



Big Data con Spark

Cloud Computing. Servicios y Aplicaciones

José Antonio Córdoba Gómez - 77201588H

Granada, España - 4 de junio de 2021
Máster en Ingeniería Informática



ugr

Universidad
de Granada

Índice general

1	Introducción	3
2	Resolución	6
2.1	Despliegue de la infraestructura	6
2.2	Obtención de columnas	9
2.3	Modelos	12
2.3.1	Evaluación de los modelos	12
2.3.2	Random Forest	13
2.3.3	Gradient Boosted Tree	16
2.3.4	Logistic Regresion	18
2.4	Comparativa de los modelos	21
2.5	Factura	21
3	Conclusiones	22
A	Código completo de los modelos	23
B	Composición de Spark	32

Índice de figuras

1.1	Estado del cluster hadoop	3
1.2	Búsqueda de los ficheros de cabecera y datos del dataset que nos corresponde . .	4
1.3	Descarga de ambos ficheros	4
2.1	Conectando a la máquina maestra del clúster. Acceso denegado	7
2.2	Añadir una regla de entrada para permitir acceso al puerto 22 desde cualquier IP .	8
2.3	Conectando a la máquina maestra del clúster. Acceso concedido	9
2.4	Subiendo los ficheros de cabecera y datos al bucket s3://bucket-pepitoenpeligro .	9
2.5	Comprobamos que están en el bucket s3://bucket-pepitoenpeligro	10
2.6	Resultados del modelo Random Forest con los primeros parámetros	14
2.7	Resultados del modelo Random Forest con otros parámetros	15
2.8	Resultados del modelo Gradient Boost Tree con los primeros parámetros	17
2.9	Resultados del modelo Gradient Boost Tree con otros parámetros	18
2.10	Resultados del modelo de regresión logística con los primeros parámetros	19
2.11	Resultados del modelo de regresión logística con otros parámetros	20
2.12	Factura de los costes asociados a la realización de la práctica en un entorno cloud real	21

Capítulo 1

Introducción

Durante el desarrollo de este documento, y por tanto, de la resolución de la práctica propuesta, se va a tratar de resolver problemas de clasificación mediante técnicas computacionales basadas en **Big Data**. El marco de trabajo empleado es **Spark** usando la librería de aprendizaje profundo *MLLib* y usando el **wrapper** de **python: pyspark**.

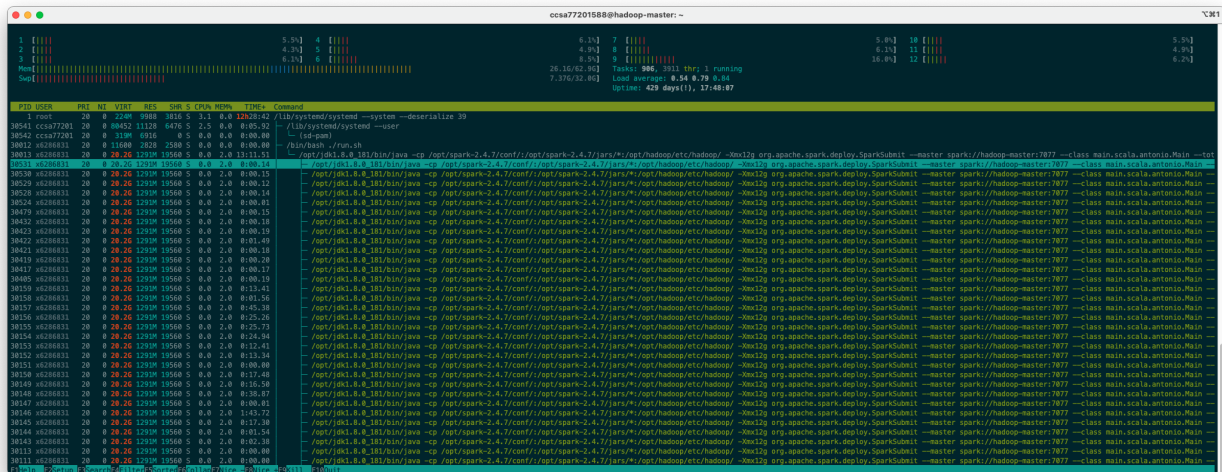
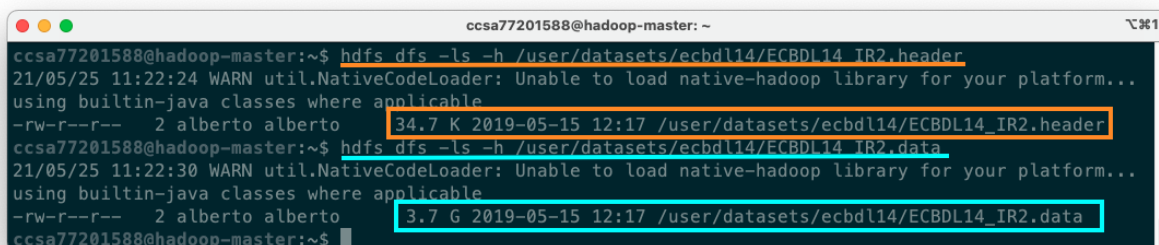


Figura 1.1: Estado del cluster hadoop

A título personal, sé que se nos ha dado acceso a un clúster de computación distribuido, conocido en la red como **hadoop.ugr.es**, que tienen la configuración del sistema de archivos distribuidos **HDFS**, pero he desistido usarlo, porque es imposible ejecutar ningún proceso allí.

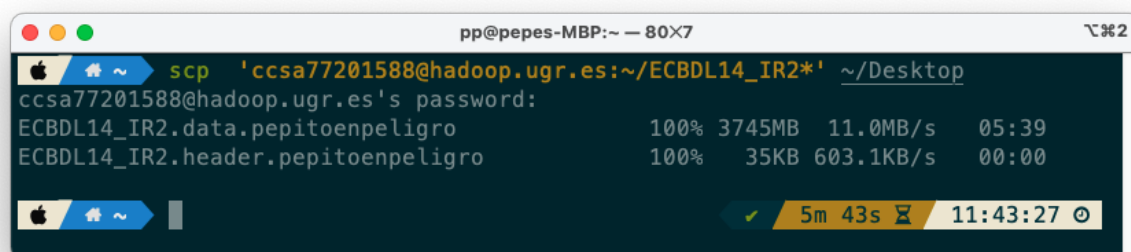
Por tanto, sólo hemos usado ese clúster para la captura del dataset y hemos trabajado en local y posteriormente, en un entorno cloud propio con el proveedor **AWS**.

Por tanto, hemos ido a buscar los ficheros que necesitamos y los hemos capturado:



```
ccsa77201588@hadoop-master: ~  
ccsa77201588@hadoop-master:~$ hdfs dfs -ls -h /user/datasets/ecbd14/ECBDL14_IR2.header  
21/05/25 11:22:24 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...  
using builtin-java classes where applicable  
-rw-r--r-- 2 alberto alberto 34.7 K 2019-05-15 12:17 /user/datasets/ecbd14/ECBDL14_IR2.header  
ccsa77201588@hadoop-master:~$ hdfs dfs -ls -h /user/datasets/ecbd14/ECBDL14_IR2.data  
21/05/25 11:22:30 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...  
using builtin-java classes where applicable  
-rw-r--r-- 2 alberto alberto 3.7 G 2019-05-15 12:17 /user/datasets/ecbd14/ECBDL14_IR2.data  
ccsa77201588@hadoop-master:~$
```

Figura 1.2: Búsqueda de los ficheros de cabecera y datos del dataset que nos corresponde



```
pp@pepes-MBP:~ — 80x7  
scp 'ccsa77201588@hadoop.ugr.es:~/ECBDL14_IR2*' ~/Desktop  
ccsa77201588@hadoop.ugr.es's password:  
ECBDL14_IR2.data.pépitoenpeligro 100% 3745MB 11.0MB/s 05:39  
ECBDL14_IR2.header.pépitoenpeligro 100% 35KB 603.1KB/s 00:00  
✓ 5m 43s 11:43:27
```

Figura 1.3: Descarga de ambos ficheros



Al comienzo, hemos implementado una composición de servicios que se puede encontrar en el anexo y en la entrega, que consiste en tres contenedores de Spark con la configuración de la empresa sevillana Bitnami, para poder realizar pruebas de ejecución antes de montar el sistema cloud que describimos en el siguiente capítulo. Sólo hay que tener en cuenta que hay que instalar el paquete numpy antes de usarlo:



```
docker exec -it --user root pepitoenpeligro_spark_1 pip install numpy  
docker exec -it pepitoenpeligro_spark_1 spark-submit --master spark://spark:7071
```

Capítulo 2

Resolución

2.1 Despliegue de la infraestructura

Para el despliegue de la infraestructura hemos usado el **aws cli** para generar el recurso *Elastic Map Reduce* con tres instancias de tipo *m5.xlarge*, dos de esclavo y una de *master*.

```
1  aws emr create-cluster --applications Name=Spark Name=Zeppelin
   ↳ --ec2-attributes
   ↳ '{"KeyName":"spark","InstanceProfile":"EMR_EC2_DefaultRole",
2    "SubnetId":"subnet-731b7d19",
3    "EmrManagedSlaveSecurityGroup":"sg-0a2bf65c779ae46d3",
4    "EmrManagedMasterSecurityGroup":"sg-0f45869b481f32b74"}'
5
6  --service-role EMR_DefaultRole --enable-debugging --release-label
   ↳ emr-6.3.0 --log-uri 's3n://bucket-pepitoenpeligro/' --name
   ↳ 'PepeCluster' --instance-groups
   ↳ '[{"InstanceCount":1,"EbsConfiguration":{"EbsBlockDeviceConfigs":
7    [{"VolumeSpecification":{"SizeInGB":32,"VolumeType":"gp2"},
8    "VolumesPerInstance":2}]}],
9
   ↳ "InstanceGroupType":"MASTER","InstanceType":"m5.xlarge","Name":"Master
   ↳ Instance Group"}, {"InstanceCount":2,"EbsConfiguration"
10    : {"EbsBlockDeviceConfigs":
11    [{"VolumeSpecification":
12    {"SizeInGB":32,"VolumeType":"gp2"},
13    "VolumesPerInstance":2}]}],
14    "InstanceGroupType":"CORE","InstanceType":"m5.xlarge",
15    "Name":"Core Instance Group"}] '
```

16 `--configurations '[{"Classification":"spark","Properties":{}}]'`
 `--scale-down-behavior TERMINATE_AT_TASK_COMPLETION --region`
 `eu-central-1`

Probamos a conectarnos por ssh a la máquina maestra pero no pudimos porque el puerto 22 está por defecto sólo disponible dentro de la **VPC** de **AWS**.

