INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

UNIDAD DE APRENDIZAJE: MINERÍA DE DATOS
Semestre Escolar: Septiembre 2020–Enero 2021
Grupo: 3CV2

Prof. Dra Fabiola Ocampo Botello

Proyecto número 3 Árboles

Integrantes:
Alcántara Luna Diego Alexis
Torres Barajas Bryan Oswaldo
Islas Diez de Sollano Brandon Jovani

INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

UNIDAD DE APRENDIZAJE: MINERÍA DE DATOS
Semestre Escolar: Septiembre 2020–Enero 2021
Grupo: 3CV2

Prof. Dra Fabiola Ocampo Botello

Proyecto número 1 de Árboles

Árbol de decisión ID3

La base de datos con la que se trabajó se obtuvo de la página del Gobierno de México, proporcionada por la secretaría de salud, donde se proporcionan datos abiertos con información referente a casos COVID-19 en México, publicada el 14 de Abril del año 2020 la cual se puede encontrar en la siguiente dirección.

- Fuente:
  https://datos.gob.mx/busca/dataset/informacion-referente-a-casos-covid-19-en-mexico

Clasificación

Han, Kamber & Pei (2012) establecen que la clasificación es una forma de analizar datos para generar modelos que describen importantes clases de datos. Estos modelos se llaman clasificadores, permiten predecir etiquetas de clases categóricas (discretas, desordenadas).

La clasificación tiene numerosas aplicaciones, incluida la detección de fraudes, el marketing de objetivos, la predicción del rendimiento, la fabricación y el diagnóstico médico (Han, Kamber & Pei, 2012).

La clasificación de datos es un proceso de dos pasos (Han, Kamber & Pei, 2012):

1) El primer paso, de aprendizaje, es construir un modelo de clasificación.

> Un algoritmo de clasificación construye el clasificador analizando o "aprendiendo de" un conjunto de entrenamiento compuesto por tuplas de base de datos y sus etiquetas de clase asociadas.
>
> Una tupla, X, está representada por un vector de atributos de n dimensiones, X = {x1, x2, …, xn}, que representa las n mediciones realizadas en la tupla que contiene “n” atributos de la base de datos, respectivamente, A1, A2, …, An.
> Este primer paso del proceso de clasificación también puede verse como el aprendizaje de un mapeo o función, y = f(X) que puede predecir la etiqueta de clase asociada y de una tupla X dada.

En esta vista, se desea aprender un mapeo o función que separa las clases de datos.

Normalmente, este mapeo se representa en forma de reglas de clasificación, árboles de decisión o fórmulas matemáticas.

2) Un segundo paso, de clasificación, en el cual el modelo se usa para predecir etiquetas de clase para otros datos.

En el segundo paso se considera:

- El modelo se utiliza para la clasificación.

- Primero, se estima la precisión predictiva del clasificador.

- Si se usara el conjunto de entrenamiento para medir la precisión del clasificador, esta estimación probablemente sería optimista.

- Por lo tanto, se utiliza un conjunto de prueba, formado por tuplas de prueba y sus etiquetas de clase asociadas. Son independientes de las tuplas de entrenamiento, lo que significa que no se utilizaron para construir el clasificador.

- La precisión de un clasificador en un conjunto de prueba dado es el porcentaje de tuplas de conjuntos de prueba que el clasificador clasifica correctamente.

- La etiqueta de clase asociada de cada tupla de prueba se compara con la predicción de clase del clasificador aprendido para esa tupla.

- Si la precisión del clasificador se considera aceptable, el clasificador se puede utilizar para clasificar tuplas de datos futuras para las que no se conoce la etiqueta de clase. (Estos datos también se denominan en la literatura sobre aprendizaje automático (machine learning) como datos "desconocidos" o "no vistos anteriormente").

Árboles de decisión

Han, Kamber & Pei (2012) establecen que Un árbol de decisión es una estructura de árbol similar a un diagrama de flujo, donde cada nodo interno (nodo no hoja) denota una prueba en un atributo, cada rama representa un resultado de la prueba y cada nodo hoja (o nodo terminal) tiene una etiqueta de clase.

Los nodos internos se indican con rectángulos y los nodos de hoja se indican con óvalos. Algunos algoritmos de árboles de decisión producen solo árboles binarios (donde cada nodo interno se ramifica exactamente a otros dos nodos), mientras que otros pueden producir árboles no binarios.

Un algoritmo de inducción, o más concretamente un inductor (también conocido como aprendiz), es una entidad que obtiene un conjunto de entrenamiento y forma un modelo que generaliza la relación entre los atributos de entrada y el atributo objetivo.

Los inductores de árboles de decisión son algoritmos que construyen automáticamente un árbol de decisiones a partir de un conjunto de datos determinado (Rokach, L. & Maimon, O., 2015).

La exactitud de la clasificación se expresa como uno menos el error de generalización.

El error de entrenamiento es definido como el porcentaje de ejemplos en el conjunto de entrenamiento que fueron correctamente clasificados en el árbol de clasificación

Matriz de confusión

La evaluación del desempeño de un modelo de clasificación considera dos aspectos:

- La cantidad de registros previstos por el modelo de forma adecuada
- La cantidad de registros previstos por el modelo de forma inadecuada

Lo anterior se presenta en una matriz de confusión.

Contiene el número de elementos que se han clasificado correcta o incorrectamente para cada clase. Un beneficio de una matriz de confusión es que es fácil ver si el sistema confunde dos clases (es decir, etiquetar comúnmente una como otra).

Table 4.1 A confusion matrix.

