Instituto Tecnologico de Tijuana

Ingeniería en Sistemas Computacionales

Materia
Datos Masivos

Proyecto Final

Profesor:
Jose Christian Romero Hernandez

Alumno
16211976 Bermudez Ornelas Alberto
16212372 Zuñiga Sosa Ruben

Fecha
14/06/2020

TECNOLÓGICO NACIONAL DE MÉXICO

Índice

Introducción

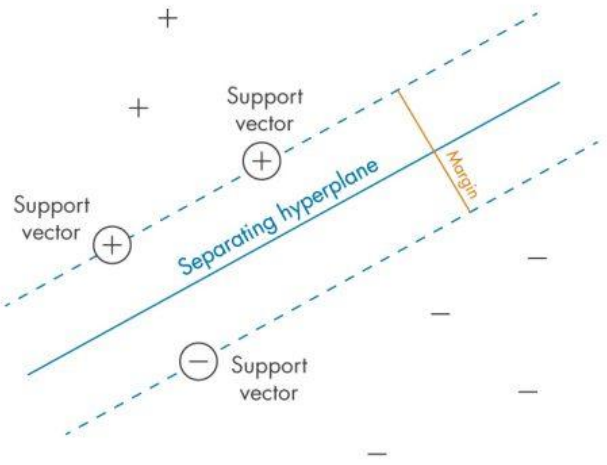
En el siguiente documento podremos apreciar las diferencias destacables que existen al comparar diferentes algoritmos con base en una serie de pruebas hechas sobre un data frame específico y una determinada cantidad de repeticiones. Obteniendo de esta forma datos como la precisión del algoritmo entre otros datos interesantes.

Marco Teórico

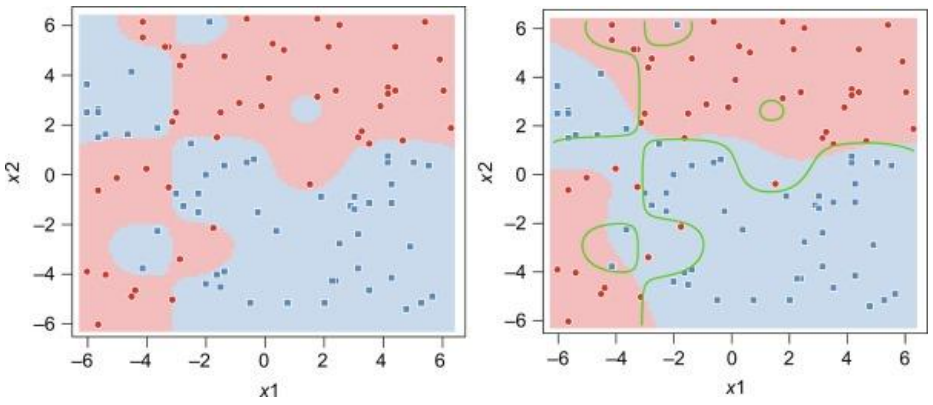
SVM

Las máquinas de vectores de apoyo (SVM) son poderosas herramientas de aprendizaje de máquinas para la clasificación y predicción de datos (Vapnik, 1995). El problema de la separación de dos clases se resuelve utilizando un hiperplano que maximiza el margen entre las clases. Los puntos de datos que se encuentran en los márgenes se llaman vectores de apoyo.

El algoritmo SVM busca encontrar el hiperplano que crea el mayor margen entre los puntos de formación de las dos clases.