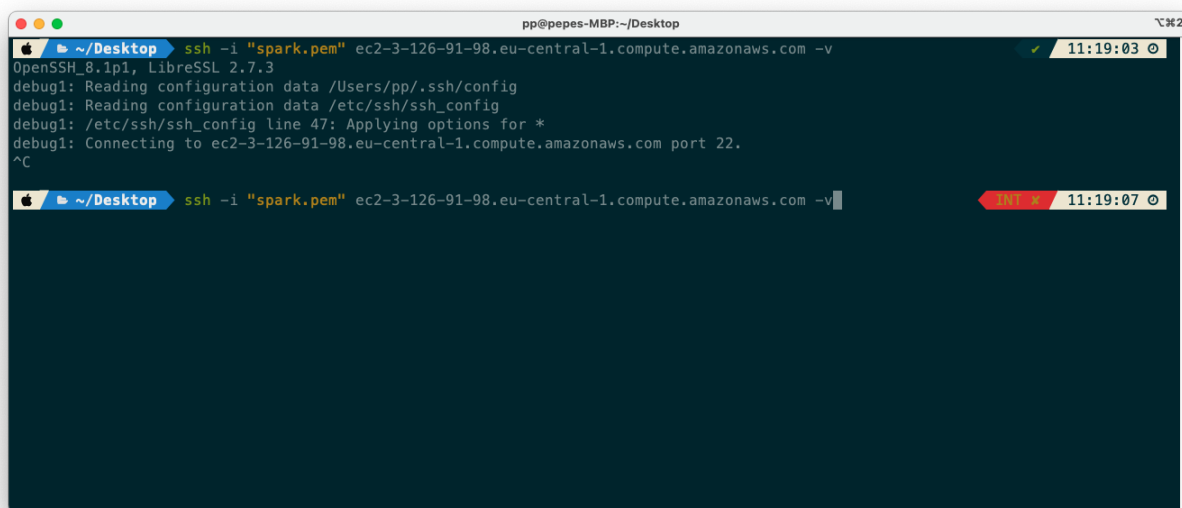


Figura 2.1: Conectando a la máquina maestra del clúster. Acceso denegado

Hemos adaptado las reglas de entrada de la máquina máster para poder conectar por ssh desde fuera de la **VPC** de **AWS**. Por simplicidad y rapidez, hemos ido a realizarlo mediante el cliente web de la consola.

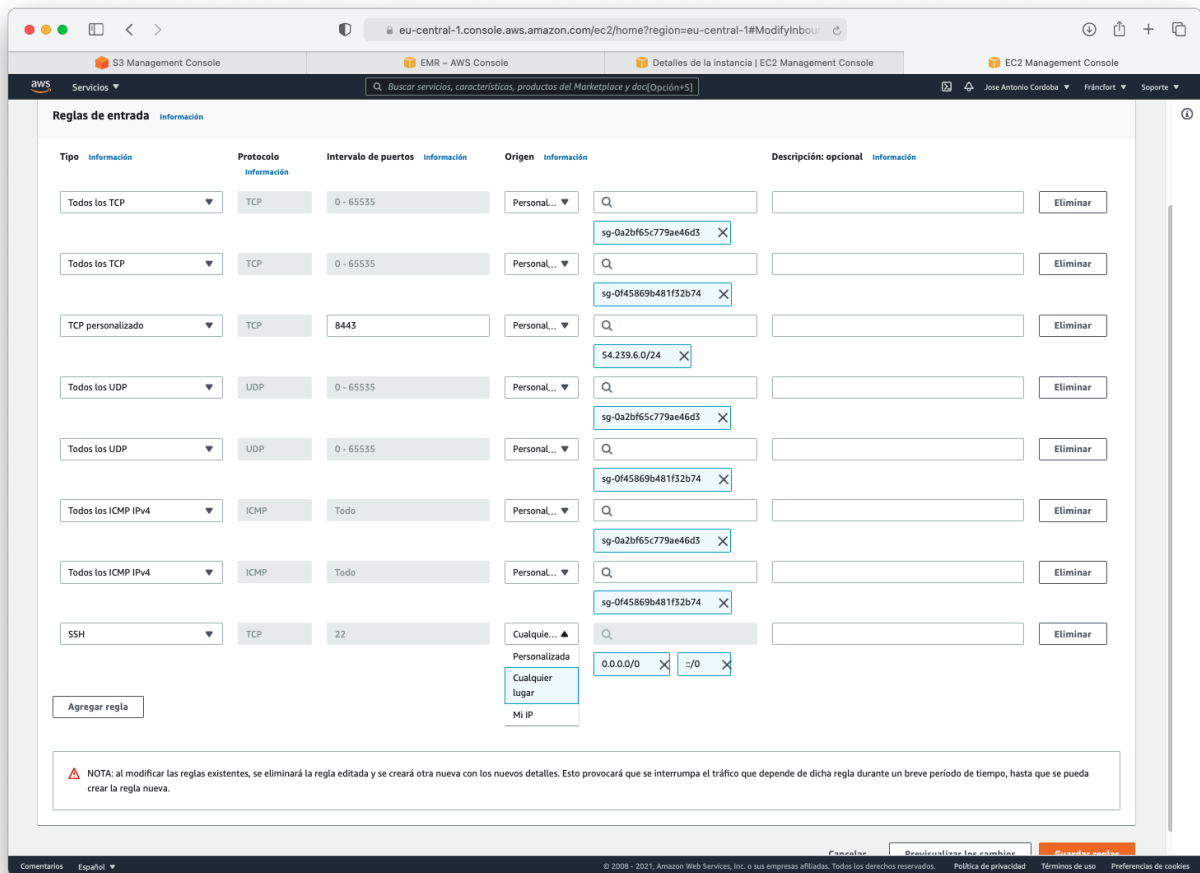


Figura 2.2: Añadir una regla de entrada para permitir acceso al puerto 22 desde cualquier IP

```
ec2-user@ip-172-31-20-57:~$ ssh -i "spark.pem" ec2-user@ec2-3-126-91-98.eu-central-1.compute.amazonaws.com
Last login: Mon May 31 09:23:51 2021 from 85.136.54.242.dyn.user.ono.com

 _ _ | _ _ | _ _ |
 _ _ | ( _ _ | /
 _ _ | \ _ _ | _ _ |

Amazon Linux 2 AMI

https://aws.amazon.com/amazon-linux-2/
17 package(s) needed for security, out of 41 available
Run "sudo yum update" to apply all updates.

EEEEEEEEEEEEEEEEEEEE MMMMMMMM MMMMMMMM RRRRRRRRRRRRRRRR
E:::::::::::::::::::::E M::::::::M M::::::::M R:::::::::R
EE::::::::::::::::::::E M::::::::M M::::::::M R:::::::::R
E:::::E EEEEE M::::::::M M::::::::M RR::::::::R R::::R
E:::::E M::::::::M M::::::::M M::::::::M R::::::::R
E:::::E M::::::::M M::::::::M M::::::::M R::::::::R
E::::::::::::::::::::E M::::::::M M::::::::M M::::::::M R::::::::R
E::::::::::::::::::::E M::::::::M M::::::::M M::::::::M R::::::::R
E:::::E M::::::::M M::::::::M M::::::::M R::::::::R
E:::::E EEEEE M::::::::M M::::::::M M::::::::M R::::::::R
EE::::::::::::::::::::E M::::::::M M::::::::M R::::::::R
E::::::::::::::::::::E M::::::::M M::::::::M RR::::::::R R::::::::R
EEEEEEEEEEEEEEEEEEEE MMMMMMMM MMMMMMMM RRRRRRRR RRRRRR

[ec2-user@ip-172-31-20-57 ~]$
```

Figura 2.3: Conectando a la máquina maestra del clúster. Acceso concedido

2.2 Obtención de columnas

Para la obtención de las columnas que nos interesan hemos subido los ficheros de cabecera y de datos a un bucket S3 de AWS y luego hemos aplicado un proceso de extracción de columnas.

```
aws s3 cp ./ECBDL14_IR2.data s3://bucket-pepitoenpeligro/raw_data/
upload: ./ECBDL14_IR2.header to s3://bucket-pepitoenpeligro/raw_data/ECBDL14_IR2.header
aws s3 cp ./ECBDL14_IR2.data s3://bucket-pepitoenpeligro/raw_data/
Completed 534.8 MiB/3.7 GiB (11.3 MiB/s) with 1 file(s) remaining
```

Figura 2.4: Subiendo los ficheros de cabecera y datos al bucket s3://bucket-pepitoenpeligro

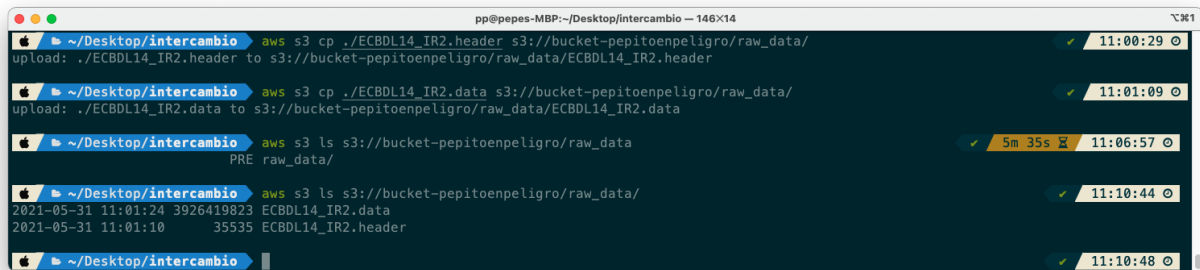


Figura 2.5: Comprobamos que están en el bucket s3://bucket-pepitoenpeligro

A continuación aprovechamos el entorno cloud para directamente leer del bucket definido anteriormente y realizar la extracción de las columnas que nos han sido asignadas:

- PredCN_central_2
- PredSS_r1_4
- PSSM_r1_3_T
- AA_freq_central_M
- PSSM_r2_-1_L
- PSSM_r1_-2_R

Para ello hemos modelado el siguiente script de python que se resumen crear un contexto de ejecución de spark, leer desde el bucker la cabecera y el fichero de datos, realizar la operación de map y reducir con los datos en sí posteriormente, seleccionar las columnas que nos interesan y exportar el dataframe a un bucket.

```

1  import sys
2  import time
3  from pyspark import SparkContext, SparkConf, sql
4  from pyspark.ml.classification import LogisticRegression
5  from functools import reduce
6
7  configurationSpark = SparkConf().setAppName("CC-P4-Preprocesado")
8  sparkContexto = SparkContext.getOrCreate(conf=configurationSpark)
9  sqlContext = sql.SQLContext(sparkContexto)

```

```

10 name_output="pepitoenpeligro"
11
12 if __name__ == "__main__":
13     start = time.time()
14     print("Comenzando el preprocesado")
15     ficheroCabeceras = sparkContext.textFile("
16     s3n://bucket-pepitoenpeligro/raw_data/ECBDL14_IR2.header").collect()
17     cabecerasFiltradas = filter(lambda line: "@attribute" in line
18     ↪ ,ficheroCabeceras)
19     print("Mapeando")
20     mapHeaders = list(map(lambda line: line.split()[1],
21     ↪ cabecerasFiltradas))
22
23     print("Leyendo en un dataframe los datos del dataset pesado")
24     df =
25     ↪ sqlContext.read.csv("s3://bucket-pepitoenpeligro/raw_data/ECBDL14_IR2.data",
26     header=False,sep=",",inferSchema=True)
27     print("Reduciendo")
28     dfReducido = reduce(lambda data, idx:
29     ↪ data.withColumnRenamed(df.schema.names[idx], mapHeaders[idx]),
30     ↪ range(len(df.schema.names)), df)
31
32     dfReducido.createOrReplaceTempView("sql_dataset")
33
34     columns= ['`PredCN_central_2`', '`PredSS_r1_4`', '`PSSM_r1_3_T`',
35     ↪ '`AA_freq_central_M`', '`PSSM_r2_1_L`', '`PSSM_r1_2_R`']
36     print("Seleccionando las columnas {%s, %s, %s, %s, %s, %s}" %
37     ↪ (columns[0], columns[1], columns[2], columns[3], columns[4],
38     ↪ columns[5]))
39     sqlDF = sqlContext.sql('SELECT %s, %s, %s, %s, %s, %s, class FROM
40     ↪ sql_dataset' % (columns[0], columns[1], columns[2], columns[3],
41     ↪ columns[4], columns[5]))
42
43     print("Escribiendo en el fichero csv")
44     sqlDF.write.format('csv').option('header',True)
45     .save('s3n://bucket-pepitoenpeligro/%s' % (name_output))
46
47     print("Fin del preprocesado")
48     print("[3] Hemos Seleccionando las columnas {%s, %s, %s, %s, %s, %s}" %
49     ↪ (columns[0], columns[1], columns[2], columns[3], columns[4],
50     ↪ columns[5]))
51     end = time.time()

```

```

40     print("Tiempo consumido en la seleccion de columnas: %s" % (end -
        ↪ start))
41     sparkContexto.stop()

```

2.3 Modelos

Definimos una proporción de entrenamiento del 80% y 20% de test. Usamos todas las filas del conjunto de datos. Para cada modelo medimos el tiempo que tarda en entrenar y evaluar para poder hacer una comparación más adelante.

