Comparación Del Rendimiento De Los Algoritmos SVM, Logistic Regression, Multilayer Perceptron

Enciso Maldonado Aileen Yurely, Garcia Bautista Ana Laura

Resumen – En el siguiente trabajo se pretende hacer un análisis de los siguientes algoritmos que son Logistic Regression, Decision three, Multilayer Perceptron, por lo cual se espera hacer una comparación de su rendimiento de dichos algoritmos a analizar para poder determinar cual de estos algoritmos está el más rápido y preciso. Este trabajo esta siendo realizado para la materia de datos masivos impartida por el Dr. José Christian Romero Hernández.

Palabras Claves: Logistic Regression, Decision three, Multilayer Perceptron.

I. INTRODUCCION

En este trabajo se hace un análisis de datos que es una implementación de datos masivos más utilizada en nuestra actualidad debido la cantidad de datos que se manejan hoy en día que son una gran cantidad de datos a analizar por lo cual al hacer la comparación de los algoritmos Logistic Regression, Decision three, Multilayer Perceptron, son los algoritmos que pretendemos utilizar para realizar esta investigación. En base a estos algoritmos se pretende hacer un análisis de los datos de un sistema bancario para analizar que algoritmo puede ser mas eficiente para el sistema de banco basado en su comportamiento. Para la realización de la comparación del trabajo de utilizo la herramienta de Spark-Scala que nos ayuda tener un mejor control de nuestros datos a utilizar.

II. MARCO TEÓRICO

Hoy en día el análisis de datos es algo de lo más común cuando se requiere analizar datos de gran magnitud por lo cual en este análisis de datos de un sistema bancario se realizará una clasificación de sistema supervisado que se define como clases o etiqueta base a los datos a analizar. Para poder determinar que algoritmo favorece al sistema bancario se analizara el tiempo y presión de cada uno de los algoritmos. En base a nuestro trabajo los datos proporcionados por el sistema bancario son un conjunto de datos que se analizara el sistema bancario para así tener un mejor control de su sistema.

Algoritmo SVM

Las SVM son una de las técnicas más poderosas del aprendizaje automático. Consiste en construir un hiperplano en un espacio de dimensionalidad muy alta (o incluso infinita) que separe las clases que tenemos. Una buena separación entre las clases permitirá una clasificación correcta de la nueva muestra, es decir, necesitamos encontrar la máxima separación a los puntos más cercanos a este hiperplano.

El objetivo de los problemas de clasificación que aplican este tipo algoritmos de aprendizaje supervisado es el siguiente; dado un conjunto de entrenamiento con sus etiquetas de clase, entrenar una SVM para construir un modelo que prediga la clase de una nueva muestra o conjunto de test.

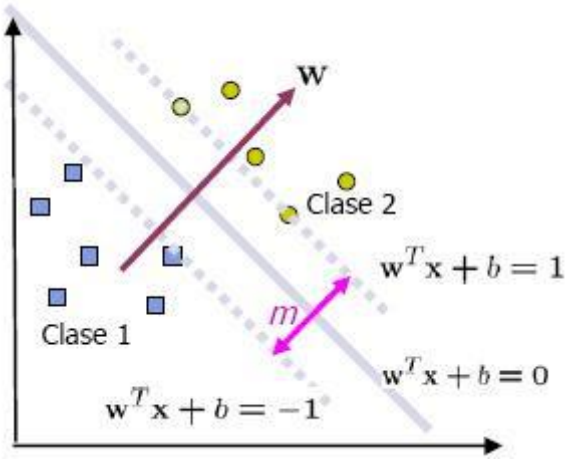

Figure 1 Algoritmo SVM

Algoritmo Decision Three

Árbol de decisión o Decisión Tree Classification es un tipo de algoritmo de aprendizaje supervisado que se utiliza principalmente en problemas de clasificación, aunque funciona para variables de entrada y salida categóricas como continuas.

Los algoritmos de aprendizaje basados en árbol se consideran uno de los mejores y más utilizados métodos de aprendizaje supervisado. Los métodos basados en árboles potencian

modelos predictivos con alta precisión, estabilidad y facilidad de interpretación. A diferencia de los modelos lineales, mapean bastante bien las relaciones no lineales.

