TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE TIJUANA

SUBDIRECCIÓN ACADÉMICA
DEPARTAMENTO DE SISTEMAS Y COMPUTACIÓN

SEMESTRE:
Enero - Junio 2020

CARRERA:
Ing. Tecnologías de la Información y Comunicaciones

MATERIA:
Datos Masivos

TÍTULO:
Rendimiento de algoritmos de Machine Learning

UNIDAD A EVALUAR:
Unidad 4

NOMBRE Y NÚMERO DE CONTROL DEL ALUMNO:
Diaz Martinez Ruben Emilio # 15210791

Flores Acosta Alfredo # 15210331

NOMBRE DEL MAESTRO (A):
Dr. Jose Christian Romero Hernandez

Jueves 18 de Junio 2020

II .

Índice

III. Introducción

En este documento se mostrará la comparación de algoritmos de Machine Learning donde se muestran los resultados de manera en que se puedan distinguir entre si, para eso se proyectarán de forma tabulada para una mejor expresión de ellos, por ende podremos optar por el resultado más favorable para esto se tomarán aspectos como su rapidez y precisión, para ello ocuparemos un DATASET llamado " BANK MARKETING DATA SET "

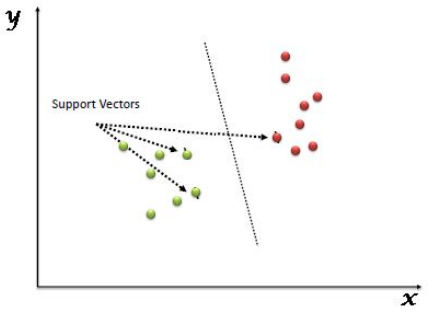
Ahí nos podrán proporcionar las descarga del DATASET y como la del Data Set Description para poder extraer los valores que se necesita extraer más adelante. Una vez dicho esto comenzaremos con los puntos establecidos que se muestra en el índice de la página 2 de esta documentación.

IV. Marco Teórico de los Algoritmos

1. SVM

Support Vector Machine "(SVM) es un algoritmo de aprendizaje automático supervisado que se puede utilizar para desafíos de clasificación o regresión. Sin embargo, se usa principalmente en problemas de clasificación.

En el algoritmo SVM, graficamos cada elemento de datos como un punto en el espacio n-dimensional (donde n es el número de entidades que tiene) con el valor de cada entidad como el valor de una coordenada particular. Luego, realizamos la clasificación al encontrar el hiperplano que diferenciar muy bien las dos clases.



[Imagen 1 : Clasificación de los valores]

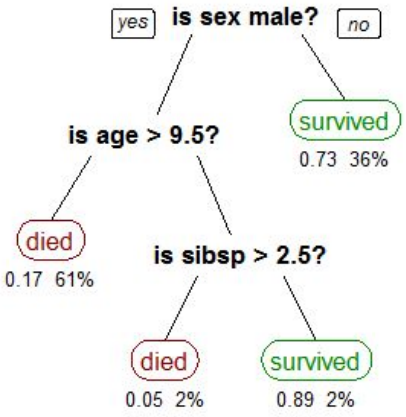
5. Decision Tree

Un árbol tiene muchas analogías en la vida real, y resulta que ha influido en un área amplia del aprendizaje automático, cubriendo tanto la clasificación como la regresión. En el análisis de decisiones, se puede usar un árbol de decisiones para representar visual y explícitamente las decisiones y la toma de decisiones. Como su nombre indica, utiliza un modelo de decisiones en forma de árbol. Aunque es una herramienta de uso común en la minería de datos para derivar una estrategia para alcanzar un objetivo en particular, también se usa ampliamente en el aprendizaje automático, que será el enfoque principal de este artículo.

