

Etapas 4 – Aprendizaje supervisado

Manuel Sebastian Quimbayo Barbosa

Ing. José Alfair Morales Barrera

ECBTI

Análisis de datos

Universidad Nacional Abierta y a Distancia

7 de noviembre de 2023

Introducción

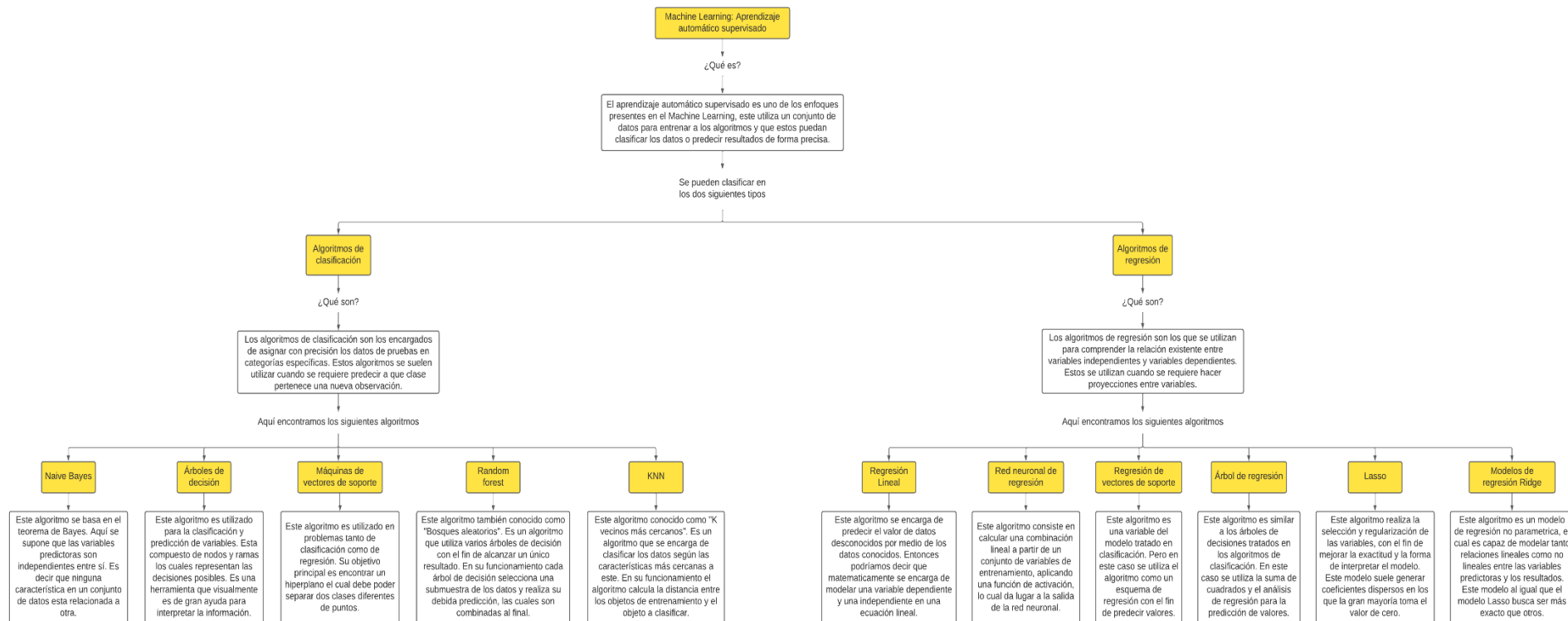
El Machine Learning es una rama de la inteligencia artificial que cada vez coge más fuerza en el mercado actual, ya que permite que las máquinas aprendan a partir de los datos que se les proporcionan, logrando de este modo herramientas que ayudan a predecir valores o que nos permiten buscar información de manera mucho más sencilla a la que estamos acostumbrados en la actualidad.

De igual forma el Machine Learning tiene diferentes enfoques, uno de ellos es el aprendizaje supervisado, aquí el objetivo es entrenar al modelo con el fin de que este pueda clasificar los datos o predecir resultados de forma precisa.

En el presente trabajo vamos a realizar un mapa conceptual con el fin de comprender mejor el enfoque de aprendizaje supervisado; además, se realizarán tres ejercicios con diferentes modelos los cuales son árboles de decisión, modelo KNN y modelo Naive Bayes, analizando los resultados obtenidos y sacando conclusiones del dataset proporcionado.

Mapa conceptual

https://lucid.app/lucidchart/89475d5b-7fa3-437d-925d-01bce25368fe/edit?viewport_loc=-2150%2C-606%2C6938%2C3156%2C0_0&invitationId=inv_daf88dee-f66f-4809-af60-749054a1f90e



Configuración del dataset para los ejercicios

Para empezar, vamos a recordar que volveremos a trabajar en la herramienta de Knime, la cual nos facilitará el análisis de la información debido a que cuenta con una interfaz gráfica amigable y fácil de usar. Ahora, lo primero que debemos realizar es subir el archivo de datos a la herramienta, para ello utilizamos la función “**CSV Reader**” y la configuramos, es decir, seleccionamos la ruta donde se encuentra el archivo que contiene los datos de los pacientes con Covid. En las siguientes imágenes podemos ver el proceso realizado:

CSV Reader



Dialog - 3:1 - CSV Reader

File

Settings Transformation Advanced Settings Limit Rows Encoding Flow Variables Job Manager Selection Memory Policy

Input location

Read from: Local File System

Mode: ☒ File ☐ Files in folder

File: C:\Users\Sebastian Quimbayo\Documents\Ingeniería de sistemas\archive\Cleaned-Data.csv Browse...