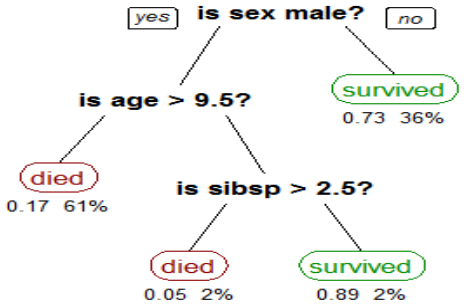

Figure 2 Decision Three

Algoritmo Logistic Regression

La regresión logística o Logistic Regression es un algoritmo de clasificación que se utiliza para predecir la probabilidad de una variable dependiente categórica. Como todos los análisis de regresión, la regresión logística es un análisis predictivo. Se usa para describir datos y explicar la relación entre una variable binaria dependiente y una o más variables independientes nominales, ordinales, de intervalo o de nivel de razón

Este modelo logístico binario se utiliza para estimar la probabilidad de una respuesta binaria basada en una o más variables predictoras o independientes. Permite decir que la presencia de un factor de riesgo aumenta la probabilidad de un resultado dado un porcentaje específico.

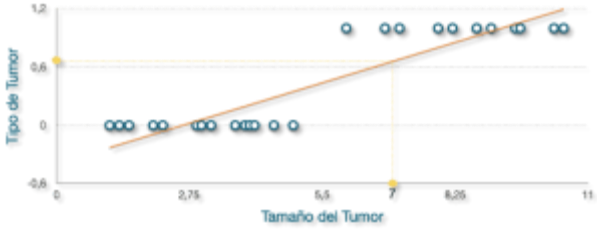

Figure 3 Logistic Regression

Algoritmo Multilayer Perceptron

El perceptrón multicapa es el hola mundo del aprendizaje profundo: un buen lugar para comenzar cuando estás aprendiendo sobre el aprendizaje profundo.

Un perceptrón multicapa (MLP) es una red neuronal profunda y artificial. Está compuesto por más de un perceptrón. Están compuestos por una capa de entrada para recibir la señal, una capa de salida que toma una decisión o predicción sobre la entrada, y entre esos dos, un número arbitrario de capas ocultas que son el verdadero motor computacional del MLP. Los MLP con una capa oculta son capaces de aproximar cualquier función continua.

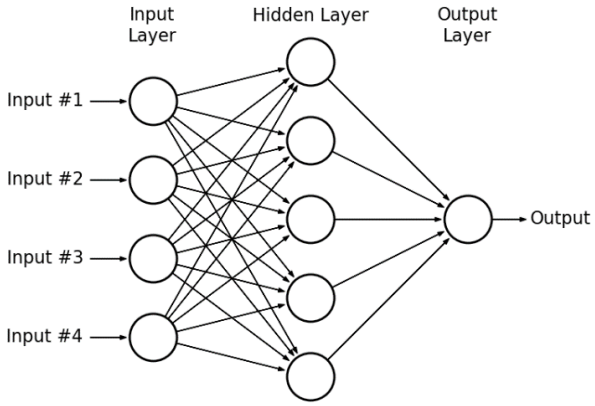

Figure 4 Multilayer Perceptron

III. IMPLANTACIÓN

Apche Spark

Spark es un motor ultrarrápido para el almacenamiento, procesamiento y análisis de grandes volúmenes de datos. Es de código abierto y se encuentra gestionado por la Apache Software Fundación. Por tanto, la herramienta se conoce como Apache Spark y es uno de sus proyectos más activos.

Figure 5 Spark

Visual Studio Code

Este software se utilizó como interpretación de lenguaje de scala

Figure 6 Visual Code

.

IV. Resultados

SVM

El código de SVM nos permitirá encontrar a que categoría pertenecen los conjuntos de datos ya que este algoritmo es un modelo basado predicciones de datos y poder organizarlos por categorías.

