Análisis Predictivo Mediante Clasificación

INTELIGENCIA DE NEGOCIO
Práctica 1

Teresa Cabrera del Moral
GIIADE 2023/2024

CONTENIDO

1 Introducción 3
1.1 Hotel Registration 3
1.2 Identificación de Gestos 3
1.3 Bank Marketing 4
2 Procesado de Datos 5
3 Resultados Obtenidos 11
4 Configuración de Algoritmos 16
5 Análisis de Resultados 24
6 Interpretación de los Datos 26
7 Contenido Adicional 28
8 Bibliografía 30

1 INTRODUCCIÓN

En la siguiente práctica, se abordarán tres problemas diferentes y, utilizando algoritmos de aprendizaje supervisado de clasificación, realizaremos análisis predictivos sobre dichos problemas.

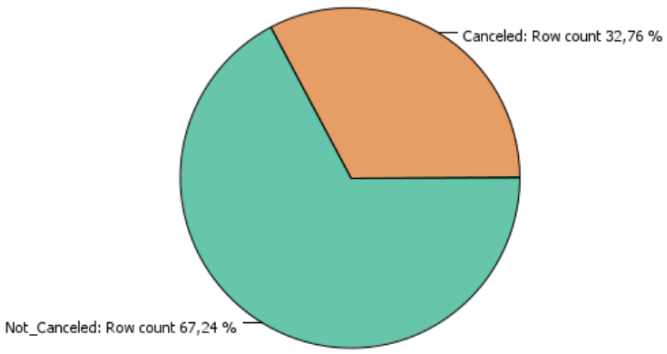
1.1 HOTEL REGISTRATION

Se trata de un problema cuya solución está cada día más demandada por las compañías hoteleras. Se pide realizar una predicción de cancelaciones de reservas en un hotel a partir de la información proporcionada en cada reserva.

Echando un vistazo a las 5 primeras filas de nuestro conjunto de datos, podemos hacernos una idea de con qué tipo de datos estamos trabajando. Tenemos tanto datos numéricos como categóricos. Por tanto, para ciertos algoritmos necesitaremos pasar la base de datos por un proceso de normalización.

Row ID	Booking...	...	n...	...	...	type_o...	...	room_t...	l...	ar...	...	...	mark...	...	...	...	...	avg...	...	Target
Row0	INN00001	2	0	1	2	Meal Plan 1	0	Room_Type 1	224	2017	10	2	Offline	0	0	0	65	0	Not_Canceled
Row1	INN00002	2	0	2	3	Not Selected	0	Room_Type 1	5	2018	11	6	Online	0	0	0	106.68	1	Not_Canceled
Row2	INN00003	1	0	2	1	Meal Plan 1	0	Room_Type 1	1	2018	2	28	Online	0	0	0	60	0	Canceled
Row3	INN00004	2	0	0	2	Meal Plan 1	0	Room_Type 1	211	2018	5	20	Online	0	0	0	100	0	Canceled
Row4	INN00005	2	0	1	1	Not Selected	0	Room_Type 1	48	2018	4	11	Online	0	0	0	94.5	0	Canceled

En esta gráfica podemos ver que existen muchos más casos en los que no ha habido cancelaciones. Estamos ante una base de datos desbalanceada que puede suponernos un problema para los algoritmos de clasificación pues predecirán con peor precisión. Más adelante trataremos de arreglar este problema para obtener mejores resultados.



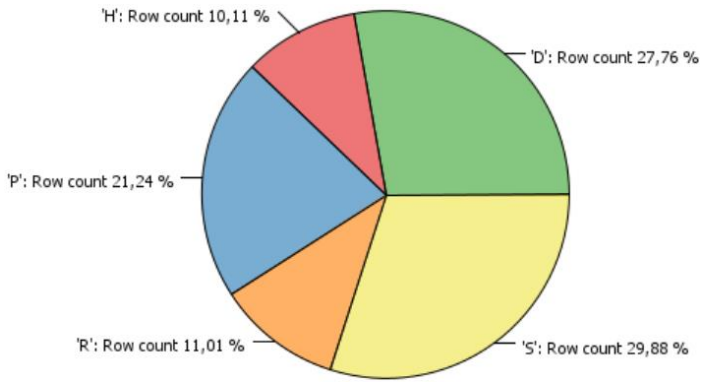
1.2 IDENTIFICACIÓN DE GESTOS

A partir de diversos vídeos en los que se hacen gestos, se han obtenido velocidades y aceleraciones vectoriales y se pide identificar las distintas fases de un gesto clasificándoles en 5 categorías: D (descanso), P (preparación), S (ahogo), H (quieto) y R (vuelta).

Podemos ver que, en este problema, todos los atributos son de tipo Double. Esto lo tendremos en cuenta a la hora de elegir algoritmos y preprocesar los datos.

Row ID	0	X11	X12	X13	X14	X15	X16	X17	X18	X19	X20	X21	X22	X23	X24	X25	X26	X27	X28	X29	X30	X31	X32	Phase
Row0		-0.001	0	-0.005	-0.001	0	0.001	0	0.001	0	0	-0	-0	-0	0	0.005	0.01	0.001	0.008	0.005	0.001	0	0	D
Row1		0.002	0	0.001	0	-0	-0	-0	-0	0	-0	0	-0	0	-0	0.005	0.006	0.001	0.003	0.001	0	0	0	D
Row2		0.002	0	-0.002	-0	0	-0.001	-0.001	-0	0	0	0	0	-0	-0	0.002	0.003	0	0.003	0.002	0.001	0	0	D
Row3		0.001	0	0	-0	-0	-0.001	-0	-0	-0	-0	0	-0.001	-0	-0	0.001	0.001	0	0.001	0	0.001	0	0.001	D
Row4		0.001	0	0	0	-0	0	0	-0	0	0	-0	0	0	-0	0	0.002	0	0.002	0	0	0	0	D

Fijándonos también en el reparto de la clase objetivo en la base de datos con la que trabajaremos vemos que hay algo de desbalanceo pues tenemos muchos más casos S (ahogo). Tomaremos esta clase como la 'positiva' a la hora de predecir y medir el rendimiento de los algoritmos.



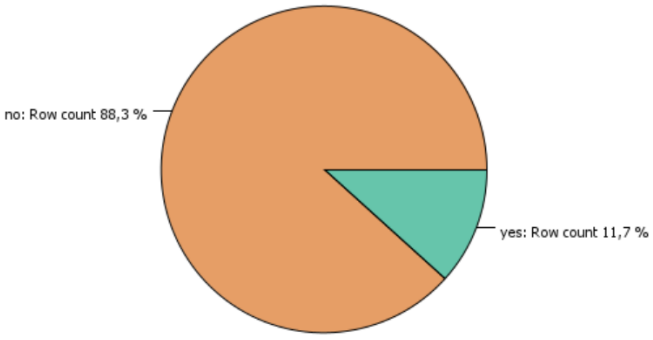
1.3 BANK MARKETING

Un banco portugués quiere predecir a partir de los datos personales de cada uno de sus clientes, si estarán interesados en un nuevo producto que se les quiere ofrecer. Se trata de 20 atributos y, el objetivo (y), es de tipo binario: el cliente se ha suscrito, o no se ha suscrito.

Este es un tipo de problema muy similar al de Hotel Registration porque también tenemos atributos tanto numéricos como categóricos, y también se nos pide predecir una clase binaria.

Row ID	I age	S job	S marital	S education	S default	I balance	S housing	S loan	S contact	I day	S month	I duration	I campaign	I pdays	I previous	S poutcome	S Target
Row0	58	management	married	tertiary	no	2143	yes	no	unknown	5	may	261	1	-1	0	unknown	no
Row1	44	technician	single	secondary	no	29	yes	no	unknown	5	may	151	1	-1	0	unknown	no
Row2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	may	76	1	-1	0	unknown	no
Row3	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	may	92	1	-1	0	unknown	no
Row4	33	unknown	single	unknown	no	1	no	no	unknown	5	may	198	1	-1	0	unknown	no

Además, podemos ver que el desbalanceo de la clase objetivo en este caso es mucho mayor, habiendo muchos más clientes que no están interesados en el nuevo producto.



Estando ante dos problemas tan similares, los abordaré con los mismos recursos, aplicándolos según el caso.

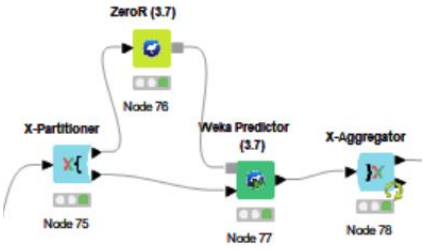
2 PROCESADO DE DATOS

El Preprocesado de Datos engloba a todas aquellas técnicas de análisis que permiten mejorar la calidad de un conjunto de datos de modo que las técnicas de minería puedan obtener mayor y mejor información.

Por ahora, abordaremos sólo el preprocesado justo y necesario para que los algoritmos funcionen dando resultados que, aunque no sean óptimos, son suficientes para comenzar a predecir.

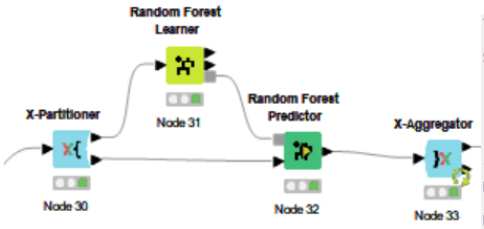
Hotel Registration y Bank Marketing, como hemos comentado en el apartado anterior, son problemas muy similares con datos numéricos y nominales, y buscan predecir una clasificación binaria. Por tanto, utilizaremos los mismos algoritmos para ambos:

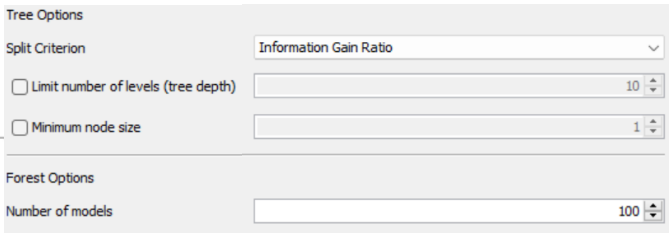
- **ZeroR**
  Predice siempre la clase mayoritaria. Nos sirve para compararlo con el resto de algoritmos y, si éstos obtienen peor resultados que ZeroR, algo falla.