2.3.1 Evaluación de los modelos

Para la evaluación de los modelos, hemos definido una función que necesita el modelo, el grid de parámetros, el conjunto de entrenamiento y el conjunto de test. Dentro ajustamos el modelo al conjunto de entrenamiento y obtenemos las predicciones y sacamos los valores de **precisión, f1, auc y recall**.

```

1  def predictions(estimator, paramGrid, dataTrain, dataTest):
2      # binary clasification
3      # https://spark.apache.org/docs/latest/
4      # mllib-evaluation-metrics.html#binary-classification
5      train_validator = TrainValidationSplit(estimator=estimator,
        ↪ estimatorParamMaps=paramGrid,
        ↪ evaluator=BinaryClassificationEvaluator(), trainRatio=portionTrain)
6      model = train_validator.fit(dataTrain)
7      predictions = model.transform(dataTest)
8      predictionAndLabel = predictions.select("prediction", "label")
9
10     # convierte labels y prediccones a float
11     predictionAndLabel = predictionAndLabel.withColumn("prediction",
        ↪ func.round(predictionAndLabel['prediction']).cast('float'))
12     predictionAndLabel = predictionAndLabel.withColumn("label",
        ↪ func.round(predictionAndLabel['label']).cast('float'))
13     metrics=MulticlassMetrics(predictionAndLabel
14     .select("prediction", "label").rdd.map(tuple))
15
16
17     evaluator = BinaryClassificationEvaluator()
18     auRocRF = evaluator.evaluate(predictions)
19
20
21     # la matriz de confusion revienta

```

```

22     cnf_matrix = metrics.confusionMatrix()
23     accuracy = round(metrics.accuracy*100, 3)
24     f1 = metrics.fMeasure(1.0)
25     recall = metrics.recall(1.0)
26
27     print("Results of model %s" % (estimator.__dict__['uid']))
28     print("Accuracy %s" % accuracy)
29     print("F1 %s" % f1)
30     print("Recall %s" % recall)
31     print("AUC %s" % auRocRF)
32
33     return predictions, model

```

2.3.2 Random Forest

```

1  def random_forest_2(trainingData,testData):
2      print("[Random Forest] init")
3      start_time = time()
4      rf = RandomForestClassifier(labelCol="label", featuresCol="features",
5      ↪ seed=12345)
6      # ParamGridBuilder params:
7      # https://spark.apache.org/docs/latest/ml-tuning.html
8      paramGridRF = ParamGridBuilder().addGrid(rf.numTrees, [5, 10,
9      ↪ 20]).addGrid(rf.maxDepth, [2, 3, 6]).build()
10
11     predictionsRF, mRF = predictions(rf,paramGridRF,trainingData,testData)
12     end_time = time()
13     elapsed_time = end_time - start_time
14     print("[Random Forest] With params %s" % paramGridRF)
15     print("[Random Forest] time %s" %(elapsed_time))

```

```
root@ip-172-31-20-57:/home/ec2-user  100%
ize: 2.4 KiB, free: 4.8 GiB)
21/05/31 13:07:23 INFO BlockManagerInfo: Removed broadcast_332_piece0 on ip-172-31-31-43.eu-central-1.compute.internal:37935 in memory (si
ze: 2.4 KiB, free: 4.8 GiB)
21/05/31 13:07:23 INFO BlockManagerInfo: Removed broadcast_295_piece0 on ip-172-31-20-57.eu-central-1.compute.internal:38843 in memory (si
ze: 30.0 KiB, free: 911.6 MiB)
21/05/31 13:07:23 INFO BlockManagerInfo: Removed broadcast_295_piece0 on ip-172-31-22-227.eu-central-1.compute.internal:41567 in memory (s
ize: 30.0 KiB, free: 4.8 GiB)
21/05/31 13:07:23 INFO BlockManagerInfo: Removed broadcast_295_piece0 on ip-172-31-31-43.eu-central-1.compute.internal:37935 in memory (si
ze: 30.0 KiB, free: 4.8 GiB)
21/05/31 13:07:23 INFO BlockManagerInfo: Removed broadcast_334_piece0 on ip-172-31-20-57.eu-central-1.compute.internal:38843 in memory (si
ze: 3.5 KiB, free: 911.6 MiB)
21/05/31 13:07:23 INFO BlockManagerInfo: Removed broadcast_334_piece0 on ip-172-31-22-227.eu-central-1.compute.internal:41567 in memory (s
ize: 3.5 KiB, free: 4.8 GiB)
21/05/31 13:07:23 INFO BlockManagerInfo: Removed broadcast_334_piece0 on ip-172-31-31-43.eu-central-1.compute.internal:37935 in memory (si
ze: 3.5 KiB, free: 4.8 GiB)
21/05/31 13:07:23 INFO BlockManagerInfo: Removed broadcast_303_piece0 on ip-172-31-20-57.eu-central-1.compute.internal:38843 in memory (si
ze: 4.2 KiB, free: 911.6 MiB)
21/05/31 13:07:23 INFO BlockManagerInfo: Removed broadcast_303_piece0 on ip-172-31-22-227.eu-central-1.compute.internal:41567 in memory (s
ize: 4.2 KiB, free: 4.8 GiB)
21/05/31 13:07:23 INFO BlockManagerInfo: Removed broadcast_303_piece0 on ip-172-31-31-43.eu-central-1.compute.internal:37935 in memory (si
ze: 4.2 KiB, free: 4.8 GiB)
Results of model RandomForestClassifier_ac43a8619786
Accuracy 62.203
F1 0.6215991186704432
Recall 0.6201583006260928
AUC 0.654666644864211
[Random Forest] With params [{Param(parent='RandomForestClassifier_ac43a8619786', name='numTrees', doc='Number of trees to train (>= 1).')
: 5, Param(parent='RandomForestClassifier_ac43a8619786', name='maxDepth', doc='Maximum depth of the tree. (>= 0) E.g., depth 0 means 1 lea
f node; depth 1 means 1 internal node + 2 leaf nodes.'): 2}, {Param(parent='RandomForestClassifier_ac43a8619786', name='numTrees', doc='Nu
mber of trees to train (>= 1).'): 5, Param(parent='RandomForestClassifier_ac43a8619786', name='maxDepth', doc='Maximum depth of the tree.
(>= 0) E.g., depth 0 means 1 leaf node; depth 1 means 1 internal node + 2 leaf nodes.'): 3}, {Param(parent='RandomForestClassifier_ac43a86
19786', name='numTrees', doc='Number of trees to train (>= 1).'): 5, Param(parent='RandomForestClassifier_ac43a8619786', name='maxDepth',
doc='Maximum depth of the tree. (>= 0) E.g., depth 0 means 1 leaf node; depth 1 means 1 internal node + 2 leaf nodes.'): 6}, {Param(parent
='RandomForestClassifier_ac43a8619786', name='numTrees', doc='Number of trees to train (>= 1).'): 10, Param(parent='RandomForestClassifier
_ac43a8619786', name='maxDepth', doc='Maximum depth of the tree. (>= 0) E.g., depth 0 means 1 leaf node; depth 1 means 1 internal node + 2
leaf nodes.'): 2}, {Param(parent='RandomForestClassifier_ac43a8619786', name='numTrees', doc='Number of trees to train (>= 1).'): 10, Par
am(parent='RandomForestClassifier_ac43a8619786', name='maxDepth', doc='Maximum depth of the tree. (>= 0) E.g., depth 0 means 1 leaf node;
depth 1 means 1 internal node + 2 leaf nodes.'): 3}, {Param(parent='RandomForestClassifier_ac43a8619786', name='numTrees', doc='Number of
trees to train (>= 1).'): 10, Param(parent='RandomForestClassifier_ac43a8619786', name='maxDepth', doc='Maximum depth of the tree. (>= 0)
E.g., depth 0 means 1 leaf node; depth 1 means 1 internal node + 2 leaf nodes.'): 6}, {Param(parent='RandomForestClassifier_ac43a8619786',
name='numTrees', doc='Number of trees to train (>= 1).'): 20, Param(parent='RandomForestClassifier_ac43a8619786', name='maxDepth', doc='M
aximum depth of the tree. (>= 0) E.g., depth 0 means 1 leaf node; depth 1 means 1 internal node + 2 leaf nodes.'): 2}, {Param(parent='Rand
omForestClassifier_ac43a8619786', name='numTrees', doc='Number of trees to train (>= 1).'): 20, Param(parent='RandomForestClassifier_ac43a
8619786', name='maxDepth', doc='Maximum depth of the tree. (>= 0) E.g., depth 0 means 1 leaf node; depth 1 means 1 internal node + 2 leaf
nodes.'): 3}, {Param(parent='RandomForestClassifier_ac43a8619786', name='numTrees', doc='Number of trees to train (>= 1).'): 20, Param(par
ent='RandomForestClassifier_ac43a8619786', name='maxDepth', doc='Maximum depth of the tree. (>= 0) E.g., depth 0 means 1 leaf node; depth
1 means 1 internal node + 2 leaf nodes.'): 6}]
[Random Forest] time 100.46077370643616
FIN
21/05/31 13:07:23 INFO BlockManagerInfo: Removed broadcast_281_piece0 on ip-172-31-20-57.eu-central-1.compute.internal:38843 in memory (si
ze: 35.1 KiB, free: 911.6 MiB)
21/05/31 13:07:23 INFO BlockManagerInfo: Removed broadcast_309_piece0 on ip-172-31-20-57.eu-central-1.compute.internal:38843 in memory (si
```

Figura 2.6: Resultados del modelo Random Forest con los primeros parámetros

```
1 def random_forest_2(trainingData,testData):
2     print("[Random Forest] init")
3     start_time = time()
4     rf = RandomForestClassifier(labelCol="label", featuresCol="features",
5                               ↪ seed=2021)
6     # ParamGridBuilder params:
7     # https://spark.apache.org/docs/latest/ml-tuning.html
```



```

7      paramGridRF = ParamGridBuilder().addGrid(rf.numTrees, [10, 30,
      ↪ 60]).addGrid(rf.maxDepth, [3, 6, 12]).build()
8      predictionsRF, mRF = predictions(rf,paramGridRF,trainingData,testData)
9      end_time = time()
10     elapsed_time = end_time - start_time
11     print("[Random Forest] With params %s" % paramGridRF)
12     print("[Random Forest] time %s" %(elapsed_time))