Reader options

Format

Autodetect format ⚙

Column delimiter: , Row delimiter: ☒ Line break ☐ Custom ␣␣

Quote char: " Quote escape char: \"

Comment char

☒ Has column header ☐ Has row ID

☐ Support short data rows ☐ Prepend file index to row ID

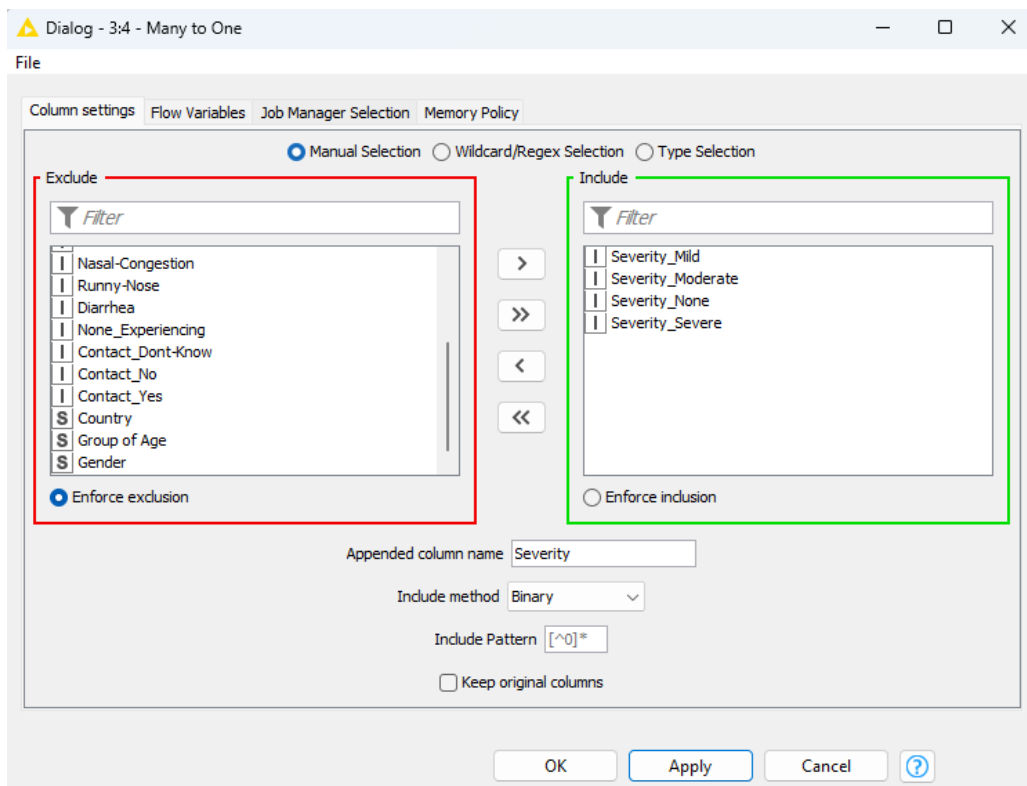
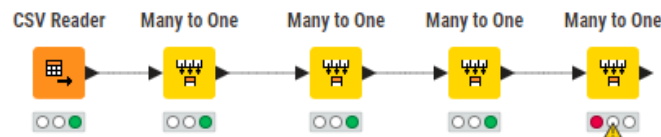
Preview

The suggested column types are based on the first 10000 rows only. See 'Advanced Settings' tab.

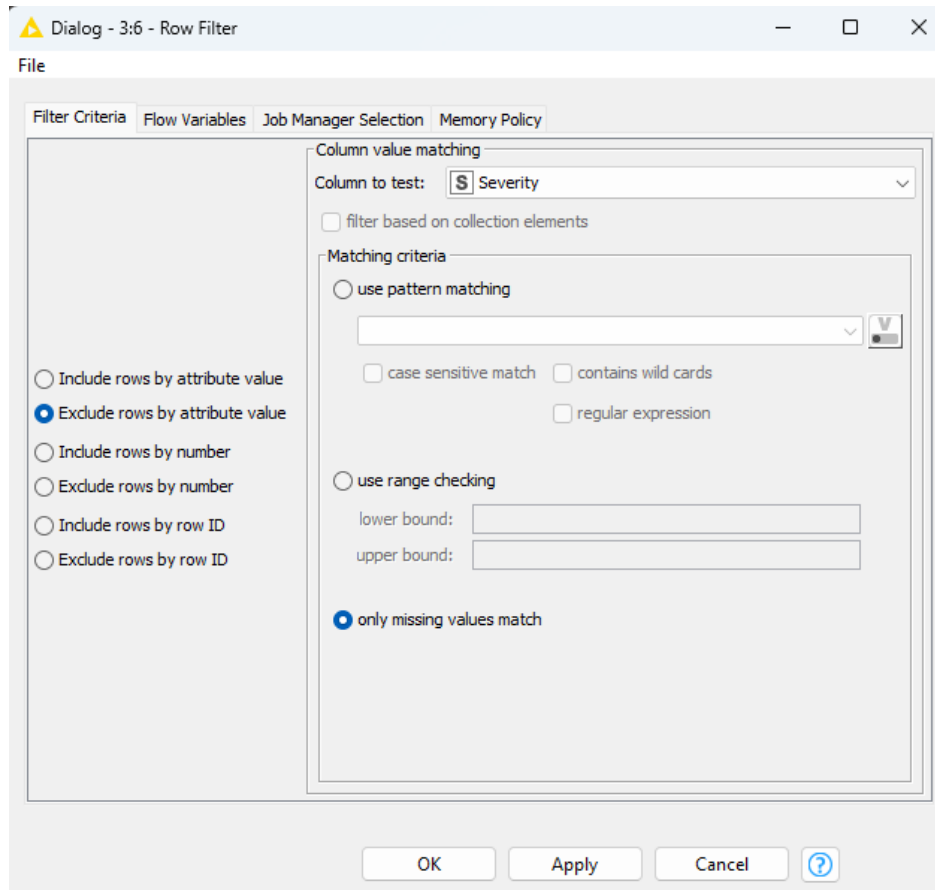
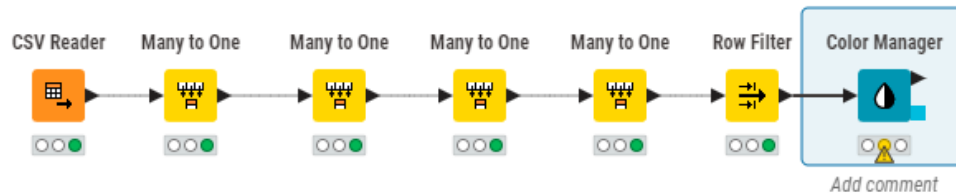
Row ID	Fever	Tiredness	Dry-Co...	Difficult...	Sore-T...	None_S...	Pains	Nasal-C...	Runny-...	Diarrhea	None_E...	Age
Row0	1	1	1	1	1	0	1	1	1	1	0	1
Row1	1	1	1	1	1	0	1	1	1	1	0	1
Row2	1	1	1	1	1	0	1	1	1	1	0	1
Row3	1	1	1	1	1	0	1	1	1	1	0	1