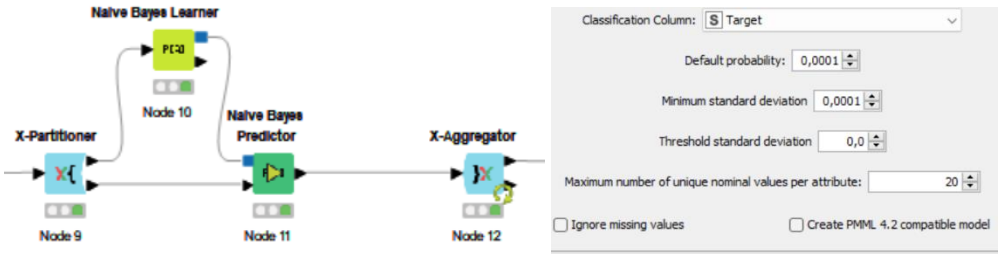
- **Random Forest**

  Combina múltiples árboles de decisión para, finalmente en un estudio de la clasificación de cada uno de ellos, devolver la predicción final. Puede manejar datos numéricos y categóricos y es resistente al sobreajuste, por lo que suponemos que tendrá un buen comportamiento incluso con el inicial desbalanceo.



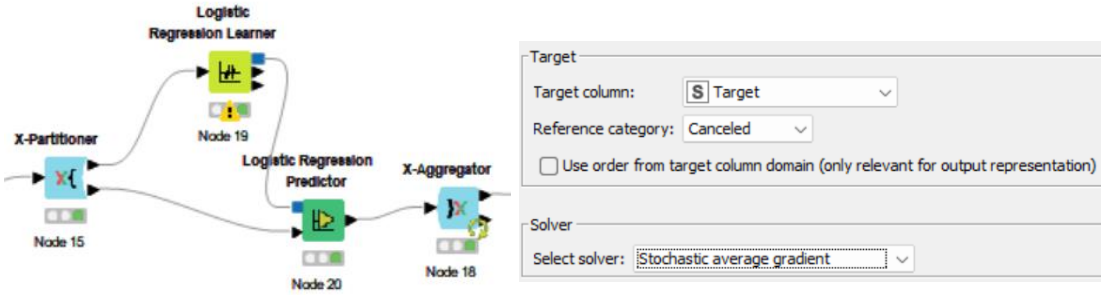


- **Naïve Bayes**
  Algoritmo probabilístico, basado en el teorema de Bayes. Permite variables no numéricas y también valores nulos, por lo que por ahora no tocaremos los datos. Este algoritmo supone que todos los atributos son independientes entre sí, sin embargo, es considerado un estándar y sus resultados son competitivos, pese a esta hipótesis poco realista.



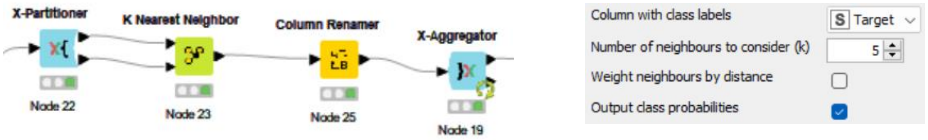
- **Regresión Logística**

  Utilizaremos este modelo lineal para estudiar si las características guardan una relación lineal con la predicción final. Debemos transformar los datos categóricos a numéricos y normalizarlos para que funcione.

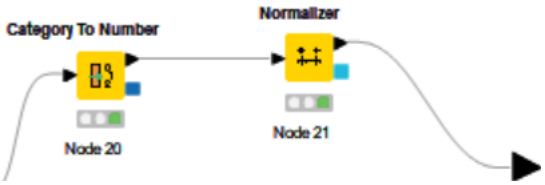


- **kNN**

  Utiliza las distancias entre vecinos para clasificar a una clase. Dado que el cálculo de "cercanía" se hace en función a la distancia euclídea, es necesario pasar los datos categóricos a número y normalizarlos para obtener mejores resultados.

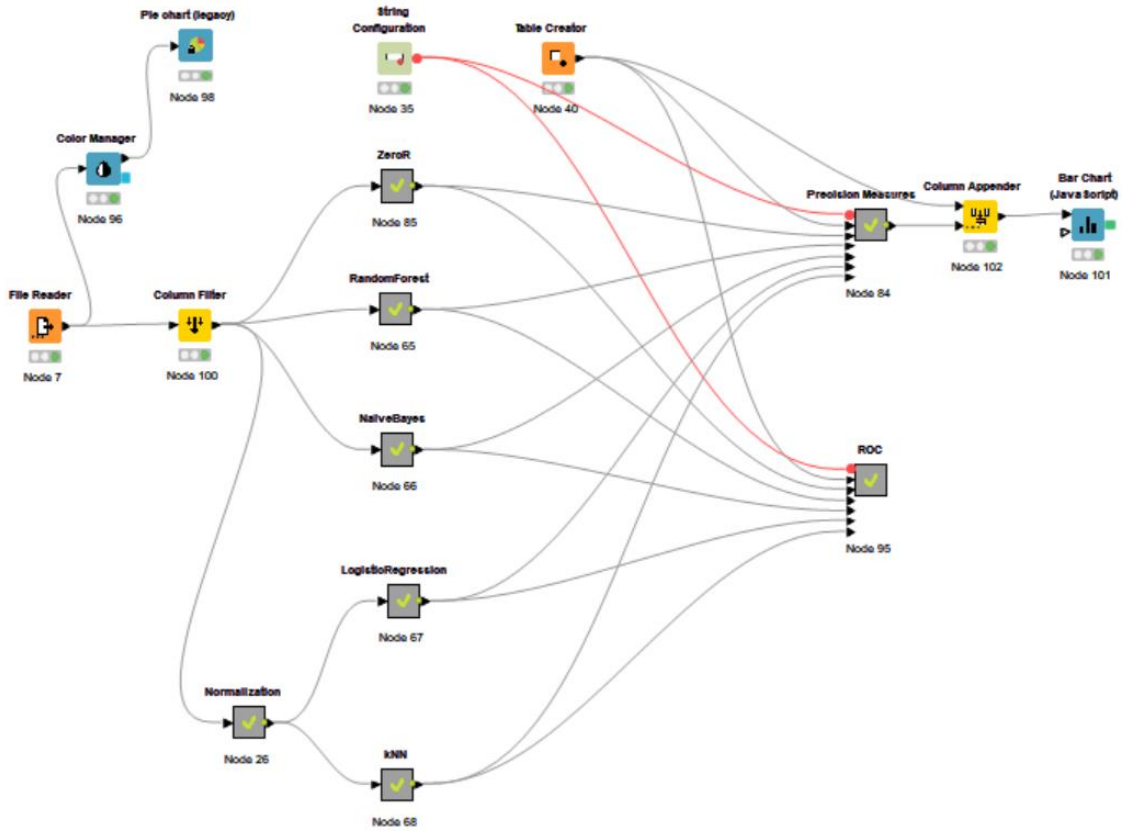


Para normalizar los datos antes de pasar por los algoritmos de Regresión Logística y kNN he utilizado el nodo Category To Number que asigna a cada categoría de los atributos un número y luego el Normalizer para normalizar los atributos, que ahora son totalmente numéricos, entre 0 y 1.

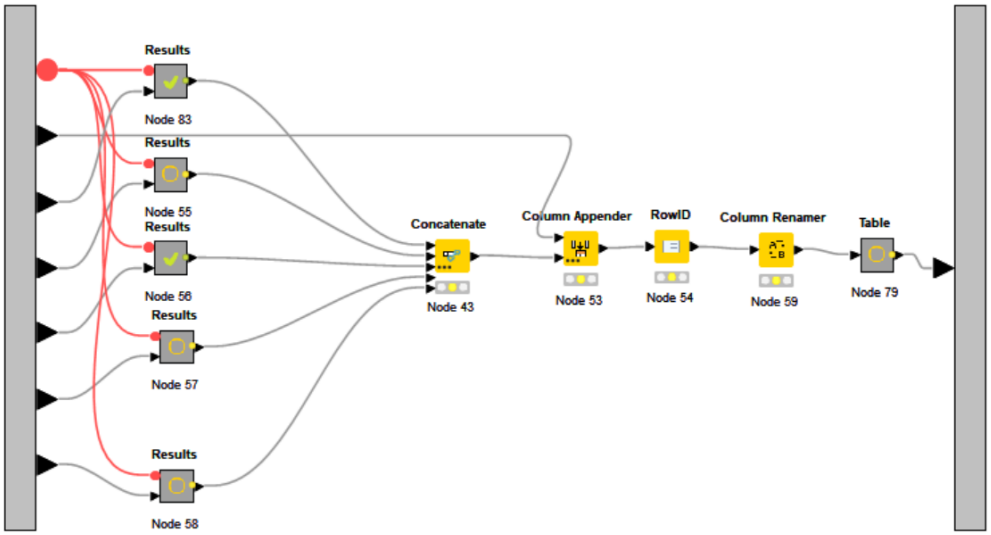


En el caso de Hotel Registration, como casi todos los *booking_ids* son únicos, no tiene sentido crear un número identificativo para cada uno de ellos, y como no es información relevante para la predicción, me desharé de esa columna con un nodo Column Filter.

Por otro lado, en Bank Marketing se nos avisa en la descripción sobre nuestro *dataset*, de que el atributo *duration* afecta en gran medida a nuestro atributo objetivo (puesto que, si la duración es 0, 'y' será no). Pero no conocemos la duración antes de hacer la llamada, y después de hacerla 'y' será conocido. Por tanto, es mejor ignorar este atributo para nuestro propósito en concreto.

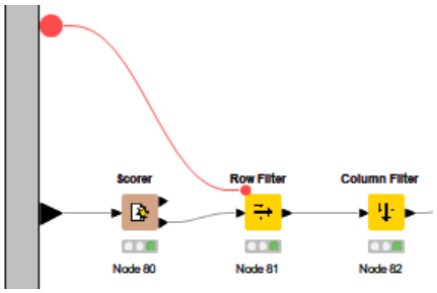


Aquí podemos ver el aspecto de nuestro primer *workflow* tanto para Hotel Registration como para Bank Marketing. Con el metanodo *Precision Measures* obtendremos una tabla con la matriz de confusión y medidas de rendimiento calculadas a partir de ella para cada algoritmo como la precisión, el F1-Score y la G-mean.