```

```

root@ip-172-31-20-57:/home/ec2-user
21/05/31 12:38:22 INFO MapOutputTrackerMasterEndpoint: Asked to send map output locations for shuffle 255 to 172.31.31.43:45324
21/05/31 12:38:22 INFO TaskSetManager: Finished task 0.0 in stage 1272.0 (TID 1266) in 6 ms on ip-172-31-22-227.eu-central-1.compute.intern
al (executor 1) (1/2)
21/05/31 12:38:22 INFO TaskSetManager: Finished task 1.0 in stage 1272.0 (TID 1267) in 7 ms on ip-172-31-31-43.eu-central-1.compute.intern
al (executor 2) (2/2)
21/05/31 12:38:22 INFO YarnScheduler: Removed TaskSet 1272.0, whose tasks have all completed, from pool
21/05/31 12:38:22 INFO DAGScheduler: ResultStage 1272 (collectAsMap at MulticlassMetrics.scala:61) finished in 0,009 s
21/05/31 12:38:22 INFO DAGScheduler: Job 342 is finished. Cancelling potential speculative or zombie tasks for this job
21/05/31 12:38:22 INFO YarnScheduler: Killing all running tasks in stage 1272: Stage finished
21/05/31 12:38:22 INFO DAGScheduler: Job 342 finished: collectAsMap at MulticlassMetrics.scala:61, took 5,087786 s
Results of model RandomForestClassifier_3d2444fb22f4
Accuracy 62.708
F1 0.6230463927593538
Recall 0.6170545125405121
AUC 0.6665406425074919
[Random Forest] With params [{Param(parent='RandomForestClassifier_3d2444fb22f4', name='numTrees', doc='Number of trees to train (>= 1).')
: 10, Param(parent='RandomForestClassifier_3d2444fb22f4', name='maxDepth', doc='Maximum depth of the tree. (>= 0) E.g., depth 0 means 1 le
af node; depth 1 means 1 internal node + 2 leaf nodes.'): 3}, {Param(parent='RandomForestClassifier_3d2444fb22f4', name='numTrees', doc='N
umber of trees to train (>= 1).'): 10, Param(parent='RandomForestClassifier_3d2444fb22f4', name='maxDepth', doc='Maximum depth of the tree
. (>= 0) E.g., depth 0 means 1 leaf node; depth 1 means 1 internal node + 2 leaf nodes.'): 6}, {Param(parent='RandomForestClassifier_3d244
4fb22f4', name='numTrees', doc='Number of trees to train (>= 1).'): 10, Param(parent='RandomForestClassifier_3d2444fb22f4', name='maxDepth
', doc='Maximum depth of the tree. (>= 0) E.g., depth 0 means 1 leaf node; depth 1 means 1 internal node + 2 leaf nodes.'): 12}, {Param(pa
rent='RandomForestClassifier_3d2444fb22f4', name='numTrees', doc='Number of trees to train (>= 1).'): 30, Param(parent='RandomForestClassi
fier_3d2444fb22f4', name='maxDepth', doc='Maximum depth of the tree. (>= 0) E.g., depth 0 means 1 leaf node; depth 1 means 1 internal node
+ 2 leaf nodes.'): 3}, {Param(parent='RandomForestClassifier_3d2444fb22f4', name='numTrees', doc='Number of trees to train (>= 1).'): 30,
Param(parent='RandomForestClassifier_3d2444fb22f4', name='maxDepth', doc='Maximum depth of the tree. (>= 0) E.g., depth 0 means 1 leaf no
de; depth 1 means 1 internal node + 2 leaf nodes.'): 6}, {Param(parent='RandomForestClassifier_3d2444fb22f4', name='numTrees', doc='Number
of trees to train (>= 1).'): 30, Param(parent='RandomForestClassifier_3d2444fb22f4', name='maxDepth', doc='Maximum depth of the tree. (>=
0) E.g., depth 0 means 1 leaf node; depth 1 means 1 internal node + 2 leaf nodes.'): 12}, {Param(parent='RandomForestClassifier_3d2444fb2
2f4', name='numTrees', doc='Number of trees to train (>= 1).'): 60, Param(parent='RandomForestClassifier_3d2444fb22f4', name='maxDepth', d
oc='Maximum depth of the tree. (>= 0) E.g., depth 0 means 1 leaf node; depth 1 means 1 internal node + 2 leaf nodes.'): 3}, {Param(parent=
'RandomForestClassifier_3d2444fb22f4', name='numTrees', doc='Number of trees to train (>= 1).'): 60, Param(parent='RandomForestClassifier_
3d2444fb22f4', name='maxDepth', doc='Maximum depth of the tree. (>= 0) E.g., depth 0 means 1 leaf node; depth 1 means 1 internal node + 2
leaf nodes.'): 6}, {Param(parent='RandomForestClassifier_3d2444fb22f4', name='numTrees', doc='Number of trees to train (>= 1).'): 60, Para
m(parent='RandomForestClassifier_3d2444fb22f4', name='maxDepth', doc='Maximum depth of the tree. (>= 0) E.g., depth 0 means 1 leaf node; de
pth 1 means 1 internal node + 2 leaf nodes.'): 12}]
[Random Forest] time 451.3554310798645
FIN
21/05/31 12:38:22 INFO SparkContext: Invoking stop() from shutdown hook
21/05/31 12:38:22 INFO AbstractConnector: Stopped Spark@2a018c36{HTTP/1.1, (http/1.1)}{0.0.0.0:4040}
21/05/31 12:38:22 INFO SparkUI: Stopped Spark web UI at http://ip-172-31-20-57.eu-central-1.compute.internal:4040
21/05/31 12:38:22 INFO YarnClientSchedulerBackend: Interrupting monitor thread
21/05/31 12:38:22 INFO YarnClientSchedulerBackend: Shutting down all executors
21/05/31 12:38:22 INFO YarnSchedulerBackend$YarnDriverEndpoint: Asking each executor to shut down
21/05/31 12:38:22 INFO YarnClientSchedulerBackend: YARN client scheduler backend Stopped
21/05/31 12:38:22 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
21/05/31 12:38:22 INFO MemoryStore: MemoryStore cleared
21/05/31 12:38:22 INFO BlockManager: BlockManager stopped
21/05/31 12:38:22 INFO BlockManagerMaster: BlockManagerMaster stopped
21/05/31 12:38:22 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
21/05/31 12:38:22 INFO SparkContext: Successfully stopped SparkContext
21/05/31 12:38:22 INFO ShutdownHookManager: Shutdown hook called

```

Figura 2.7: Resultados del modelo Random Forest con otros parámetros

2.3.3 Gradient Boosted Tree

```
1 def gradient_boosted_tree_1(trainingData, testData):
2     print("[Gradient Boosted Tree] init")
3     start_time = time()
4     gbt = GBTClassifier(labelCol="label", featuresCol="features",
5         ↪ seed=2021)
6     #paramGridGBT = ParamGridBuilder().addGrid(gbt.maxIter, [10, 15,
7         ↪ 20]).addGrid(gbt.maxDepth, [3, 6, 12]).build()
8     paramGridGBT = ParamGridBuilder().addGrid(gbt.maxIter, [5, 10,
9         ↪ 15]).addGrid(gbt.maxDepth, [2, 3, 9]).build()
10    predictionsGBT, mGBT =
11    ↪ predictions(gbt,paramGridGBT,trainingData,testData)
12    end_time = time()
13    elapsed_time = end_time - start_time
14    print("[Gradient Boosted Tree] With params %s" % paramGridGBT)
15    print("[Gradient Boosted Tree] time %s" %(elapsed_time))
```

```

root@ip-172-31-20-57:/home/ec2-user
21/05/31 21:14:24 INFO MemoryStore: Block broadcast_1856 stored as values in memory (estimated size 4.0 KiB, free 907.6 MiB)
21/05/31 21:14:24 INFO MemoryStore: Block broadcast_1856_piece0 stored as bytes in memory (estimated size 2.3 KiB, free 907.6 MiB)
21/05/31 21:14:24 INFO BlockManagerInfo: Added broadcast_1856_piece0 in memory on ip-172-31-20-57.eu-central-1.compute.internal:43671 (size: 2.3 KiB, free: 911.2 MiB)
21/05/31 21:14:24 INFO SparkContext: Created broadcast 1856 from broadcast at DAGScheduler.scala:1479
21/05/31 21:14:24 INFO DAGScheduler: Submitting 2 missing tasks from ResultStage 2593 (ShuffledRDD[3077] at reduceByKey at MulticlassMetrics.scala:61) (first 15 tasks are f
or partitions Vector(0, 1))
21/05/31 21:14:24 INFO YarnScheduler: Adding task set 2593.0 with 2 tasks resource profile 0
21/05/31 21:14:24 INFO TaskSetManager: Starting task 0.0 in stage 2593.0 (TID 2663) (ip-172-31-22-227.eu-central-1.compute.internal, executor 2, partition 0, NODE_LOCAL, 46
18 bytes) taskResourceAssignments Map()
21/05/31 21:14:24 INFO TaskSetManager: Starting task 1.0 in stage 2593.0 (TID 2664) (ip-172-31-31-43.eu-central-1.compute.internal, executor 1, partition 1, NODE_LOCAL, 461
8 bytes) taskResourceAssignments Map()
21/05/31 21:14:24 INFO BlockManagerInfo: Added broadcast_1856_piece0 in memory on ip-172-31-22-227.eu-central-1.compute.internal:44487 (size: 2.3 KiB, free: 4.8 GiB)
21/05/31 21:14:24 INFO MapOutputTrackerMasterEndpoint: Asked to send map output locations for shuffle 621 to 172.31.22.227:40412
21/05/31 21:14:24 INFO MapOutputTrackerMasterEndpoint: Asked to send map output locations for shuffle 621 to 172.31.31.43:52324
21/05/31 21:14:24 INFO TaskSetManager: Finished task 1.0 in stage 2593.0 (TID 2664) in 6 ms on ip-172-31-31-43.eu-central-1.compute.internal (executor 1) (1/2)
21/05/31 21:14:24 INFO TaskSetManager: Finished task 0.0 in stage 2593.0 (TID 2663) in 6 ms on ip-172-31-22-227.eu-central-1.compute.internal (executor 2) (2/2)
21/05/31 21:14:24 INFO YarnScheduler: Removed TaskSet 2593.0, whose tasks have all completed, from pool
21/05/31 21:14:24 INFO DAGScheduler: ResultStage 2593 (collectASMap at MulticlassMetrics.scala:61) finished in 0.009 s
21/05/31 21:14:24 INFO DAGScheduler: Job 665 is finished. Cancelling potential speculative or zombie tasks for this job
21/05/31 21:14:24 INFO YarnScheduler: Killing all running tasks in stage 2593: Stage finished
21/05/31 21:14:24 INFO DAGScheduler: Job 665 finished: collectASMap at MulticlassMetrics.scala:61, took 3.143727 s
Results of model GBTClassifier_0783115b7593
Accuracy 62.68
F1 0.6223499661854216
Recall 0.6157639409124961
AUC 0.6652824575904002
[Gradient Boosted Tree] With params [(Param(parent='GBTClassifier_0783115b7593', name='maxIter', doc='max number of iterations (>= 0).'): 5, Param(parent='GBTClassifier_078
3115b7593', name='maxDepth', doc='Maximum depth of the tree. (>= 0) E.g., depth 0 means 1 leaf node; depth 1 means 1 internal node + 2 leaf nodes.'): 2), (Param(parent='GBT
Classifier_0783115b7593', name='maxIter', doc='max number of iterations (>= 0).'): 5, Param(parent='GBTClassifier_0783115b7593', name='maxDepth', doc='Maximum depth of the
tree. (>= 0) E.g., depth 0 means 1 leaf node; depth 1 means 1 internal node + 2 leaf nodes.'): 3), (Param(parent='GBTClassifier_0783115b7593', name='maxIter', doc='max numb
er of iterations (>= 0).'): 5, Param(parent='GBTClassifier_0783115b7593', name='maxDepth', doc='Maximum depth of the tree. (>= 0) E.g., depth 0 means 1 leaf node; depth 1 m
eans 1 internal node + 2 leaf nodes.'): 9), (Param(parent='GBTClassifier_0783115b7593', name='maxIter', doc='max number of iterations (>= 0).'): 10, Param(parent='GBTClassifi
er_0783115b7593', name='maxDepth', doc='Maximum depth of the tree. (>= 0) E.g., depth 0 means 1 leaf node; depth 1 means 1 internal node + 2 leaf nodes.'): 2), (Param(par
ent='GBTClassifier_0783115b7593', name='maxIter', doc='max number of iterations (>= 0).'): 10, Param(parent='GBTClassifier_0783115b7593', name='maxDepth', doc='Maximum dept
h of the tree. (>= 0) E.g., depth 0 means 1 leaf node; depth 1 means 1 internal node + 2 leaf nodes.'): 3), (Param(parent='GBTClassifier_0783115b7593', name='maxIter', doc=
'max number of iterations (>= 0).'): 10, Param(parent='GBTClassifier_0783115b7593', name='maxDepth', doc='Maximum depth of the tree. (>= 0) E.g., depth 0 means 1 leaf node;
depth 1 means 1 internal node + 2 leaf nodes.'): 9), (Param(parent='GBTClassifier_0783115b7593', name='maxIter', doc='max number of iterations (>= 0).'): 15, Param(parent='
GBTClassifier_0783115b7593', name='maxDepth', doc='Maximum depth of the tree. (>= 0) E.g., depth 0 means 1 leaf node; depth 1 means 1 internal node + 2 leaf nodes.'): 2),
(Param(parent='GBTClassifier_0783115b7593', name='maxIter', doc='max number of iterations (>= 0).'): 15, Param(parent='GBTClassifier_0783115b7593', name='maxDepth', doc='Ma
ximum depth of the tree. (>= 0) E.g., depth 0 means 1 leaf node; depth 1 means 1 internal node + 2 leaf nodes.'): 3), (Param(parent='GBTClassifier_0783115b7593', name='maxI
ter', doc='max number of iterations (>= 0).'): 15, Param(parent='GBTClassifier_0783115b7593', name='maxDepth', doc='Maximum depth of the tree. (>= 0) E.g., depth 0 means 1
leaf node; depth 1 means 1 internal node + 2 leaf nodes.'): 9)]
[Gradient Boosted Tree] time 199.18023228645325
FIN
21/05/31 21:14:24 INFO SparkContext: Invoking stop() from shutdown hook
21/05/31 21:14:24 INFO AbstractConnector: Stopped Spark@Sec8166a(HTTP/1.1, (http://172.31.20.57:4040))
21/05/31 21:14:24 INFO SparkUI: Stopped Spark web UI at http://ip-172-31-20-57.eu-central-1.compute.internal:4040
21/05/31 21:14:24 INFO YarnClientSchedulerBackend: Interrupting monitor thread
21/05/31 21:14:24 INFO YarnClientSchedulerBackend: Shutting down all executors
21/05/31 21:14:24 INFO YarnSchedulerBackend$YarnDriverEndpoint: Asking each executor to shut down
21/05/31 21:14:24 INFO YarnClientSchedulerBackend: YARN client scheduler backend Stopped