Código

```
import org.apache.spark.ml.classification.LinearSVC
val change = df.withColumnRenamed("y", "label")
val ft = change.select("label","features")
ft.show()
val cs1 = ft.withColumn("label",when(col("label").equalTo("1"),0).otherwise(col("label")))
val cs2 = cs1.withColumn("label",when(col("label").equalTo("2"),1).otherwise(col("label")))
val cs3 = cs2.withColumn("label",'label.cast("Int"))
val lsvc = new LinearSVC().setMaxIter(10).setRegParam(0.1)
val lsvcModel = lsvc.fit(cs3)
println(s"Coefficients: ${lsvcModel.coefficients} Intercept: ${lsvcModel.intercept}")
```

Ilustración 8 Código SVM

Resultado

```
+--------+-----+--------+--------+---+--------------------+-----+
|campaign|pdays|previous|poutcome|  y|            features|label|
+--------+-----+--------+--------+---+--------------------+-----+
|       1|   -1|       0| unknown|  2|[2143.0,5.0,261.0...|[2.0]|
|       1|   -1|       0| unknown|  2|[29.0,5.0,151.0,-...|[2.0]|
|       1|   -1|       0| unknown|  2|[2.0,5.0,76.0,-1....|[2.0]|
|       1|   -1|       0| unknown|  2|[1506.0,5.0,92.0,...|[2.0]|
|       1|   -1|       0| unknown|  2|[1.0,5.0,198.0,-1...|[2.0]|
+--------+-----+--------+--------+---+--------------------+-----+
```

Ilustración 9 Resultado SVM

Decision Three

En este algoritmo nos predice en conjunto de datos utilizando arboles de decisión con base a estas clasificaciones nos va prediciendo y categorizando los datos dependiendo a su clasificación del algoritmo.

Código

```
import org.apache.spark.ml.Pipeline
import org.apache.spark.ml.evaluation.RegressionEvaluator
import org.apache.spark.ml.regression.DecisionTreeRegressionModel
import org.apache.spark.ml.regression.DecisionTreeRegressor


val featureIndexer = new VectorIndexer().setInputCol("features")
.setOutputCol("indexedFeatures").setMaxCategories(4).fit(c55)
val Array(trainingData, testData) = c55.randomSplit(Array(0.7, 0.3))
val dt = new DecisionTreeRegressor().setLabelCol("label").setFeaturesCol("indexedFeatures")
val pipeline = new Pipeline().setStages(Array(featureIndexer, dt))
val model43 = pipeline.fit(trainingData)
val predictions = model.transform(testData)
predictions.select("prediction", "label", "features").show(5)
```

Ilustración 10 Codigo DT

Resultado

```
+--------------+-----+--------------------+
|predictedLabel|label|            features|
+--------------+-----+--------------------+
|           0.0|  0.0|[1944.0,10.0,122....|
|           0.0|  1.0|[103.0,10.0,104.0...|
|           0.0|  0.0|[1803.0,10.0,59.0...|
|           0.0|  0.0|[66.0,19.0,75.0,-...|
|           0.0|  0.0|[336.0,5.0,133.0,...|
|           0.0|  1.0|[76.0,18.0,639.0,...|
|           0.0|  0.0|[129.0,13.0,190.0...|
|           0.0|  0.0|[-103.0,13.0,180....|
|           0.0|  0.0|[292.0,4.0,45.0,-...|
|           0.0|  0.0|[130.0,11.0,88.0,...|
+--------------+-----+--------------------+
only showing top 10 rows

evaluator: org.apache.spark.ml.evaluation.Mult
accuracy: Double = 0.8878825505806734
Test Error = 0.11211744941932655
```

Ilustración 11 Resultado DT

Logistic Regression

En este algoritmo podremos ver el análisis estadístico de la regresión lineal que al igual que los demás nos va a categorizar los datos dependiendo de estos y a su vez nos limita una a una cierta cantidad de números, pero nos muestra lo esencial que es el poder categorizar y predecir nuestros datos de manera de conjuntos lineales.

Código

```
val evaluator = new RegressionEvaluator().setLabelCol("label")
.setPredictionCol("prediction").setMetricName("rmse")
val  e4= predictions.withColumn("prediction",'prediction.cast("Double"))
val rmse = evaluator.evaluate(e4)
```

Ilustración 12 Código LR

Resultado

```
test: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [
lr: org.apache.spark.ml.classification.LogisticRegression = logr
pipeline: org.apache.spark.ml.Pipeline = pipeline_16ff6dfd724b
model: org.apache.spark.ml.PipelineModel = pipeline_16ff6dfd724b
results: org.apache.spark.sql.DataFrame = [age: int, job: string
//////////////////////////  R  //////////////////////////
predictionAndLabels: org.apache.spark.rdd.RDD[(Double, Double)]
metrics: org.apache.spark.mllib.evaluation.MulticlassMetrics = o
11958.0  202.0
1309.0   285.0
0.8901410498763996
```

Ilustración 13 Resultado LR

Multilayer Perceptron

En este algoritmo se realiza la precisión a través de capas separando los datos y categorizándolos dependiendo del problema establecido por los datos dado que no se pueden separar de manera línea si no como se menciona que se realiza la categorización por capas y predicción de los datos de manera más fácil y de forma óptima.