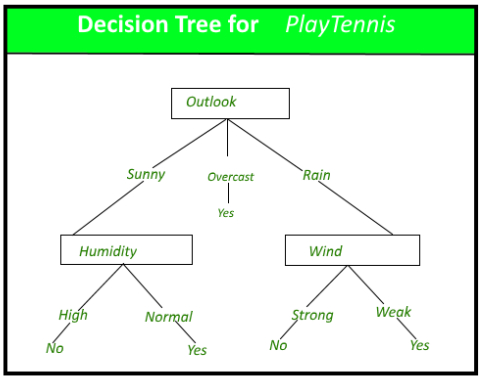


También penaliza la distancia total de los puntos que se encuentran en el lado equivocado de su margen siempre que haya superposición entre las dos clases de datos. Esto permite que se tolere un número limitado de clasificaciones erróneas cerca del margen. La otra característica clave en el SVM es el uso de las funciones del núcleo y el parámetro de penalización para convertir los límites no lineales en el espacio de los parámetros de las entradas en límites lineales en algún espacio transformado de mayor dimensión. En la siguiente figura se ilustra la representación de un problema de dos clases en un espacio bidimensional utilizando SVM. Aquí, la demarcación de los límites entre las clases roja y azul (panel izquierdo) muestra un espacio predominantemente continuo para la clase roja con bolsas azules incrustadas. [1]



Decision Tree

Un Árbol de Decisión (o Árboles de Decisiones) es un método analítico que a través de una representación esquemática de las alternativas disponible facilita la toma de mejores decisiones, especialmente cuando existen riesgos, costos, beneficios y múltiples opciones. El nombre se deriva de la apariencia del modelo parecido a un árbol y su uso es amplio en el ámbito de la toma de decisiones bajo incertidumbre (Teoría de Decisiones) junto a otras herramientas como el Análisis del Punto de Equilibrio. [2]

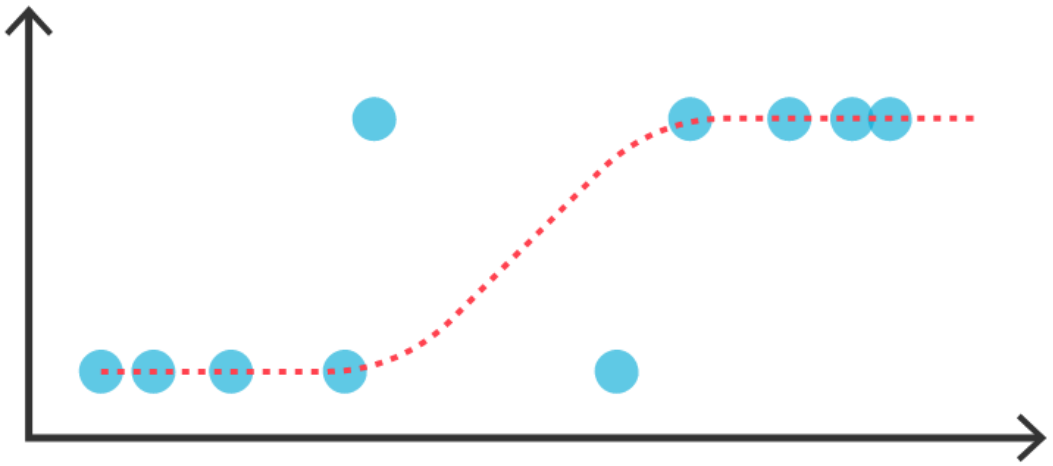


Un árbol se puede "aprender" dividiendo el conjunto de fuentes en subconjuntos según una prueba de valor de atributo. Este proceso se repite en cada subconjunto derivado de una manera recursiva denominada partición recursiva. La recursividad se completa cuando el subconjunto en un nodo tiene el mismo valor de la variable objetivo, o cuando la división ya no agrega valor a las predicciones. Los árboles de decisión pueden manejar datos de gran dimensión. En general, el clasificador de árbol de decisión tiene buena precisión. La inducción del árbol de decisiones es un enfoque inductivo típico para aprender conocimientos sobre clasificación.[3]