	Predicted negative	Predicted positive
Negative Examples	A	B
Positive Examples	C	D

- Accuracy is: $(a+d)/(a+b+c+d)$
- Misclassification rate is: $(b+c)/(a+b+c+d)$
- Precision is: $d/(b+d)$
- True positive rate (Recall) is: $d/(c+d)$
- False positive rate is: $b/(a+b)$
- True negative rate (Specificity) is: $a/(a+b)$
- False negative rate is: $c/(c+d)$

Sobreajuste y Subajuste

Rokach, L. & Maimon, O. (2015) mencionan que se refiere a la situación en la que el algoritmo de inducción genera un clasificador que se ajusta perfectamente a los datos de entrenamiento pero ha perdido la capacidad de generalizar a instancias no presentadas durante el entrenamiento. En otras palabras, en lugar de aprender, el clasificador simplemente memoriza las instancias de entrenamiento. El sobreajuste se reconoce generalmente como una violación del principio de la navaja de Occam.

Árbol de decisión ID3

Evidentemente, la construcción del árbol de decisión no es única, y si aplicamos una estrategia u otra a la hora de decidir en qué orden se hacen las preguntas sobre los atributos podemos

encontrar árboles muy dispares. De entre todos los posibles árboles, estamos interesados en encontrar aquellos que cumplan las mejores características como máquinas de predicción, e intentaremos dar un mecanismo automático de construcción del árbol a partir de los ejemplos.
En el año 1979, J. Ross Quinlan presenta un método para construir árboles de decisión que presentan muy buenas características, como son un buen balanceado y un tamaño pequeño (el menor número de preguntas posibles para poder encontrar la respuesta en todos los casos, si es posible). Para ello, hace uso de la Teoría de la Información desarrollada por C. Shannon en 1948. Aunque el mérito se le asocia a Quinlan, la verdad es que de forma paralela aparecieron varios métodos de construcción que eran prácticamente equivalentes. El algoritmo de construcción que presenta se llama ID3, y hasta el momento ha sido el más utilizado y en el que se han basado la mayoría de las mejoras posteriores introducidas por el mismo Quinlan y por otros autores.
El ID3 construye un árbol de decisión de arriba a abajo, de forma directa, sin hacer uso de backtracking, y basándose únicamente en los ejemplos iniciales proporcionados. Para ello, usa el concepto de Ganancia de Información para seleccionar el atributo más útil en cada paso. En cierta forma, sigue un método voraz para decidir la pregunta que mayor ganancia da en cada paso, es decir, aquella que permite separar mejor los ejemplos respecto a la clasificación final.

- Es un algoritmo desarrollado por Ross Quinlan

- ID3 se basa en la navaja de afeitar de Occam.

- Se prefieren los árboles de decisión pequeños.

- Sólo acepta atributos categóricos

- Para construir un árbol de decisión, la ganancia de información se calcula para cada atributo y el atributo con la mayor ganancia de información se convierte en el nodo raíz.

Objetivo

Dentro de la realización del árbol de decisión ID3 tomaremos como columna objetivo de predicción la columna intubado, esto con la finalidad de poder predecir si, por ejemplo, si una persona padece obesidad será intubado o si padece Asma que probabilidad hay de que esta persona se entubada.

Diccionario de datos

Nombre	Tipo de Dato	Dominio
Intubado	Entero	1- Sí 2- No 97 - No aplica 98 - Se ignora 99 - No se especifica
Neumonia	Entero	1- Sí 2- No 97 - No aplica 98 - Se ignora 99 - No se especifica
Diabetes	Entero	1- Sí 2- No 97 - No aplica 98 - Se ignora 99 - No se especifica
EPOC	Entero	1- Sí 2- No 97 - No aplica 98 - Se ignora 99 - No se especifica
Asma	Entero	1- Sí 2- No 97 - No aplica 98 - Se ignora 99 - No se especifica
Obesidad	Entero	1- Sí 2- No 97 - No aplica 98 - Se ignora 99 - No se especifica
Tabaquismo	Entero	1- Sí 2- No 97 - No aplica 98 - Se ignora 99 - No se especifica

Características de la Base de datos

Nuestra base de datos utilizada para este árbol, consta de información sobre pacientes que han contraído el virus COVID-19, tal como la fecha de los síntomas, la fecha del ingreso, características sobre su salud, si tiene Asma, si fuma o si padece de enfermedades relacionadas con el sistema cardiorespiratorio

Tratamiento de los datos

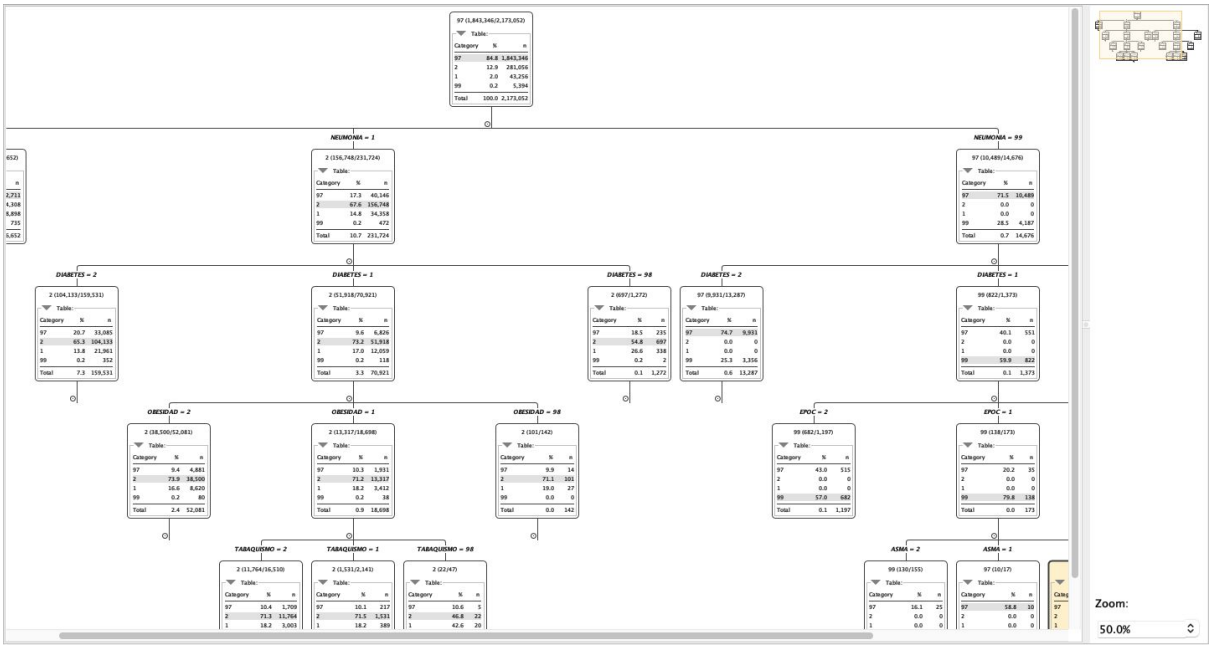
Una vez filtradas nuestras columnas, procedemos a buscar los datos faltantes y en este caso, eliminaremos las filas que tengan esta característica, pues al ser información muy sensible, si llegamos a insertar algún valor, podría darnos un resultado muy alejado de la realidad.