Código

```
import org.apache.spark.ml.classification.MultilayerPerceptronClassifier
import org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator
val splits = categories.randomSplit(Array(0.6, 0.4), seed = 1234L)
val train = splits(0)
val test = splits(1)
val layers = Array[Int](2, 1, 3, 3)
val trainer = new MultilayerPerceptronClassifier().setLayers(layers)
.setBlockSize(128).setSeed(1234L).setMaxIter(100)
val  c4= categories.withColumn("prediction",'prediction.cast("Double"))
val  c5= c4.withColumn("label",'prediction.cast("Double"))
val predictionAndLabels = c5 .select("prediction", "label")
val evaluator = new MulticlassClassificationEvaluator().setMetricName("accuracy")
println(s"Test set accuracy = ${evaluator.evaluate(predictionAndLabels)}")
```

Ilustración 14 Código MP

Resultado

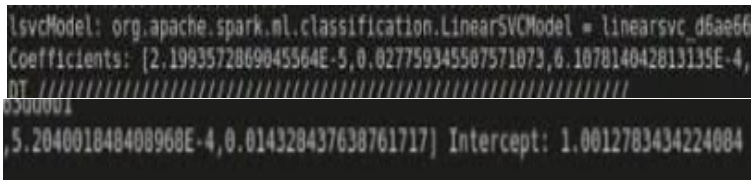

Ilustración 15 Resultado MP

Tabla

DataSet	Algoritmo	Tiempo	Repeticiones	Resultados
Bank.csv	MP	28	12	0.88
Bank.csv	SVM	30	12	0.88
Bank.csv	LP	25	12	0.89
Bank.csv	DT	27	12	1.0

Ilustración 16 Tabulación

V. Conclusiones

Como resultado de nuestro proyecto basado en los algoritmos de aprendizaje automático presenciamos que el algoritmo más eficiente es el Multiplayer Percepcion debido a que fue el que menos tiempo nos da análisis de datos y con mejor percepción al procesar el dataset y nos clasifico el 85 % de los datos utilizados de manera correcta esto trabajo se realizó para la materia de datos masivos impartida por el maestro José Christian Romero Hernández

VI. Referencias

I. Algoritmos SVM para problemas sobre Big Data, Yvonne Gala García, José Ramon Dorronsoro Ibero, 25 de septiembre de 2013 https://repositorio.uam.es/bitstream/handle/10486/14108/66152_Yvonne_Gala_Garcia.pdf?sequence=1&isAllowed=y [1].

II. Aprendizaje Supervisado: Logistic Regression, Ligdi González, 22 Marzo, 2018 https://ligdigonzalez.com/aprendizaje-supervisado-logistic-regression/#:~:text=La%20regresi%C3%B3n%20log%C3%ADstica%20o%20Logistic,de%20una%20variable%20dependiente%20categ%C3%B3rica.&text=Permite%20decir%20que%20la%20presencia,resultado%20dado%20un%20porcentaje%20espec%C3%ADfico.[2]

III. Aprendizaje Supervisado: Decision Tree Classification, Ligdi González, 23 Marzo, 2018 https://ligdigonzalez.com/aprendizaje-supervisado-decision-tree-classification/#:~:text=%C3%81rbol%20de%20decisi%C3%B3n%20o%20Decisi%C3%B3n,y%20salida%20categ%C3%B3ricas%20como%20continuas.&text=Las%20ventajas%20que%20tiene%20este,F%C3%A1cil%20de%20entender[3].

IV. Una guía para principiantes de perceptrones multicapa (MLP), Chris Nicholson https://pathmind.com/wiki/multilayer-perceptron [4].

V. ¿Qué es Spark y cómo revoluciona al Big Data y al Machine Learning?, Yhorman Sierra, 2018 https://blog.mdcloud.es/que-es-spark-big-data-y-machine-learning/