetro de regularización (previene *overfitting*).
- **elasticNetParam**: parámetro de regularización elástica de la red.

El código más relevante es (se expone la primera parametrización):

```
# Regresión Logística 1
lr = LogisticRegression(maxIter=10, regParam=0.2, elasticNetParam=0.7,
    family="binomial")
lrModel_1 = lr.fit(train)
predictions = lrModel_1.transform(test)
# Accuracy
evaluator = MulticlassClassificationEvaluator(labelCol="label",
    predictionCol="prediction", metricName="accuracy")
f.write("Accuracy RL1 (test): " + str(evaluator.evaluate(predictions)) + "\n")
```

### RL1: maxIter=10, regParam=0.2, elasticNetParam=0.7

Los resultados obtenidos son los siguientes:

Accuracy	0.499
----------	-------

### RL2: maxIter=20, regParam=0.25, elasticNetParam=0.8

Los resultados obtenidos son los siguientes:

Accuracy	0.499
----------	-------

A pesar de tener parámetros distintos no hay diferencias significativas en el acierto obtenido. **Spark** devuelve por defecto 17 decimales de precisión y en ellos son todos iguales. En la discusión de resultados hablaremos del resultado obtenido.

## 2.4. Naive-Bayes

Este algoritmo se basa en el teorema de Bayes y asume independencia de las variables para clasificar los datos.

Los parámetros más relevantes son:

- **smoothing**: parámetro de suavizado laplaciano que se realiza.

El código más relevante es (se expone la primera parametrización):

```
# Naive Bayes 1
nb = NaiveBayes(smoothing=2.0, featuresCol="scaled_features",
    labelCol="label", modelType="multinomial")
nbModel_1 = nb.fit(train)
predictions = nbModel_1.transform(test)
# Accuracy
evaluator = MulticlassClassificationEvaluator(labelCol="label",
    predictionCol="prediction", metricName="accuracy")
accuracy = evaluator.evaluate(predictions)
f.write("Accuracy NBI (test): " + str(evaluator.evaluate(predictions)) + "\n")
```

### NB1: smoothing=2.0

Los resultados obtenidos son los siguientes:

<b>Accuracy</b>	0.548
-----------------	-------

### NB2: smoothing=5.0

Los resultados obtenidos son los siguientes:

<b>Accuracy</b>	0.548
-----------------	-------

De nuevo las dos parametrizaciones ofrecen los mismos resultados de acierto.

## 2.5. Support Vector Machine

Este algoritmo intenta separar de la mejor forma posible los datos de ambas clases. Lo hace buscando el hiperplano que tenga el margen más amplio de distancia con cada una de las clases.

Los parámetros más relevantes son:

- **maxIter**: número máximo de iteraciones.
- **regParam**: parámetro de regularización (previene *overfitting*).

El código más relevante es el siguiente (se expone la primera parametrización):

```
# Support Vector Machine 1
lsvc = LinearSVC(maxIter=10, regParam=0.05)
lsvcModel_1 = lsvc.fit(train)
predictions = lsvcModel_1.transform(test)
# Accuracy
evaluator = MulticlassClassificationEvaluator(labelCol="label",
    predictionCol="prediction", metricName="accuracy")
f.write("Accuracy SVM1 (test): " + str(evaluator.evaluate(predictions)) + "\n")
```

### SVM1: maxIter=10, regParam=0.05

Los resultados obtenidos son los siguientes:

<b>Accuracy</b>	0.538
-----------------	-------

### SVM2: maxIter=20, regParam=0.04

Los resultados obtenidos son los siguientes:

<b>Accuracy</b>	0.549
-----------------	-------

Aquí sí existe diferencia significativa de resultados. Se comentan a continuación.



## 2.6. Resultados

Los resultados de los diferentes modelos quedan recogidos en la siguiente tabla:

Modelo	Parámetros	Accuracy
RL1	maxIter=10 regParam=0.2 elsaticNetParam=0.7	0.499
RL2	maxIter=20 regParam=0.25 elsaticNetParam=0.8	0.499
NB1	smoothing=2.0	0.548
NB2	smoothing=5.0	0.548
SVM1	maxIter=10 regParam=0.05	0.538
SVM2	maxIter=20 regParam=0.04	0.549

### 3. Conclusiones

El Machine Learning con Big Data es cada vez más usual y es una de las grandes partes del Cloud Computing evidentemente. Este paradigma nos dota de unos recursos que antes no se disponían. En este caso para evitar los problemas del servidor he traído los datos a local y posteriormente he levantado un contenedor donde he podido usar **Spark**.

Como trabajo futuro me gustaría usar **Databricks** que fue comentada por el profesor de la asignatura y parece tener un uso asequible a la vez que una herramienta potente.

Mediante tres modelos y varias parametrizaciones de los mismos se ha resuelto un problema de clasificación binaria. El acierto obtenido es decente pero sabiendo que hay dos clases debería ser superior. Seguramente manejando más atributos y usando técnicas de preprocesamiento más avanzadas intentar mejorarlo. Asimismo hacer un grid de parámetros en donde obtengamos la mejor parametrización de un modelo.

El mejor modelo es *Linear Support-Vector-Machine* con un acierto en el conjunto de test de 0.549. Como ya se ha dicho a nivel de inteligencia artificial no es satisfactorio pero el uso de herramientas Cloud y el aprendizaje adquirido son altos.

### 4. Bibliografía

- [1] esic. *Apache Spark: Introducción, qué es y cómo funciona*. URL: <https://www.esic.edu/rethink/tecnologia/apache-spark-introduccion-que-es-y-como-funciona>.
- [2] stackoverflow. *How to copy file from HDFS to the local file system*. URL: <https://stackoverflow.com/questions/17837871/how-to-copy-file-from-hdfs-to-the-local-file-system>.