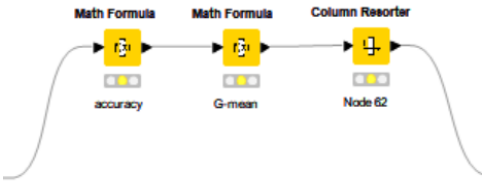


Dentro de cada metanodo *Results* hay un Scorer, que nos mostrará la matriz de confusión y las estadísticas de precisión como sensibilidad o especificidad, y un Row y Column Filter para

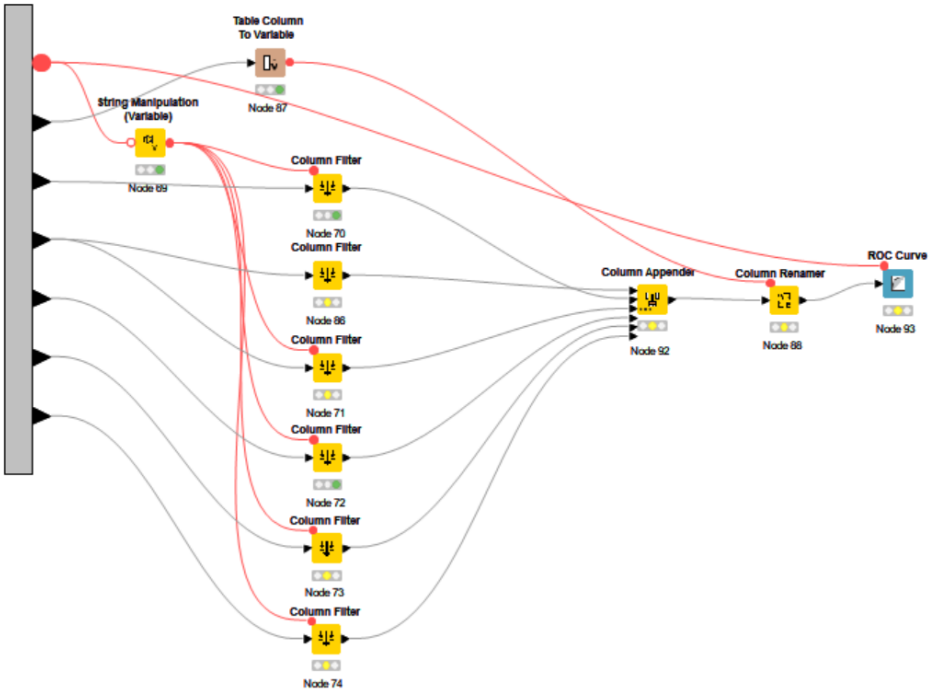
quedarnos con una tabla final sólo con las medidas que nos interesan, de manera que todas las tablas queden uniformes para después concatenarlas.



En el metanodo Table final, tendremos dos Math Formulas para calcular la *accuracy* y *g-mean* mediante las siguientes expresiones: `($TP$+$TN$)/($TP$+$FP$+$TN$+$FN$)` y `sqrt($TPR$*$TNR$)` respectivamente.

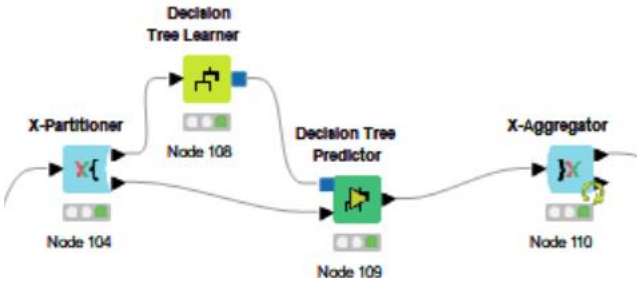


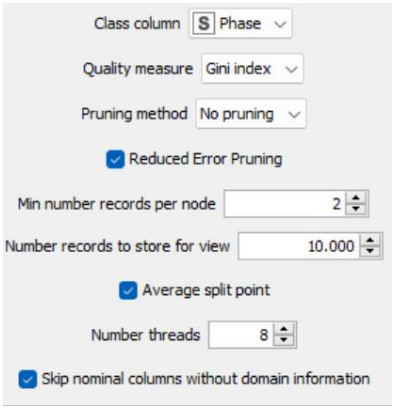
Y con *ROC* obtendremos la curva *Receiver Operating Characteristic*, que nos ayuda a representar la relación entre la tasa de verdaderos positivos (Sensibilidad) y la tasa de falsos positivos (1 - Especificidad).



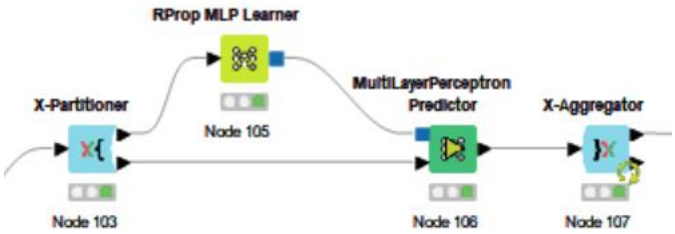
En la Identificación de Gestos, al tratarse de un problema multiclase con la totalidad de sus atributos numéricos. He decidido hacer otro enfoque.

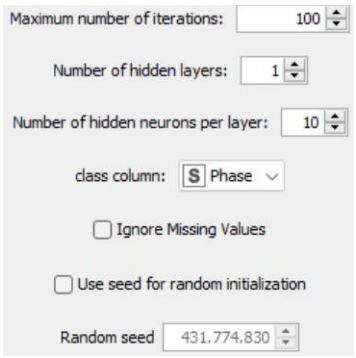
- **ZeroR**
  Lo añado de nuevo pues nos va a permitir medir el rendimiento de los algoritmos.
- **Árbol de Decisión**
  Se trata de un algoritmo fácil de utilizar y que puede resultar eficiente en este problema multiclase. Lo configuraremos con el criterio de elección de atributo Information Gain Ratio para que así, a la hora de construir el árbol, éste elija siempre primero aquella variable que predecirá una mayor cantidad de información.





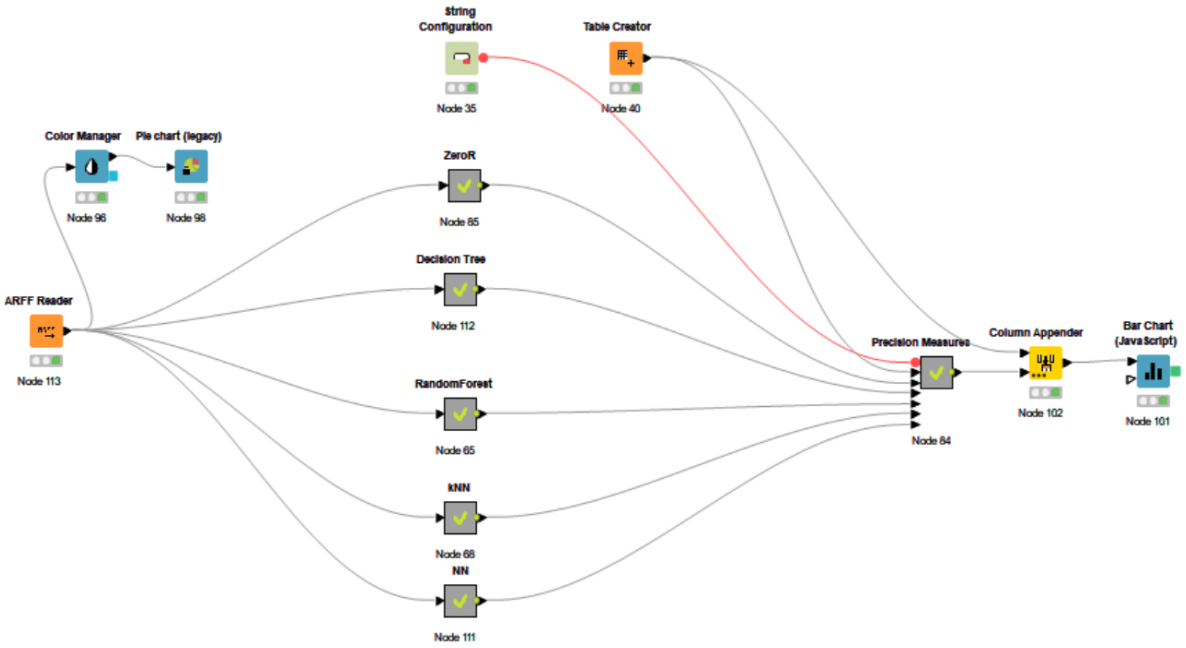
- **Random Forest**
  Volvemos a utilizarlo para este problema porque una técnica *ensemble* siempre aporta robustez a la predicción y, además, podremos comparar el rendimiento del Árbol de Decisión en solitario y en conjunto.
- **kNN**
  También volvemos a utilizar este algoritmo porque, en este problema en concreto, mi inicial suposición es que los gestos similares pueden tender a agruparse en el mismo espacio de características y, por tanto, se clasificarán como sus k vecinos más cercanos. Veremos cómo rinde y si es cierto.
- **Redes Neuronales**
  Basado en el funcionamiento del cerebro humano. Resulta un buen candidato para predecir este problema multiclase. Además, es más robusto que los árboles de decisión porque aportan peso a las conexiones entre las capas de nodos.





En principio sólo preprocesaremos los datos antes del algoritmo kNN para normalizarlos, puesto que se obtendrá un resultado más adecuado. Sin embargo, no es necesario ningún otro procesado de datos para que el resto de algoritmos comiencen a funcionar.