Logistic Regression

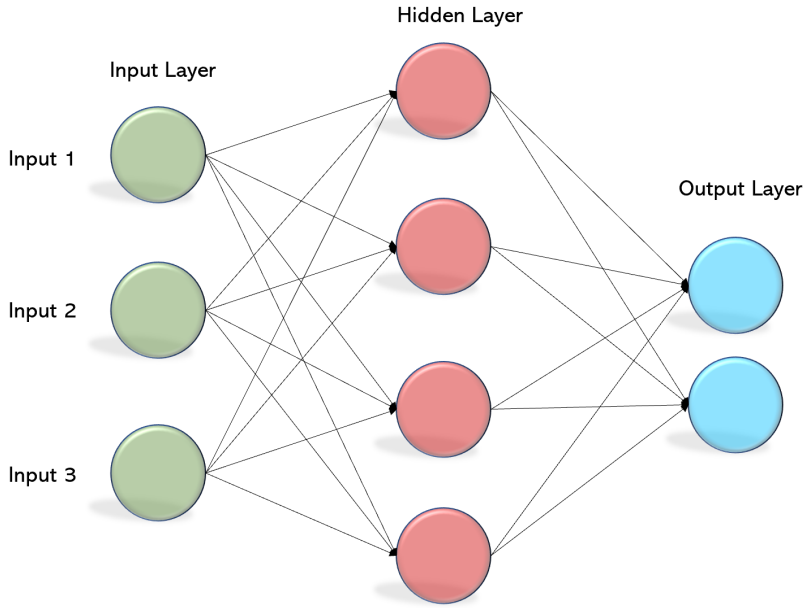
La regresión logística es un método popular para predecir una respuesta categórica. Es un caso especial de modelos lineales generalizados que predice la probabilidad de los resultados. En spark.ml, la regresión logística se puede utilizar para predecir un resultado binario mediante el uso de la regresión logística binomial, o se puede utilizar para predecir un resultado multiclase mediante el uso de la regresión

logística multinomial. Use el parámetro de familia para seleccionar entre estos dos algoritmos, o déjelo sin configurar y Spark deducirá la variante correcta.



Multilayer Perceptron

Multilayer perceptron classifier (MLPC) is a classifier based on the feedforward artificial neural network. MLPC consists of multiple layers of nodes. Each layer is fully connected to the next layer in the network. Nodes in the input layer represent the input data. All other nodes map inputs to outputs by a linear combination of the inputs with the node's weights w and bias b and applying an activation function.Implementación



Implementación

Se decidió hacer uso del lenguaje spark para este proyecto debido a su gran versatilidad a la hora de manipular grandes cantidades de datos, logrando analizarlos en un tiempo bastante bueno. Spark proporciona API para Python, Java y Scala, elegimos Scala porque es un lenguaje funcional que nos permite implementar el paradigma MapReduce de manera fácil y rápida. Scala trabaja en la JVM, lo que nos permite tener las múltiples bibliotecas creadas para Java. También se utilizo visual studio code como visualizador como editor de código ya que nos permite abrir varias instancias de terminales, Así como su integración con gitHub lo que nos permite trabajar en parejas. El Sistema operativo utilizado fue windows, el cual fue por mera comodidad, ya que no tendríamos que iniciar alguna máquina virtual con linux para poder trabajar lo cual no ahorró mucho tiempo y agiliza el proceso.

Resultados

SVM

De este algoritmo se realizaron 30 iteraciones. De las cuales podemos sacar un promedio de 88.30 % de Precisión al igual que podemos ver el Tiempo de ejecución el cual tenemos un promedio de 6.55 segundos

Grado de exactitud	**Tiempo de Ejecución** **(Segundos)**
0.883541165	6.5448933
0.885151582	6.5351829
0.885589778	6.4524035
0.881819533	6.5175121
0.88007694	6.3783402
0.883693661	6.3496224
0.885699641	6.2947765
0.880173976	6.2885279
0.88002696	6.2763336
0.885318396	6.8735895
0.885834939	6.3747923
0.882702743	8.0565045
0.880892936	6.3632332
0.885989113	6.2066037
0.881884539	6.3908401
0.886205878	6.4389906
0.882466424	6.3527104

0.87896402	6.5385412
0.881046905	6.4727403
0.885396872	6.7915596
0.882637009	6.4450255
0.883091149	6.6239326
0.882965462	6.5288968
0.878616165	6.5316187
0.888643881	6.7322588
0.884210526	6.4527874
0.880966455	6.9409944
0.88282961	6.7817002
0.879734709	6.8601687
0.885151582	6.5351829
0.883693661	6.3783402

