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

Universidad de Granada

Índice general

Índice de figuras

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**.

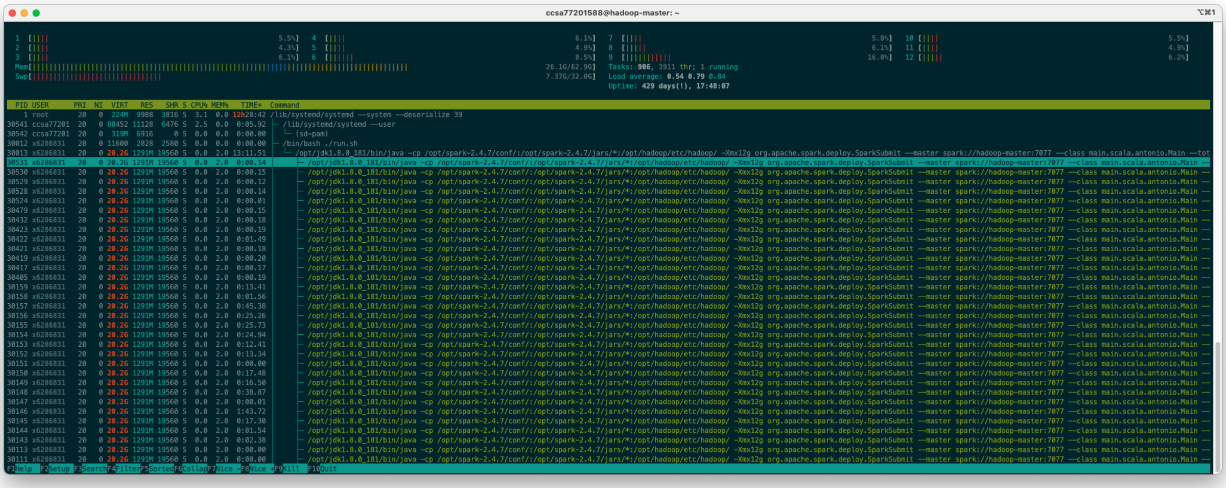


Figura 1.1: Estado del cluster hadoop

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:

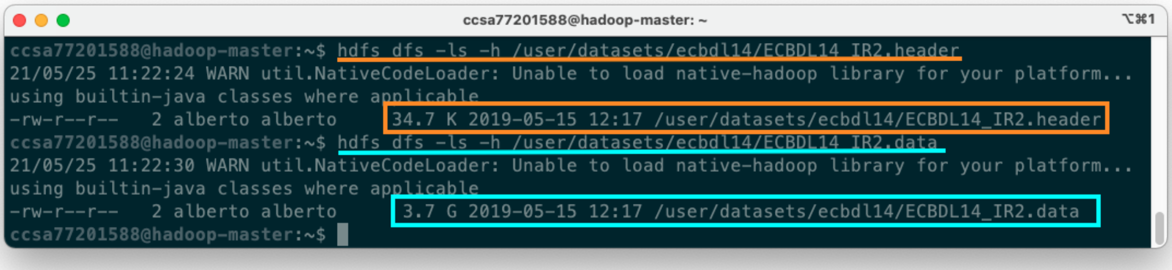


Figura 1.2: Búsqueda de los ficheros de cabecera y datos del dataset que nos corresponde

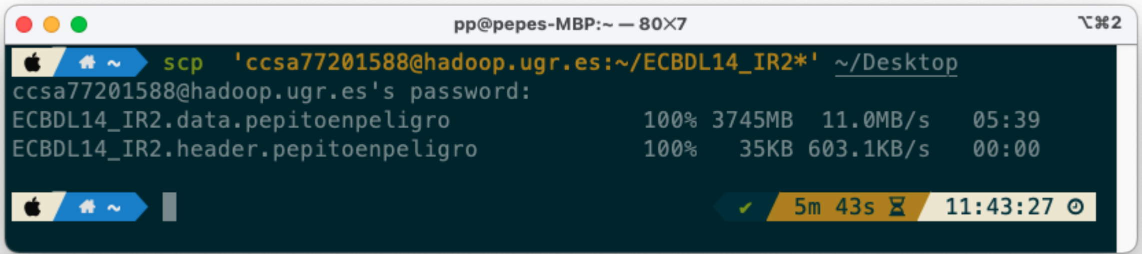


Figura 1.3: Descarga de ambos ficheros

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://707
```

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*.

```
aws emr create-cluster --applications Name=Spark Name=Zeppelin --ec2-attributes '{"KeyName":"spark","InstanceProfile":"EMR_EC2_DefaultRole",
    "SubnetId":"subnet-731b7d19",
    "EmrManagedSlaveSecurityGroup":"sg-0a2bf65c779ae46d3",
    "EmrManagedMasterSecurityGroup":"sg-0f45869b481f32b74"}'

    --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":
        [{"VolumeSpecification":{"SizeInGB":32,"VolumeType":"gp2"},
        "VolumesPerInstance":2}]},
    "InstanceGroupType":"MASTER","InstanceType":"m5.xlarge","Name":"Master Instance Group"},{"InstanceCount":2,"EbsConfiguration"
        :{"EbsBlockDeviceConfigs":
            [{"VolumeSpecification":
                {"SizeInGB":32,"VolumeType":"gp2"},
                "VolumesPerInstance":2}]},
                "InstanceGroupType":"CORE","InstanceType":"m5.xlarge",
                "Name":"Core Instance Group"}]'
```

```
      --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**.

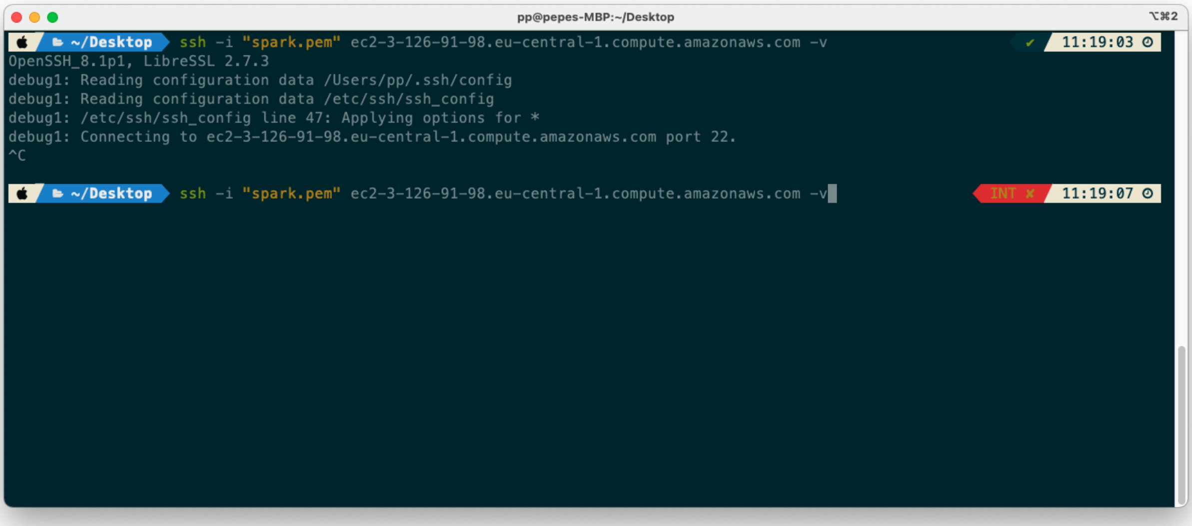


Figura 2.1: Conectando a la máquina maestra del clúster. Acceso denegado

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.