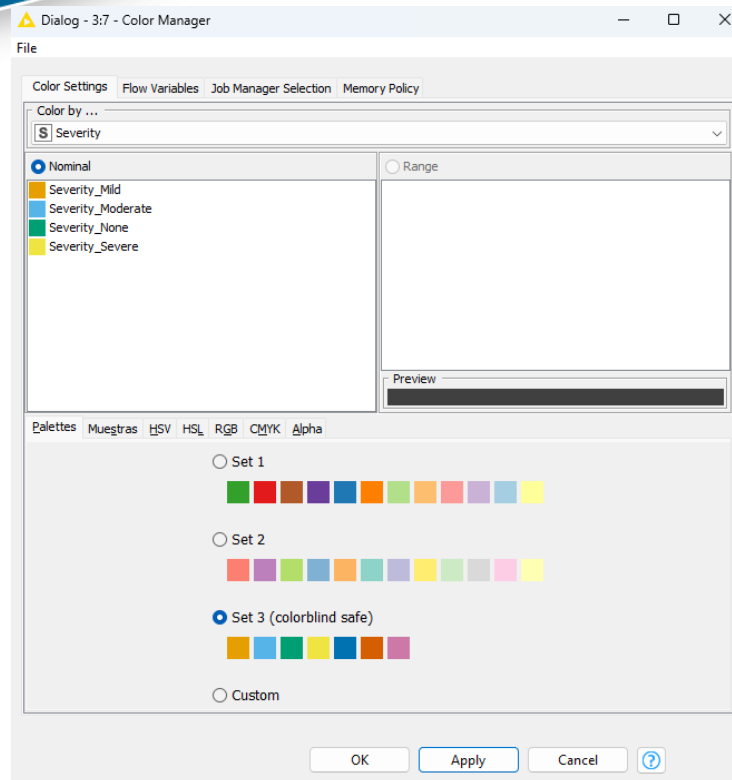
OK Apply Cancel ?

Ahora, al ver que el Dataset cuenta con muchas columnas (27) vamos a utilizar la función “Many to One” con el fin de agrupar cierto tipo de información en una sola columna. En este caso utilizamos 4 veces la función para agrupar lo siguiente: grupo de edades, genero del paciente, severidad del caso e información sobre si el paciente estuvo en contacto con alguien que tenía la enfermedad. Todo esto hace que el Dataset se reduzca a 16 columnas y podamos continuar con el análisis de la información de manera más sencilla.

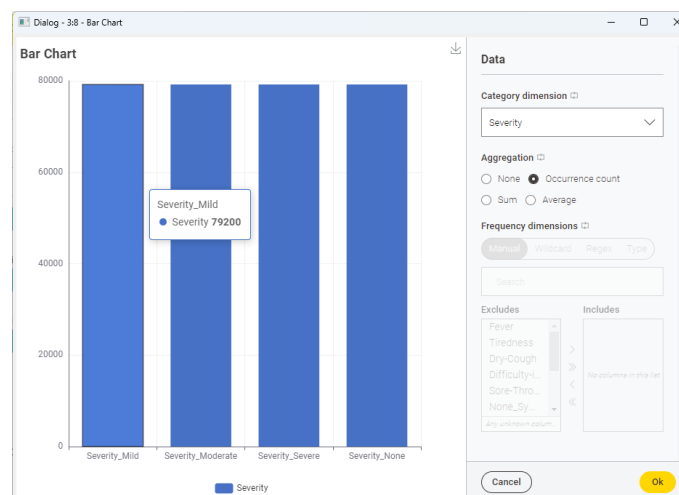


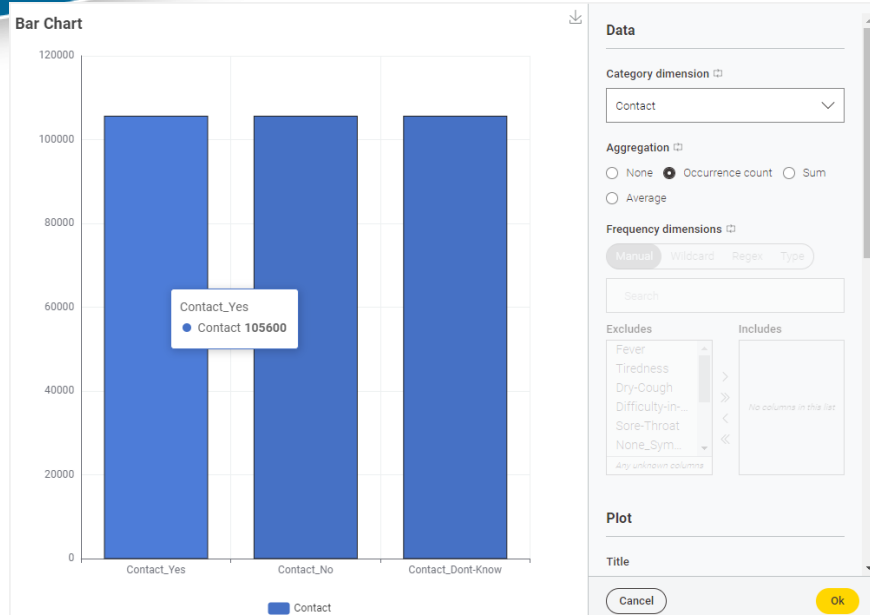
Ahora aplicamos dos funciones una es **“Row Filter”** y la otra **“Color Manager”**, la primera nos ayuda a eliminar todas las filas que tengan espacios vacíos en algunos de sus campos y la segunda función nos permite aplicar un color a la variable objetivo la cual en este caso es **“Severidad”** del caso de Covid. En las siguientes imágenes podemos ver su configuración.