Decision Tree

Del siguiente algoritmo se hicieron 30 pruebas(de las cuales solo mostraremos las 20 primeras), obteniendo un promedio de 89.3 % de exactitud. También podemos observar su Tiempo de ejecución siendo bastante bueno teniendo un promedio de 0.66 segundos

Grado de exactitud	**Tiempo de Ejecución (Segundos)**
0.8922151240528896	0.6547962
0.8937486193947427	0.6291957
0.8922633562403015	0.622835

0.8911349603057924	0.6392308
0.8921705826029315	0.6630638
0.89023221525986	0.6855751
0.8927032357022862	0.647094
0.8971441320838911	0.6320468
0.8946307221268016	0.632234
0.89313819224829	0.6661928
0.8907032440670586	0.6186124
0.8905377607992947	0.6455503
0.8946047678795483	0.6545471
0.8942886507120834	0.7117374
0.8914340428666127	0.647828
0.8938514217656692	0.7188458
0.8971082225792317	0.7214284
0.8906719425514765	0.6422044
0.8968306493124495	0.9607727

Logistic Regression

con este algoritmo logramos conseguir en general un grado de exactitud del 88.7% y un promedio de 1.155 seg

Las pruebas realizadas en este caso fueron un total de 20, las cuales arrojaban el mismo resultado exactamente. un supuesto escenario donde cambiaría sería correr el código en una segunda máquina.

Coeficientes	Intercepciones	Grado de exactitud	Tiempo de Ejecución (Segundos)
[2.6075947650098506E-5,-0.0036213550908734578,0.0019938415466320835,0.0013493218793036 4,0.04008022407390256]	-2.696757175407947	0.8871120157010977	1.1778629
[2.6075947650098506E-5,-0.0036213550908734578,0.0019938415466320835,0.0013493218793036 4,0.04008022407390256]	-2.696757175407947	0.8871120157010977	1.168612
[2.6075947650098506E-5,-0.0036213550908734578,0.0019938415466320835,0.0013493218793036 4,0.04008022407390256]	-2.696757175407947	0.8871120157010977	1.1563647
[2.6075947650098506E-5,-0.0036213550908734578,0.0019938415466320835,0.0013493218793036 4,0.04008022407390256]	-2.696757175407947	0.8871120157010977	1.1329513
[2.6075947650098506E-5,-0.0036213550908734578,0.0019938415466320835,0	-2.696757175407947	0.8871120157010977	1.2443289

.0013493218793036 4,0.0400802240739 0256]			
[2.60759476500985 06E-5,-0.003621355 0908734578,0.0019 938415466320835,0 .0013493218793036 4,0.0400802240739 0256]	-2.6967571754079 47	0.88711201570109 77	1.1600901
[2.60759476500985 06E-5,-0.003621355 0908734578,0.0019 938415466320835,0 .0013493218793036 4,0.0400802240739 0256]	-2.6967571754079 47	0.88711201570109 77	1.1668557
[2.60759476500985 06E-5,-0.003621355 0908734578,0.0019 938415466320835,0 .0013493218793036 4,0.0400802240739 0256]	-2.6967571754079 47	0.88711201570109 77	1.1108608
[2.60759476500985 06E-5,-0.003621355 0908734578,0.0019 938415466320835,0 .0013493218793036 4,0.0400802240739 0256]	-2.6967571754079 47	0.88711201570109 77	1.1076868
[2.60759476500985 06E-5,-0.003621355 0908734578,0.0019 938415466320835,0 .0013493218793036 4,0.0400802240739 0256]	-2.6967571754079 47	0.88711201570109 77	1.1355112
[2.60759476500985 06E-5,-0.003621355 0908734578,0.0019 938415466320835,0 .0013493218793036 4,0.0400802240739	-2.6967571754079 47	0.88711201570109 77	1.1311578