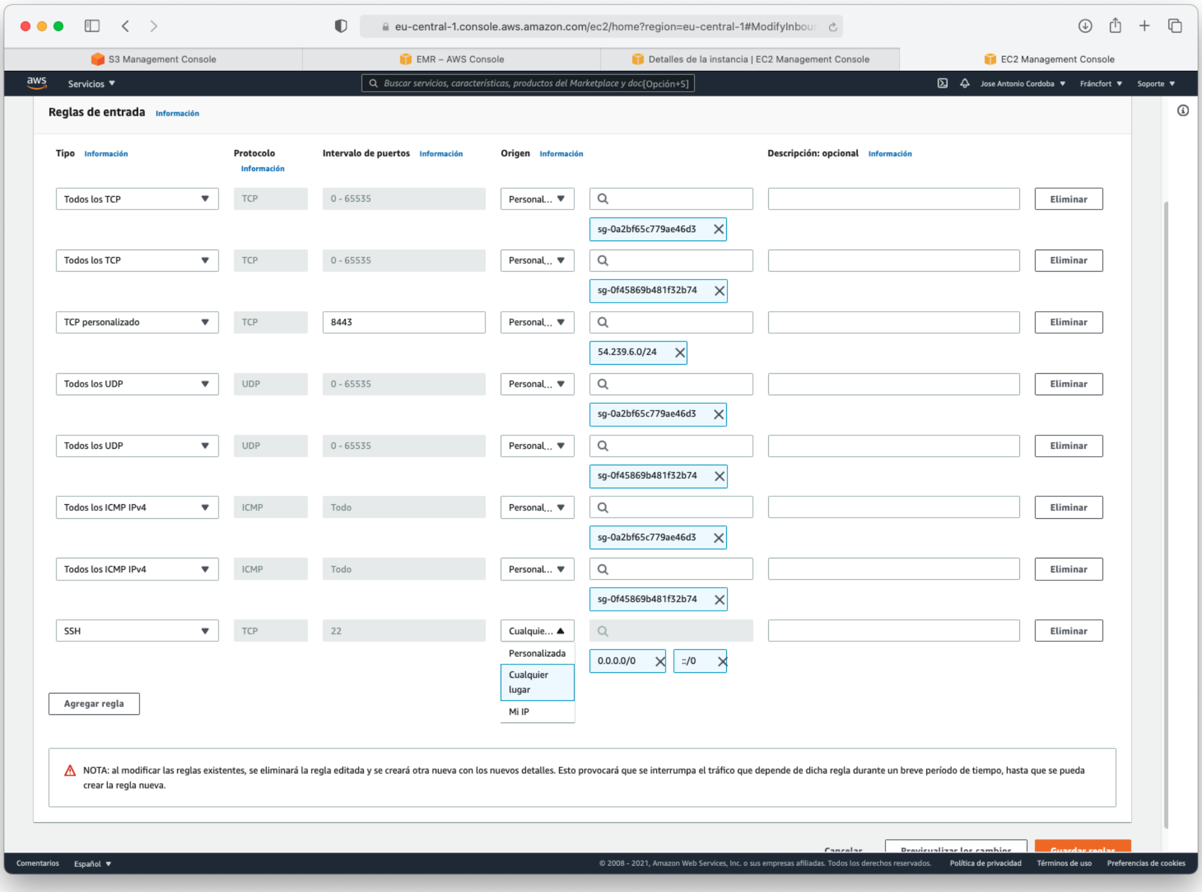


Figura 2.2: Añadir una regla de entrada para permitir acceso al puerto 22 desde cualquier IP

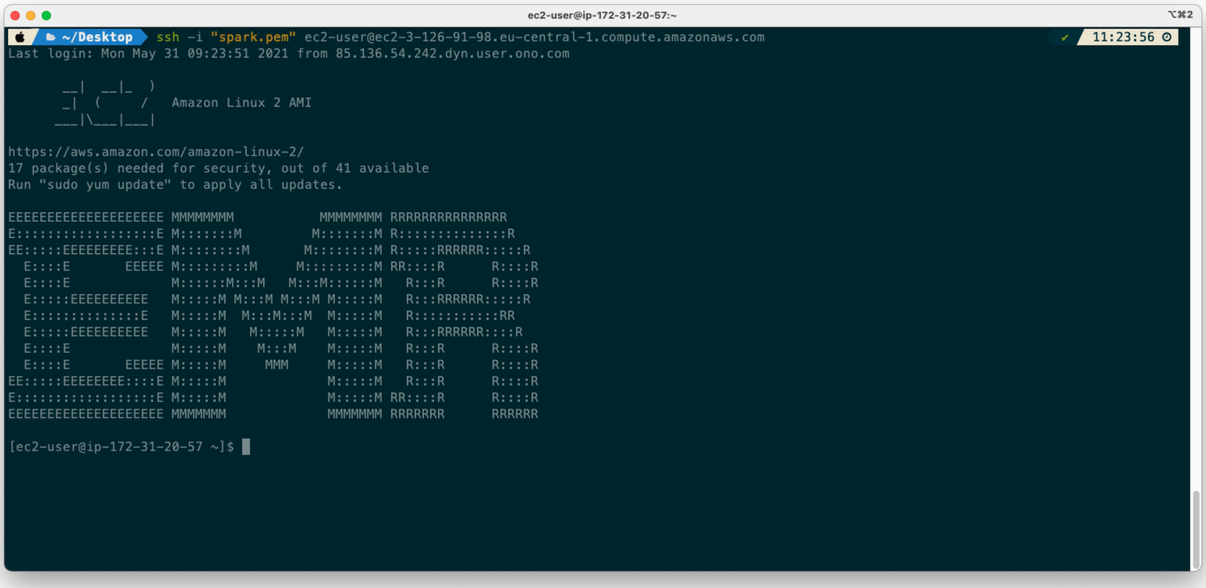


Figura 2.3: Conectando a la máquina maestra del clúster. Acceso concedido

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.

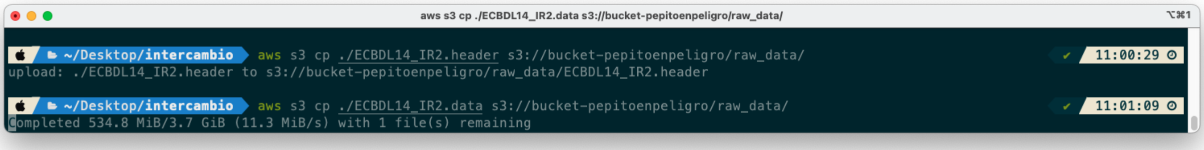


Figura 2.4: Subiendo los ficheros de cabecera y datos al bucket s3://bucket-pepitoenpeligro

