

UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

FACULTAD DE CONTADURÍA Y ADMINISTRACIÓN

**MAESTRÍA EN GESTIÓN DE TECNOLOGÍAS DE LA INFORMACIÓN Y LA
COMUNICACIÓN**



Big Data

Alumnos:

Ledezma Ibarra Juan Francisco

López Ramírez Flavio Hiram

Práctica Final – Modelo Multilayer Perceptron Classifier.

Noviembre 2023, Tijuana Baja California, México

Contenido

Introducción	3
Implementación.....	4
Resultados	7
Conclusiones	8

Introducción

Los datos del dataset con que se trabajó, están relacionados con campañas de marketing directo (llamadas telefónicas) de una institución bancaria portuguesa. El objetivo es predecir la probabilidad de que un cliente abra una cuenta de inversión implementando el modelo Multilayer Perceptron Classifier.

Implementación

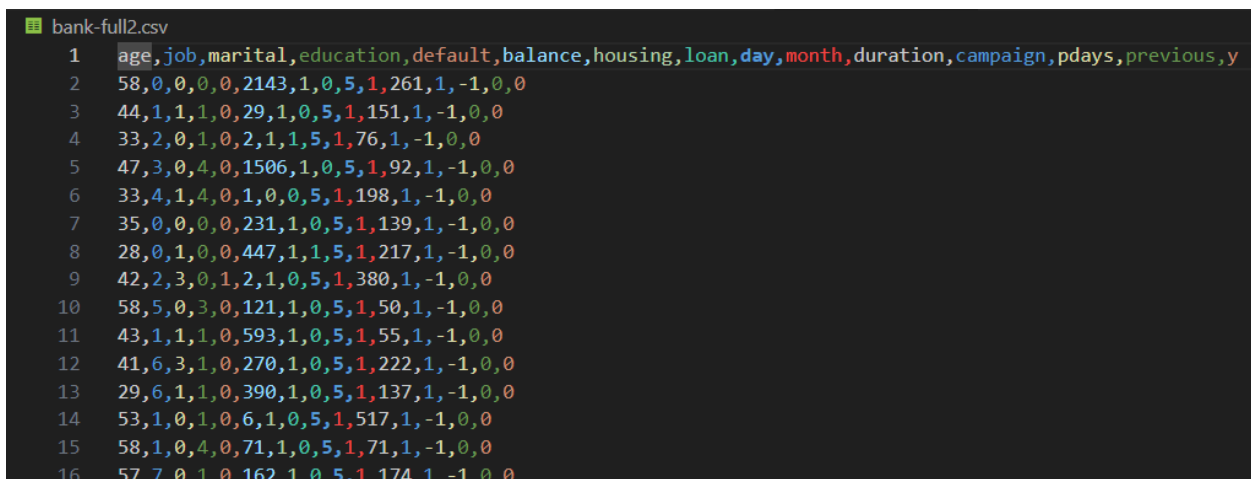
Para este proyecto que consiste en predecir la probabilidad de que un cliente abra una cuenta de inversión implementando el modelo Multilayer Perceptron Classifier, para la limpieza de datos se utilizó el programa de Microsoft Excel, para la generación del código se utilizó Scala, finalmente este código se ejecuto Spark-Shell.

Limpieza de datos:

1.- Se realizó limpieza de dataset, eliminando las columnas “contact” y “poutcome”, ya que se consideran irrelevantes para el objetivo de la clasificación.

2.- Se reemplazaron las “.” a “,” para su correcta importación a scala.

3.- Se convirtieron los datos categóricos a numéricos dando como resultado la estructura mostrada en la siguiente imagen.



```
1 age,job,marital,education,default,balance,housing,loan,day,month,duration,campaign,pdays,previous,y
2 58,0,0,0,0,2143,1,0,5,1,261,1,-1,0,0
3 44,1,1,1,0,29,1,0,5,1,151,1,-1,0,0
4 33,2,0,1,0,2,1,1,5,1,76,1,-1,0,0
5 47,3,0,4,0,1506,1,0,5,1,92,1,-1,0,0
6 33,4,1,4,0,1,0,0,5,1,198,1,-1,0,0
7 35,0,0,0,0,231,1,0,5,1,139,1,-1,0,0
8 28,0,1,0,0,447,1,1,5,1,217,1,-1,0,0
9 42,2,3,0,1,2,1,0,5,1,380,1,-1,0,0
10 58,5,0,3,0,121,1,0,5,1,50,1,-1,0,0
11 43,1,1,1,0,593,1,0,5,1,55,1,-1,0,0
12 41,6,3,1,0,270,1,0,5,1,222,1,-1,0,0
13 29,6,1,1,0,390,1,0,5,1,137,1,-1,0,0
14 53,1,0,1,0,6,1,0,5,1,517,1,-1,0,0
15 58,1,0,4,0,71,1,0,5,1,71,1,-1,0,0
16 57,7,0,1,0,162,1,0,5,1,174,1,-1,0,0
```