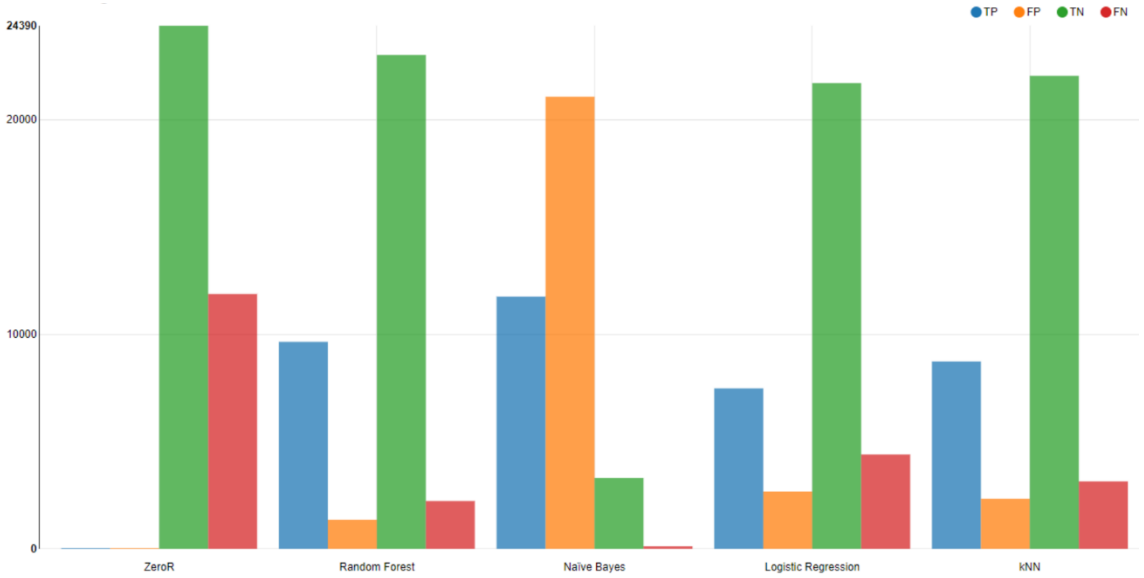
Aquí abajo se puede representado el *workflow* con el que empezaremos a trabajar para Identificación de Gestos.



3 RESULTADOS OBTENIDOS

Tras realizar una primera predicción con un procesamiento de datos simple obtenemos los siguientes resultados:

3.1 HOTEL REGISTRATION



En esta gráfica comparamos la matriz de confusión generada con cada uno de los algoritmos. Lo que buscamos es maximizar las columnas *True* (la azul y la verde), pues son los casos predichos con exactitud.

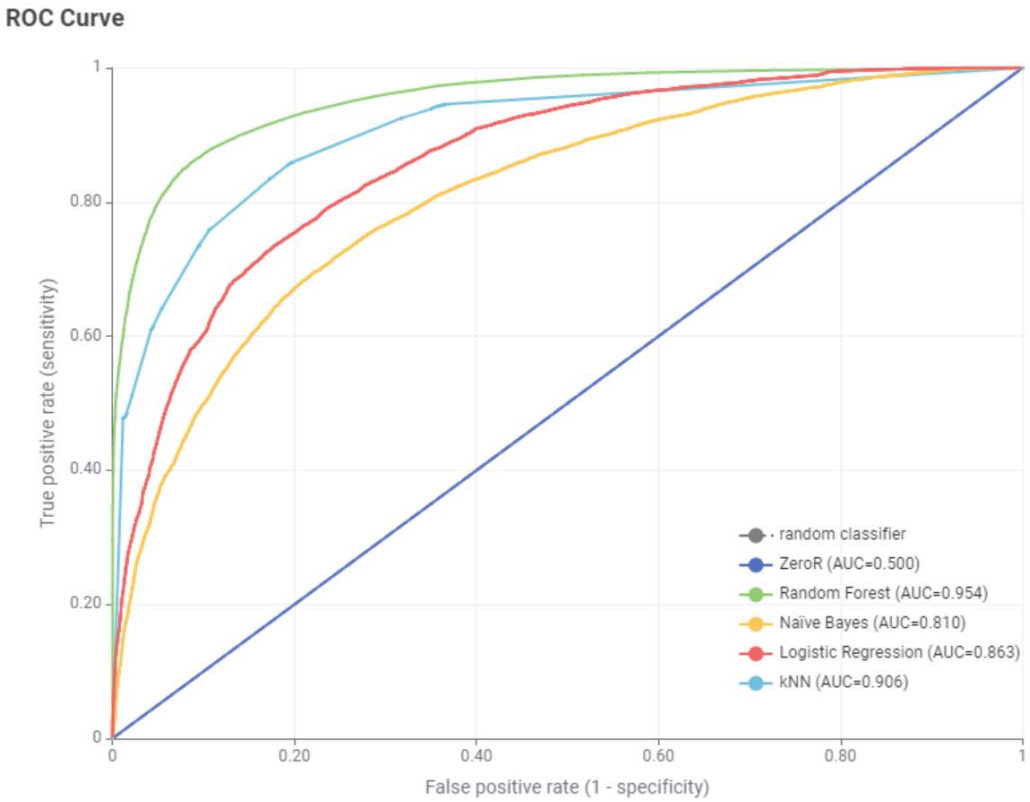
Se puede comprobar que los algoritmos que inicialmente han tenido un mejor comportamiento han sido Random Forest y kNN. Es evidente que todas han predicho muchos más casos 'no cancelados' a 'cancelados', esto se debe al desbalanceo de la clase, por lo tanto, lo tendremos en cuenta para mejorar los resultados.

Row ID	TP	FP	TN	FN	TPR	TNR	PPV	Accuracy	F1-score	G-mean
ZeroR	0	0	24390	11885	0	1	?	0.672	?	0
Random Forest	9665	1340	23050	2220	0.813	0.945	0.878	0.902	0.844	0.877
Naïve Bayes	11765	21072	3318	120	0.99	0.136	0.358	0.416	0.526	0.367
Logistic Regre...	7483	2674	21716	4402	0.63	0.89	0.737	0.805	0.679	0.749
kNN	8743	2293	22097	3142	0.736	0.906	0.792	0.85	0.763	0.816

La **precisión** por sí sola puede ser engañosa, ya que, por ejemplo, si nos fijamos en ZeroR, se obtiene una precisión de 0,672, es decir, una precisión por encima de 0,5, sólo por el hecho de que clasifica en función de la clase mayoritaria y por tanto, es evidente que al ser una base de datos desbalanceada en la que el 67,2% de los casos son 'No Cancelados', sea más probable que no se equivoque yendo a por esa mayoría. Pero su precisión en una base de datos balanceada realmente debería ser de 0,5. Por ello, para tener más conocimiento del rendimiento de un algoritmo, además de la medida de precisión, también tenemos la de **F1-score**. Ésta combina la precisión de las clases positivas (TP/TP+FP) con la capacidad del modelo para recuperar correctamente ejemplos de la clase positiva (TP/TP+FN).

Por tanto, esto nos dice que Random Forest al tener el F1-score más alto de 0.844 es el que mejor rendimiento presenta en términos de clasificación de la clase de interés: *Canceled*. Al

contrario que Naïve Bayes que con un 0.526, ha clasificado más falsos positivos que verdaderos positivos.



Cuanto más cerca esté la curva ROC del vértice superior izquierdo del gráfico (donde la sensibilidad es 1 y la especificidad es 1), mejor será su rendimiento.

Los algoritmos con mejor AUC (Area Under Curve) son efectivamente Random Forest y kNN.

Vemos que el que peor resultado nos da es Naïve Bayes. Como sabemos, este parte del supuesto de que todos los atributos son independientes. Por tanto, no es de extrañar que en este tipo de problemas tenga un mal desempeño, esto quiere decir, que realmente la decisión de cancelar o no cancelar una reserva sí que está muy relacionada con los atributos presentes en la base de datos: edad, número de días, número de adultos, de niños, etc. Lo que nos lleva a entender por qué kNN ha obtenido tan buenos resultados, pues, las variables con misma clasificación se encuentran en el mismo espacio multidimensional respecto a sus características.

Deduzco que Random Forest es el que mejores resultados ha obtenido puesto que, es eficaz para manejar el desequilibrio entre casos cancelados y no cancelados, ya que puede ajustar los pesos de las clases y asignar más importancia a las clases minoritarias.

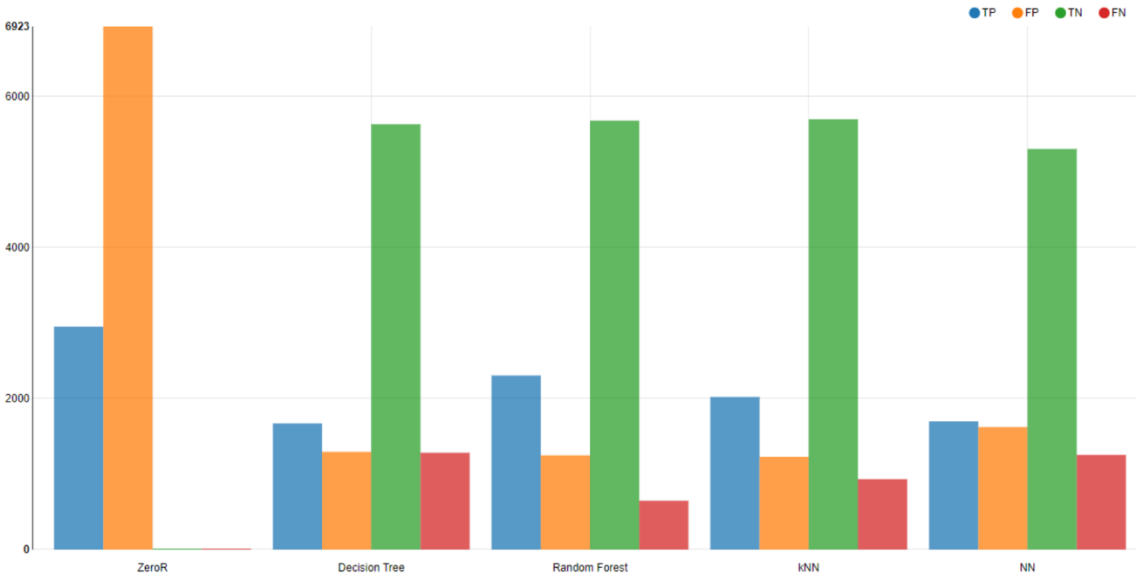
3.2 IDENTIFICACIÓN DE GESTOS

Row ID	I TP	I FP	I TN	I FN	D TPR	D TNR	D PPV	D Accuracy	D F1-score	D G-mean
ZeroR	2950	6923	0	0	1	0	0.299	0.299	0.46	0
Decision Tree	1669	1293	5630	1281	0.566	0.813	0.563	0.739	0.565	0.678
Random Forest	2304	1246	5677	646	0.781	0.82	0.649	0.808	0.709	0.8
kNN	2019	1228	5695	931	0.684	0.823	0.622	0.781	0.652	0.75
NN	1697	1621	5302	1253	0.575	0.766	0.511	0.709	0.541	0.664

Analizando los resultados obtenidos tras este primer intento de predicción vemos que al ser sólo una de las 5 clases, la positiva, será más probable predecir que una instancia es negativa (cualquiera de los otros cuatro estados de manos). Por ello, obtenemos buenos datos en