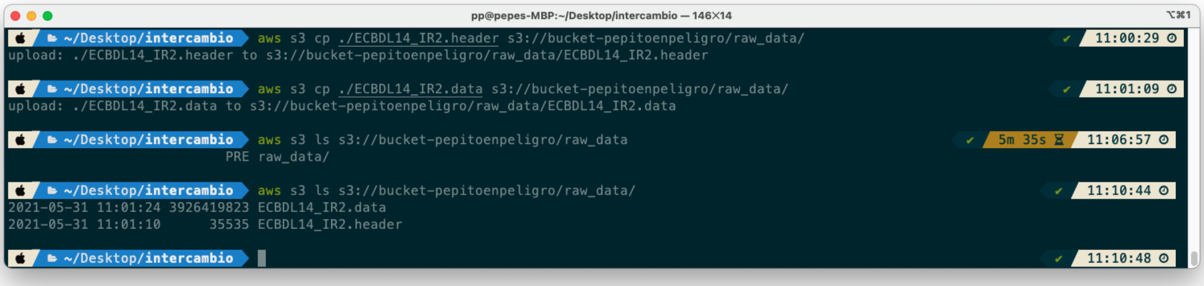


Figura 2.5: Comprobamos que están en el bucket s3://bucket-pepitoenpeligro

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.

```python
import sys
import time
from pyspark import SparkContext, SparkConf, sql
from pyspark.ml.classification import LogisticRegression
from functools import reduce

configurationSpark = SparkConf().setAppName("CC-P4-Preprocesado")
sparkContexto = SparkContext.getOrCreate(conf=configurationSpark)
sqlContext = sql.SQLContext(sparkContexto)
```

```python
name_output="pepitoenpeligro"

if __name__ == "__main__":
    start = time.time()
    print("Comenzando el preprocesado")
    ficheroCabeceras = sparkContexto.textFile("
    s3n://bucket-pepitoenpeligro/raw_data/ECBDL14_IR2.header").collect()
    cabecerasFiltradas = filter(lambda line: "@attribute" in line ,ficheroCabeceras)
    print("Mapeando")
    mapHeaders = list(map(lambda line: line.split()[1], cabecerasFiltradas))

    print("Leyendo en un dataframe los datos del dataset pesado")
    df = sqlContext.read.csv("s3://bucket-pepitoenpeligro/raw_data/ECBDL14_IR2.data",
    header=False,sep=",",inferSchema=True)
    print("Reduciendo")
    dfReducido = reduce(lambda data, idx: data.withColumnRenamed(df.schema.names[idx], mapHeaders[idx]), range(len(df.schema.names)), df)

    dfReducido.createOrReplaceTempView("sql_dataset")

    columns= ['`PredCN_central_2`', '`PredSS_r1_4`', '`PSSM_r1_3_T`', '`AA_freq_central_M`', '`PSSM_r2_-1_L`', '`PSSM_r1_-2_R`']
    print("Seleccionando las columnas {%s, %s, %s, %s, %s, %s}" % (columns[0], columns[1], columns[2], columns[3], columns[4], columns[5]))
    sqlDF = sqlContext.sql('SELECT %s, %s, %s, %s, %s, %s, class FROM sql_dataset' % (columns[0], columns[1], columns[2], columns[3], columns[4], columns[5]))

    print("Escribiendo en el fichero csv")
    sqlDF.write.format('csv').option('header',True)
    .save('s3n://bucket-pepitoenpeligro/%s' % (name_output))

    print("Fin del preprocesado")
    print("[3] Hemos Seleccionando las columnas {%s, %s, %s, %s, %s, %s}" % (columns[0], columns[1], columns[2], columns[3], columns[4], columns[5]))
    end = time.time()
```

```
    print("Tiempo consumido en la seleccion de columnas: %s" % (end -
     ↪  start))
    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

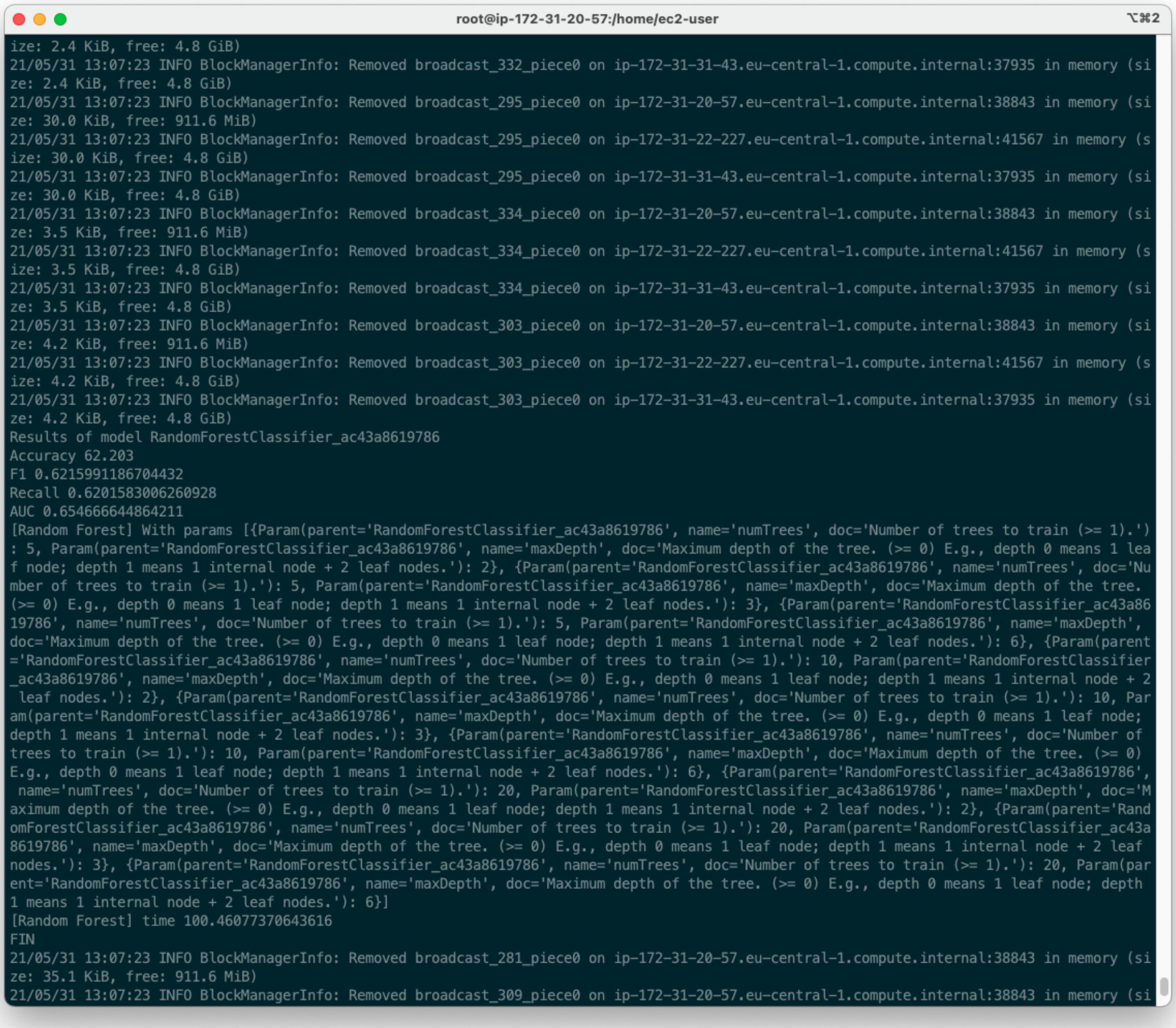
Para la evaluación de los modelos, hemos definido una función que necesita el modelo, el grid de parámetros, el conjunto de entrenamiento y el conjunto de test. Dentro ajustamos el modelo al conjunto de entrenamiento y obtenemos las predicciones y sacamos los valores de **precisión**, **f1**, **auc**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

```
def predictions(estimator, paramGrid, dataTrain, dataTest):
    # binary clasification
    # https://spark.apache.org/docs/latest/
    # mllib-evaluation-metrics.html#binary-classification
    train_validator = TrainValidationSplit(estimator=estimator,
     ↪  estimatorParamMaps=paramGrid,
     ↪  evaluator=BinaryClassificationEvaluator(), trainRatio=portionTrain)
    model = train_validator.fit(dataTrain)
    predictions = model.transform(dataTest)
    predictionAndLabel = predictions.select("prediction","label")

    # convierte labels y prediccones a float
    predictionAndLabel = predictionAndLabel.withColumn("prediction",
     ↪  func.round(predictionAndLabel['prediction']).cast('float'))
    predictionAndLabel = predictionAndLabel.withColumn("label",
     ↪  func.round(predictionAndLabel['label']).cast('float'))
    metrics=MulticlassMetrics(predictionAndLabel
    .select("prediction","label").rdd.map(tuple))


    evaluator = BinaryClassificationEvaluator()
    auRocRF = evaluator.evaluate(predictions)