Se dibuja un árbol de decisión al revés con su raíz en la parte superior. En la imagen de la izquierda, el texto en negrita en negro representa una condición / nodo interno, en función del cual el árbol se divide en ramas / bordes. El final de la rama que ya no se divide es la decisión / hoja, en este caso, si el pasajero murió o sobrevivió, representado como texto rojo y verde respectivamente..

¿Cómo se puede representar un algoritmo como un árbol?

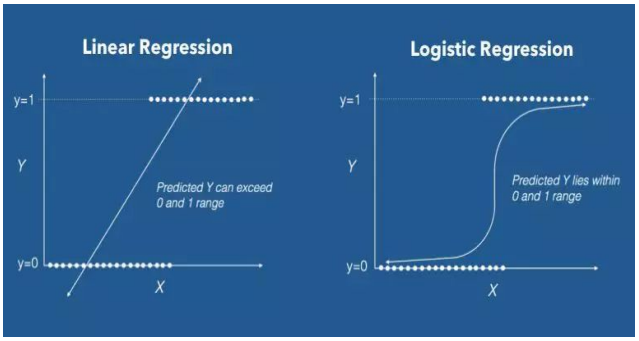
Para esto, consideremos un ejemplo muy básico que utiliza un conjunto de datos titánico para predecir si un pasajero sobrevivirá o no. El siguiente modelo utiliza 3 características / atributos / columnas del conjunto de datos, a saber, sexo, edad y sibsp (número de cónyuges o hijos).



[Imagen 3 : Ejemplo de Algoritmo Decision Three]

3. Logistic Regression

La regresión logística es un algoritmo de modelado predictivo que se usa cuando la variable Y es binaria categórica. Es decir, solo puede tomar dos valores como 1 o 0. El objetivo es determinar una ecuación matemática que se pueda usar para predecir la probabilidad del evento 1. Una vez que se establece la ecuación, se puede usar para predecir la Y cuando solo Las X son conocidas.



[Imagen 4 : Ejemplo de Logistic Regression]

4. Multilayer Perceptron

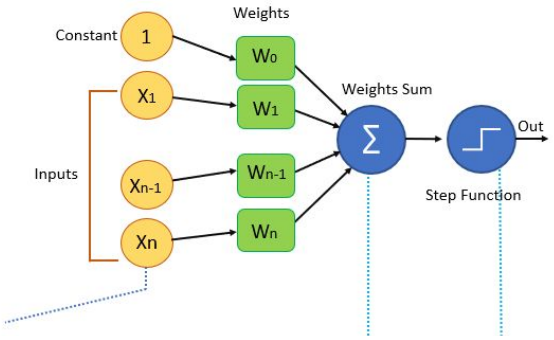
El siguiente artículo ofrece un resumen del Algoritmo de aprendizaje del Perceptrón. ¿A qué se refiere la palabra Perceptrón en la industria del aprendizaje automático? Perceptrón es una unidad de red neuronal artificial que realiza cálculos para comprender mejor los datos. ¿Qué es una unidad de red neuronal? Un grupo de neuronas artificiales interconectadas entre sí a través de conexiones sinápticas se conoce como una red neuronal.

¿Qué es una neurona artificial?

Teniendo en cuenta el estado del mundo de hoy y para resolver los problemas que nos rodean, estamos tratando de determinar las soluciones al comprender cómo funciona la naturaleza, esto también se conoce como biomimética. Del mismo modo, para funcionar como cerebros humanos, las personas desarrollan neuronas artificiales que funcionan de manera similar a las neuronas biológicas en un ser humano. Una neurona artificial es una función matemática compleja, que toma datos y pesos por separado, los fusiona y los pasa a través de la función matemática para producir resultados.

Algoritmo de aprendizaje perceptrónico

El algoritmo del Perceptrón se usa en un dominio de aprendizaje automático supervisado para la clasificación. En la clasificación, hay dos tipos de clasificación lineal y clasificación no lineal. La clasificación lineal no es más que si podemos clasificar el conjunto de datos dibujando una línea recta simple, entonces se puede llamar un clasificador binario lineal. Mientras que si no podemos clasificar el conjunto de datos dibujando una línea recta simple, entonces se puede llamar un clasificador binario no lineal.