Posteriormente, como nuestros datos son de tipo entero (int) procedemos a convertirlos a tipo string, esto con la finalidad de que la herramienta knime pueda trabajar con los datos.

Después de realizar un column rename procedemos a particionar los datos, en este caso se utiliza el 80% de los datos para el entrenamiento del árbol y el 20% para prueba del mismo.

Posteriormente esta información se envía a nuestros nodos “Decision tree learner” y “Decision tree predictor”, obteniendo los siguientes resultados.

Árbol



Medidas

Table "default" – Rows: 8 | Spec - Columns: 11 | Properties | Flow Variables

Row ID	TruePositives	FalsePositives	TrueNegatives	FalseNegatives	Recall	Precision	Sensitivity	Specificity	F-measure	Accuracy	Cohen's kappa
2	52722	5205	41108	444228	0.106	0.91	0.106	0.888	0.19	?	?
1	0	6	498603	44654	0	0	0	1	NaN	?	?
98	0	0	541604	1659	0	?	0	1	?	?	?
97	0	484934	58329	0	?	0	?	0.107	?	?	?
99	0	396	542867	0	?	0	?	0.999	?	?	?
Overall	?	?	?	?	?	?	?	?	?	0.097	-0.001

INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

UNIDAD DE APRENDIZAJE: MINERÍA DE DATOS
Semestre Escolar: Septiembre 2020–Enero 2021
Grupo: 3CV2

Prof. Dra Fabiola Ocampo Botello

Proyecto número 1 de Árboles

Árbol de decisión C4.5 - con Weka.

La base de datos utilizada se obtuvo del sitio Kaggle la cual cuenta con datos nominales sobre las regiones y estados de Brasil, así como el número de casos y muertes en cada una de estas desde febrero del 2020 hasta el 17 de noviembre del 2020.

- Fuente:https://www.kaggle.com/unanimad/corona-virus-brazil

Introducción.

Esta base de datos cuenta con distintos atributos tanto nominales como ordinales comunes como la región del país, la fecha, el estado, número de casos y número de muertes.

Para realizar este estudio se consideraron casi todos los atributos ya que son pocos y nos funciona bien para poder realizar nuestro árbol c4.5, solo se eliminó la fecha ya que arrojaba demasiados datos y no era posible visualizar el árbol de forma correcta, y que de cualquier modo no nos interesa ese atributo para el fin del estudio realizado.

En este caso, la base de datos se encontraba poblada en su totalidad y sin discrepancias o errores a corregir, por lo que la limpieza de datos fue omitida.

Dentro del software de Weka podemos seleccionar el preprocesamiento que realizaremos en nuestra base de datos antes de clasificarlos y de crear el árbol c4.5, en donde el único paso realizado fue la eliminación de la columna de fecha para poder trabajar mejor con los atributos restantes.

Ahora, para la realización de un árbol C4.5 necesitamos seleccionar un atributo nominal como columna objetivo para que a partir de esto pueda realizar el árbol y su clasificación, en este caso tomamos la columna de “Estado” ya que se siguió la estrategia de elegir el atributo más común.

Como resultado obtuvimos un árbol que clasifica a todos los estados dependiendo de su región y más a parte depende del número de casos y el número de muertes. Obteniendo asi una representación fácil a la vista, en la cual podemos observar que estados son los estados con menos casos y con menos muertes en el país de Brasil.

Diccionario de datos.

Nombre	Tipo	Descripción	Dominio
date	Nominal	Fecha en la que se registra el número de casos y muertes por Covid-19 en Brasil.	0000-00-00 - 9999-99-99
region	Nominal	Indica la región en la que se encuentra el estado.	1- Centro - Oeste 2- Noreste 3- Norte 4- Sureste 5- Sur
state	Nominal	Indica el estado del país Brasil.	1- DF 2- GO 3- MS 4- MT 5- AL 6- BA 7- CE 8- MA 9- PB 10- PE 11- PI 12- RN 13- SE 14- AC 15- AM 16- AP 17- PA 18- RO 19- RR 20- TO 21- ES 22- MG