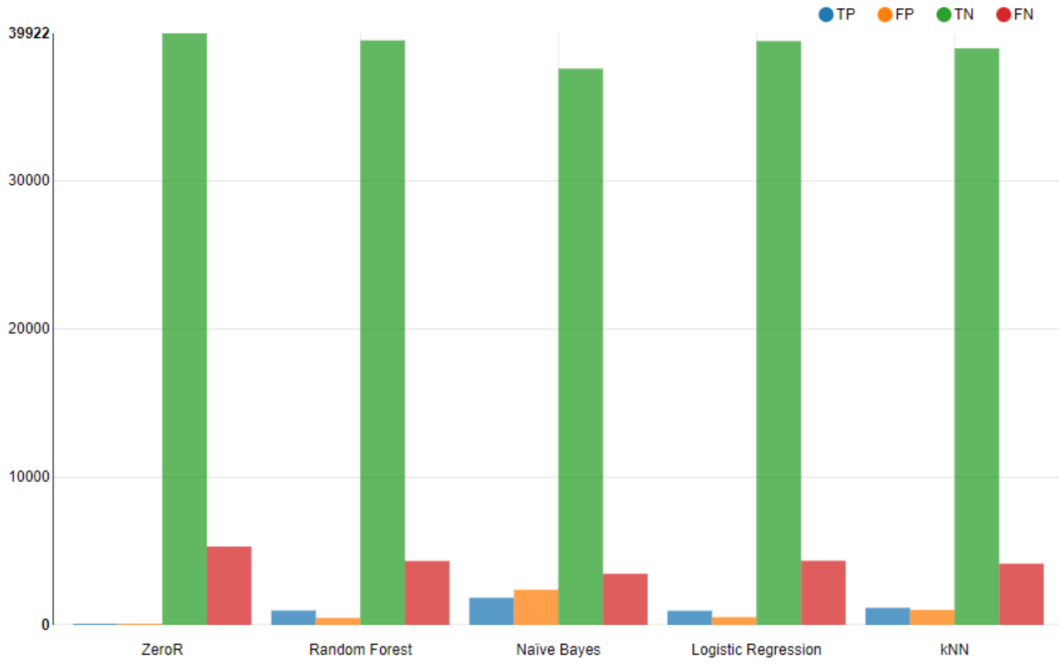
cuanto a los TN. Como nos dicta ZeroR, sabemos que tenemos 2950 casos 'S' en total; el algoritmo que más se ha acercado a esa cifra ha sido Random Forest que, con sus 100 árboles, ha conseguido ponerse en el primer puesto por encima del resto.

El peor, no obstante, han sido las redes neuronales que, a pesar de ser un algoritmo reconocido por su efectividad y buena tasa de acierto, no parece que esté rindiendo de forma óptima, consiguiendo sólo un 0.541 de F1-score.



3.3 Bank Marketing

Al igual que hemos visto con el problema de Hotel Registration, ningún algoritmo funciona muy bien con bases de datos desbalanceadas, puesto que tienden a clasificar la clase mayoritaria con mucha más frecuencia. Por lo que, si con Hotel Registration podíamos obtener unos primeros resultados malos con un desbalanceo del 67,2%, con Bank Marketing serán peores pues la clase objetivo tiene una proporción 88/12.



Aquel que en ambos casos suele rendir mejor es Random Forest ya que, al ser un algoritmo que combina muchos árboles de decisión mediante muestreo aleatorio, es robusto frente al sobreajuste.

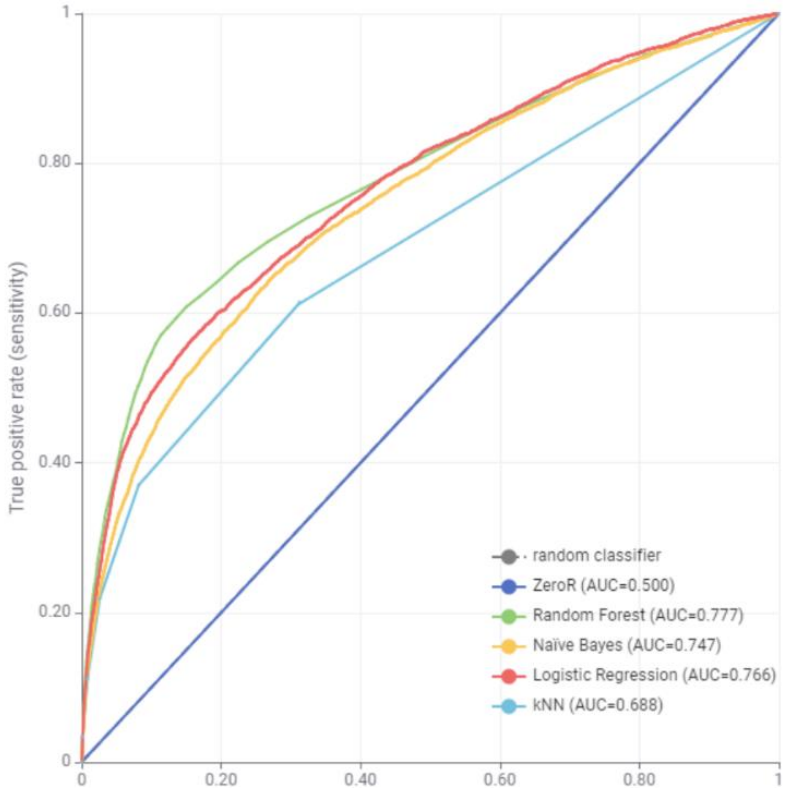
En el caso de kNN, esta predicción sesgada se debe a que hay una mayor probabilidad de que la mayoría de los k puntos cercanos sean de la clase mayoritaria. Como resultado, la clase minoritaria puede estar infrarrepresentada en las predicciones, lo que lleva a un rendimiento deficiente en la clasificación de la clase minoritaria.

Por tanto, aunque vemos que hay un alto porcentaje de acierto para los casos Verdaderos Negativos acompañado de unos números de Accuracy que parecen bastante buenos; los Verdaderos Positivos son prácticamente equiparables a los Falsos Positivos, y eso se ve reflejado en nuestros índices pésimos de F1-Score.

Row ID	I TP	I FP	I TN	I FN	D TPR	D TNR	D PPV	D Accuracy	D F1-score	D G-mean
ZeroR	0	0	39922	5289	0	1	?	0.883	?	0
Random Forest	951	461	39461	4338	0.18	0.988	0.674	0.894	0.284	0.422
Naïve Bayes	1691	1992	37930	3598	0.32	0.95	0.459	0.876	0.377	0.551
Logistic Regre...	945	503	39419	4344	0.179	0.987	0.653	0.893	0.281	0.42
kNN	1131	1019	38903	4158	0.214	0.974	0.526	0.885	0.304	0.456

En cuanto a la curva ROC, vemos que todos los algoritmos tienen mucho margen de mejora.

Sí que se puede divisar que hay una curva cuya evolución se distingue mucho del resto, la de kNN. Vemos como todas evolucionan más rápido y alto para intentar alcanzar ese vértice superior izquierdo, pero, kNN sigue una progresión más lineal y estancada. Su rendimiento, como hemos dicho antes, no parece estar siendo el óptimo con un AUC de 0.688.



Como hemos explicado en el apartado anterior, para hacer que k-NN funcione con atributos categóricos, hemos tenido que transformarlos en valores numéricos, lo que implica asignar números a las categorías. Sin embargo, la forma en que asignamos esos números puede influir

significativamente en el rendimiento del algoritmo. Es decir, la elección de cómo asignamos los números a las categorías puede afectar la forma en que k-NN percibe las distancias entre los puntos en el espacio de atributos. Es posible que, en este caso, esa asignación no haya sido la más adecuada.

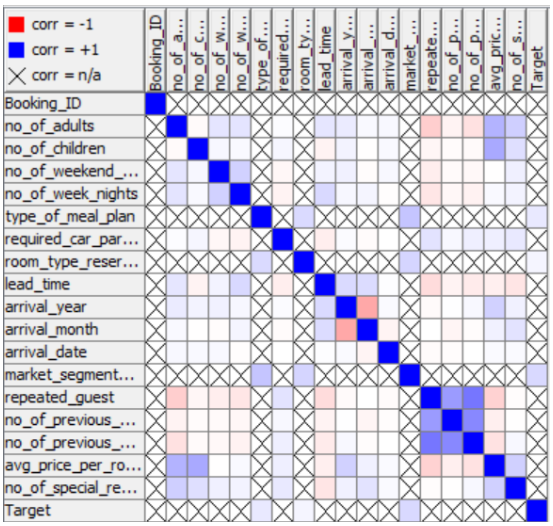
Espero que, una vez equilibrada la clase, todos los AUCs crezcan y se puedan sacar conclusiones más precisas sobre la razón del comportamiento de cada uno.

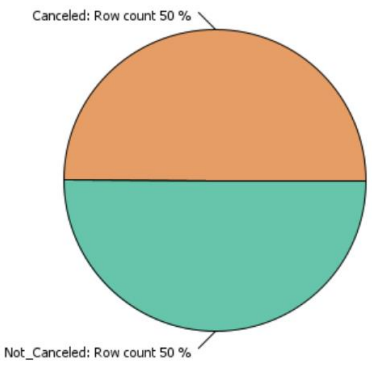
4 CONFIGURACIÓN DE ALGORITMOS

4.1 HOTEL REGISTRATION

Normalmente, lo primero que haríamos sería eliminar los *missing values* de nuestra base de datos. Es decir, aquellos datos cuyo valor es *unknown* o '?' y, por tanto, perjudiquen a la predicción. Sin embargo, nos encontramos con que la base de datos Hotel Registration no tiene ningún valor desconocido.

Lo mismo pasa con la correlación entre atributos. He probado a utilizar los nodos *Linear Correlation* y *Correlation Filter* para eliminar atributos redundantes, pero no existía ningún caso en el que dos variables se correlacionasen más de un 70%. Por tanto, también nos saltamos este paso.