[Imagen 5: Perceptron Algorithm Block Diamgram]

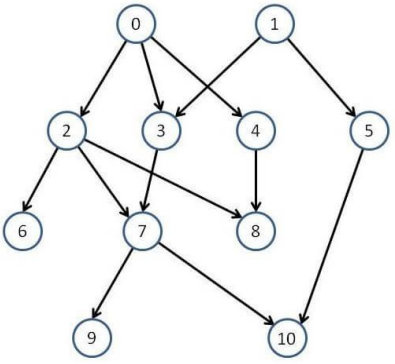
V. Implementación

En este proceso de la comparación del rendimiento en cada uno de los algoritmos se requerirá de las siguientes herramientas de software para poderlo realizar. Se explicara el porque el uso de ellas en los algoritmos de Machine Learning

Apache Spark combina un sistema de computación distribuida a través de clusters de ordenadores con una manera sencilla y elegante de escribir programas. Fue creado en la Universidad de Berkeley en California y es considerado el primer software de código abierto que hace la programación distribuida realmente accesible a los científicos de datos.

Es sencillo entender Spark si lo comparamos con su predecesor, MapReduce, el cual revolucionó la manera de trabajar con grandes conjuntos de datos ofreciendo un modelo relativamente simple para escribir programas que se podían ejecutar paralelamente en cientos y miles de máquinas al mismo tiempo. Gracias a su arquitectura, MapReduce logra prácticamente una relación lineal de escalabilidad, ya que si los datos crecen es posible añadir más máquinas y tardar lo mismo.

Spark mantiene la escalabilidad lineal y la tolerancia a fallos de MapReduce, pero amplía sus bondades gracias a varias funcionalidades: **DAG** y **RDD**.



[Imagen 7 : Ejemplo de un DAG con 10 nodos]

V. ¿Por que el uso de estas herramientas (Scala - Spark) ?

¿Por qué spark?

Dado que casi todas las computadoras personales hoy en día tienen muchos Gigabytes de RAM (y está en un crecimiento acelerado) y CPU y GPU potentes, muchos problemas de aprendizaje automático del mundo real se pueden resolver con una sola computadora y marcos como Scikit Learn, sin necesidad de un sistema distribuido, esto es, un grupo de muchas computadoras. A veces, sin embargo, los datos crecen y siguen creciendo. ¿Quién nunca escuchó el término "Big Data"? Cuando sucede, una solución no distribuida / escalable puede resolverse por un corto tiempo, pero luego será necesario revisar dicha solución y tal vez cambiarla significativamente

¿Por qué Scala?

Scala es un lenguaje de programación hermoso y muy bien diseñado, con una sólida formación científica del equipo de investigación del profesor Martin Odersky en la Ecole Polytechnique Fédérale de Lausanne.

En términos más técnicos, Scala se creó con un fuerte paradigma funcional, pero también totalmente compatible con el imperativo paradigma orientado a objetos de la plataforma JVM, aprovechando todas las décadas de evolución y madurez de JVM. En resumen, todo lo que uno hace en Java se puede hacer en Scala y mucho más con un código mucho más corto y limpio.

No es sorprendente que Spark esté construido precisamente sobre Scala, aunque también proporciona interfaces de programación para Python, R y, naturalmente, Java.a

VI. Resultados (Tabulación)

En esta parte haremos las comparaciones con cada uno de los algoritmos tomando en cuenta su performance como lo son sus tiempos de respuesta y su precisión.