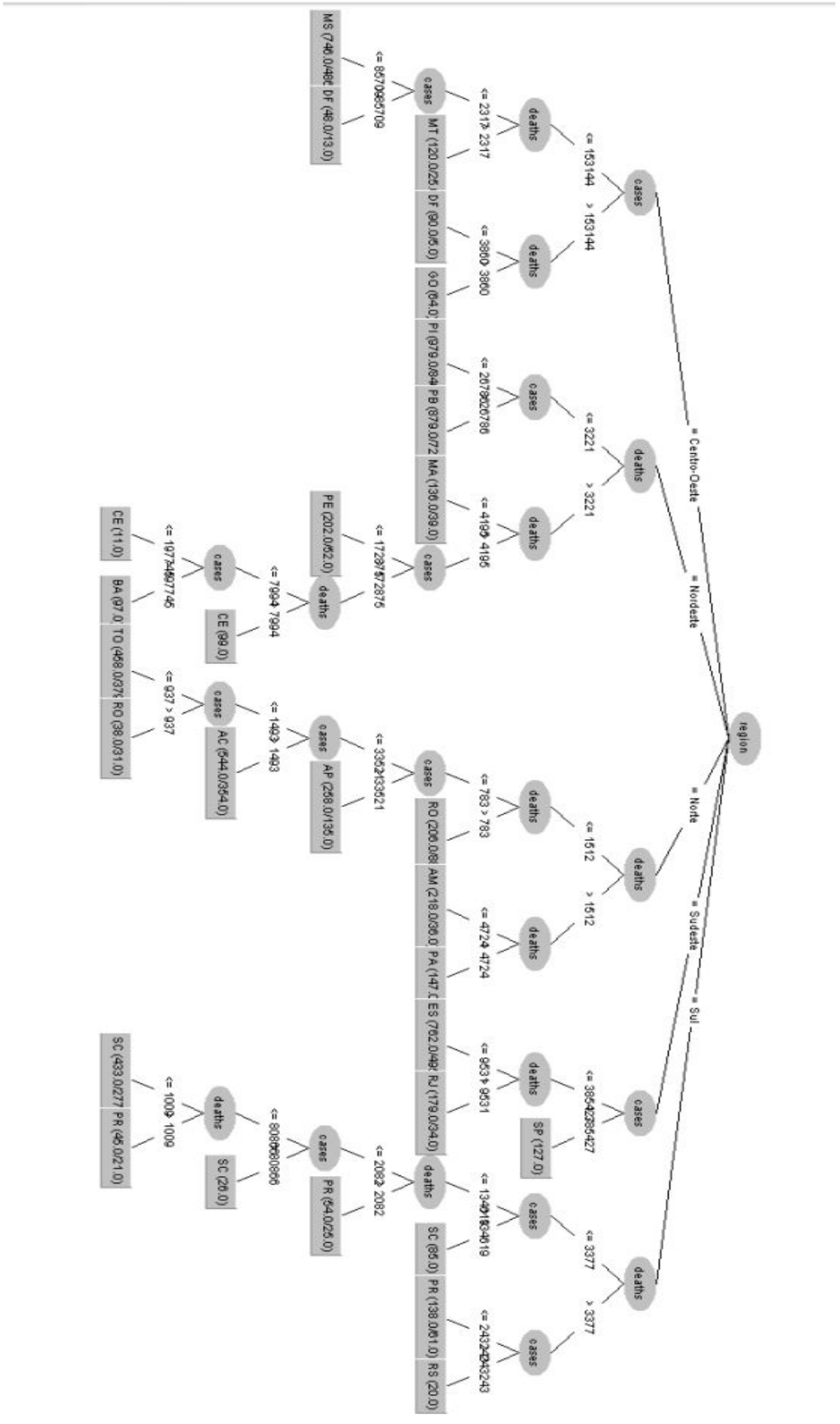
			23- RJ 24- SP 25- PR 26- RS 27- SC
cases	Numérico	Indica el número de casos activos de Covid-19 en un estado de Brasil.	0-N
deaths	Numérico	Indica el número de muertes causadas por Covid-19 en un estado de Brasil.	0-N

Objetivos.

El objetivo principal de este análisis, es el encontrar una clasificación sobre nuestra columna objetivo (state), es decir sobre los estados del país Brasil, dependiendo de la región (region) en la que se encuentran, del número de casos activos (cases) de Covid-19 y del número de muertes causadas por Covid-19 dentro del estado (deaths).

El algoritmos C4.5 genera una decisión a partir de atributos discretos y continuos, mediante particiones realizadas de forma recursiva. Este árbol se construye mediante la estrategia Depth-First, o Profundidad-Primero.
El algoritmo considera todas las pruebas posibles que puedan dividir al conjunto de datos y selecciona la prueba que resulta en la mayor ganancia de información. Para cada atributo discreto, se considera una prueba con n resultados, siendo n el número de valores posibles que puede tomar el atributo. Para cada atributo continuo, se realiza una prueba binaria sobre cada uno de los valores que toma el atributo en los datos. En cada nodo, el sistema debe decidir cuál prueba escoge para dividir los datos.

Árbol realizado con Weka.



Medidas.