    # la matriz de confusion revienta
```

```
    cnf_matrix = metrics.confusionMatrix()
    accuracy = round(metrics.accuracy*100, 3)
    f1 = metrics.fMeasure(1.0)
    recall = metrics.recall(1.0)

    print("Results of model %s"  % (estimator.__dict__['uid']))
    print("Accuracy %s" % accuracy)
    print("F1 %s" % f1)
    print("Recall %s" % recall)
    print("AUC %s" % auRocRF)

    return predictions, model
```

2.3.2 Random Forest

```
def random_forest_2(trainingData,testData):
    print("[Random Forest] init")
    start_time = time()
    rf = RandomForestClassifier(labelCol="label", featuresCol="features",
     ↪ seed=12345)
    # ParamGridBuilder params:
    # https://spark.apache.org/docs/latest/ml-tuning.html
    paramGridRF = ParamGridBuilder().addGrid(rf.numTrees, [5, 10,
     ↪ 20]).addGrid(rf.maxDepth, [2, 3, 6]).build()

    predictionsRF, mRF = predictions(rf,paramGridRF,trainingData,testData)
    end_time = time()
    elapsed_time = end_time - start_time
    print("[Random Forest] With params %s" % paramGridRF)
    print("[Random Forest] time %s" %(elapsed_time))
```

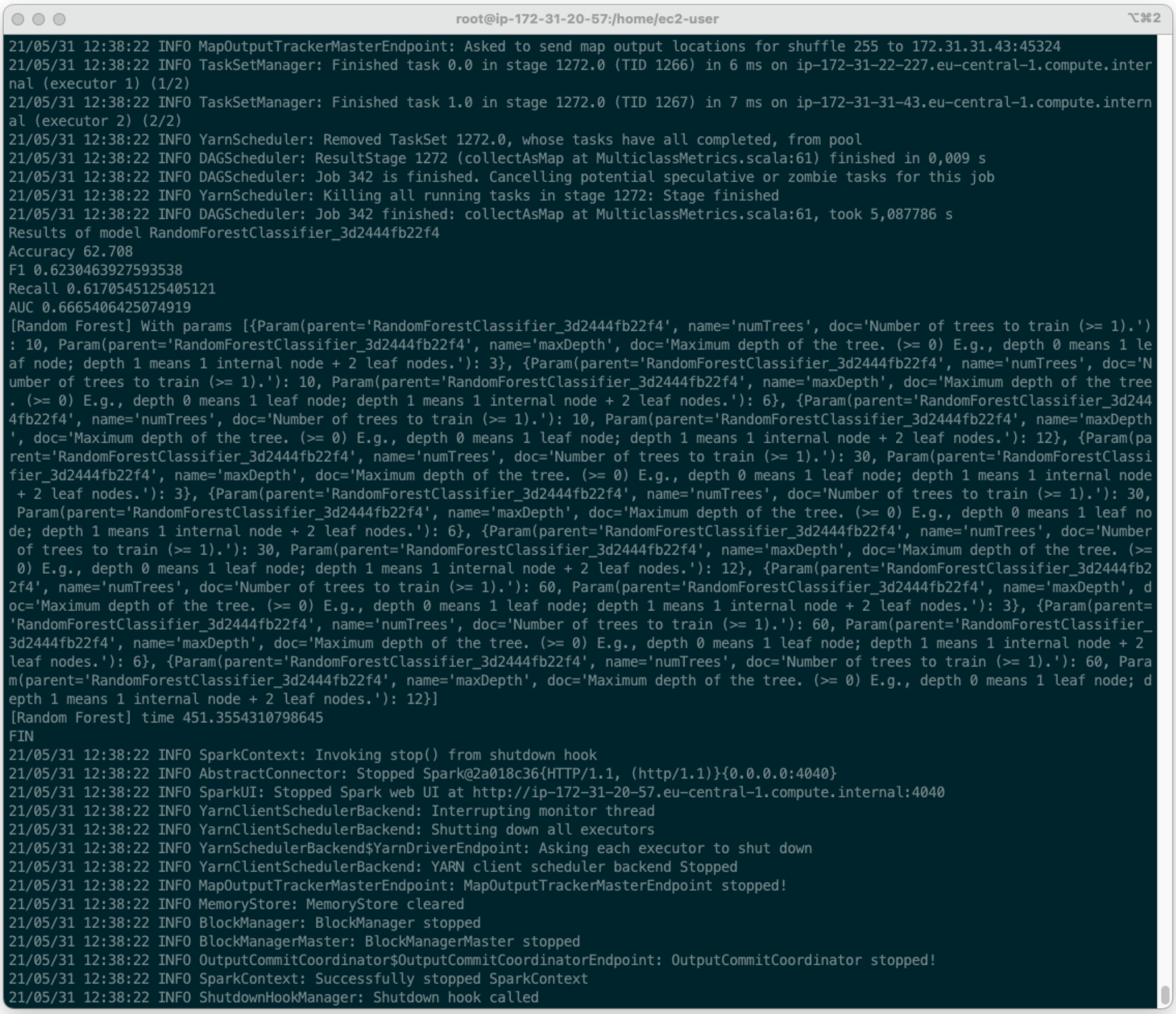


```
root@ip-172-31-20-57:/home/ec2-user
ize: 2.4 KiB, free: 4.8 GiB)
21/05/31 13:07:23 INFO BlockManagerInfo: Removed broadcast_332_piece0 on ip-172-31-31-43.eu-central-1.compute.internal:37935 in memory (si
ze: 2.4 KiB, free: 4.8 GiB)
21/05/31 13:07:23 INFO BlockManagerInfo: Removed broadcast_295_piece0 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

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

```python
    paramGridRF = ParamGridBuilder().addGrid(rf.numTrees, [10, 30,
     ↪ 60]).addGrid(rf.maxDepth, [3, 6, 12]).build()
    predictionsRF, mRF = predictions(rf,paramGridRF,trainingData,testData)
    end_time = time()
    elapsed_time = end_time - start_time
    print("[Random Forest] With params %s" % paramGridRF)
    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.inter
nal (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; d
epth 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

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

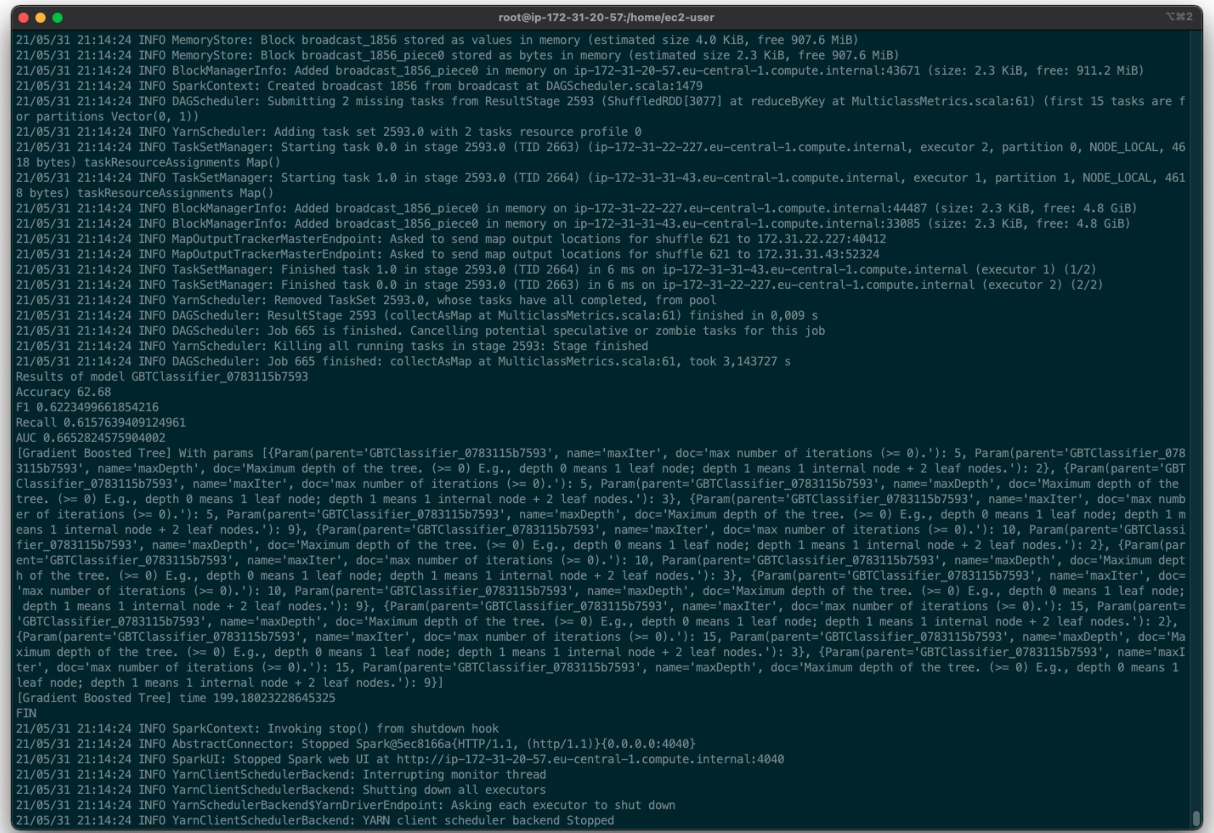


Figura 2.8: Resultados del modelo Gradient Boost Tree con los primeros parámetros