Dataset Limpio

4.- En las siguientes cuatro imágenes se ejecuta el código de escala el cual contiene las 10 corridas con las diferentes Seeds utilizada; de igual manera se presenta la evidencia del archivo readme.md y de los commits realizados en github:



Imagen 1 - Archivo Readme en Github

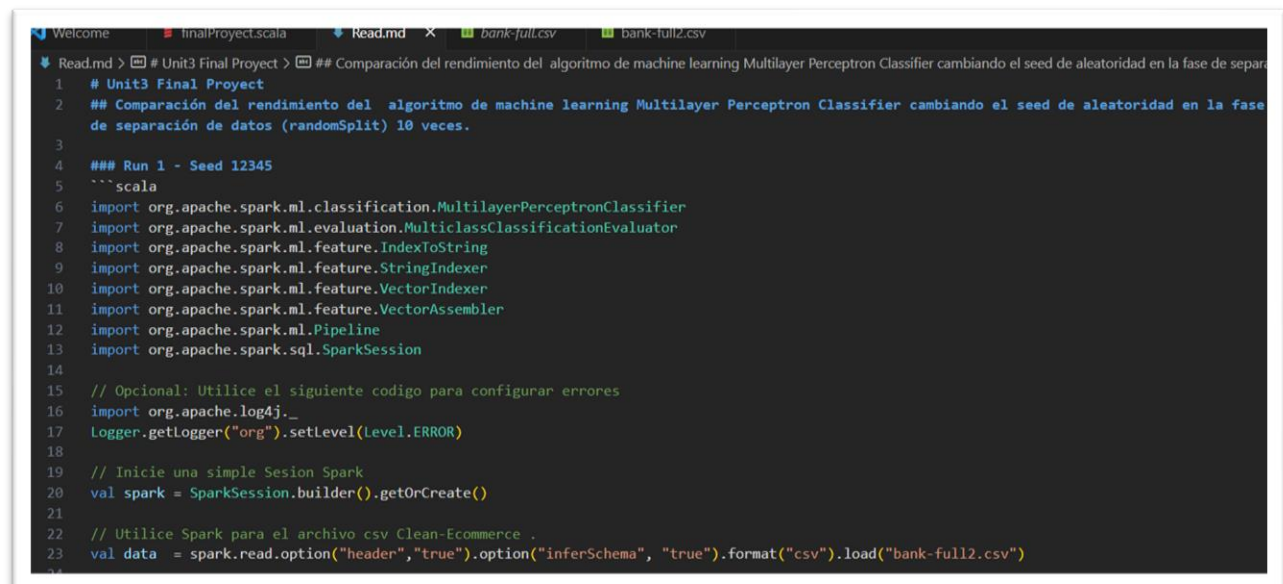


Imagen 2 - Archivo Fuente de Readme (Visual Code)