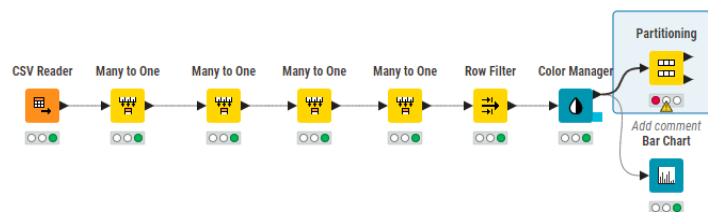


Ahora vamos a representar la variable objetivo la cual es “Severidad” por medio de un gráfico de barras. En el gráfico podemos ver que todas las variables tienen el mismo número de datos (79200). Esto nos hace ver que el Dataset tiene como objetivo contar con el mismo número de casos en cuanto a severidad y las demás variables, ya que como vemos en la segunda imagen los datos de “Contacto” también cuentan con el mismo número de casos (105600).





Lo que sigue es utilizar la función de **“Partitioning”**, esta función nos permite especificar el número de datos que se utilizará para entrenar al algoritmo, en nuestro caso trabajaremos con el 80% de los datos como entrenamiento.



Dialog - 3:9 - Partitioning

File

First partition Flow Variables Job Manager Selection Memory Policy

Choose size of first partition

☐ Absolute 100

☒ Relative[%] 80

☐ Take from top

☐ Linear sampling

☐ Draw randomly

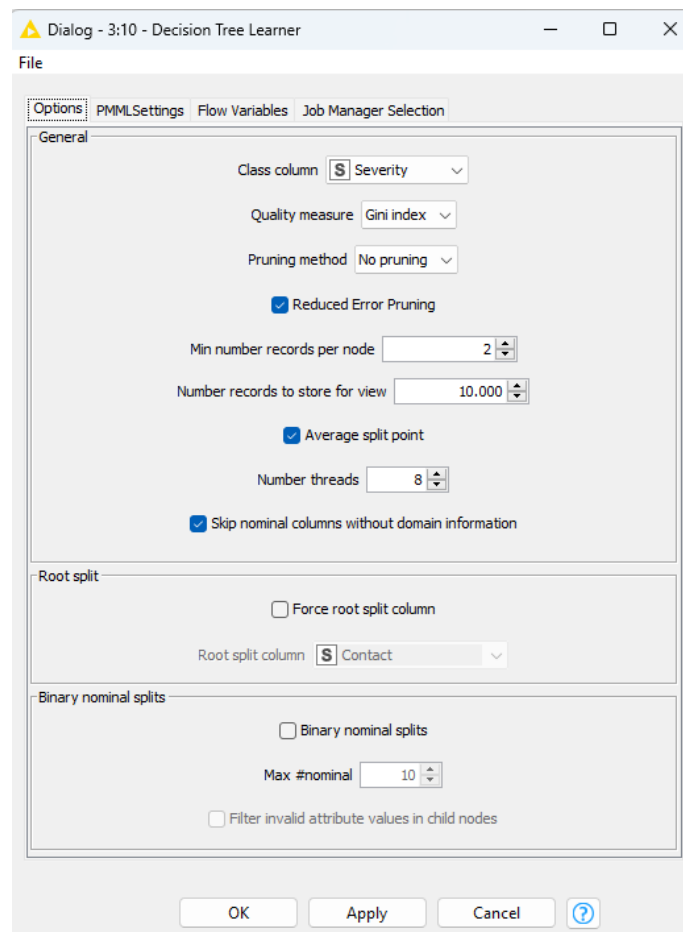
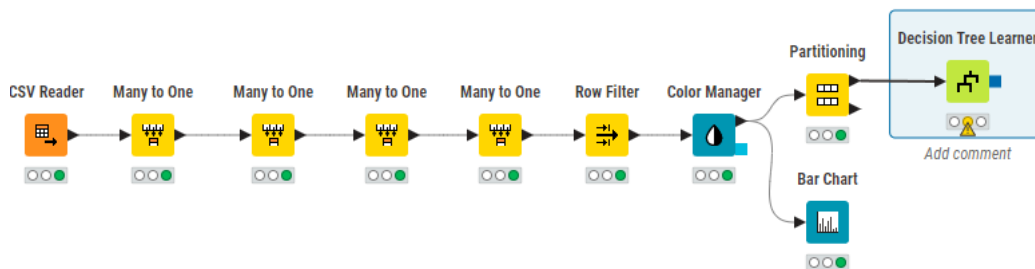
☒ Stratified sampling \$ Severity

☐ Use random seed 1.699.918.084.1

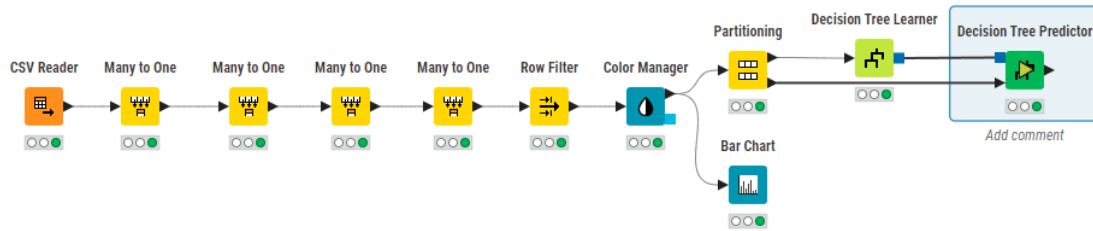
OK Apply Cancel ?