0256]			
[2.60759476500985 06E-5,-0.003621355 0908734578,0.0019 938415466320835,0 .0013493218793036 4,0.0400802240739 0256]	-2.6967571754079 47	0.88711201570109 77	1.1619209
[2.60759476500985 06E-5,-0.003621355 0908734578,0.0019 938415466320835,0 .0013493218793036 4,0.0400802240739 0256]	-2.6967571754079 47	0.88711201570109 77	1.1540226
[2.60759476500985 06E-5,-0.003621355 0908734578,0.0019 938415466320835,0 .0013493218793036 4,0.0400802240739 0256]	-2.6967571754079 47	0.88711201570109 77	1.1418843
[2.60759476500985 06E-5,-0.003621355 0908734578,0.0019 938415466320835,0 .0013493218793036 4,0.0400802240739 0256]	-2.6967571754079 47	0.88711201570109 77	1.1344772
[2.60759476500985 06E-5,-0.003621355 0908734578,0.0019 938415466320835,0 .0013493218793036 4,0.0400802240739 0256]	-2.6967571754079 47	0.88711201570109 77	1.1464259
[2.60759476500985 06E-5,-0.003621355 0908734578,0.0019 938415466320835,0 .0013493218793036 4,0.0400802240739 0256]	-2.6967571754079 47	0.88711201570109 77	1.1707406

Multilayer Perceptron

De la misma manera que en el anterior algoritmo se llevaron a cabo 30 pruebas del código en scala, lamentablemente la comparación resultó tener nula variación entre resultados dando como resultado en todo caso la precisión de 88.2% y al igual que el caso anterior se prevé que el resultado cambie según la máquina donde se ejecute el código por no más de 1 o 2%.

Test set accuracy	**Tiempo de Ejecución (Segundos)**
0.8828888399088972	7.40998291

Conclusiones

Los diferentes algoritmos que revisamos tendían a variar muy poco, esto pudo ser por diferentes motivos tales como:el sistema en el que se probó, la cantidad de datos seleccionados, los tipos de datos seleccionados, incluso la capacidad del algoritmo generado en base a nuestros conocimientos.

Como un conocimiento general podemos concluir que estos algoritmos trabajan de manera estable aún en presencia de una cantidad muy grande de datos. Y en particular el lenguaje elegido para esto hace que todo sea llevado de una manera más ligera y cómoda para nuestra máquina, en este caso Scala. Al Final podemos concluimos que el algoritmo de Decisión Tree es el más eficiente ya que presenta un promedio de velocidad de 0.66 segundos junto con una precisión de 89.3 %, la cual es la más alta de entre todos los demás algoritmos.

TECNOLÓGICO
NACIONAL DE MÉXICO

TIJUANA, BAJA CALIFORNIA

Referencias

Mishra, S. (2017, 21 julio). Support Vector Machine - an overview | ScienceDirect Topics. Https://Www.Sciencedirect.Com. https://www.sciencedirect.com/topics/earth-and-planetary-sciences/support-vector-machine [1]

Tutoriales, G. (2016, 7 marzo). Árbol de Decisión (Qué es y para qué sirve). Gestión de Operaciones. https://www.gestiondeoperaciones.net/procesos/arbol-de-decision/ [2]

GeeksforGeeks. (2021, 22 junio). Decision Tree. https://www.geeksforgeeks.org/decision-tree/ [3]

Connelly, L. (2020). Logistic regression. *Medsurg Nursing*, *29*(5), 353-354. https://www.proquest.com/openview/e8e7564d6f02ac54d757f3b74422f0ef/1?pq-origsite=gscholar&cbl=30764

Logistic Regression https://www.statisticssolutions.com/free-resources/directory-of-statistical-analyses/what-is-logistic-regression/#:~:text=Logistic%20regression%20is%20the%20appropriate,variable%20is%20dichotomous%20(binary).&text=Logistic%20regression%20is%20used%20to,or%20ratio%2Dlevel%20independent%20variables.

sciencedirect 2021 Elsevier B.V Multilayer Perceptron https://www.sciencedirect.com/topics/computer-science/multilayer-perceptrons