```

Figura 2.8: Resultados del modelo Gradient Boost Tree con los primeros parámetros

```

1 def gradient_boosted_tree_2(trainingData, testData):
2     print("[Gradient Boosted Tree] init")
3     start_time = time()
4     gbt = GBTClsifier(labelCol="label", featuresCol="features",
5         ↪ seed=2021)
6     paramGridGBT = ParamGridBuilder().addGrid(gbt.maxIter, [10, 15,
7         ↪ 20]).addGrid(gbt.maxDepth, [3, 6, 12]).build()
8
9     predictionsGBT, mGBT =
10    ↪ predictions(gbt,paramGridGBT,trainingData,testData)
11
12     end_time = time()
13     elapsed_time = end_time - start_time
14     print("[Gradient Boosted Tree] With params %s" % paramGridGBT)

```

```
11 print("[Gradient Boosted Tree] time %s" %(elapsed_time))
```

```
or partitions Vector(0, 1))
21/05/31 21:36:32 INFO YarnScheduler: Adding task set 4633.0 with 2 tasks resource profile 0
21/05/31 21:36:32 INFO TaskSetManager: Starting task 0.0 in stage 4633.0 (TID 4703) (ip-172-31-22-227.eu-central-1.compute.internal, executor 2, partition 0, NODE_LOCAL, 46
18 bytes) taskResourceAssignments Map()
21/05/31 21:36:32 INFO TaskSetManager: Starting task 1.0 in stage 4633.0 (TID 4704) (ip-172-31-31-43.eu-central-1.compute.internal, executor 1, partition 1, NODE_LOCAL, 461
8 bytes) taskResourceAssignments Map()
21/05/31 21:36:32 INFO BlockManagerInfo: Added broadcast_3386_piece0 in memory on ip-172-31-22-227.eu-central-1.compute.internal:34663 (size: 2.3 KiB, free: 4.8 GiB)
21/05/31 21:36:32 INFO BlockManagerInfo: Added broadcast_3386_piece0 in memory on ip-172-31-31-43.eu-central-1.compute.internal:34331 (size: 2.3 KiB, free: 4.8 GiB)
21/05/31 21:36:32 INFO MapOutputTrackerMasterEndpoint: Asked to send map output locations for shuffle 1131 to 172.31.22.227:48892
21/05/31 21:36:32 INFO MapOutputTrackerMasterEndpoint: Asked to send map output locations for shuffle 1131 to 172.31.31.43:42986
21/05/31 21:36:32 INFO TaskSetManager: Finished task 0.0 in stage 4633.0 (TID 4703) in 6 ms on ip-172-31-22-227.eu-central-1.compute.internal (executor 2) (1/2)
21/05/31 21:36:32 INFO TaskSetManager: Finished task 1.0 in stage 4633.0 (TID 4704) in 6 ms on ip-172-31-31-43.eu-central-1.compute.internal (executor 1) (2/2)
21/05/31 21:36:32 INFO YarnScheduler: Removed TaskSet 4633.0, whose tasks have all completed, from pool
21/05/31 21:36:32 INFO DAGScheduler: ResultStage 4633 (collectASMap at MulticlassMetrics.scala:61) finished in 0,008 s
21/05/31 21:36:32 INFO DAGScheduler: Job 1175 is finished. Cancelling potential speculative or zombie tasks for this job
21/05/31 21:36:32 INFO YarnScheduler: Killing all running tasks in stage 4633: Stage finished
21/05/31 21:36:32 INFO DAGScheduler: Job 1175 finished: collectASMap at MulticlassMetrics.scala:61, took 3,154722 s
Results of model GBTCClassifier_234b195706b7
Accuracy 62.48
F1 0.616575378580768
Recall 0.605748317690163
AUC 0.6607271755809723
[Gradient Boosted Tree] With params [(Param(parent='GBTCClassifier_234b195706b7', name='maxIter', doc='max number of iterations (>= 0.): 10, Param(parent='GBTCClassifier_23
4b195706b7', name='maxDepth', doc='Maximum depth of the tree. (>= 0) E.g., depth 0 means 1 leaf node; depth 1 means 1 internal node + 2 leaf nodes.'): 3), {Param(parent='GB
TCClassifier_234b195706b7', name='maxIter', doc='max number of iterations (>= 0.): 10, Param(parent='GBTCClassifier_234b195706b7', name='maxDepth', doc='Maximum depth of th
e tree. (>= 0) E.g., depth 0 means 1 leaf node; depth 1 means 1 internal node + 2 leaf nodes.'): 6}, {Param(parent='GBTCClassifier_234b195706b7', name='maxIter', doc='max nu
mber of iterations (>= 0.): 10, Param(parent='GBTCClassifier_234b195706b7', name='maxDepth', doc='Maximum depth of the tree. (>= 0) E.g., depth 0 means 1 leaf node; depth
1 means 1 internal node + 2 leaf nodes.'): 12}, {Param(parent='GBTCClassifier_234b195706b7', name='maxIter', doc='max number of iterations (>= 0.): 15, Param(parent='GBTCCL
assifier_234b195706b7', name='maxDepth', doc='Maximum depth of the tree. (>= 0) E.g., depth 0 means 1 leaf node; depth 1 means 1 internal node + 2 leaf nodes.'): 3}, {Param
(parent='GBTCClassifier_234b195706b7', name='maxIter', doc='max number of iterations (>= 0.): 15, Param(parent='GBTCClassifier_234b195706b7', name='maxDepth', doc='Maximum
depth of the tree. (>= 0) E.g., depth 0 means 1 leaf node; depth 1 means 1 internal node + 2 leaf nodes.'): 6}, {Param(parent='GBTCClassifier_234b195706b7', name='maxIter',
doc='max number of iterations (>= 0.): 15, Param(parent='GBTCClassifier_234b195706b7', name='maxDepth', doc='Maximum depth of the tree. (>= 0) E.g., depth 0 means 1 leaf n
ode; depth 1 means 1 internal node + 2 leaf nodes.'): 12}, {Param(parent='GBTCClassifier_234b195706b7', name='maxIter', doc='max number of iterations (>= 0.): 20, Param(pa
rent='GBTCClassifier_234b195706b7', name='maxDepth', doc='Maximum depth of the tree. (>= 0) E.g., depth 0 means 1 leaf node; depth 1 means 1 internal node + 2 leaf nodes.'):
3}, {Param(parent='GBTCClassifier_234b195706b7', name='maxIter', doc='max number of iterations (>= 0.): 20, Param(parent='GBTCClassifier_234b195706b7', name='maxDepth', do
c='Maximum depth of the tree. (>= 0) E.g., depth 0 means 1 leaf node; depth 1 means 1 internal node + 2 leaf nodes.'): 6}, {Param(parent='GBTCClassifier_234b195706b7', name=
'maxIter', doc='max number of iterations (>= 0.): 20, Param(parent='GBTCClassifier_234b195706b7', name='maxDepth', doc='Maximum depth of the tree. (>= 0) E.g., depth 0 mea
ns 1 leaf node; depth 1 means 1 internal node + 2 leaf nodes.'): 12}]
[Gradient Boosted Tree] time 359.99056029319763
FIN
21/05/31 21:36:32 INFO SparkContext: Invoking stop() from shutdown hook
21/05/31 21:36:32 INFO AbstractConnector: Stopped Spark@4962fa87(HTTP/1.1, (http://1.1.1.1){0.0.0.0:4040})
21/05/31 21:36:32 INFO SparkUI: Stopped Spark web UI at http://ip-172-31-20-57.eu-central-1.compute.internal:4040
21/05/31 21:36:32 INFO YarnClientSchedulerBackend: Interrupting monitor thread
21/05/31 21:36:32 INFO YarnClientSchedulerBackend: Shutting down all executors
21/05/31 21:36:32 INFO YarnSchedulerBackend$YarnDriverEndpoint: Asking each executor to shut down
21/05/31 21:36:32 INFO YarnClientSchedulerBackend: YARN client scheduler backend Stopped
21/05/31 21:36:33 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
21/05/31 21:36:33 INFO MemoryStore: MemoryStore cleared
21/05/31 21:36:33 INFO BlockManager: BlockManager stopped
21/05/31 21:36:33 INFO BlockManagerMaster: BlockManagerMaster stopped
21/05/31 21:36:33 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
```

Figura 2.9: Resultados del modelo Gradient Boost Tree con otros parámetros

2.3.4 Logistic Regression

```
1 def logistic_regression1(trainingData, testData):
2     print("[Logistic Regression] init")
3     start_time = time()
4     lr =
5         LogisticRegression(featuresCol="features", labelCol="label", maxIter=100, family="multinomial")
6     lrGrid = ParamGridBuilder().addGrid(lr.regParam, [0.1, 0.01, 0.001]).addGrid(lr.elasticNetParam, [0.5, 0.6, 0.8]).build()
7     predictionsRL, mRL = predictions(lr, lrGrid, trainingData, testData)
8     end_time = time()
```

```

8     elapsed_time = end_time - start_time
9     print("[Logistic Regression] With params %s" % predictionsRL)
10    print("[Logistic Regression] time %s" %(elapsed_time))