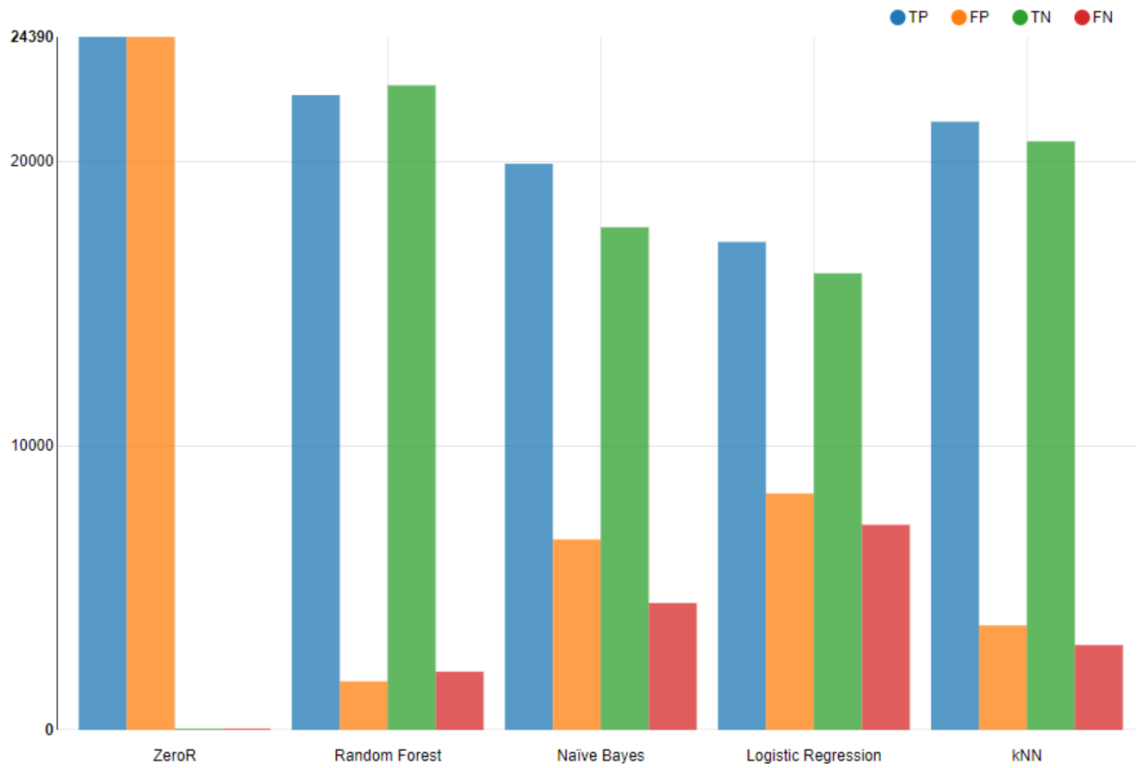


Procedemos entonces a algo que sí sabemos que necesita esta base de datos: equilibrio entre clases. Utilizaremos una técnica *oversampling* para aumentar la cantidad de ejemplos de la clase minoritaria *Canceled*. Para ello añadimos el nodo SMOTE que crea nuevas instancias 'artificiales' a partir de las existentes.

Este gráfico circular a la izquierda representa el nuevo balanceo perfecto ahora existente en nuestros datos, y con el que vamos a trabajar en adelante.

También tendremos en cuenta que para un óptimo funcionamiento de Naïve Bayes es conveniente discretizar la base de datos de entrada. Para ello añadimos el nodo CAIM Binner.

Analizando los resultados obtenidos tras tomar estas medidas mencionadas para un mejor preprocesado de los datos, nos damos cuenta de que todos los algoritmos han mejorado su rendimiento.



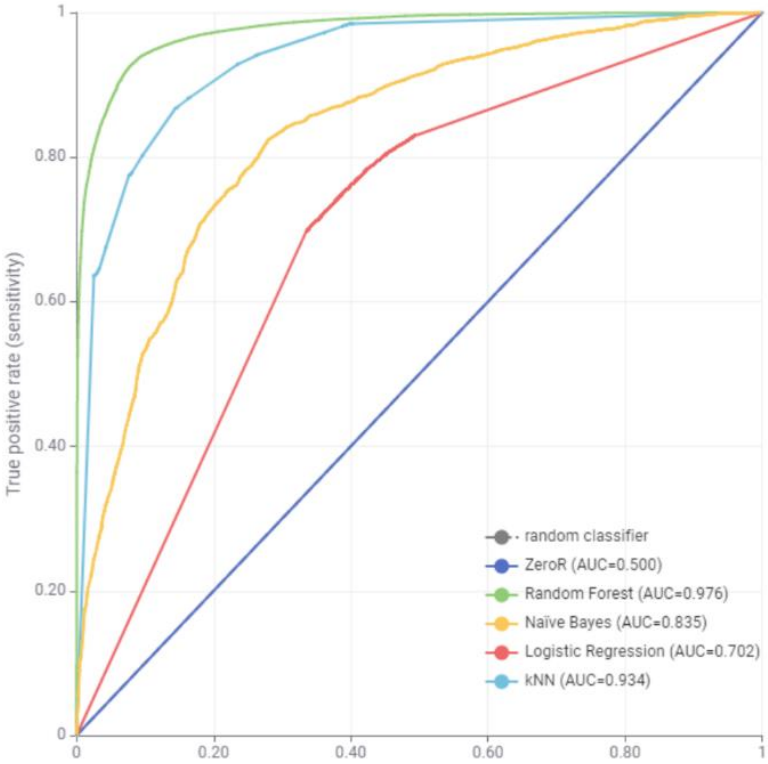
Todas las columnas azules han comenzado a ascender, lo que significa que ahora nuestros algoritmos predicen mucho mejor los casos *Canceled*.

Algoritmos como Naïve Bayes, que antes era el que ocupaba el último puesto, después de balanceo y discretización, ha mejorado considerablemente superando ahora a la Regresión Logística. La cual sorprende en este caso pues, no sólo no se ha mantenido, sino que ha empeorado.

El único cambio que ha sufrido el algoritmo de Regresión Logística ha sido la aplicación de SMOTE. Sabiendo los posibles efectos secundarios que conlleva esta técnica, es normal pensar que ha podido introducir instancias no tan convenientes que, o bien se asemejan demasiado a las originales y resultan redundantes, o bien se tratan de ruido que puede confundir a los algoritmos. No obstante, analizando la importante mejora del resto, descarto estas posibilidades, pues habría afectado a todos.

Mi siguiente suposición, pues, es que SMOTE haya introducido relaciones no lineales entre las características que haya perjudicado a la Regresión Logística debido a su capacidad limitada de sólo percibir aquellas relaciones lineales entre las características y la variable objetivo. El AUC empeorado a 0.702 indica que a lo mejor sería conveniente otro tipo de medidas específicas para este algoritmo.

Random Forest y kNN, una vez más, presentan un rendimiento envidiable mejorando sus ya resultados anteriores buenos.

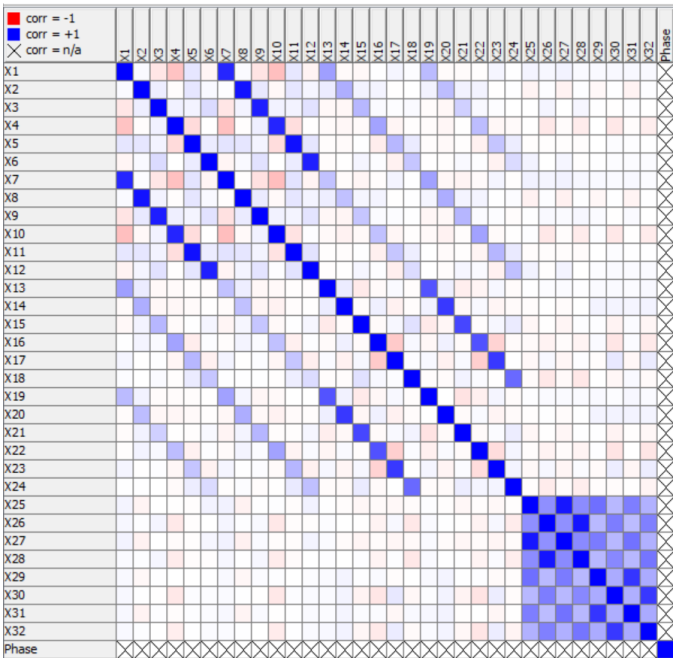


4.2 IDENTIFICACIÓN DE GESTOS

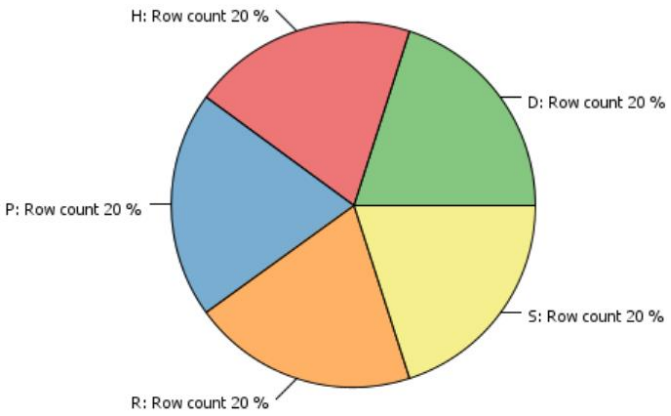
En este problema en concreto, no tenemos *missing values* entre nuestros datos, por lo que no habrá que hacer ningún tipo de tratamiento al respecto.

Al ser una base de datos con tantos atributos, me gustaría reducir aquellos que no sean tan importantes o que resulten redundantes para hacer el proceso más eficiente y acotado.

He utilizado los nodos Linear Correlation y Correlation Filter, configurando un umbral de 0,8, y he logrado filtrar 8 columnas y quedarme con un modelo de 24 atributos, eliminando aquellas variables de coordenadas y de velocidad que hacían referencia a las muñecas, pues no son necesarias teniendo las de las manos.