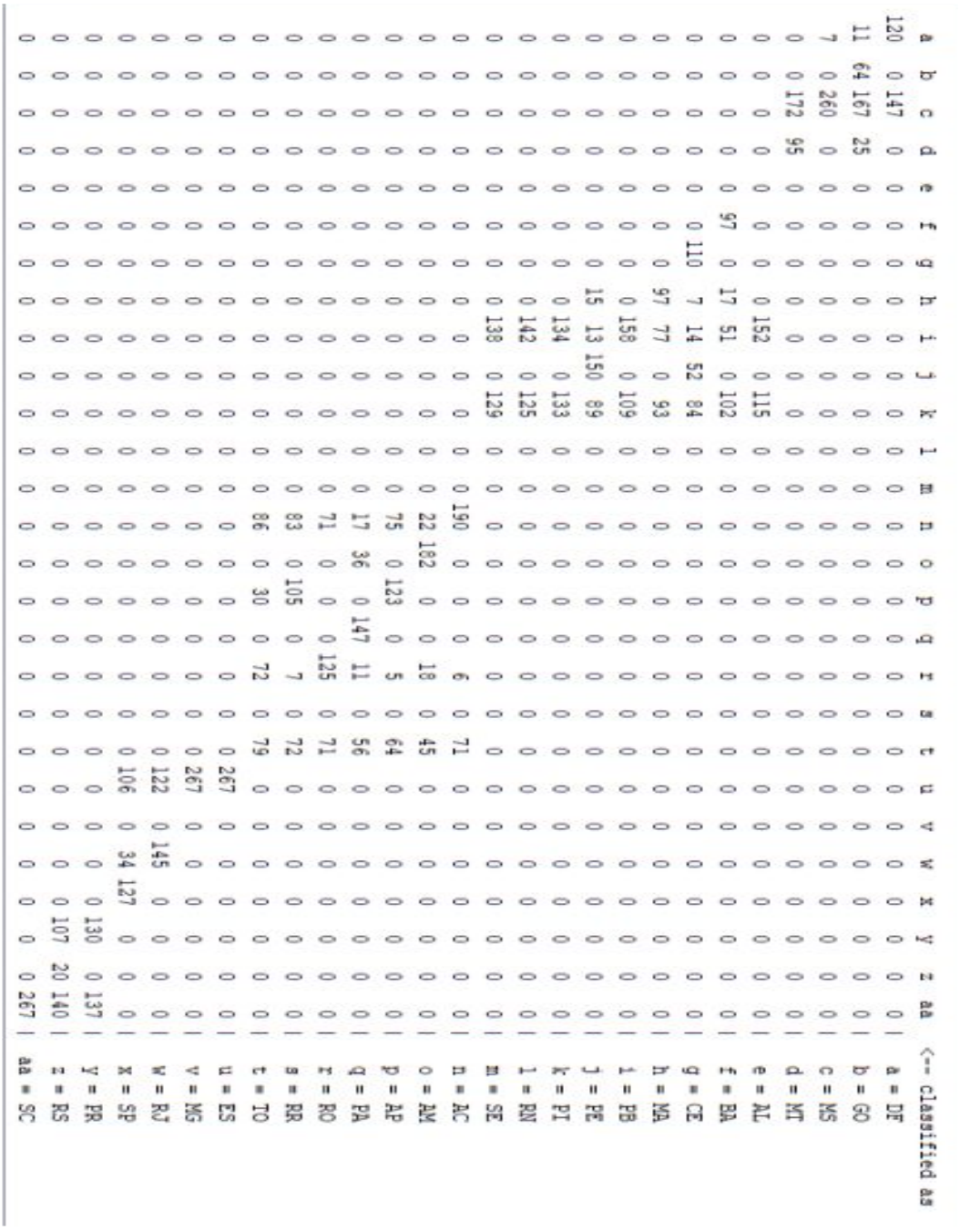
=== Summary ===

Correctly Classified Instances	3086	42.8076 %
Incorrectly Classified Instances	4123	57.1924 %
Kappa statistic	0.4061	
Mean absolute error	0.0468	
Root mean squared error	0.153	
Relative absolute error	65.667 %	
Root relative squared error	81.0352 %	
Total Number of Instances	7209	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0.449	0.003	0.870	0.449	0.593	0.616	0.974	0.581	DF
	0.240	0.000	1.000	0.240	0.387	0.483	0.961	0.455	GO
	0.974	0.070	0.349	0.974	0.513	0.560	0.964	0.348	MS
	0.356	0.004	0.792	0.356	0.491	0.520	0.970	0.480	MT
	0.000	0.000	?	0.000	?	?	0.898	0.160	AL
	0.363	0.000	1.000	0.363	0.533	0.595	0.930	0.487	BA
	0.412	0.000	1.000	0.412	0.584	0.635	0.954	0.584	CE
	0.363	0.006	0.713	0.363	0.481	0.496	0.920	0.357	MA
	0.592	0.104	0.180	0.592	0.276	0.282	0.901	0.165	PB
	0.562	0.007	0.743	0.562	0.640	0.634	0.957	0.515	PE
	0.498	0.122	0.136	0.498	0.213	0.207	0.889	0.148	PI
	0.000	0.000	?	0.000	?	?	0.893	0.153	RN
	0.000	0.000	?	0.000	?	?	0.891	0.151	SE
	0.712	0.051	0.349	0.712	0.469	0.472	0.958	0.324	AC
	0.682	0.005	0.835	0.682	0.751	0.746	0.976	0.666	AM
	0.461	0.019	0.477	0.461	0.469	0.449	0.950	0.344	AP
	0.551	0.000	1.000	0.551	0.710	0.736	0.973	0.700	PA
	0.468	0.017	0.512	0.468	0.489	0.471	0.956	0.398	RO
	0.000	0.000	?	0.000	?	?	0.945	0.300	RR
	0.296	0.055	0.172	0.296	0.218	0.187	0.929	0.235	TO
	1.000	0.071	0.350	1.000	0.519	0.570	0.964	0.350	ES
	0.000	0.000	?	0.000	?	?	0.964	0.350	MG
	0.543	0.005	0.810	0.543	0.650	0.653	0.975	0.570	RJ
	0.476	0.000	1.000	0.476	0.645	0.683	0.972	0.642	SP
	0.487	0.015	0.549	0.487	0.516	0.499	0.977	0.475	PR
	0.075	0.000	1.000	0.075	0.139	0.269	0.975	0.492	RS
	1.000	0.040	0.491	1.000	0.658	0.686	0.988	0.702	SC
Weighted Avg.	0.428	0.022	?	0.428	?	?	0.948	0.412	