Ejercicio con el algoritmo de árboles de decisión

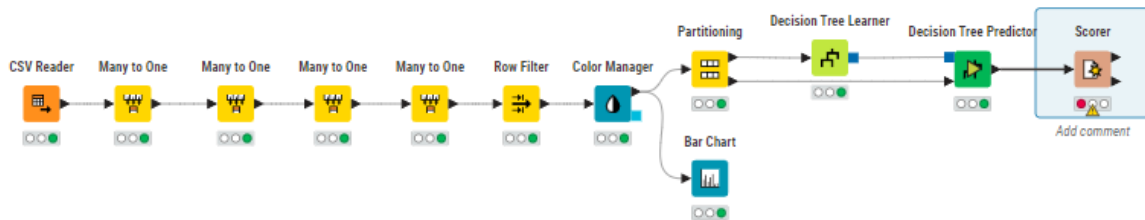
Para empezar lo primero que vamos a hacer es conectar los datos de entrenamiento de la función “**Partitioning**” a la función “**Decision Tree Learner**”, esto con el fin de que el algoritmo sepa cual es la variable objetivo y cuantos datos se usaran para que pueda entrenarse. En las siguientes imágenes podemos ver su proceso de configuración.



Ahora vamos a poner la función **“Decision Tree Predictor”**, esta función la conectamos con los datos de prueba y con la función **“Decision Tree Learner”** y le damos en ejecutar.



Después de realizar esto vamos a usar la función **“Scorer”**, esta función nos permite ver la matriz con los resultados de las predicciones. En las siguientes imágenes se puede ver su configuración.



Dialog - 3:12 - Scorer

File

Scorer | Flow Variables | Job Manager Selection | Memory Policy

First Column
Severity

Second Column
Prediction (Severity)

Sorting of values in tables
Sorting strategy: Insertion order ☐ Reverse order

Provide scores as flow variables
☐ Use name prefix

Missing values
In case of missing values: ☒ Ignore ☐ Fail

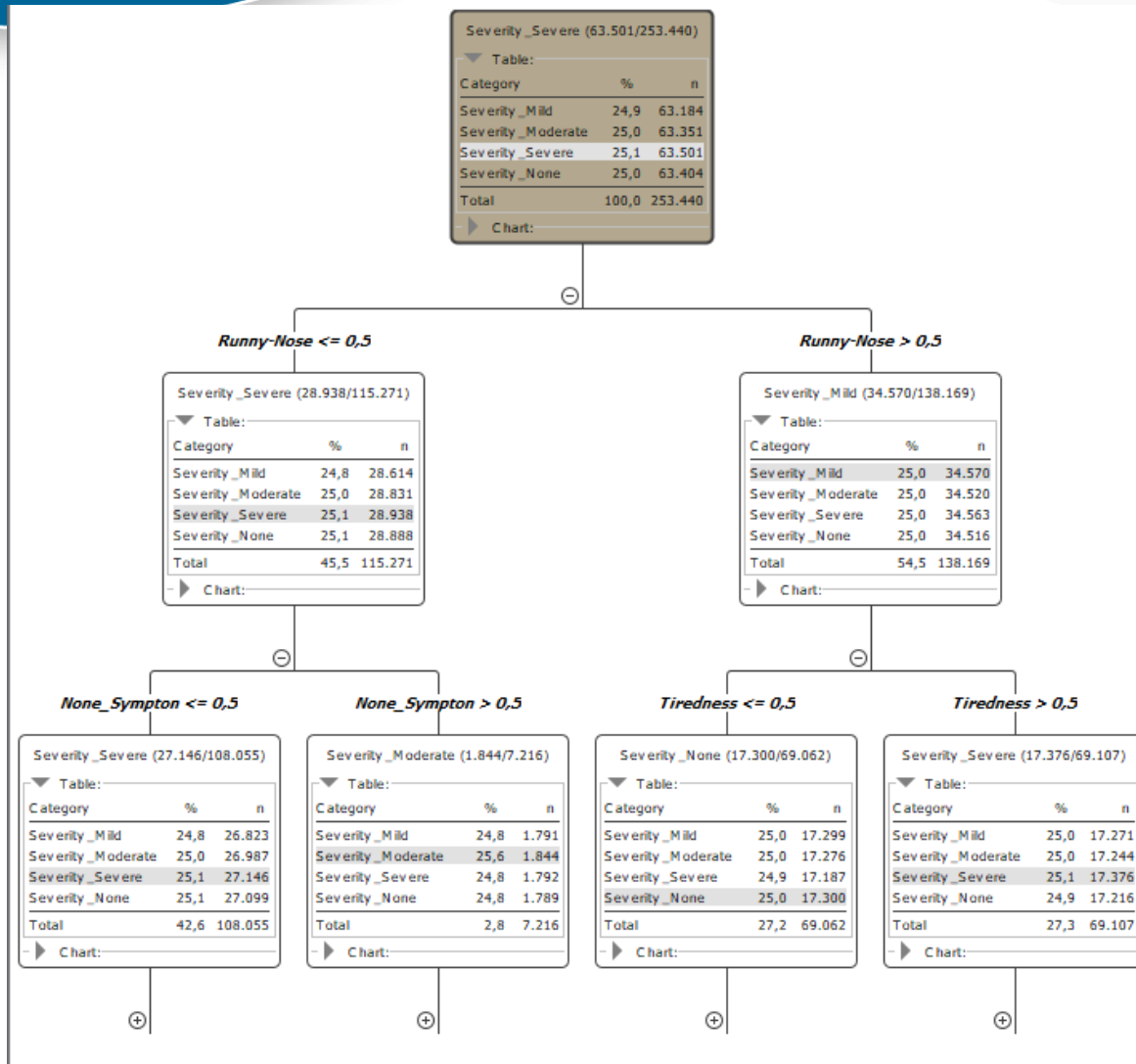
OK Apply Cancel ?

Ahora vamos a ver que los resultados fueron muy negativos, ya que la matriz de prueba nos indica que el algoritmo cuenta con una precisión del 0,33%. Debido a esto considero que se debe analizar mejor que variables están afectando al algoritmo y quitarlas, con el fin de obtener una precisión de predicción mayor.