Imagen 3 - Historial de Commits Generados en Github

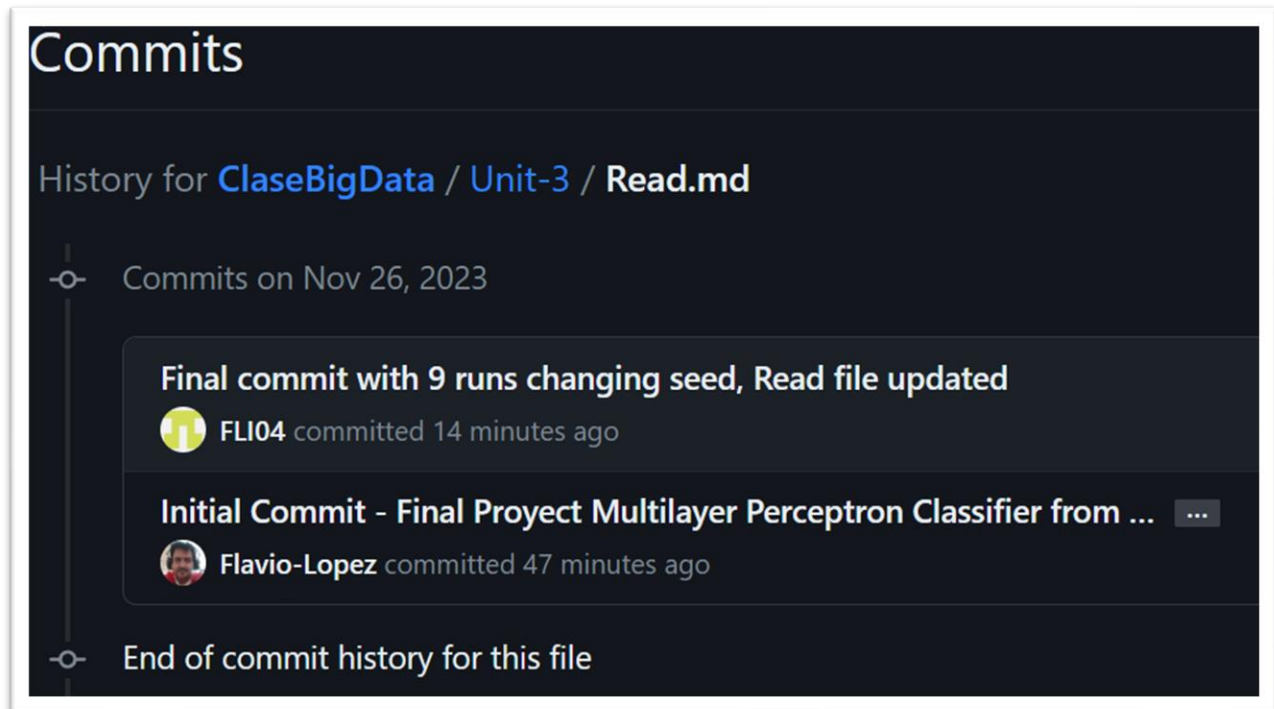


Imagen 4 - Código en Scala

```
Welcome  finalProject.scala X  Read.md  bank-full.csv  bank-full2.csv
finalProject.scala
38 //Transforma los datos label categoricos a numericos
39 val indexerLabel = new StringIndexer().setInputCol("y").setOutputCol("indexedLabel").fit(features)
40
41 //Transforma los datos features categoricos a numericos con limitante de 4
42 val indexerFeatures = new VectorIndexer().setInputCol("features").setOutputCol("indexedFeatures").setMax
43
44 //Se crean los arreglos que seran usados para training y test 70% y 30%
45 val Array(training, test) = features.randomSplit(Array(0.7, 0.3), seed = 12345) // vuelta 1
46 val Array(training, test) = features.randomSplit(Array(0.7, 0.3), seed = 54321) // vuelta 2
47 val Array(training, test) = features.randomSplit(Array(0.7, 0.3), seed = 41872) // vuelta 3
48 val Array(training, test) = features.randomSplit(Array(0.7, 0.3), seed = 24947) // vuelta 4
49 val Array(training, test) = features.randomSplit(Array(0.7, 0.3), seed = 45868) // vuelta 5
50 val Array(training, test) = features.randomSplit(Array(0.7, 0.3), seed = 17875) // vuelta 6
51 val Array(training, test) = features.randomSplit(Array(0.7, 0.3), seed = 93172) // vuelta 7
52 val Array(training, test) = features.randomSplit(Array(0.7, 0.3), seed = 22287) // vuelta 8
53 val Array(training, test) = features.randomSplit(Array(0.7, 0.3), seed = 65444) // vuelta 9
54 val Array(training, test) = features.randomSplit(Array(0.7, 0.3), seed = 80975) // vuelta 10
55
56
57 //Se crean las capas
58 val layers = Array[Int](14,6,2,2)
59
60 //Se crea la estructura del modelo Multilayer Perceptron.
61 val trainer = new MultilayerPerceptronClassifier().setLayers(layers).setLabelCol("indexedLabel").setFeat
62
```

Resultados

En los resultados se puede observar como la exactitud del modelo cambia respecto a los Seeds utilizados.

Run Seed Accuracy (%)		
1	12345	88.22580645
2	54321	88.61542996
3	41872	88.45173042
4	24947	88.99096054
5	45868	88.32555037
6	17875	87.98876488
7	93172	87.91420118
8	22287	88.48350019
9	65444	88.430724
10	80975	88.36809449

Conclusiones

Se llegó a la conclusión que el modelo entrenado con el Seed 24947 (run 4) fue el que nos arrojó el mejor porcentaje de exactitud con el 88.99%.