Matriz de confusión.



```
   a   b   c   d   e   f   g   h   i   j   k   l   m   n   o   p   q   r   s   t   u   v   w   x   y   z  aa  <-- classified as
 120   0 147   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |  a = DF
  11  64 167  25   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |  b = GO
   7   0 260   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |  c = MS
   0   0 172  95   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |  d = MT
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |  e = AL
   0   0   0   0   0  97   0  17  51   0 102   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |  f = BA
   0   0   0   0   0   0 110   7  14  52  84   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |  g = CE
   0   0   0   0   0   0   0  97  77   0  93   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |  h = MA
   0   0   0   0   0   0   0   0 158   0 109   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |  i = PB
   0   0   0   0   0   0   0  15  13 150  89   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |  j = PE
   0   0   0   0   0   0   0   0 134   0 133   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |  k = PI
   0   0   0   0   0   0   0   0   0 142   0 125   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |  l = RN
   0   0   0   0   0   0   0   0 138   0 129   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |  m = SE
   0   0   0   0   0   0   0   0   0   0   0   0 190   0   0   0   6   0  71   0   0   0   0   0   0   0   0 |  n = AC
   0   0   0   0   0   0   0   0   0   0   0   0  22 102   0   0  18   0  45   0   0   0   0   0   0   0   0 |  o = AM
   0   0   0   0   0   0   0   0   0   0   0   0  75   0 123   0   5   0  64   0   0   0   0   0   0   0   0 |  p = AP
   0   0   0   0   0   0   0   0   0   0   0   0  17  36   0 147  11   0  56   0   0   0   0   0   0   0   0 |  q = PA
   0   0   0   0   0   0   0   0   0   0   0   0  71   0   0   0 125   0  71   0   0   0   0   0   0   0   0 |  r = RO
   0   0   0   0   0   0   0   0   0   0   0   0  83   0 105   0   7   0  72   0   0   0   0   0   0   0   0 |  s = RR
   0   0   0   0   0   0   0   0   0   0   0   0  86   0  30   0  72   0  79   0   0   0   0   0   0   0   0 |  t = TO
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 267   0   0   0   0   0   0   0 |  u = ES
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 267   0   0   0   0   0   0 |  v = MG
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 145   0   0   0   0   0 |  w = RJ
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 106   0  34 127   0   0   0   0 |  x = SP
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |  y = PR
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |  z = RS
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 | aa = SC
```

Resultados.

Obtuvimos datos muy dispersos en cuanto a la precisión, ya que al tener demasiadas clases evaluadas era complicado clasificarlas dentro de los casos positivos de manera completamente correcta, es por esto que obtenemos un árbol con un error de 57%.

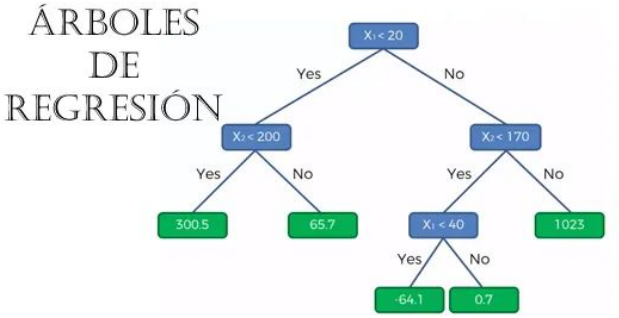
INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

UNIDAD DE APRENDIZAJE: MINERÍA DE DATOS
Semestre Escolar: Septiembre 2020–Enero 2021
Grupo: 3CV2

Prof. Dra Fabiola Ocampo Botello

Proyecto número 1 de Árboles

Árbol de regresión CART Herramienta Python



El repositorio de base de datos se obtuvo del sitio Kaggle donde se tiene una relación por país de una serie de datos cuantitativos en el cual se llevaba una bitácora de los casos que iban surgiendo, como se estaban tratando y cuántas muertes, entre otros datos más este registro se realizó de enero del 2020 hasta agosto del 2020 por lo que los datos presentados no son actualizados hasta la fecha.

- Fuente:https://www.kaggle.com/imdevskp/corona-virus-report?select=worldometer_data.csv

Introducción.

En este dataset se contaba con una serie de atributos muy interesantes, entre ellos estaban; Total de muertes, Total de casos activos, Número de casos críticos o serios, entonces se planteó el problema de realizar un árbol CART ya que sus características nos podían ayudar a predecir o realizar una pequeña clasificación de las variables que se mencionan, se tomaron en concreto las columnas:

- Serious,Critical (variable objetivo o variable dependiente)
- ActiveCases
- TotalRecovered
- TotalDeaths
- TotalCases

Como sabemos una regresión nos permite predecir un dato con base en la probabilidad que se den ciertas variables para que ocurra ese dato que estamos esperando.
Se tiene un conjunto de variables independientes y una dependiente, en este caso las variables independientes son (ActiveCases, TotalRecovered, TotalDeaths, TotalCases) y la variable dependiente es Serious,Critical.
Antes de comenzar con el análisis se seleccionaron las columnas objetivo del dataset y luego se realizó una limpieza de datos la cual se muestra en la figura 3.1.

```
In [68]: predataframe2=predataframe1.iloc [:, [2,3,4,5,6,7,8,9,10,11]]

In [14]: predataframe2

Out[14]:
```

	Population	TotalCases	TotalDeaths	TotalRecovered	ActiveCases	Serious,Critical	Tot Cases/1M pop	Deaths/1M pop	TotalTests
0	3.311981e+08	5032179	162804.0	2576668	2292707	18296.0	15194.0	492.0	63139605
1	2.127107e+08	2917562	98644.0	2047660	771258	8318.0	13716.0	464.0	13206188
2	1.381345e+09	2025409	41638.0	1377384	606387	8944.0	1466.0	30.0	22149351
3	1.459409e+08	871894	14606.0	676357	180931	2300.0	5974.0	100.0	29716907
4	5.938157e+07	538184	9604.0	387316	141264	539.0	9063.0	162.0	3149807
...	...	...	...	...	...	...	...	...	...
204	4.992000e+03	13	1.0	10	2	0.0	2604.0	200.0	61
205	2.624700e+04	13	0.0	7	6	0.0	495.0	0.0	424
206	3.489000e+03	13	0.0	13	0	0.0	3726.0	0.0	1816
207	8.010000e+02	12	0.0	12	0	0.0	14981.0	0.0	12
208	5.986820e+05	10	1.0	8	1	0.0	17.0	2.0	8

Figura 3.1 Selección de Columnas del Dataset.