Confusion Matrix - 3:12 - Scorer				
File Hilite				
Severity \ Prediction...	Severity_Mild	Severity_...	Severity_S...	Severity_N...
Severity_Mild	81	6675	5044	4134
Severity_Moderate	6643	52	5060	4020
Severity_Severe	6602	5192	52	4083
Severity_None	6625	5173	3898	26
<div> <div>Correct classified: 211</div> <div>Wrong classified: 63.149</div> <div>Accuracy: 0,333%</div> <div>Error: 99,667%</div> <div>Cohen's kappa (κ): -0,329%</div> </div>				

Entonces ahora lo que voy a realizar es trabajar solo con las variables de los síntomas para que según estas se pueda saber el grado de severidad del caso de Covid y ver si esto permite que el algoritmo aumente su precisión a la hora de realizar las predicciones.

A continuación, en las siguientes imágenes podemos ver el árbol de decisión proporcionado por la función “**Decision Tree Predictor**” y la matriz de confusión con la información clave de las predicciones. Además, analizaremos los resultados obtenidos con el fin de saber si el algoritmo es fiable.



A partir de lo que podemos ver en el árbol de clasificación, la primera característica que divide el conjunto de datos es “Secreción nasal”. Esto nos indica que 115.271 casos no presentan este síntoma lo cual es el 45,5% de todos los casos y que 138.169 (54,5%) si lo presentan. En este aspecto no podemos profundizar más, ya que los casos de severidad tienen casi que el mismo porcentaje.

Ahora de los 115.271 casos que no presentaron “Secreción nasal”, 108.055 presentaron alguno de los otros síntomas peligrosos, como puede ser “Dificultad al respirar” y tan solo 7.216 no presentaron ninguno de los síntomas peligrosos del Covid.

Por otro lado, de los 138.169 casos que presentaron el síntoma de “Secreción nasal”, 69.062 no presentaron el síntoma de “Cansancio”, pero 69.107 si lo presentaron; dejando un poco marcado que los que no presentaron este síntoma llegaron a ser caso con ninguna severidad.

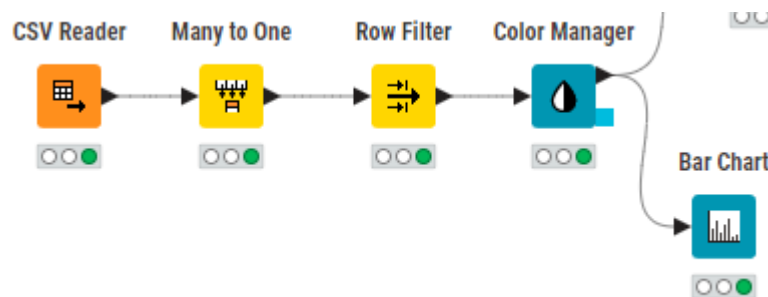
Confusion Matrix - 3:12 - Scorer				
File	Hilite			
Severity \ Prediction ...	Severity_Mild	Severity_...	Severity_S...	Severity_N...
Severity_Mild	3349	4224	4485	3958
Severity_Moderate	3787	3722	4463	3877
Severity_Severe	3751	4184	3943	3821
Severity_None	3770	4189	4425	3412
<div> <div>Correct classified: 14.426</div> <div>Wrong classified: 48.934</div> <div>Accuracy: 22,768%</div> <div>Error: 77,232%</div> <div>Cohen's kappa (κ): -0,03%</div> </div>				

A partir de la matriz de confusión podemos ver que el algoritmo, aunque tuvo un aumento en su precisión, pasando de un 0,33% a un 22,77%. Aún es un algoritmo poco fiable teniendo como porcentaje de error un valor de 77,23% y clasificando 48.934 datos de manera incorrecta y tan solo 14.426 de forma correcta.

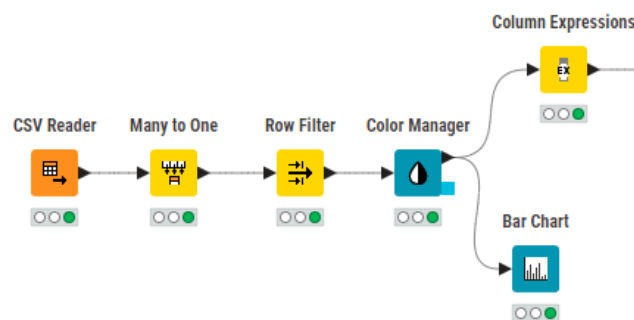
Ejercicio con el algoritmo KNN o vecino más cercano

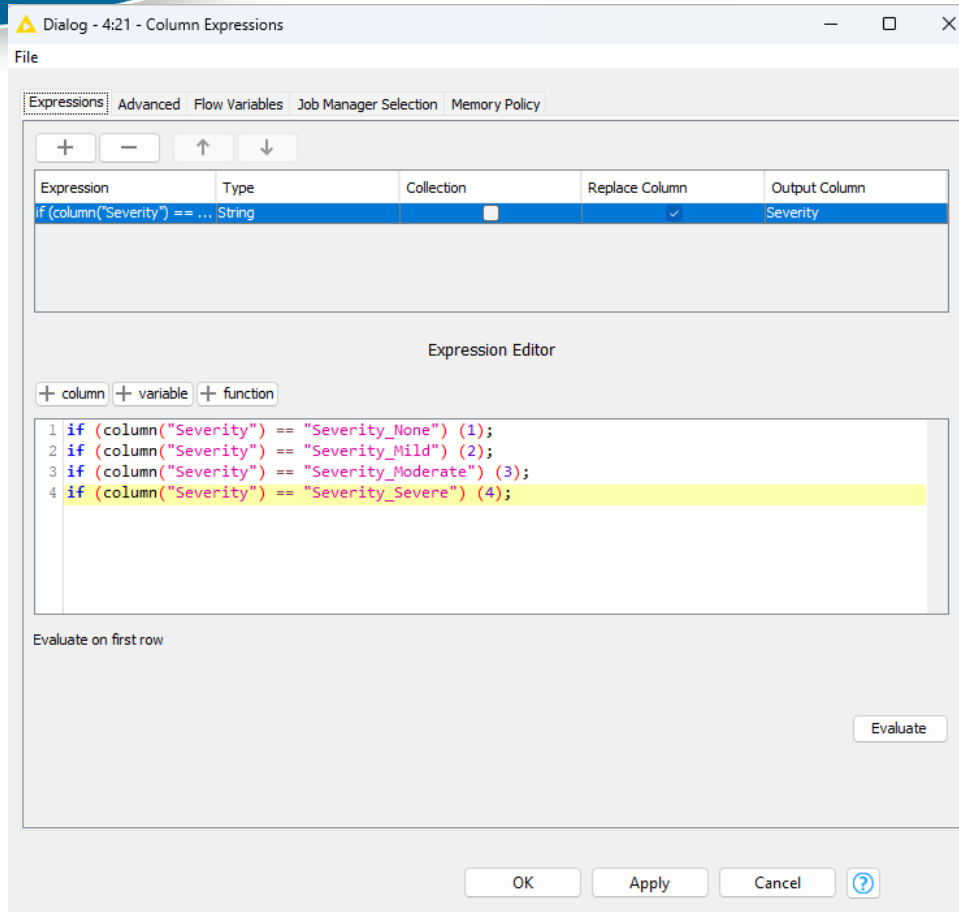
Ahora vamos a revisar y analizar el mismo Dataset del Covid, pero esta vez lo haremos con el algoritmo KNN, el cual busca los vecinos más cercanos a la hora de realizar predicciones.