Recordaremos que para cada algoritmo se tuvo que realizar una cantidad de 10 iteraciones para ver el cambio de resultados en cada una de ellas

Comenzaremos con el primer algoritmo Support

1. Vector Machine (SVM)

Para este algoritmo, los resultados fueron los siguientes:

Coeficientes:**[-2.125897501491213E,-0.135172 27458849872,7.51402188801716E-4.2.7023337 5064008964E,0.0111177544540215354]**
Intercep : -1.084924165339881

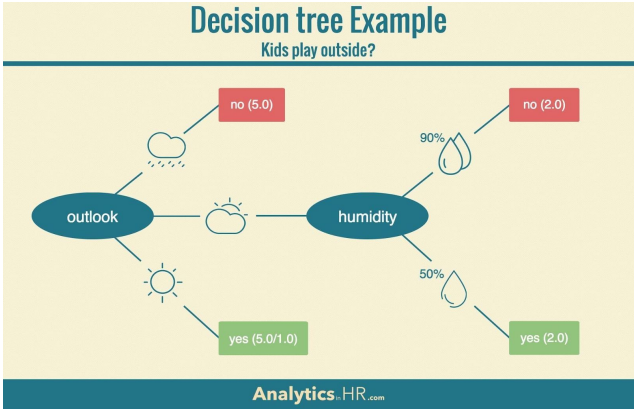
Con una Precisión de : **0.8881574025585397**

```
scala> svma()
20/06/16 16:46:07 WARN BLAS: Failed to load implementation from:
20/06/16 16:46:07 WARN BLAS: Failed to load implementation from:

Algoritmo Linear Support Vector Machine Accuracy

Precision:
0.8881574025585397

scala>
```

[Imagen 8 : Resultados del algoritmo SVM]

2. Decision Tree

Para este segundo Algoritmo, es un poco más complejo debido a las interacciones que se tiene que realizar con cada uno de los nodos hasta llegar a un resultado, esto debido a que tu proceso se interpreta como un árbol ya que cada nodo hace referencia a las hojas del árbol, haciendo de su proceso un poco más exhausto



[Imagen 9 : Ejemplo de un Árbol Decision Tree]

Los resultados este algoritmo nos arroja un una prueba de error de : **0.11107837360047146**

Como era de esperarse nos arrojó el esquema del árbol, acomodandolo dependiendo su valor Mayor > o < Menor

```
scala> println(s"Learned classification tree model:\n\n ${treeModel.toDebugStr
Learned classification tree model:

 DecisionTreeClassificationModel (uid=dtc_aa04c7392942) of depth 5 with 37 nod
  If (feature 2 <= 544.5)
   If (feature 3 <= 12.5)
    Predict: 0.0
   Else (feature 3 > 12.5)
    If (feature 2 <= 179.5)
     Predict: 0.0
    Else (feature 2 > 179.5)
     If (feature 3 <= 188.5)
      If (feature 3 <= 96.5)
       Predict: 1.0
      Else (feature 3 > 96.5)
       Predict: 0.0
     Else (feature 3 > 188.5)
      If (feature 3 <= 465.0)
       Predict: 0.0
      Else (feature 3 > 465.0)
       Predict: 1.0
  Else (feature 2 > 544.5)
   If (feature 2 <= 862.0)
    If (feature 3 <= 0.0)
     If (feature 2 <= 659.5)
      Predict: 0.0
     Else (feature 2 > 659.5)
```

[Imagen 10 : Resultado del esquema Tree]

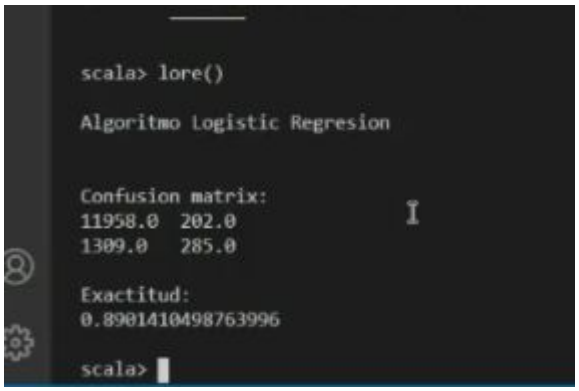
```
OUTPUT  TERMINAL  DEBUG CONSOLE  PROBLEMS

        Predict: 1.0
      Else (feature 2 > 877.0)
       If (feature 4 <= 1.5)
        Predict: 1.0
       Else (feature 4 > 1.5)
        If (feature 1 <= 1.5)
         Predict: 0.0
        Else (feature 1 > 1.5)
         Predict: 1.0

Exactitud
0.8889216263995285
```