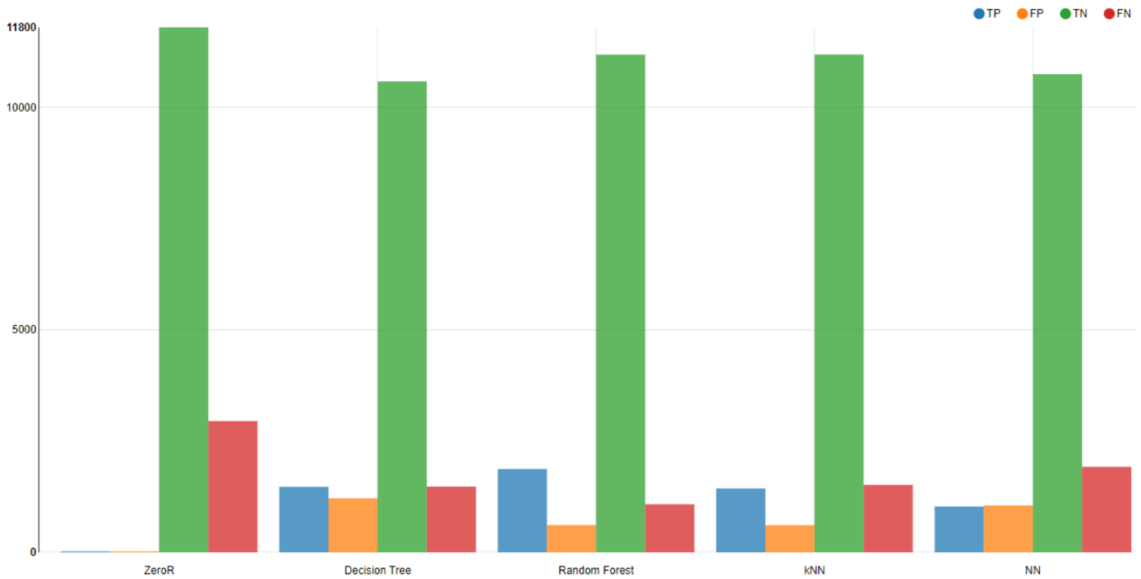
Además, en un principio pensé en balancear todas las clases de manera equitativa con SMOTE esperando un mejor rendimiento.



Sin embargo, tras un primer intento de predicción los resultados no han sido los esperados.

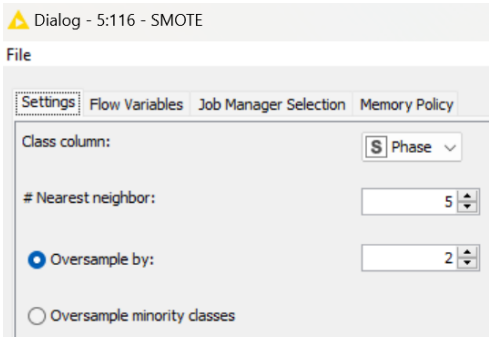
Row ID	TP	FP	TN	FN	TPR	TNR	PPV	Accuracy	F1-score	G-mean
ZeroR	0	0	11800	2950	0	1	?	0.8	?	0
Decision Tree	1472	1215	10585	1478	0.499	0.897	0.548	0.817	0.522	0.669
Random Forest	1871	611	11189	1079	0.634	0.948	0.754	0.885	0.689	0.775
kNN	1493	600	11200	1457	0.506	0.949	0.713	0.861	0.592	0.693
NN	1025	1053	10747	1925	0.347	0.911	0.493	0.798	0.408	0.563

Todas nuestras medidas *Accuracy* han mejorado pero *F1-score* y *G-mean* han empeorado. Al darle más frecuencia a estados como 'H' o 'R', los algoritmos clasifican mejor los True Negative, que son todos estos casos que antes eran minoritarios con respecto a 'S', el caso positivo. Sin embargo, nuestros True Positives caen por la misma razón, equilibrando todas las clases, hemos aminorado el porcentaje de casos positivos 'S'.



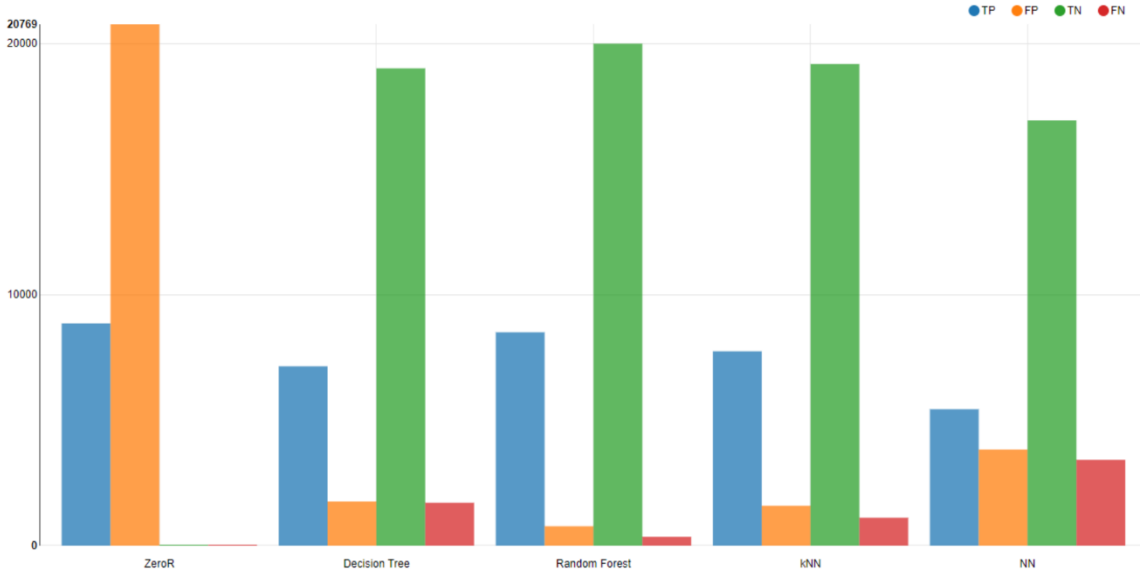
Esto explica por qué nuestro Accuracy es mejor, ya que han aumentado los verdaderos negativos. Pero nuestro F1-score ha empeorado, pues es la medida que más tiene en cuenta los verdaderos positivos. Y el G-mean también ha caído debido a que han empeorado tanto la especificidad (TNR = TN / (TN + FP)) como la sensibilidad (TPR = TP / (TP + FN)).

Sabiendo que estos resultados no son demasiado buenos y puedo obtener mejores. He decidido cambiar la configuración de mi SMOTE y en vez de equilibrar todas las clases, duplicar las instancias de cada clase para tener más datos con los que trabajar y, por tanto, mejorar la precisión de todos los algoritmos.



Volviendo así a la proporción de las clases que teníamos originalmente con un 29,88% de casos positivos.

Row ID	TP	FP	TN	FN	TPR	TNR	PPV	Accuracy	F1-score	G-mean
ZeroR	8850	20769	0	0	1	0	0.299	0.299	0.46	0
Decision Tree	7142	1757	19012	1708	0.807	0.915	0.803	0.883	0.805	0.859
Random Forest	8499	772	19997	351	0.96	0.963	0.917	0.962	0.938	0.962
kNN	7741	1585	19184	1109	0.875	0.924	0.83	0.909	0.852	0.899
NN	5433	3828	16941	3417	0.614	0.816	0.587	0.755	0.6	0.708



Los resultados ahora son notablemente mejores. Vemos que nuestros Verdaderos Positivos crecen de forma relevante al tener más casos de 'S' con los que trabajar.

Random Forest, una vez más, presenta los mejores índices de precisión seguido de kNN y el árbol de decisión. Al igual que en los casos anteriores, Random Forest tiene tan buena tasa de acierto gracias al muestreo aleatorio de todos sus árboles de decisión que lo hace resistente al sobreajuste.

También puedo entender que kNN le haya seguido con los segundos mejores resultados porque aquellas imágenes que representen el mismo estado, ya sea S, D, P, R o H, tendrán unas características de posición y velocidad de manos muy similares entre ellas.

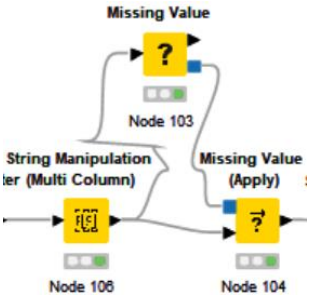
Es evidente que el árbol de decisión ha obtenido peores resultados que el Random Forest al tratarse de uno sólo y no de 100. No obstante, vemos como un solo árbol resulta más eficiente al construirse más rápido que el Random Forest y, aunque no es igual de efectivo, tiene valores bastante competitivos.

Las redes neuronales han mejorado con respecto al intento anterior, pero creo que aún no estamos exprimiendo todo su potencial. Tal vez añadiendo más capas y más nodos por cada capa obtendríamos mejores resultados. Sin embargo, esto a lo mejor requeriría demasiado

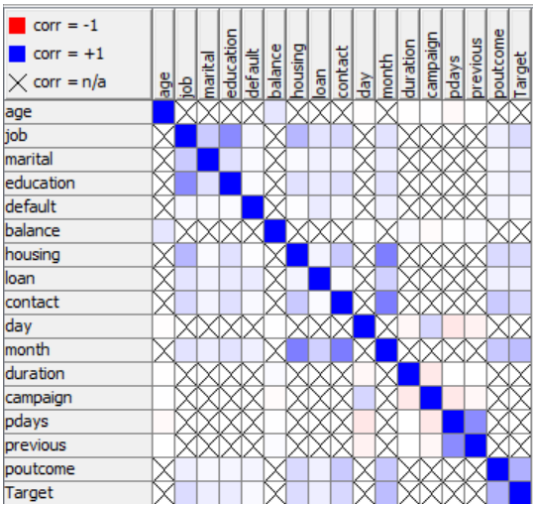
tiempo y potencia para mi portátil. Esto es uno de los inconvenientes de las redes neuronales, no son tan eficientes como otros modelos.

4.3 BANK MARKETING

Al echar un vistazo en la base de datos original también nos damos cuenta de que existen muchos *missing values*. Muchos de ellos en la columna *contact*, la cual simplemente especifica si el contacto fue por teléfono móvil, fijo o desconocido. Asumimos que este dato no es demasiado relevante para nuestra predicción dado que, además, la mayoría de los valores de esta variable son desconocidos. Por tanto, eliminaremos este atributo. Con el resto de *missing values* que encontramos en *education*, *poutcome*, *job*, etc, aplicaremos el nodo Missing Value para convertirlos en el valor más frecuente de su clase.



También me he interesado por la correlación lineal, para ver si pueden existir dos variables tan relacionadas entre sí que resulten redundantes. No obstante, utilizando el nodo Linear Correlation me he topado con que no hay dos variables que superen un umbral de 0,7. Por lo que no eliminaré ningún atributo más.