```

```

root@ip-172-31-20-57:/home/ec2-user
21/05/31 13:58:04 INFO DAGSchedulr: ShuffleMapStage 926 (map at MulticlassMetrics.scala:52) finished in 2,678 s
21/05/31 13:58:04 INFO DAGSchedulr: looking for newly runnable stages
21/05/31 13:58:04 INFO DAGSchedulr: running: Set()
21/05/31 13:58:04 INFO DAGSchedulr: waiting: Set(ResultStage 927)
21/05/31 13:58:04 INFO DAGSchedulr: failed: Set()
21/05/31 13:58:04 INFO DAGSchedulr: Submitting ResultStage 927 (ShuffledRDD[1036] at reduceByKey at MulticlassMetrics.scala:61), which has no missing parents
21/05/31 13:58:04 INFO MemoryStore: Block broadcast_581 stored as values in memory (estimated size 4.0 KiB, free 907.2 MiB)
21/05/31 13:58:04 INFO MemoryStore: Block broadcast_581_piece0 stored as bytes in memory (estimated size 2.3 KiB, free 907.2 MiB)
21/05/31 13:58:04 INFO BlockManagerInfo: Added broadcast_581_piece0 in memory on ip-172-31-20-57.eu-central-1.compute.internal:46759 (size: 2.3 KiB, free: 911.8 MiB)
21/05/31 13:58:04 INFO SparkContext: Created broadcast 581 from broadcast at DAGSchedulr.scala:1479
21/05/31 13:58:04 INFO DAGSchedulr: Submitting 2 missing tasks from ResultStage 927 (ShuffledRDD[1036] at reduceByKey at MulticlassMetrics.scala:61) (first 15 tasks are for partitions Vector(0, 1))
21/05/31 13:58:04 INFO YarnScheduler: Adding task set 927.0 with 2 tasks resource profile 0
21/05/31 13:58:04 INFO TaskSetManager: Starting task 0.0 in stage 927.0 (TID 800) (ip-172-31-31-43.eu-central-1.compute.internal, executor 2, partition 0, NODE_LOCAL, 4618 bytes) taskResourceAssignments Map()
21/05/31 13:58:04 INFO TaskSetManager: Starting task 1.0 in stage 927.0 (TID 801) (ip-172-31-22-227.eu-central-1.compute.internal, executor 1, partition 1, NODE_LOCAL, 4618 bytes) taskResourceAssignments Map()
21/05/31 13:58:04 INFO BlockManagerInfo: Added broadcast_581_piece0 in memory on ip-172-31-31-43.eu-central-1.compute.internal:32829 (size: 2.3 KiB, free: 4.8 GiB)
21/05/31 13:58:04 INFO BlockManagerInfo: Added broadcast_581_piece0 in memory on ip-172-31-22-227.eu-central-1.compute.internal:40579 (size: 2.3 KiB, free: 4.8 GiB)
21/05/31 13:58:04 INFO MapOutputTrackerMasterEndpoint: Asked to send map output locations for shuffle 60 to 172.31.31.43:53294
21/05/31 13:58:04 INFO MapOutputTrackerMasterEndpoint: Asked to send map output locations for shuffle 60 to 172.31.22.227:48092
21/05/31 13:58:04 INFO TaskSetManager: Finished task 1.0 in stage 927.0 (TID 801) in 8 ms on ip-172-31-22-227.eu-central-1.compute.internal (executor 1) (1/2)
21/05/31 13:58:04 INFO TaskSetManager: Finished task 0.0 in stage 927.0 (TID 800) in 8 ms on ip-172-31-31-43.eu-central-1.compute.internal (executor 2) (2/2)
21/05/31 13:58:04 INFO YarnScheduler: Removed TaskSet 927.0, whose tasks have all completed, from pool
21/05/31 13:58:04 INFO DAGSchedulr: ResultStage 927 (collectAsMap at MulticlassMetrics.scala:61) finished in 0,010 s
21/05/31 13:58:04 INFO DAGSchedulr: Job 298 is finished. Cancelling potential speculative or zombie tasks for this job
21/05/31 13:58:04 INFO YarnScheduler: Killing all running tasks in stage 927: Stage finished
21/05/31 13:58:04 INFO DAGSchedulr: Job 298 finished: collectAsMap at MulticlassMetrics.scala:61, took 2,693090 s
Results of model LogisticRegression_de785cf88721
Accuracy 55.294
F1 0.5881916252566529
Recall 0.6383517695903377
AUC 0.5633849535791051
[Logistic Regression] With params DataFrame[features: vector, label: int, rawPrediction: vector, probability: vector, prediction: double]
[Logistic Regression] time 79.06433081626892
FIN
21/05/31 13:58:04 INFO SparkContext: Invoking stop() from shutdown hook
21/05/31 13:58:04 INFO AbstractConnector: Stopped Spark@69c9f77(HTTP/1.1, (http/1.1)){0.0.0.0:4040}
21/05/31 13:58:04 INFO SparkUI: Stopped Spark web UI at http://ip-172-31-20-57.eu-central-1.compute.internal:4040
21/05/31 13:58:04 INFO YarnClientSchedulerBackend: Interrupting monitor thread
21/05/31 13:58:04 INFO YarnClientSchedulerBackend: Shutting down all executors
21/05/31 13:58:04 INFO YarnSchedulerBackend$YarnDriverEndpoint: Asking each executor to shut down
21/05/31 13:58:04 INFO YarnClientSchedulerBackend: YARN client scheduler backend Stopped
21/05/31 13:58:04 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
21/05/31 13:58:04 INFO MemoryStore: MemoryStore cleared
21/05/31 13:58:04 INFO BlockManager: BlockManager stopped

```

Figura 2.10: Resultados del modelo de regresión logística con los primeros parámetros

```

1  def logistic_regression_2(trainingData, testData):
2      print("[Logistic Regression] init")
3      start_time = time()

```

```

4      lr =
      ↳ LogisticRegression(featuresCol="features",labelCol="label",maxIter=100,family="mul
5      lrGrid = ParamGridBuilder().addGrid(lr.regParam, [0.1, 0.01,
      ↳ 0.001]).addGrid(lr.elasticNetParam, [0.6, 0.7, 0.9]).build()
6      predictionsRL, mRL = predictions(lr,lrGrid,trainingData,testData)
7      end_time = time()
8      elapsed_time = end_time - start_time
9      print("[Logistic Regression] With params %s" % predictionsRL)
10     print("[Logistic Regression] time %s" %(elapsed_time))

```

```

root@ip-172-31-20-57:/home/ec2-user
21/05/31 22:06:34 INFO DAGSchedulr: looking for newly runnable stages
21/05/31 22:06:34 INFO DAGSchedulr: running: Set()
21/05/31 22:06:34 INFO DAGSchedulr: waiting: Set(ResultStage 966)
21/05/31 22:06:34 INFO DAGSchedulr: failed: Set()
21/05/31 22:06:34 INFO DAGSchedulr: Submitting ResultStage 966 (ShuffledRDD[1049] at reduceByKey at MulticlassMetrics.scala:61), which has no missing parents
21/05/31 22:06:34 INFO MemoryStore: Block broadcast_607 stored as values in memory (estimated size 4.0 KiB, free 903.8 MiB)
21/05/31 22:06:34 INFO MemoryStore: Block broadcast_607_piece0 stored as bytes in memory (estimated size 2.3 KiB, free 903.8 MiB)
21/05/31 22:06:34 INFO BlockManagerInfo: Added broadcast_607_piece0 in memory on ip-172-31-20-57.eu-central-1.compute.internal:39889 (size: 2.3 KiB, free: 911.3 MiB)
21/05/31 22:06:34 INFO SparkContext: Created broadcast 607 from broadcast at DAGSchedulr.scala:1479
21/05/31 22:06:34 INFO DAGSchedulr: Submitting 2 missing tasks from ResultStage 966 (ShuffledRDD[1049] at reduceByKey at MulticlassMetrics.scala:61) (first 15 tasks are fo
r partitions Vector(0, 1))
21/05/31 22:06:34 INFO YarnScheduler: Adding task set 966.0 with 2 tasks resource profile 0
21/05/31 22:06:34 INFO TaskSetManager: Starting task 0.0 in stage 966.0 (TID 826) (ip-172-31-22-227.eu-central-1.compute.internal, executor 1, partition 0, NODE_LOCAL, 4618
bytes) taskResourceAssignments Map()
21/05/31 22:06:34 INFO TaskSetManager: Starting task 1.0 in stage 966.0 (TID 827) (ip-172-31-31-43.eu-central-1.compute.internal, executor 2, partition 1, NODE_LOCAL, 4618
bytes) taskResourceAssignments Map()
21/05/31 22:06:34 INFO BlockManagerInfo: Added broadcast_607_piece0 in memory on ip-172-31-22-227.eu-central-1.compute.internal:42783 (size: 2.3 KiB, free: 4.8 GiB)
21/05/31 22:06:34 INFO MapOutputTrackerMasterEndpoint: Asked to send map output locations for shuffle 60 to 172.31.22.227:53920
21/05/31 22:06:34 INFO MapOutputTrackerMasterEndpoint: Asked to send map output locations for shuffle 60 to 172.31.31.43:53722
21/05/31 22:06:34 INFO TaskSetManager: Finished task 0.0 in stage 966.0 (TID 826) in 7 ms on ip-172-31-22-227.eu-central-1.compute.internal (executor 1) (1/2)
21/05/31 22:06:34 INFO TaskSetManager: Finished task 1.0 in stage 966.0 (TID 827) in 8 ms on ip-172-31-31-43.eu-central-1.compute.internal (executor 2) (2/2)
21/05/31 22:06:34 INFO YarnScheduler: Removed TaskSet 966.0, whose tasks have all completed, from pool
21/05/31 22:06:34 INFO DAGSchedulr: ResultStage 966 (collectAsMap at MulticlassMetrics.scala:61) finished in 0.010 s
21/05/31 22:06:34 INFO DAGSchedulr: Job 311 is finished. Cancelling potential speculative or zombie tasks for this job
21/05/31 22:06:34 INFO YarnScheduler: Killing all running tasks in stage 966: Stage finished
21/05/31 22:06:34 INFO DAGSchedulr: Job 311 finished: collectAsMap at MulticlassMetrics.scala:61, took 2,615697 s
Results of model LogisticRegression_8b7ef7ea1ee
Accuracy 55.4
F1 0.5887562378882051
Recall 0.6377267651793895
AUC 0.5632648305163078
[Logistic Regression] With params DataFrame[features: vector, label: int, rawPrediction: vector, probability: vector, prediction: double]
[Logistic Regression] time 80.29311227798462
FIN
21/05/31 22:06:34 INFO SparkContext: Invoking stop() from shutdown hook
21/05/31 22:06:34 INFO AbstractConnector: Stopped Spark06a6e3a73(HTTP/1.1, (http://1.1)) {0.0.0.0:4040}
21/05/31 22:06:34 INFO SparkUI: Stopped Spark web UI at http://ip-172-31-20-57.eu-central-1.compute.internal:4040
21/05/31 22:06:34 INFO YarnClientSchedulerBackend: Interrupting monitor thread
21/05/31 22:06:34 INFO YarnClientSchedulerBackend: Shutting down all executors
21/05/31 22:06:34 INFO YarnSchedulerBackend$YarnDriverEndpoint: Asking each executor to shut down
21/05/31 22:06:34 INFO YarnClientSchedulerBackend: YARN client scheduler backend Stopped
21/05/31 22:06:34 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
21/05/31 22:06:34 INFO MemoryStore: MemoryStore cleared
21/05/31 22:06:34 INFO BlockManager: BlockManager stopped
21/05/31 22:06:34 INFO BlockManagerMaster: BlockManagerMaster stopped
21/05/31 22:06:34 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
21/05/31 22:06:34 INFO SparkContext: Successfully stopped SparkContext
21/05/31 22:06:34 INFO ShutdownHookManager: Shutdown hook called
21/05/31 22:06:34 INFO ShutdownHookManager: Deleting directory /mnt/tmp/spark-65757822-54a2-43c9-b04a-3d64d1a42b0b
21/05/31 22:06:34 INFO ShutdownHookManager: Deleting directory /mnt/tmp/spark-65757822-54a2-43c9-b04a-3d64d1a42b0b/pyspark-8b92e8b9-83d9-4e73-82a9-7d5830f6959e
21/05/31 22:06:34 INFO ShutdownHookManager: Deleting directory /mnt/tmp/spark-2f3ec298-18ac-4ef4-a3e4-105c5a148e72

```

Figura 2.11: Resultados del modelo de regresión logística con otros parámetros

2.4 Comparativa de los modelos

	Precisión	AUC	F1	Recall	Tiempo
Random Forest 1	62 %	0.65	0.62	0.62	101 segundos
Random Forest 2	62 %	0.66	0.62	0.61	451 segundos
Gradient Boost Tree 1	62.68 %	0.66	0.62	0.61	200 segundos
Gradient Boost Tree 2	62.48 %	0.66	0.61	0.60	359 segundos
Linear Regression 1	55.95 %	0.56	0.58	0.63	79 segundos
Linear Regression 2	58.88 %	0.56	0.58	0.63	81 segundos

Cuadro 2.1: Tal y como podemos ver, de todas las variaciones de parámetros que hemos realizado en todos los modelos, el único que ha encontrado mejoría es el de la regresión lineal. Comparando los tiempos de entrenamiento y su compromiso con la precisión y el área bajo la curva ROC, podemos encontrar que la solución más inteligente sería usar o el RandomForest o el Gradient Boost Tree con los parámetros más sencillos.