```python
def gradient_boosted_tree_2(trainingData, testData):
    print("[Gradient Boosted Tree] init")
    start_time = time()
    gbt = GBTClassifier(labelCol="label", featuresCol="features",
     ↪  seed=2021)
    paramGridGBT = ParamGridBuilder().addGrid(gbt.maxIter, [10, 15,
     ↪  20]).addGrid(gbt.maxDepth, [3, 6, 12]).build()

    predictionsGBT, mGBT =
     ↪  predictions(gbt,paramGridGBT,trainingData,testData)
    end_time = time()
    elapsed_time = end_time - start_time
    print("[Gradient Boosted Tree] With params %s" % paramGridGBT)
```

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

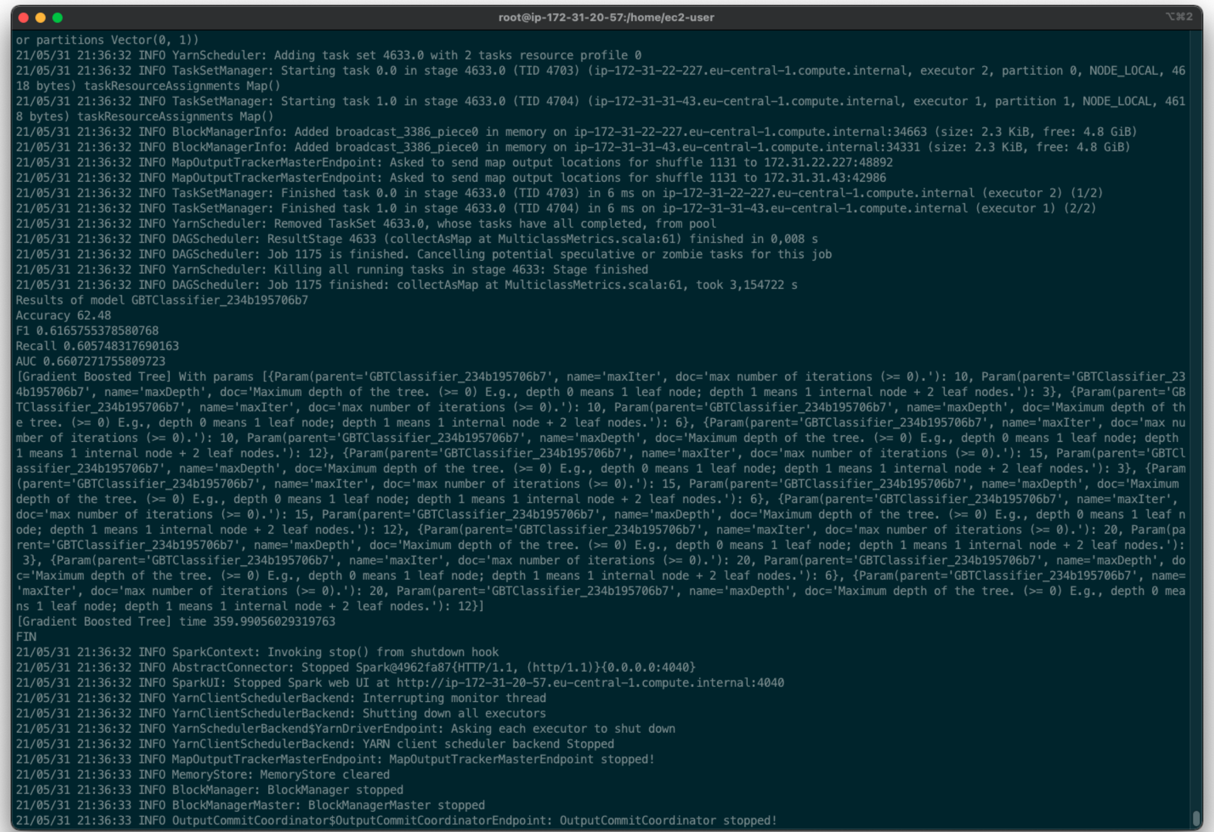


Figura 2.9: Resultados del modelo Gradient Boost Tree con otros parámetros

2.3.4 Logistic Regresion

```
def logistic_regresion1(trainingData, testData):
    print("[Logistic Regression] init")
    start_time = time()
    lr = LogisticRegression(featuresCol="features",labelCol="label",maxIter=100,family="mul
    lrGrid = ParamGridBuilder().addGrid(lr.regParam, [0.1, 0.01, 0.001]).addGrid(lr.elasticNetParam, [0.5, 0.6, 0.8]).build()
    predictionsRL, mRL = predictions(lr,lrGrid,trainingData,testData)
    end_time = time()
```

```python
    elapsed_time = end_time - start_time
    print("[Logistic Regression] With params %s" % predictionsRL)
    print("[Logistic Regression] time %s" %(elapsed_time))
```

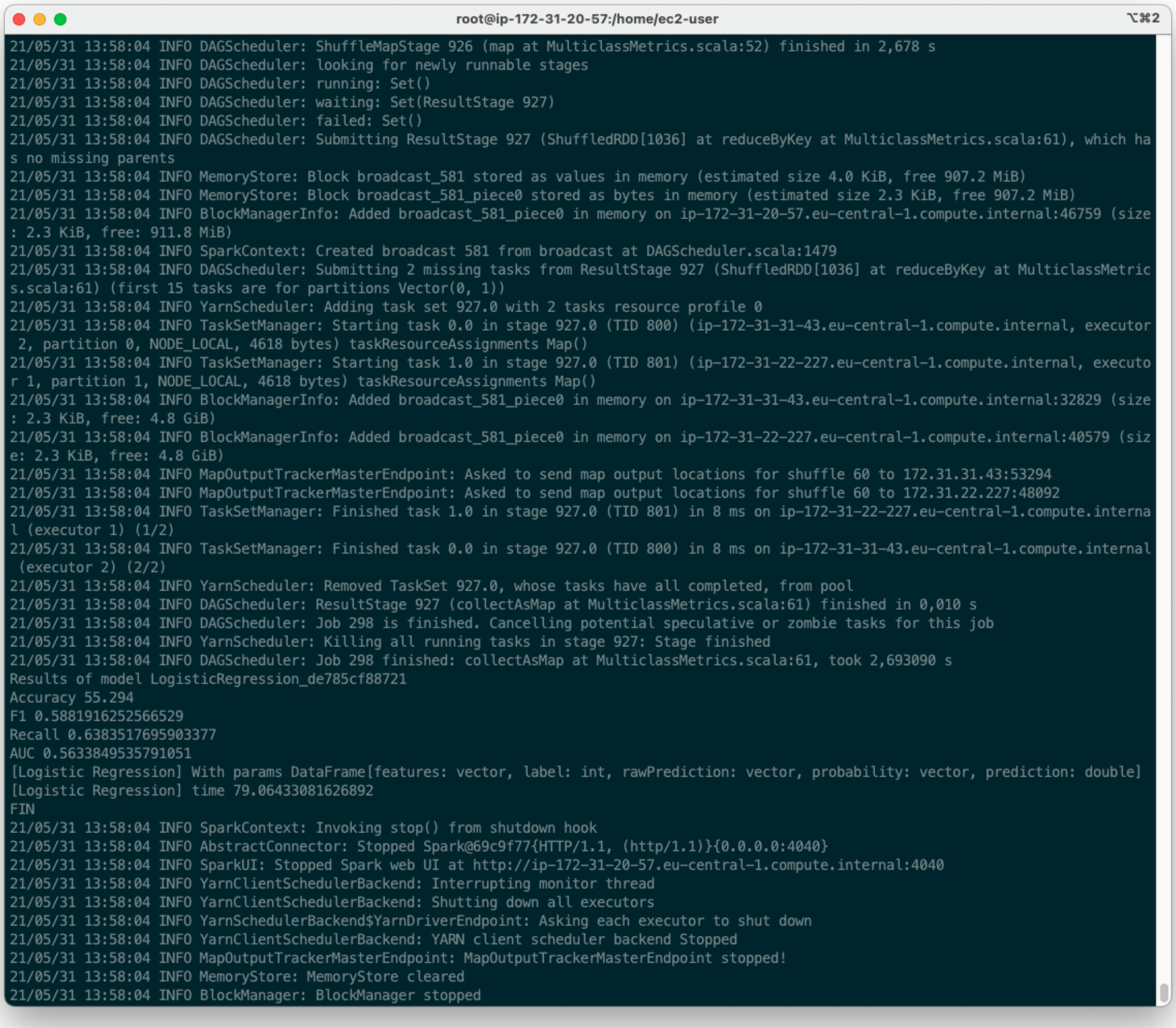


Figura 2.10: Resultados del modelo de regresión logística con los primeros parámetros