Limpieza de datos

La limpieza fue un tema difícil de tratar ya se tuvo que investigar la naturaleza de los datos faltantes y muchos cambios se hicieron en archivo csv, debido a que se debía observar campo por campo su valor faltante y no era recomendable borrar registros ya que son países importantes y era muy buena tenerlos en cuenta.

Se hizo lo siguiente:

- TotalRecovered: se realizó una resta entre el total de casos y el el total de muertes
- ActiveCases: se realizó una investigación donde aproximadamente el 25% de los casos recuperados se encuentran activos
- Seriuos/Critical: se eliminaron los datos vacíos y se hizo un promedio de los que quedaban activo
- TotalDeaths: se sustituyeron los nulos por un cero

Diccionario de datos.

La relación se muestra en la figura 3.2.

Variable	Tipo de dato	Dominio
Population	Continuo	0 - (1 x e9)
TotalCases	Continuo	0- (1 x e7)
TotalDeaths	Continuo	0- (1 x e6)

TotalRecovered	Continuo	0- (1 x e8)
ActiveCases	Numérico	0- (1 x e7)
Serious,Critical	Numérico	0- (1 x e5)
Tot Cases/1M pop	Numérico	0- (1 x e5)
Deaths/1M pop	Numérico	0- (1 x e5)
TotalTests	Numérico	0- (1 x e6)

Figura 3.1 Selección de Columnas del Dataset.

Objetivo.

Este análisis se realizó para conocer la relación que puede haber entre nuestra variable objetivo (Serious,Critical) con las variables TotalCases, TotalDeaths, TotalRecovered, ActiveCases, y dependiendo de la relación poder predecir qué ocurrirá con esa relación, por ejemplo si tomamos la relación Serious,Critical-TotalCases, se puede analizar cuántos de los casos totales fueron casos graves, en cambio la relación Serious,Critical-TotalDeaths podemos observar cuántos de esos decesos estuvieron graves, en el caso de Serious,Critical- TotalRecovered nos dice la cuántos de esos casos críticos pudieron recuperarse de la enfermedad.

El algoritmo CART es un procedimiento estadístico que se utiliza para producir modelos de clasificacion y regresion con una estructura basada en árboles, se basa en jerarquizar las decisiones de una forma binaria, es decir las ramificaciones se van creando por medio de la decisión SI una variable cumple una condición o NO cumple la condición.

El objetivo del algoritmo CART es encontrar un arbol cercano al tamaño de arbol optimo.

Árbol.

Cabe mencionar que al realizar una serie de pruebas con los árboles de regresión en Python ocurrieron dos vertientes en el aspecto de encontrar el árbol de tamaño óptimo.

Si se dejaba que el árbol corriera sin ningún tipo de restricción

en el caso de python con el parámetro “max_depth” se creaba un árbol con 100% de efectividad y se mostraba la matriz de confusión pero la imagen era ilegible ya que generaba una cantidad de nodos muy grande, por lo cual se optó por utilizar ese parámetro para crear una especie de restricción, el parámetro “max_depth” nos especifica agregar un tamaño de profundidad especificado por el programador.

Implementación del árbol de regresión

Para realizar la implementación se agregaron las siguientes librerías:

```
from sklearn.model_selection import train_test_split

from sklearn.tree import DecisionTreeRegressor
```

Posteriormente se seleccionaron las variables independientes y la variable dependiente:

```
X= predataframe2.iloc[:,1:4] ----------> Selección de variables independientes

y= predataframe2.iloc[:,5]-------------> Selección de variable dependiente
```

Luego se definen variables de entrenamiento y de pruebas tomando como relación el 80% para entrenamiento y el 20% de pruebas.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

Después se define la variable de tipo Árbol de Regresión de Decisión con el parámetro max_depth=5:

```
adr = DecisionTreeRegressor(max_depth = 5)
```

Enseguida se manda a entrenar el algoritmo:

```
adr.fit(X_train, y_train)
```

Y se obtiene la predicción del algoritmo:

```
prediccion = adr.predict(X_test)
```

A continuación en la figura 3.3 se muestra el árbol obtenido:

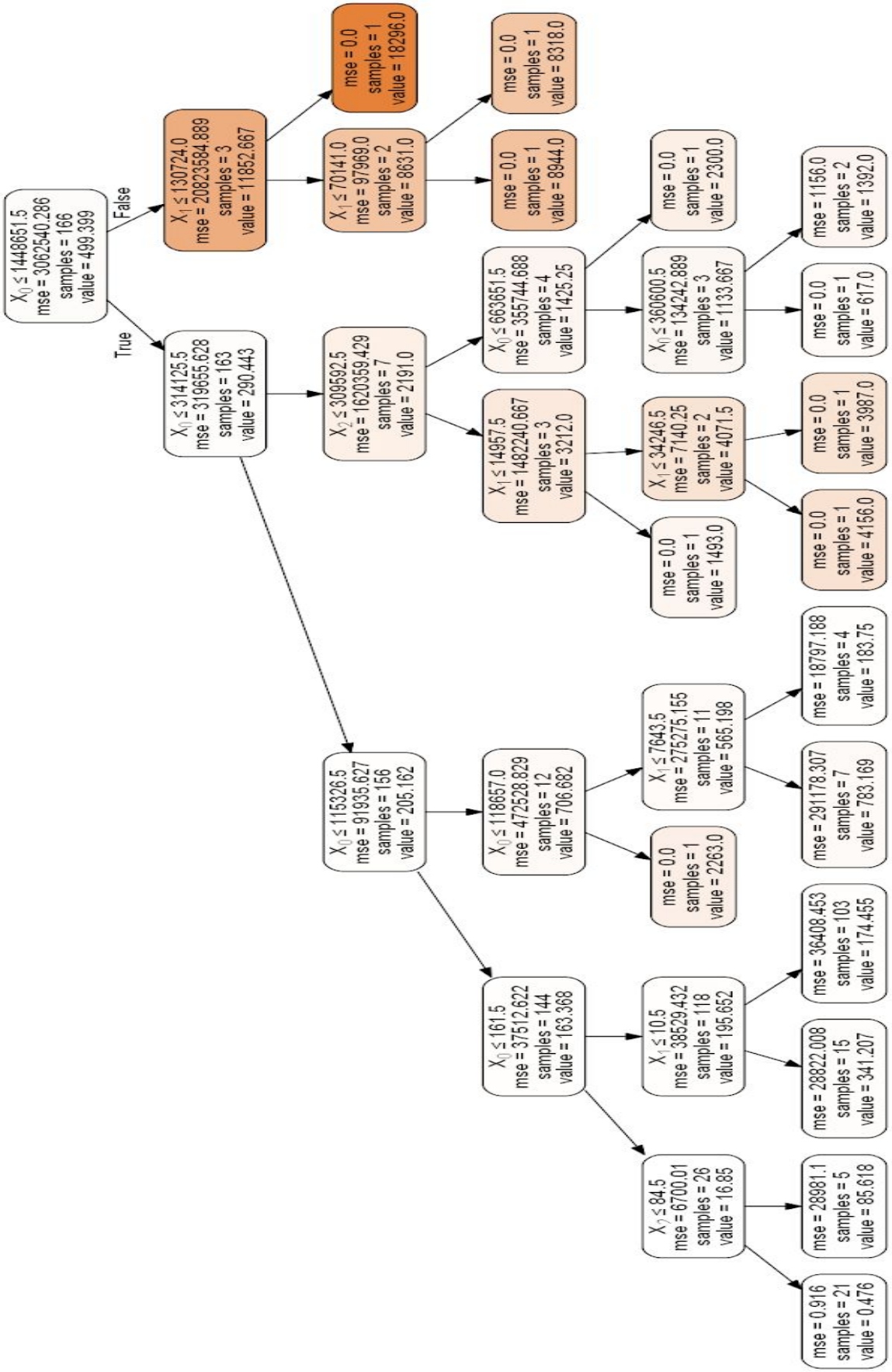


Figura 3.3 Árbol de Regresión Obtenido.

Medidas.

Como se mencionó anteriormente al mover la profundidad y por la forma que están especificadas las librerias “from sklearn.tree import DecisionTreeRegressor” y “from sklearn.metrics import classification_report, confusion_matrix” no se permite realizar matriz de confusión con arboles cuya variable sea continua. Por lo que se realizaron otro tipo de medidas como lo son :

- MAE(Mean Absolute error) - Media del valor absoluto de los errores:

  metrics.mean_absolute_error(y_test, prediccion)

  Salida= 197.02492008669702

- MSE (media de los errores al cuadrado)

  metrics.mean_squared_error(y_test, prediccion)

  Salida=57367.93780603838

- RMSE(raiz cuadrada de la media de los errores al cuadrado)

  np.sqrt(metrics.mean_squared_error(y_test, prediccion))

  Salida=239.51604916171772

- Nivel de error del modelo

  Esta escala va de 0 a 1 donde 1 es 100% de efectividad.

  print(adr.score(X_train, y_train))

  Salida=0.9873263051449528

Ahora con la ayuda de las gráfica de dispersión (Figura 3.4) podemos observar la relación entre los datos que se predijeron con los datos de prueba en donde podemos ver la respuesta del modelo con una pendiente y los puntos que se aproximan a esa tendencia, lo cual si se observa detalladamente no existe un sesgo tan marcado.

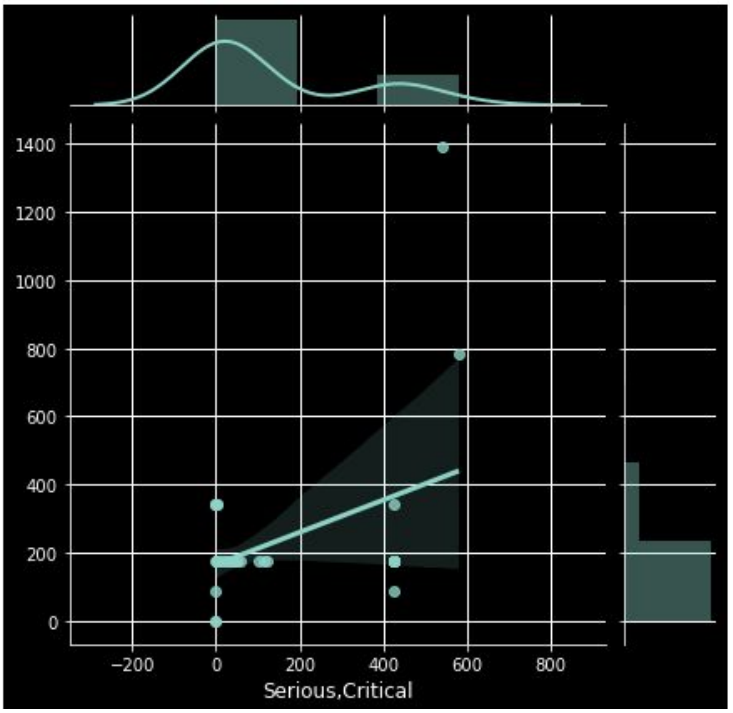


Figura 3.4 Gráfica de relación entre Variable dependiente de prueba y la predicción del modelo.