[Imagen 11: Resultado de Exactitud del Algoritmo Decision Tree]

3. Logistic Regression

Este es el algoritmo que consideramos el correcto debido al tiempo de respuesta y la exactitud de su predicción y con un poco porcentaje de errores en su modelo estructural para el resultado final.

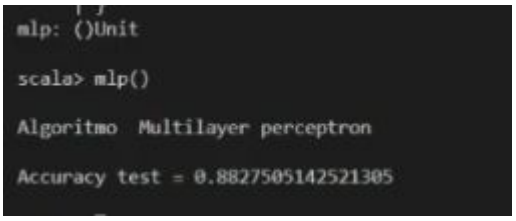
Cuando obtenemos los datos, después de la limpieza de los datos, el procesamiento previo y las disputas, el primer paso que hacemos es alimentar a un modelo sobresaliente y, por supuesto, obtener resultados en las probabilidades.



[Imagen 12 : Resultado de precisión de Logistic Regression]

4. Multilayer Perceptron

Bueno este algoritmo es un poco más complejo debido a que es un modelo neuronal, su tiempo de respuesta es un poco más extenso pero de igual manera tiene un buen performance como el resto de los algoritmos, su única desventaja seria la espera de respuesta .



[Imagen 13 : Resultado de MP]

Iteración (proceso)	SVM	TREE	LR	MLP
1	0.881	0.8942	0.8901	0.8827
2	0.881	0.8990	0.8706	0.8827
3	0.881	0.8928	0.8901	0.8827
4	0.881	0.8920	0.8991	0.8827
5	0.881	0.8946	0.8886	0.8827
6	0.881	0.8956	0.8954	0.8827
7	0.881	0.8998	0.8953	0.8827
8	0.881	0.8953	0.8954	0.8827
9	0.881	0.8928	0.8853	0.8827
10	0.881	0.8944	0.8994	0.8827

Average	0.881	0.8944	0.8994	0.8827

VII . Conclusiones

El performance de cada uno de los algoritmos son muy buenos, pero cada algoritmo es para un caso especial, si se busca una respuestas más certera es Logistic o el caso contrario si tienes un modelo más complejo con una estructura supervisada sería Multilayer, estos algoritmos de Machine Learning se adaptan dependiendo la necesidad, puesto que el algoritmo perfecto es el que uno mismo considere. Esto es una amplia gama que tiene Machine Learning al proyectar este tipo de performance sobres los diversos algoritmos que se presentaron. En este caso optamos por Logistic Regression, gracias a su performance que dio, con una velocidad de respuesta muy rápida y la exactitud de las predicciones del mismo

VIII . Referencias

Portilla, J. (2019, 13 septiembre). Scala and Spark for Big Data and Machine Learning. Recuperado 17 de junio de 2020, de https://www.udemy.com/course/scala-and-spark-for-big-data-and-machine-learning/

Ray, S. (2020, 15 abril). Understanding Support Vector Machine(SVM) algorithm from examples (along with code). Recuperado 17 de junio de 2020, de https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/

Gupta, P. (2017, 17 mayo). Decision Trees in Machine Learning. Recuperado 17 de junio de 2020, de https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052+

Prabhakaran, S. (2018, 23 abril). Logistic Regression – A Complete Tutorial With Examples in R. Recuperado 17 de junio de 2020, de https://www.machinelearningplus.com/machine-learning/logistic-regression-tutorial-examples-r/+