Para empezar, vamos a mostrar la configuración que pusimos para establecer al algoritmo que las variables a tener en cuenta son los síntomas, esto con el fin de establecer lo mismo que en el algoritmo de arboles de decisión y poder comparar en las conclusiones finales.

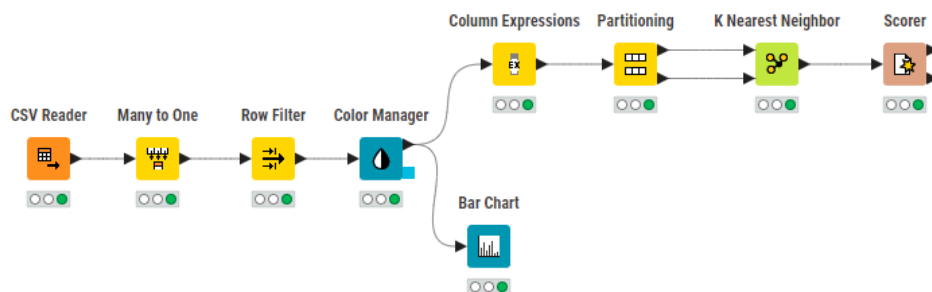


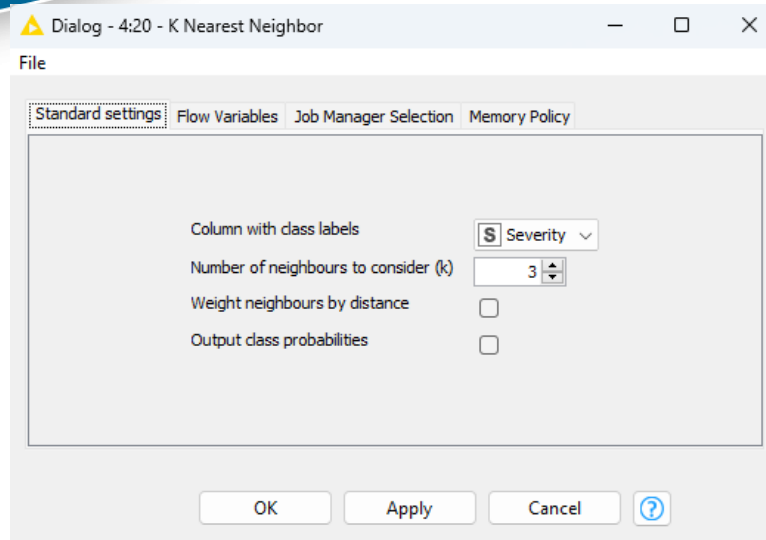
Al tener la columna de la variable objetivo ya agrupada (Severidad), tuvimos que pasar del formato string donde las variables se clasificaban de la siguiente manera: Ninguna severidad, severidad baja, severidad moderada y severidad alta. A un formato de tipo int quedando cada tipo de severidad representada por un número como podemos ver a continuación: 1 (ninguna severidad), 2 (severidad baja), 3 (severidad moderada) y 4 (severidad alta). Esto debido a que el algoritmo KNN no quería funcionar. En las siguientes imágenes vemos el proceso realizado.






Al haber realizado lo anterior ya podemos continuar con el proceso normal, para ello usamos la función **“Partitioning”** y le ponemos que el 80% de los datos serán de entrenamiento. Después colocamos la función **“K Nearest Neighbor”** y la configuramos con 3 puntos como vecinos cercanos y finalmente, ponemos la función **“Scorer”**, esto para poder ver la matriz de confusión con los resultados de las predicciones.






Confusion Matrix - 4:22 - Scorer

—
□
✕

File Hilite

Severity \ Class [kNN]	2	3	4	1
2	3301	3415	5029	4125
3	3709	2947	5117	4220
4	3705	3393	4520	4130
1	3708	3343	5109	3589

Correct classified: 14.357

Accuracy: 22,659%

Cohen's kappa (κ): -0,031%

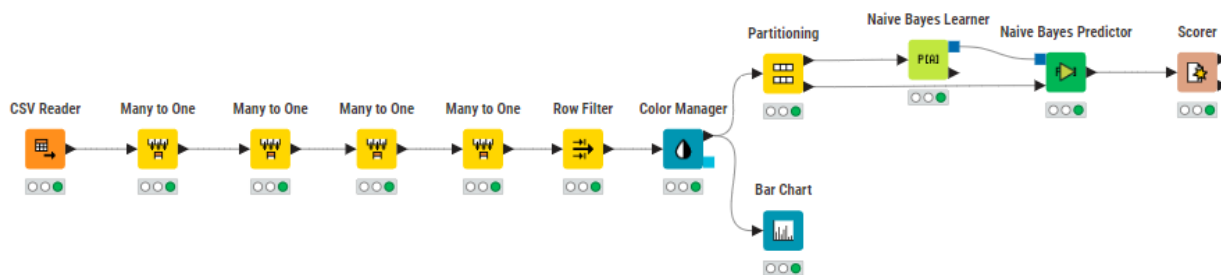
Wrong classified: 49.003

Error: 77,341%


A partir de los resultados obtenidos podemos ver que el algoritmo KNN obtuvo una precisión de predicción del 22,66% quedando un poco más abajo que la precisión del algoritmo de árboles de decisión. En este caso el algoritmo clasificó 14.357 veces el caso de severidad del Covid de manera correcta y 49.003 de forma incorrecta. Esto lo que quiere decir es que nuevamente nos encontramos con un algoritmo poco fiable para la predicción de estos datos, ya que maneja una precisión menor al 50%.