2.5 Factura

RWS	
Página de inicio de la consola	
Etiquetas de asignación de costos	
Facturación	
Facturas	
Pagos	
Créditos	
Órdenes de compra	
Preferencias	
Preferencias de facturación	
Métodos de pago	
Facturación unificada	
Configuración fiscal	

Detalles		+ Expandir todo
Cargos por servicios de AWS		\$14.59
▶ Amplify		\$0.00
▶ CloudWatch		\$0.00
▶ Cognito		\$0.00
▶ Data Transfer		\$0.00
▶ DynamoDB		\$0.00
▼ Elastic Compute Cloud		\$10.21
▼ EU (Frankfurt)		\$10.21
Amazon Elastic Compute Cloud running Linux/UNIX		\$10.21
\$0.00 per Linux t2.micro instance-hour (or partial hour) under monthly free tier	12.157 Hrs	\$0.00
\$0.194 per On Demand Linux c5.xlarge Instance Hour	0.844 Hrs	\$0.16
\$0.23 per On Demand Linux m5.xlarge Instance Hour	38.866 Hrs	\$8.94
\$0.45 per On Demand Linux z1d.xlarge Instance Hour	2.459 Hrs	\$1.11
EBS		\$0.00
\$0.00 for 1167 Mbps per z1d.xlarge instance-hour (or partial hour)	2.459 Hrs	\$0.00
\$0.00 for 800 Mbps per c5.xlarge instance-hour (or partial hour)	0.844 Hrs	\$0.00
\$0.00 per GB-month of General Purpose (SSD) provisioned storage under monthly free tier	22.146 GB-Mo	\$0.00
▼ Elastic MapReduce		\$1.85
▼ EU (Frankfurt)		\$1.85
Amazon Elastic MapReduce EUC1-BoxUsage:m5.xlarge		\$1.85
\$0.048 per hour for EMR m5.xlarge	38.574 Hrs	\$1.85
▶ Key Management Service		\$0.00
▶ Lambda		\$0.00
▶ Relational Database Service		\$0.00
▶ Secrets Manager		\$0.00
▶ Simple Notification Service		\$0.00

Figura 2.12: Factura de los costes asociados a la realización de la práctica en un entorno cloud real

Capítulo 3

Conclusiones

Después de realizar este trabajo, podemos resumir las siguientes cuestiones:

- Desplegar un ambiente cloud para el desarrollo con técnicas de Big Data bajo un gran proveedor como AWS es tremendamente recomendable, sencillo y barato.
- Las herramientas de técnica de Big Data con Spark las encuentro extremadamente rápidas y fáciles de usar, para ingentes volúmenes de datos. En nuestro caso, el entrenamiento y evaluación del modelo más pesado ha consumido 451 segundos, un tiempo que si hubiéramos invertido una máquina estándar local, con el mismo conjunto de datos, no hubiera tardado esos 451 segundos, ni de lejos, el tiempo consumido hubiese sido mucho mayor.
- Me hubiera gustado disponer de otra planificación en el máster para poder profundizar con Scala, o tener la oportunidad de haber hecho alguna evaluación de rendimiento aumentando el número de nodos esclavo. Esto último no lo he hecho por no abultar más la factura, pero lo hubiera encontrado super recomendable para saber cuál puede ser la ganancia en prestaciones de escalar en horizontal un sistema de Big Data. Ya tendré oportunidad de hacerlo en el verano.

Apéndice A

Código completo de los modelos

```
1  # En hadoop: /opt/spark-3.0.1/bin/pyspark --master
   ↪ spark://hadoop-master:7077
2  # spark-submit --conf spark.jars.ivy=/tmp/.ivy /intercambio/models.py
3  # exec(open('/intercambio/models.py', encoding="utf-8").read())
4
5  # sudo curl -L
   ↪ "https://github.com/docker/compose/releases/download/1.29.2/docker-compose
6  # -$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
7  # sudo chmod +x /usr/local/bin/docker-compose
8
9
10 import sys
11 import os.path
12 from time import *
13 import pyspark.sql.functions as func
14
15 # Librerías Core de spark
16 from pyspark import SparkContext, SparkConf, sql
17
18 from pyspark.sql.functions import udf
19 from pyspark.ml.feature import StringIndexer
20 from pyspark.sql.types import StringType, DoubleType, IntegerType
21 from pyspark.sql import SparkSession
22 from functools import reduce
23 from pyspark.mllib.evaluation import MulticlassMetrics
24 from pyspark.mllib.evaluation import BinaryClassificationMetrics
25 from pyspark.ml import Pipeline
```



```

26
27 # Libreria MLKit de Spark
28 from pyspark.ml.linalg import *
29 from pyspark.ml.feature import *
30 from pyspark.ml.tuning import *
31 from pyspark.ml.evaluation import *
32 from pyspark.ml.classification import *
33 from pyspark.ml import *
34
35 # Neural Network:
36   ↳ https://runawayhorse001.github.io/LearningApacheSpark/fnn.html
37 # Random Forest:
38   ↳ https://runawayhorse001.github.io/LearningApacheSpark/regression.html?
39   # highlight=random
40   # %20forest#random-forest-regression
41 # Decision Tree:
42   ↳ https://runawayhorse001.github.io/LearningApacheSpark/classification.html#id5
43 # Gradient Boost Tree:
44   ↳ https://runawayhorse001.github.io/LearningApacheSpark/classification.html#
45   # gradient-boosted-tree-classification
46 # Binomial Logistic Regression:
47   ↳ https://runawayhorse001.github.io/LearningApacheSpark/classification.html#
48   # binomial-logistic-regression
49
50 # docker exec -it --user root ubuntu_spark_1 pip install numpy
51 # docker exec -it ubuntu_spark_1 /bin/bash
52 # spark-submit --master spark://spark:7077 --total-executor-cores 4
53   ↳ --executor-memory 4g /intercambio/models.py
54 # docker exec -it ubuntu_spark_1 /bin/bash spark-submit --master
55   ↳ spark://spark:7077 --total-executor-cores 4 --executor-memory 8g
56   ↳ /intercambio/models.py
57
58 title = "CC-P4-Modelos"
59 name_file="/Intercambio/pepitoenpeligro-training.csv"
60
61 columns = ['PredCN_central_2', 'PredSS_r1_4', 'PSSM_r1_3_T',
62   ↳ 'AA_freq_central_M', 'PSSM_r2_1_L', 'PSSM_r1_2_R']
63 columns_asIndex= ['PredCN_central_2', 'PredSS_r1_4', 'PSSM_r1_3_T',
64   ↳ 'AA_freq_central_M', 'PSSM_r2_1_L', 'PSSM_r1_2_R']
65
66 portionTrain = 0.8
67 portionTest = 0.2

```

```

58
59 def predictions(estimator, paramGrid, dataTrain, dataTest):
60     # binary clasification
61     # https://spark.apache.org/docs/latest/
62     # mllib-evaluation-metrics.html#binary-classification
63     train_validator = TrainValidationSplit(estimator=estimator,
64         ↳ estimatorParamMaps=paramGrid,
65         ↳ evaluator=BinaryClassificationEvaluator(), trainRatio=portionTrain)
66     model = train_validator.fit(dataTrain)
67     predictions = model.transform(dataTest)
68     predictionAndLabel = predictions.select("prediction", "label")
69
70     # convierte labels y predicciones a float
71     predictionAndLabel = predictionAndLabel.withColumn("prediction",
72         ↳ func.round(predictionAndLabel['prediction']).cast('float'))
73     predictionAndLabel = predictionAndLabel.withColumn("label",
74         ↳ func.round(predictionAndLabel['label']).cast('float'))
75     metrics=MulticlassMetrics(predictionAndLabel
76         .select("prediction", "label").rdd.map(tuple))
77
78
79     evaluator = BinaryClassificationEvaluator()
80     auRocRF = evaluator.evaluate(predictions)
81
82
83     # la matriz de confusion revienta
84     cnf_matrix = metrics.confusionMatrix()
85     accuracy = round(metrics.accuracy*100, 3)
86     f1 = metrics.fMeasure(1.0)
87     recall = metrics.recall(1.0)
88
89     print("Results of model %s" % (estimator.__dict__['uid']))
90     print("Accuracy %s" % accuracy)
91     print("F1 %s" % f1)
92     print("Recall %s" % recall)
93     print("AUC %s" % auRocRF)
94
95     return predictions, model
96
97
98 # Ya tengo captura de esta ejecucion
99 def random_forest_1(trainingData, testData):
100     print("[Random Forest] init")

```

```

96     start_time = time()
97     rf = RandomForestClassifier(labelCol="label", featuresCol="features",
98         ↪ seed=12345)
99     # ParamGridBuilder params:
100     # https://spark.apache.org/docs/latest/ml-tuning.html
101     paramGridRF = ParamGridBuilder().addGrid(rf.numTrees, [5, 10,
102         ↪ 20]).addGrid(rf.maxDepth, [2, 3, 6]).build()
103
104     predictionsRF, mRF = predictions(rf,paramGridRF,trainingData,testData)
105     end_time = time()
106     elapsed_time = end_time - start_time
107     print("[Random Forest] With params %s" % paramGridRF)
108     print("[Random Forest] time %s" %(elapsed_time))
109
110     # Ya tengo captura de esta ejecucion
111     def random_forest_2(trainingData,testData):
112         print("[Random Forest] init")
113         start_time = time()
114         rf = RandomForestClassifier(labelCol="label", featuresCol="features",
115             ↪ seed=2021)
116         # ParamGridBuilder params:
117         # https://spark.apache.org/docs/latest/ml-tuning.html
118         paramGridRF = ParamGridBuilder().addGrid(rf.numTrees, [10, 30,
119             ↪ 60]).addGrid(rf.maxDepth, [3, 6, 12]).build()
120         predictionsRF, mRF = predictions(rf,paramGridRF,trainingData,testData)
121         end_time = time()
122         elapsed_time = end_time - start_time
123         print("[Random Forest] With params %s" % paramGridRF)
124         print("[Random Forest] time %s" %(elapsed_time))
125
126     def gradient_boosted_tree_1(trainingData, testData):
127         print("[Gradient Boosted Tree] init")
128         start_time = time()
129         gbt = GBTCClassifier(labelCol="label", featuresCol="features",
130             ↪ seed=2021)
131         #paramGridGBT = ParamGridBuilder().addGrid(gbt.maxIter, [10, 15,
132             ↪ 20]).addGrid(gbt.maxDepth, [3, 6, 12]).build()
133         paramGridGBT = ParamGridBuilder().addGrid(gbt.maxIter, [5, 10,
134             ↪ 15]).addGrid(gbt.maxDepth, [2, 3, 9]).build()
135         predictionsGBT, mGBT =
136             ↪ predictions(gbt,paramGridGBT,trainingData,testData)