```python
def logistic_regresion_2(trainingData, testData):
    print("[Logistic Regression] init")
    start_time = time()
```

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

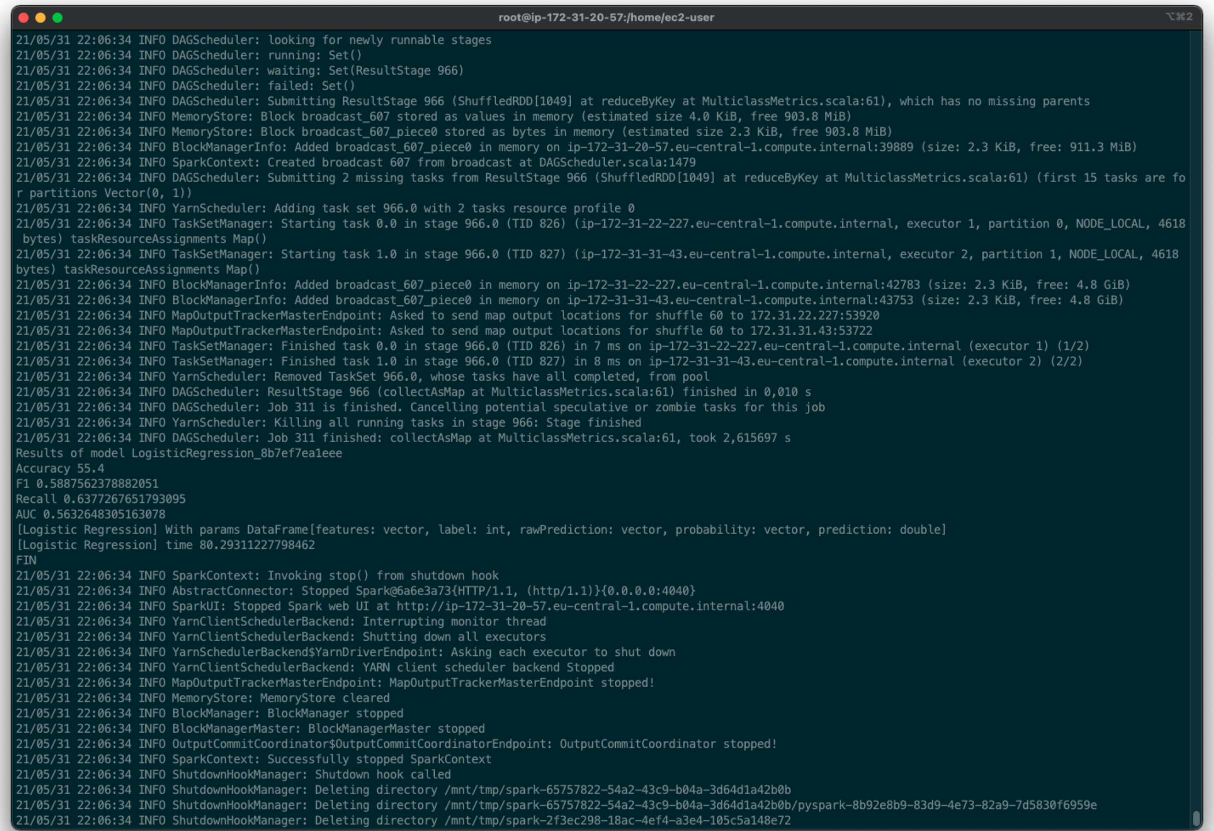

```
root@ip-172-31-20-57:/home/ec2-user
21/05/31 22:06:34 INFO DAGScheduler: looking for newly runnable stages
21/05/31 22:06:34 INFO DAGScheduler: running: Set()
21/05/31 22:06:34 INFO DAGScheduler: waiting: Set(ResultStage 966)
21/05/31 22:06:34 INFO DAGScheduler: failed: Set()
21/05/31 22:06:34 INFO DAGScheduler: 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 DAGScheduler.scala:1479
21/05/31 22:06:34 INFO DAGScheduler: 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 BlockManagerInfo: Added broadcast_607_piece0 in memory on ip-172-31-31-43.eu-central-1.compute.internal:43753 (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 DAGScheduler: ResultStage 966 (collectAsMap at MulticlassMetrics.scala:61) finished in 0,010 s
21/05/31 22:06:34 INFO DAGScheduler: 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 DAGScheduler: Job 311 finished: collectAsMap at MulticlassMetrics.scala:61, took 2,615697 s
Results of model LogisticRegression_8b7ef7ea1eee
Accuracy 55.4
F1 0.5887562378882051
Recall 0.6377267651793095
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 Spark@6a6e3a73{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

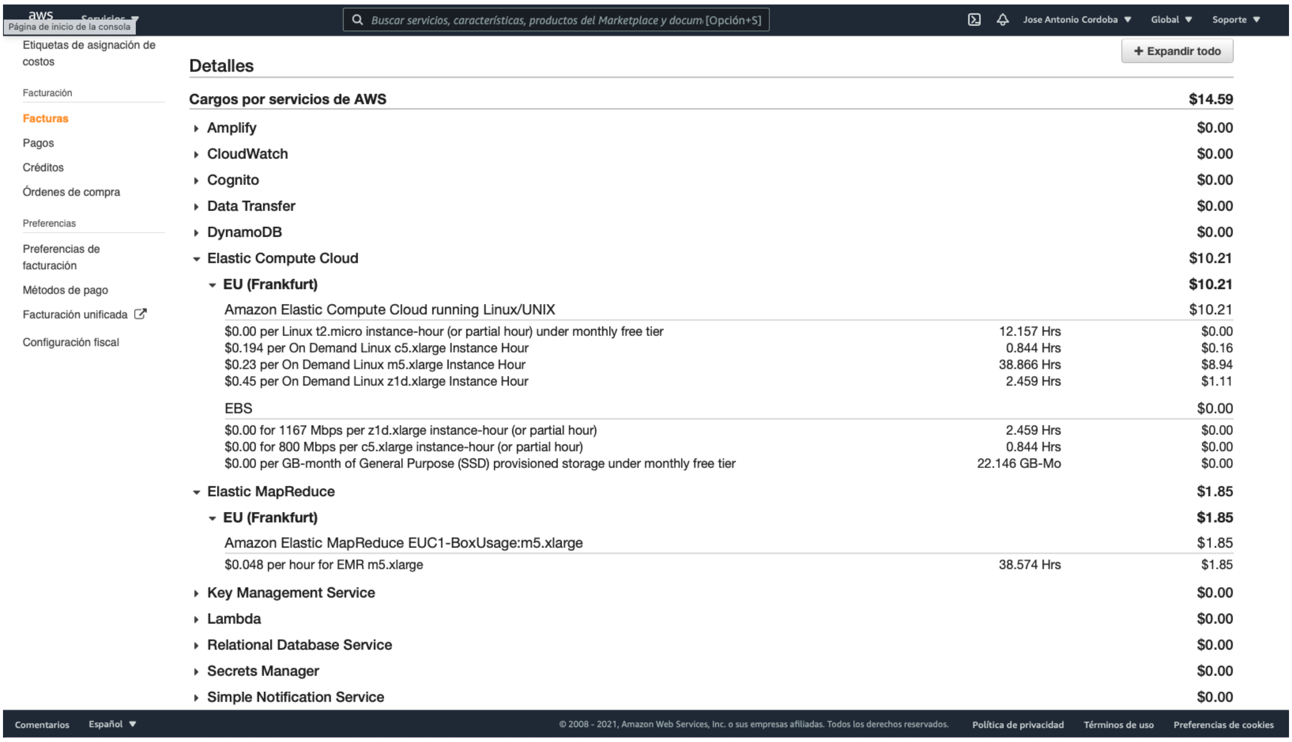


Figura 2.12: Factura de los costes asociados a la realización de la práctica en un entorno cloud real