Ejercicio con el algoritmo Naive Bayes

Después de haber utilizado la función “Partitioning” y decir que el 80% de los datos serán de entrenamiento, continuamos seleccionando la función “Naive Bayes Learner”, donde especificamos la variable objetivo. Después ponemos la función que nos realizará las predicciones de los datos la cual se conoce como “Naive Bayes Predictor” y finalmente debemos poner el “Scorer” el cual nos mostrará la matriz de confusión con los resultados de las predicciones hechas por el algoritmo.



Para empezar, vamos a mostrar la matriz de confusión con todas las variables estudiadas.


Confusion Matrix - 5:12 - Scorer

—
□
×

FileHilite

Severity \ Prediction ...	Severity_Mild	Severity_Moder...	Severity_Severe	Severity_None
Severity_Mild	2338	6208	2335	5068
Severity_Moderate	2489	5936	2356	5002
Severity_Severe	2463	6252	2223	4997
Severity_None	2491	6140	2357	4705

Correct classified: 15.202

Wrong classified: 48.158

Accuracy: 23,993%

Error: 76,007%

Cohen's kappa (κ): -0,013%

Desde aquí ya podemos ver que el algoritmo de Naive Bayes tiene una precisión mayor que la de los dos anteriores, ya que cuenta con un valor de 23,99%. De igual forma aún es un porcentaje muy bajo de precisión.

Ahora vamos a trabajar como lo hicimos en los dos algoritmos anteriores, es decir, solo con las variables de los síntomas y posteriormente analizaremos los resultados. Primero en la siguiente imagen vemos como funciona el algoritmo Naive Bayes. Este algoritmo se basa en las probabilidades, para ello cuenta los valores, saca las medias, utiliza la desviación y realiza un porcentaje de los datos.

Class counts for Severity

Class:	Severity_Mild	Severity_Moderate	Severity_None	Severity_Severe
Count:	63481	63353	63277	63329

Total count: 253440

Threshold to used for zero probabilities: 1.0E-4

Skipped attributes: Group of Age/No records, Gender/No records, Contact/No records

Gaussian distribution for Diarrhea per class value

	Severity_Mild	Severity_Moderate	Severity_None	Severity_Severe
Count:	63481	63353	63277	63329
Mean:	0,36423	0,3633	0,36231	0,36411
Std. Deviation:	0,48122	0,48095	0,48067	0,48118
Rate:	25 %	25 %	25 %	25 %

Gaussian distribution for Difficulty-in-Breathing per class value

	Severity_Mild	Severity_Moderate	Severity_None	Severity_Severe
Count:	63481	63353	63277	63329
Mean:	0,50034	0,50058	0,49713	0,49961
Std.	0,5	0,5	0,5	0,5

Finalmente, en la siguiente imagen tenemos la matriz de confusión con las predicciones del algoritmo basándose solamente en los síntomas del Covid.

Confusion Matrix - 5:12 - Scorer				
File	Hilite			
Severity \ ...	Severity_Mild	Severity_Moderate	Severity_Severe	Severity_None
Severity_Mild	3120	2405	8146	2048
Severity_Mo...	3251	2296	8156	2144
Severity_Se...	3316	2363	8062	2130
Severity_None	3265	2388	8245	2025
<div> <div>Correct classified: 15.503</div> <div>Wrong classified: 47.857</div> <div>Accuracy: 24,468%</div> <div>Error: 75,532%</div> <div>Cohen's kappa (κ): -0,007%</div> </div>				

Con estos resultados podemos confirmar que de los 3 algoritmos utilizados el de Naive Bayes es el mas preciso, pero de igual forma sigue teniendo una precisión muy baja con un porcentaje de tan solo 24,47%, clasificando correctamente 15.503 valores y fallando 47.857 veces. Lo que quiere decir que sigue siendo un algoritmo con baja confiabilidad de predicción para este caso.

Si se quisiera tener un porcentaje mayor de precisión pues se deberá analizar cada variable una por una para o también se podría entrenar al modelo con un porcentaje mayor de los datos como por ejemplo con el 90%.

De igual forma también para los algoritmos es más complicado clasificar tantos datos entre 4 variables diferentes, por eso también las imprecisiones.

Conclusiones

En conclusión, los algoritmos como los de árboles de decisión, el KNN (vecino más cercano) o Naive Bayes, son muy interesante y muy importantes a la hora de empezar a aprender a analizar datos. Cada uno tiene sus puntos positivos y negativos y por eso se debe aprender en que momento cada uno de ellos es más oportuno de utilizar.

En cuanto a las conclusiones del ejercicio puedo decir que los tres algoritmos tuvieron un porcentaje de precisión bastante bajo, esto supongo que se debe a la gran cantidad de datos y que el Dataset no tiene muchas variables que se correlacionen y permitan saber mejor el grado de severidad de un caso.

De los tres modelos el mejor fue el de Naive Bayes supongo que por que se maneja por medio de probabilidades, logrando alcanzar una precisión del 24,47%.

Referencias

Home / KNIME. (s. f.). <https://www.knime.com/>

Posada Hernández, G. J. (2016). Elementos básicos de estadística descriptiva para el análisis de datos. Universidad Católica Luis Amigó. Recuperado de <https://elibro-net.bibliotecavirtual.unad.edu.co/es/ereader/unad/127436?page=128>

Taylor Smith. (2019). Supervised Machine Learning with Python : Develop Rich Python Coding Practices While Exploring Supervised Machine Learning. Packt Publishing.
https://bibliotecavirtual.unad.edu.co/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=2145644&lang=es&site=eds-live&scope=site&ebv=EB&ppid=pp_5