```

```

130     end_time = time()
131     elapsed_time = end_time - start_time
132     print("[Gradient Boosted Tree] With params %s" % paramGridGBT)
133     print("[Gradient Boosted Tree] time %s" %(elapsed_time))
134
135
136 def gradient_boosted_tree_2(trainingData, testData):
137     print("[Gradient Boosted Tree] init")
138     start_time = time()
139     gbt = GBTClassifier(labelCol="label", featuresCol="features",
140         ↪ seed=2021)
141     paramGridGBT = ParamGridBuilder().addGrid(gbt.maxIter, [10, 15,
142         ↪ 20]).addGrid(gbt.maxDepth, [3, 6, 12]).build()
143
144     predictionsGBT, mGBT =
145         ↪ predictions(gbt,paramGridGBT,trainingData,testData)
146     end_time = time()
147     elapsed_time = end_time - start_time
148     print("[Gradient Boosted Tree] With params %s" % paramGridGBT)
149     print("[Gradient Boosted Tree] time %s" %(elapsed_time))
150
151
152 def perceptron_1(trainingData, testData):
153     print("[Peceptron] init")
154     start_time = time()
155     mlp = MultilayerPerceptronClassifier(
156         ↪ featuresCol="features",
157         ↪ labelCol="label",
158         ↪ predictionCol="prediction",
159         ↪ maxIter=100
160     )
161     mlpGrid = ParamGridBuilder().addGrid(mlp.layers, [[7, 3, 2], [7, 9, 3,
162         ↪ 2], [7, 5, 2]]).build()
163     predictionsMLP, mMLP = predictions(mlp, mlpGrid, trainingData,
164         ↪ testData)
165     end_time = time()
166     elapsed_time = end_time - start_time
167     print("[Peceptron] With params %s" % predictionsMLP)
168     print("[Peceptron] time %s" %(elapsed_time))
169
170
171 # Ya tengo captura de esta ejecucion

```

```

167 def logistic_regression1(trainingData, testData):
168     print("[Logistic Regression] init")
169     start_time = time()
170     lr = LogisticRegression(featuresCol="features",
171                             labelCol="label",maxIter=100,family="multinomial")
172     lrGrid = ParamGridBuilder().addGrid(lr.regParam, [0.1, 0.01,
173                                                       ↳ 0.001]).addGrid(lr.elasticNetParam, [0.5, 0.6, 0.8]).build()
173     predictionsRL, mRL = predictions(lr,lrGrid,trainingData,testData)
174     end_time = time()
175     elapsed_time = end_time - start_time
176     print("[Logistic Regression] With params %s" % predictionsRL)
177     print("[Logistic Regression] time %s" %(elapsed_time))
178
179 def logistic_regression_2(trainingData, testData):
180     print("[Logistic Regression] init")
181     start_time = time()
182     lr = LogisticRegression(featuresCol="features",
183                             labelCol="label",maxIter=100,family="multinomial")
184     lrGrid = ParamGridBuilder().addGrid(lr.regParam, [0.1, 0.01,
185                                                       ↳ 0.001]).addGrid(lr.elasticNetParam, [0.6, 0.7, 0.9]).build()
185     predictionsRL, mRL = predictions(lr,lrGrid,trainingData,testData)
186     end_time = time()
187     elapsed_time = end_time - start_time
188     print("[Logistic Regression] With params %s" % predictionsRL)
189     print("[Logistic Regression] time %s" %(elapsed_time))
190
191
192 def naive_bayes_1(trainingData, testData):
193     print("[NaiveBayes] init")
194     start_time = time()
195     nb = NaiveBayes(modelType="multinomial", featuresCol="features",
196                     ↳ labelCol="label", smoothing=1.0)
196     nbGrid = ParamGridBuilder().addGrid(1.0, [0.0, 0.2, 0.4, 0.6, 0.8,
197                                               ↳ 1.0]).build()
197     predictionsNB, mNB = predictions(nb,nbGrid,trainingData,testData)
198     end_time = time()
199     elapsed_time = end_time - start_time
200     print("[NaiveBayes] With params %s" % predictionsNB)
201     print("[NaiveBayes] time %s" %(elapsed_time))
202
203
204 if __name__ == "__main__":

```

```

205 print("Iniciando el contexto de Spark %s", title)
206 configurationSpark = SparkConf().setAppName(title)
207 sparkContexto = SparkContext.getOrCreate(conf=configurationSpark)
208 sqlContext = sql.SQLContext(sparkContexto)
209
210 df_columns = sqlContext.read.csv(name_file, sep=",", header=True,
    ↪ inferSchema=True)
211 indexer = StringIndexer(inputCol="PredSS_r1_4",
    ↪ outputCol="PredSS_r1_4_indexado")
212 df_columns = indexer.fit(df_columns).transform(df_columns)
213 df_columns = df_columns.drop("PredSS_r1_4")
214 df_columns =
    ↪ df_columns.withColumnRenamed("PredSS_r1_4_indexado", "PredSS_r1_4")
215 df_columns.show(20)
216
217 clases_negativas = df_columns.filter(df_columns['class']==0).count()
218 clases_positivas = df_columns.filter(df_columns['class']==1).count()
219 print("El balanceo negativo/positivio es: %s / %s" % (clases_negativas,
    ↪ clases_positivas))
220
221 #Me quedo con el numero de clases de la menor
222 tam_partition = clases_positivas
223 if(clases_positivas > clases_negativas):
224     tam_partition = clases_negativas
225
226 # Reduzco ambos al tamaño de la particion anterior: Undersampling
227 df_0 = df_columns.filter(df_columns['class'] == 0).limit(tam_partition)
228 df_1 = df_columns.filter(df_columns['class'] == 1).limit(tam_partition)
229
230 df_balanced = df_1.union(df_0)
231 df_train, df_test = df_balanced.randomSplit([portionTrain,
    ↪ portionTest])
232 df_balanced_count = df_balanced.select('class').count()
233
234 df_train_count = df_train.select('class').count()
235 df_train_negative_count =
    ↪ df_train.filter(df_columns['class']==0).select('class').count()
236 df_train_positive_count =
    ↪ df_train.filter(df_columns['class']==1).select('class').count()
237
238 df_test_count = df_test.select('class').count()

```

```

239 df_test_negative_count =
    ↳ df_test.filter(df_columns['class']==0).select('class').count()
240 df_test_positive_count =
    ↳ df_test.filter(df_columns['class']==1).select('class').count()
241
242 print("[Global] total: %s", df_balanced_count)
243 print("[Train] positivas: %s, negativas %s, total %s" %
    ↳ (df_train_positive_count, df_train_negative_count, df_train_count
    ↳ ))
244 print("[Test] positivas: %s, negativas %s, total %s" %
    ↳ (df_test_positive_count, df_test_negative_count, df_test_count ))
245
246 # Feature Transformer VectorAssembler in PySpark ML Feature
247 # https://medium.com/@nutanbhogendrasharma/
248 # feature-transformer-vectorassembler-in-pyspark
249 # -ml-feature-part-3-b3c2c3c93ee9
250 assembler = VectorAssembler(inputCols=columns_asIndex,
    ↳ outputCol='features')
251 trainingData = assembler.transform(df_train).select("features","class")
252 .withColumnRenamed("class","label")
253 testData = assembler.transform(df_test).select("features","class")
254 .withColumnRenamed("class","label")
255
256
257 # RandomForest - OK
258 random_forest_1(trainingData, testData)
259 random_forest_2(trainingData, testData)
260
261
262 # Gradient Boosted Tree - OK
263 gradient_boosted_tree_1(trainingData,testData)
264 gradient_boosted_tree_2(trainingData,testData)
265
266 # Regresion logistica- OK
267 logistic_regression1(trainingData, testData)
268 logistic_regression_2(trainingData, testData)
269
270
271 # Perceptron multicapa - No funca
272 # perceptron_1(trainingData,testData)
273
274

```

```
275
276 # Naive Bayes - No funca
277 # naive_bayes_1(trainingData, testData)
278
279
280 # https://stackoverflow.com/questions/60772315/
281 # how-to-evaluate-a-classifier-with-pyspark-2-4-5
282 # https://stackoverflow.com/questions/41714698/
283 # how-to-get-accuracy-precision-recall
284 # -and-roc-from-cross-validation-in-spark-ml
285 print("FIN")
```


Apéndice B

Composición de Spark

```
1  version: '2'
2
3  services:
4    spark:
5      image: docker.io/bitnami/spark:3
6      environment:
7        - SPARK_MODE=master
8        - SPARK_RPC_AUTHENTICATION_ENABLED=no
9        - SPARK_RPC_ENCRYPTION_ENABLED=no
10       - SPARK_LOCAL_STORAGE_ENCRYPTION_ENABLED=no
11       - SPARK_SSL_ENABLED=no
12      ports:
13        - '8080:8080'
14      volumes:
15        - ./intercambio:/intercambio
16    spark-worker-1:
17      image: docker.io/bitnami/spark:3
18      environment:
19        - SPARK_MODE=worker
20        - SPARK_MASTER_URL=spark://spark:7077
21        - SPARK_WORKER_MEMORY=2G
22        - SPARK_WORKER_CORES=2
23        - SPARK_RPC_AUTHENTICATION_ENABLED=no
24        - SPARK_RPC_ENCRYPTION_ENABLED=no
25        - SPARK_LOCAL_STORAGE_ENCRYPTION_ENABLED=no
26        - SPARK_SSL_ENABLED=no
27      volumes:
```

```
28     - ./intercambio:/intercambio
29 spark-worker-2:
30   image: docker.io/bitnami/spark:3
31   environment:
32     - SPARK_MODE=worker
33     - SPARK_MASTER_URL=spark://spark:7077
34     - SPARK_WORKER_MEMORY=2G
35     - SPARK_WORKER_CORES=2
36     - SPARK_RPC_AUTHENTICATION_ENABLED=no
37     - SPARK_RPC_ENCRYPTION_ENABLED=no
38     - SPARK_LOCAL_STORAGE_ENCRYPTION_ENABLED=no
39     - SPARK_SSL_ENABLED=no
40   volumes:
41     - ./intercambio:/intercambio
```

Bibliografía

- [1] Feature Transformer VectorAssembler in PySpark ML Feature <https://medium.com/@nutanbhogendrasharma/feature-transformer-vectorassembler-in-pyspark-ml-feature-part-3-b3c2c3c93ee9>
- [2] How to get Accuracy precision recall and ROC <https://stackoverflow.com/questions/41714698/how-to-get-accuracy-precision-recall-and-roc-from-cross-validation-in-spark-ml>
- [3] Random Forest <https://runawayhorse001.github.io/LearningApacheSpark/regression.html?highlight=random%20forest#random-forest-regression>
- [4] Gradient Boost Tree <https://runawayhorse001.github.io/LearningApacheSpark/classification.html#id5>
- [5] Logistic Regression <https://runawayhorse001.github.io/LearningApacheSpark/classification.html#binomial-logistic-regression>
- [6] AWS CLI S3 https://docs.aws.amazon.com/es_es/cli/latest/userguide/cli-services-s3-commands.html
- [7] AWS EMR <https://aws.amazon.com/es/emr/>