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 pesao 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

```python
# En hadoop: /opt/spark-3.0.1/bin/pyspark --master spark://hadoop-master:7077
# spark-submit --conf spark.jars.ivy=/tmp/.ivy /intercambio/models.py
# exec(open('/intercambio/models.py', encoding="utf-8").read())

# sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose
# -$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
# sudo chmod +x /usr/local/bin/docker-compose


import sys
import os.path
from time import *
import pyspark.sql.functions as func

# Librerias Core de spark
from pyspark import SparkContext, SparkConf, sql

from pyspark.sql.functions import udf
from pyspark.ml.feature import StringIndexer
from pyspark.sql.types import StringType, DoubleType, IntegerType
from pyspark.sql import SparkSession
from functools import reduce
from pyspark.mllib.evaluation import MulticlassMetrics
from pyspark.mllib.evaluation import BinaryClassificationMetrics
from pyspark.ml import Pipeline
```

```python

# Libreria MLKit de Spark
from pyspark.ml.linalg import *
from pyspark.ml.feature import *
from pyspark.ml.tuning import *
from pyspark.ml.evaluation import *
from pyspark.ml.classification import *
from pyspark.ml import *

# Neural Network: https://runawayhorse001.github.io/LearningApacheSpark/fnn.html
# Random Forest: https://runawayhorse001.github.io/LearningApacheSpark/regression.html?
# highlight=random
#%20forest#random-forest-regression
# Decision Tree: https://runawayhorse001.github.io/LearningApacheSpark/classification.html#id5
# Gradient Boost Tree: https://runawayhorse001.github.io/LearningApacheSpark/classification.html#
# gradient-boosted-tree-classification
# Binomial Logistic Regression: https://runawayhorse001.github.io/LearningApacheSpark/classification.html#
# binomial-logistic-regression

# docker exec -it --user root ubuntu_spark_1 pip install numpy
# docker exec -it  ubuntu_spark_1 /bin/bash
# spark-submit --master spark://spark:7077 --total-executor-cores 4 --executor-memory 4g /intercambio/models.py
# docker exec -it  ubuntu_spark_1 /bin/bash spark-submit --master spark://spark:7077 --total-executor-cores 4 --executor-memory 8g /intercambio/models.py

title = "CC-P4-Modelos"
name_file="/Intercambio/pepitoenpeligro-training.csv"

columns = ['`PredCN_central_2`', '`PredSS_r1_4`', '`PSSM_r1_3_T`', '`AA_freq_central_M`', '`PSSM_r2_-1_L`', '`PSSM_r1_-2_R`']
columns_asIndex= ['PredCN_central_2', 'PredSS_r1_4', 'PSSM_r1_3_T', 'AA_freq_central_M', 'PSSM_r2_-1_L', 'PSSM_r1_-2_R']

portionTrain = 0.8
portionTest = 0.2
```

```
def predictions(estimator, paramGrid, dataTrain, dataTest):
    # binary clasification
    # https://spark.apache.org/docs/latest/
    # mllib-evaluation-metrics.html#binary-classification
    train_validator = TrainValidationSplit(estimator=estimator, estimatorParamMaps=paramGrid, evaluator=BinaryClassificationEvaluator(), trainRatio=portionTrain)
    model = train_validator.fit(dataTrain)
    predictions = model.transform(dataTest)
    predictionAndLabel = predictions.select("prediction","label")

    # convierte labels y prediccones a float
    predictionAndLabel = predictionAndLabel.withColumn("prediction", func.round(predictionAndLabel['prediction']).cast('float'))
    predictionAndLabel = predictionAndLabel.withColumn("label", func.round(predictionAndLabel['label']).cast('float'))
    metrics=MulticlassMetrics(predictionAndLabel
    .select("prediction","label").rdd.map(tuple))


    evaluator = BinaryClassificationEvaluator()
    auRocRF = evaluator.evaluate(predictions)


    # la matriz de confusion revienta
    cnf_matrix = metrics.confusionMatrix()
    accuracy = round(metrics.accuracy*100, 3)
    f1 = metrics.fMeasure(1.0)
    recall = metrics.recall(1.0)

    print("Results of model %s"  % (estimator.__dict__['uid']))
    print("Accuracy %s" % accuracy)
    print("F1 %s" % f1)
    print("Recall %s" % recall)
    print("AUC %s" % auRocRF)

    return predictions, model

# Ya tengo captura de esta ejecucion
def random_forest_1(trainingData,testData):
    print("[Random Forest] init")
```

```python
    start_time = time()
    rf = RandomForestClassifier(labelCol="label", featuresCol="features", seed=12345)
    # ParamGridBuilder params:
    # https://spark.apache.org/docs/latest/ml-tuning.html
    paramGridRF = ParamGridBuilder().addGrid(rf.numTrees, [5, 10, 20]).addGrid(rf.maxDepth, [2, 3, 6]).build()

    predictionsRF, mRF = predictions(rf,paramGridRF,trainingData,testData)
    end_time = time()
    elapsed_time = end_time - start_time
    print("[Random Forest] With params %s" % paramGridRF)
    print("[Random Forest] time %s" %(elapsed_time))

# Ya tengo captura de esta ejecucion
def random_forest_2(trainingData,testData):
    print("[Random Forest] init")
    start_time = time()
    rf = RandomForestClassifier(labelCol="label", featuresCol="features", seed=2021)
    # ParamGridBuilder params:
    # https://spark.apache.org/docs/latest/ml-tuning.html
    paramGridRF = ParamGridBuilder().addGrid(rf.numTrees, [10, 30, 60]).addGrid(rf.maxDepth, [3, 6, 12]).build()
    predictionsRF, mRF = predictions(rf,paramGridRF,trainingData,testData)
    end_time = time()
    elapsed_time = end_time - start_time
    print("[Random Forest] With params %s" % paramGridRF)
    print("[Random Forest] time %s" %(elapsed_time))


def gradient_boosted_tree_1(trainingData, testData):
    print("[Gradient Boosted Tree] init")
    start_time = time()
    gbt = GBTClassifier(labelCol="label", featuresCol="features", seed=2021)
    #paramGridGBT = ParamGridBuilder().addGrid(gbt.maxIter, [10, 15, 20]).addGrid(gbt.maxDepth, [3, 6, 12]).build()
    paramGridGBT = ParamGridBuilder().addGrid(gbt.maxIter, [5, 10, 15]).addGrid(gbt.maxDepth, [2, 3, 9]).build()
    predictionsGBT, mGBT = predictions(gbt,paramGridGBT,trainingData,testData)
```

```
    end_time = time()
    elapsed_time = end_time - start_time
    print("[Gradient Boosted Tree] With params %s" % paramGridGBT)
    print("[Gradient Boosted Tree] time %s" %(elapsed_time))


def gradient_boosted_tree_2(trainingData, testData):
    print("[Gradient Boosted Tree] init")
    start_time = time()
    gbt = GBTClassifier(labelCol="label", featuresCol="features",
    ↪ seed=2021)
    paramGridGBT = ParamGridBuilder().addGrid(gbt.maxIter, [10, 15,
    ↪ 20]).addGrid(gbt.maxDepth, [3, 6, 12]).build()

    predictionsGBT, mGBT =
    ↪ predictions(gbt,paramGridGBT,trainingData,testData)
    end_time = time()
    elapsed_time = end_time - start_time
    print("[Gradient Boosted Tree] With params %s" % paramGridGBT)
    print("[Gradient Boosted Tree] time %s" %(elapsed_time))


def perceptron_1(trainingData, testData):
    print("[Peceptron] init")
    start_time = time()
    mlp = MultilayerPerceptronClassifier(
       featuresCol="features",
       labelCol="label",
       predictionCol="prediction",
       maxIter=100
    )
    mlpGrid = ParamGridBuilder().addGrid(mlp.layers, [[7, 3, 2], [7, 9, 3,
    ↪ 2], [7, 5, 2]]).build()
    predictionsMLP, mMLP = predictions(mlp, mlpGrid, trainingData,
    ↪ testData)
    end_time = time()
    elapsed_time = end_time - start_time
    print("[Peceptron] With params %s" % predictionsMLP)
    print("[Peceptron] time %s" %(elapsed_time))


# Ya tengo captura de esta ejecucion
```

```python
def logistic_regresion1(trainingData, testData):
    print("[Logistic Regression] init")
    start_time = time()
    lr = LogisticRegression(featuresCol="features",
    labelCol="label",maxIter=100,family="multinomial")
    lrGrid = ParamGridBuilder().addGrid(lr.regParam, [0.1, 0.01, 0.001]).addGrid(lr.elasticNetParam, [0.5, 0.6, 0.8]).build()
    predictionsRL, mRL = predictions(lr,lrGrid,trainingData,testData)
    end_time = time()
    elapsed_time = end_time - start_time
    print("[Logistic Regression] With params %s" % predictionsRL)
    print("[Logistic Regression] time %s" %(elapsed_time))

def logistic_regresion_2(trainingData, testData):
    print("[Logistic Regression] init")
    start_time = time()
    lr = LogisticRegression(featuresCol="features",
    labelCol="label",maxIter=100,family="multinomial")
    lrGrid = ParamGridBuilder().addGrid(lr.regParam, [0.1, 0.01, 0.001]).addGrid(lr.elasticNetParam, [0.6, 0.7, 0.9]).build()
    predictionsRL, mRL = predictions(lr,lrGrid,trainingData,testData)
    end_time = time()
    elapsed_time = end_time - start_time
    print("[Logistic Regression] With params %s" % predictionsRL)
    print("[Logistic Regression] time %s" %(elapsed_time))


def naive_bayes_1(trainingData, testData):
    print("[NaiveBayes] init")
    start_time = time()
    nb = NaiveBayes(modelType="multinomial", featuresCol="features", labelCol="label", smoothing=1.0)
    nbGrid = ParamGridBuilder().addGrid(1.0, [0.0, 0.2, 0.4, 0.6, 0.8, 1.0]).build()
    predictionsNB, mNB = predictions(nb,nbGrid,trainingData,testData)
    end_time = time()
    elapsed_time = end_time - start_time
    print("[NaiveBayes] With params %s" % predictionsNB)
    print("[NaiveBayes] time %s" %(elapsed_time))


if __name__ == "__main__":
```

```python
    print("Iniciando el contexto de Spark %s", title)
    configurationSpark = SparkConf().setAppName(title)
    sparkContexto = SparkContext.getOrCreate(conf=configurationSpark)
    sqlContext = sql.SQLContext(sparkContexto)

    df_columns = sqlContext.read.csv(name_file, sep=",", header=True, inferSchema=True)
    indexer = StringIndexer(inputCol="PredSS_r1_4", outputCol="PredSS_r1_4_indexado")
    df_columns = indexer.fit(df_columns).transform(df_columns)
    df_columns = df_columns.drop("PredSS_r1_4")
    df_columns = df_columns.withColumnRenamed("PredSS_r1_4_indexado","PredSS_r1_4")
    df_columns.show(20)

    clases_negativas = df_columns.filter(df_columns['class']==0).count()
    clases_positivas = df_columns.filter(df_columns['class']==1).count()
    print("El balanceo negativo/positivio es: %s / %s" % (clases_negativas, clases_positivas))

    #Me quedo con el numero de clases de la menor
    tam_partition = clases_positivas
    if(clases_positivas > clases_negativas):
        tam_partition = clases_negativas

    # Reduzco ambos al tamaño de la particion anterior: Undersampling
    df_0 = df_columns.filter(df_columns['class'] == 0).limit(tam_partition)
    df_1 = df_columns.filter(df_columns['class'] == 1).limit(tam_partition)

    df_balanced = df_1.union(df_0)
    df_train, df_test = df_balanced.randomSplit([portionTrain, portionTest])
    df_balanced_count = df_balanced.select('class').count()

    df_train_count = df_train.select('class').count()
    df_train_negative_count = df_train.filter(df_columns['class']==0).select('class').count()
    df_train_positive_count = df_train.filter(df_columns['class']==1).select('class').count()

    df_test_count = df_test.select('class').count()
```

```
    df_test_negative_count = df_test.filter(df_columns['class']==0).select('class').count()
    df_test_positive_count = df_test.filter(df_columns['class']==1).select('class').count()

    print("[Global] total: %s", df_balanced_count)
    print("[Train] positivas: %s, negativas %s, total %s" % (df_train_positive_count, df_train_negative_count, df_train_count ))
    print("[Test] positivas: %s, negativas %s, total %s" % (df_test_positive_count, df_test_negative_count, df_test_count ))

    # Feature Transformer VectorAssembler in PySpark ML Feature
    # https://medium.com/@nutanbhogendrasharma/
    # feature-transformer-vectorassembler-in-pyspark
    # -ml-feature-part-3-b3c2c3c93ee9
    assembler = VectorAssembler(inputCols=columns_asIndex, outputCol='features')
    trainingData = assembler.transform(df_train).select("features","class")
    .withColumnRenamed("class","label")
    testData = assembler.transform(df_test).select("features","class")
    .withColumnRenamed("class","label")


    # RandomForest - OK
    random_forest_1(trainingData, testData)
    random_forest_2(trainingData, testData)


    # Gradient Boosted Tree - OK
    gradient_boosted_tree_1(trainingData,testData)
    gradient_boosted_tree_2(trainingData,testData)

    # Regresion logistica- OK
    logistic_regresion1(trainingData, testData)
    logistic_regresion_2(trainingData, testData)


    # Perceptron multicapa - No funca
    # perceptron_1(trainingData,testData)
```

```python

    # Naive Bayes - No funca
    # naive_bayes_1(trainingData, testData)


    # https://stackoverflow.com/questions/60772315/
    # how-to-evaluate-a-classifier-with-pyspark-2-4-5
    # https://stackoverflow.com/questions/41714698/
    # how-to-get-accuracy-precision-recall
    # -and-roc-from-cross-validation-in-spark-ml
    print("FIN")
```

Apéndice B

Composición de Spark

```
version: '2'

services:
  spark:
    image: docker.io/bitnami/spark:3
    environment:
      - SPARK_MODE=master
      - SPARK_RPC_AUTHENTICATION_ENABLED=no
      - SPARK_RPC_ENCRYPTION_ENABLED=no
      - SPARK_LOCAL_STORAGE_ENCRYPTION_ENABLED=no
      - SPARK_SSL_ENABLED=no
    ports:
      - '8080:8080'
    volumes:
      - ./intercambio:/intercambio
  spark-worker-1:
    image: docker.io/bitnami/spark:3
    environment:
      - SPARK_MODE=worker
      - SPARK_MASTER_URL=spark://spark:7077
      - SPARK_WORKER_MEMORY=2G
      - SPARK_WORKER_CORES=2
      - SPARK_RPC_AUTHENTICATION_ENABLED=no
      - SPARK_RPC_ENCRYPTION_ENABLED=no
      - SPARK_LOCAL_STORAGE_ENCRYPTION_ENABLED=no
      - SPARK_SSL_ENABLED=no
    volumes:
```

```
      - ./intercambio:/intercambio
  spark-worker-2:
    image: docker.io/bitnami/spark:3
    environment:
      - SPARK_MODE=worker
      - SPARK_MASTER_URL=spark://spark:7077
      - SPARK_WORKER_MEMORY=2G
      - SPARK_WORKER_CORES=2
      - SPARK_RPC_AUTHENTICATION_ENABLED=no
      - SPARK_RPC_ENCRYPTION_ENABLED=no
      - SPARK_LOCAL_STORAGE_ENCRYPTION_ENABLED=no
      - SPARK_SSL_ENABLED=no
    volumes:
      - ./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/