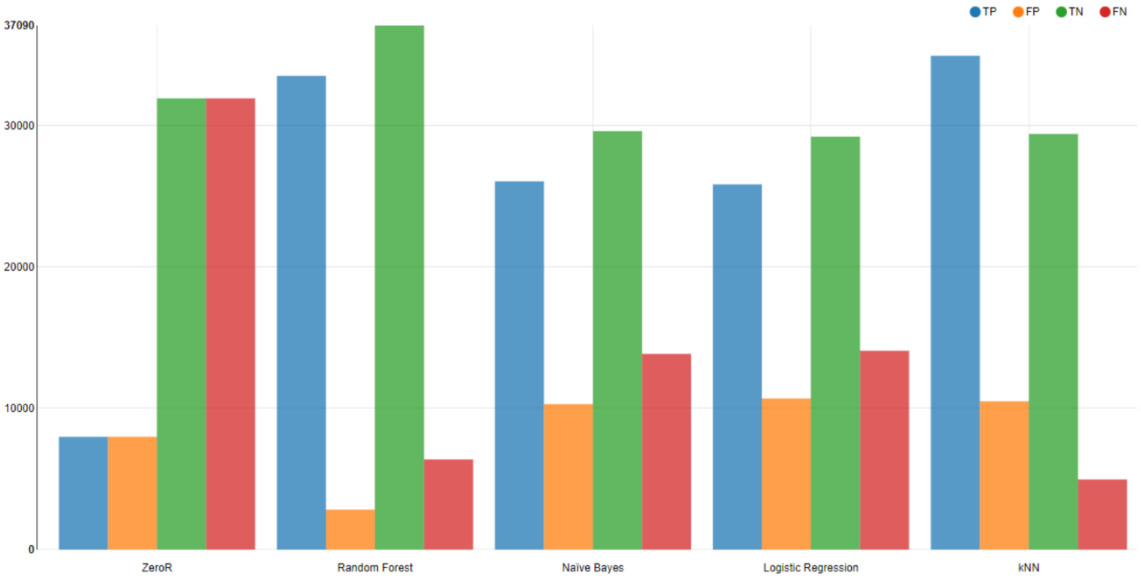


Para abordar el desbalanceo en este problema utilizaremos el nodo SMOTE, al igual que en Hotel Registration, para aumentar la cantidad de ejemplos de la clase *yes* y equilibrarla con los *no*.

Además, al estar utilizando el algoritmo Naïve Bayes también deberemos discretizar los valores antes de su predicción con CAIM Binner.

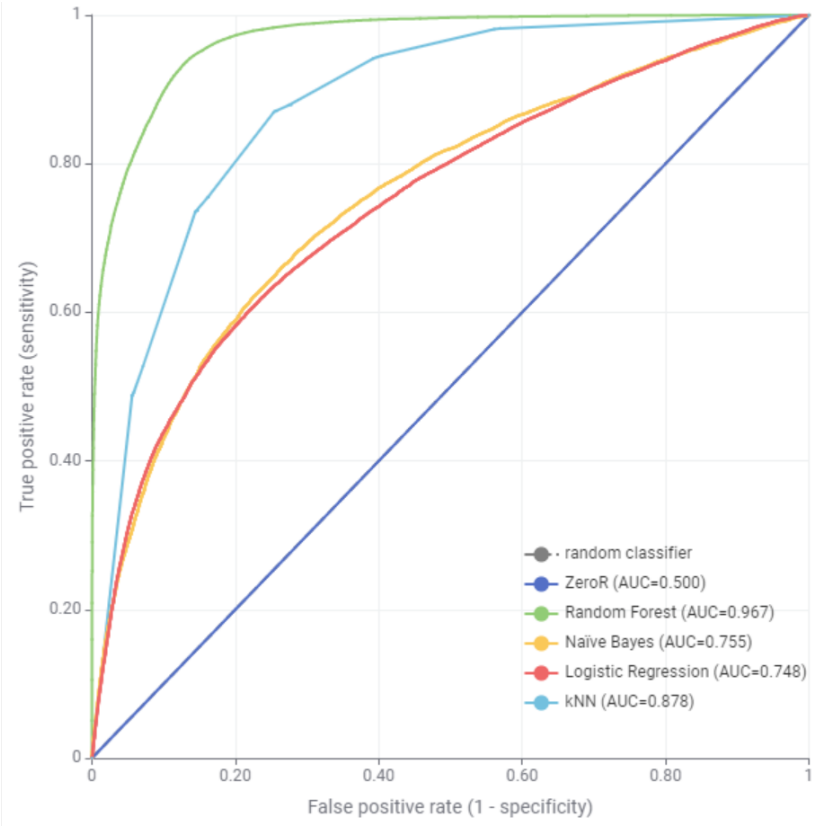
Estos son los resultados obtenidos:

Row ID	TP	FP	TN	FN	TPR	TNR	PPV	Accuracy	F1-score	G-mean
ZeroR	7984	7986	31936	31938	0.2	0.8	0.5	0.5	0.286	0.4
Random Forest	33531	2832	37090	6391	0.84	0.929	0.922	0.884	0.879	0.883
Naïve Bayes	26067	10301	29621	13855	0.653	0.742	0.717	0.697	0.683	0.696
Logistic Regre...	25848	10695	29227	14074	0.647	0.732	0.707	0.69	0.676	0.688
kNN	34953	10501	29421	4969	0.876	0.737	0.769	0.806	0.819	0.803



Después de haber balanceado la clase objetivo observamos que, en comparación con nuestro gráfico de barras anterior, empezamos a ver un poco más de color azul, es decir, de casos TP, y eso queda reflejado en nuestros valores mejorados de precisión y F1-score.

Analizando la tabla de medidas, el gráfico de barras y la curva ROC podemos evaluar la evolución de cada algoritmo con respecto a nuestra primera predicción.



Random Forest demuestra ser un algoritmo que rinde como el mejor tanto en casos desbalanceados como tras haber equilibrado las clases con SMOTE. Es sin duda el que más se ha beneficiado de este preprocesado algo más profundo respondiendo con un AUC cercano a 1. Al equilibrar las clases, el modelo ha aprendido de manera más efectiva tanto las instancias positivas como negativas.

Como ya sabemos, Naïve Bayes asume independencia condicional entre las características, así que la aplicación de SMOTE puede no haber sido beneficiosa, ya que las instancias artificiales pueden introducir correlaciones no lineales que contradicen las suposiciones de independencia condicional. No obstante, la discretización de datos a través de CAIM Binner ha simplificado el modelo y mejorado la interpretabilidad, pero el AUC de 0.755 sugiere que este algoritmo no se adapta óptimamente a la nueva estructura de datos.

Algo que no nos sorprende tras haber analizado Hotel Registration es que el AUC de la Regresión Logística ha ido a peor. Al igual que le ha pasado a Naïve Bayes, es posible que SMOTE haya introducido relaciones no lineales entre las características y esto haya perjudicado a Regresión Logística.

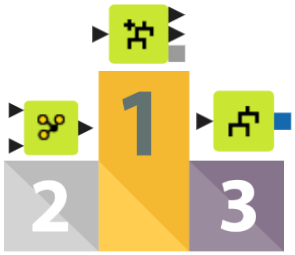
Como habíamos visto anteriormente, kNN era uno de los más perjudicados por el desbalanceo de la clase. Tras el preprocesado, se ha beneficiado significativamente de SMOTE, pues requería de suficientes ejemplos de la clase minoritaria para funcionar eficazmente. El AUC mejorado a 0.878 refleja la capacidad de kNN para adaptarse a la nueva estructura de datos y capturar relaciones no lineales.

5 ANÁLISIS DE RESULTADOS

Haciendo un ranking de los algoritmos para cada uno de los problemas antes y después del preprocesado obtenemos los siguientes puestos:

Antes de preprocesar			**Después de preprocesar**		
Hotel Registr.	Ident. Gestos	Bank Market.	Hotel Registr.	Ident. Gestos	Bank Market.
Rand. Forest	Rand. Forest	Rand. Forest	Rand. Forest	Rand. Forest	Rand. Forest
k-NN	k-NN	Regr. Logíst.	k-NN	k-NN	k-NN
Regr. Logíst.	Decision Tree	Naïve Bayes	Naïve Bayes	Decision Tree	Naïve Bayes
Naïve Bayes	Redes Neuro.	k-NN	Regr. Logíst.	Redes Neuro.	Regr. Logíst.

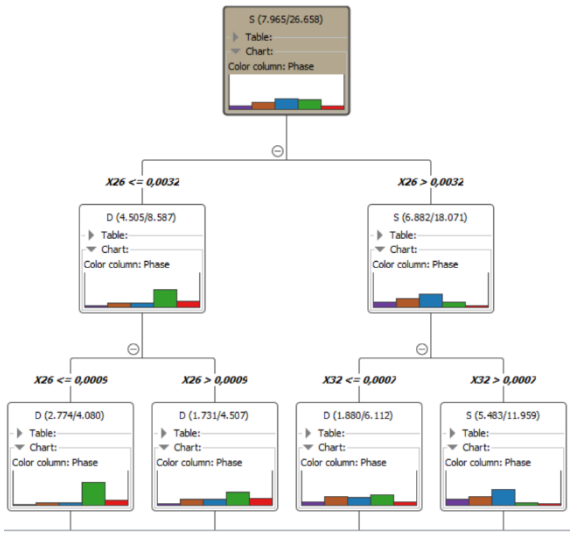
A partir de este ránking podríamos llegar a la conclusión del siguiente pódium:



- El primer puesto es claramente para **Random Forest**. Al ser una técnica *ensemble*, construye una gran cantidad de árboles de decisión mediante muestreo aleatorio que contempla una variedad de escenarios de predicción posibles y escoge los que más le benefician. Random Forest presenta numerosas ventajas que le hacen destacar frente al resto:
  - Resistencia frente al desbalanceo: algunos de sus árboles pueden centrarse en clases minoritarias y capturar patrones importantes.
  - Reducción del sobreajuste: si un árbol tiende a sobreajustarse, la contribución de ese árbol se atenúa por la combinación con otros árboles.
  - Mediante el criterio Information Gain Ratio, los atributos más importantes se pueden evaluar.

  Gracias a todas estas ventajas, hemos conseguido medidas de precisión muy favorables en nuestros problemas, sobre todo tras el preprocesado conveniente. No obstante, hemos de añadir que todas estas ventajas no vienen sin algún inconveniente y este es que Random Forest es el que más ha tardado en ejecutarse con respecto al resto. Por tanto, imagino que, para conjuntos de datos aún más grandes, aún siendo el más efectivo, puede que no sea el más eficiente.
- El segundo puesto es para **k-Nearest Neighbor**. k-NN tiene la capacidad de capturar relaciones no lineales en los datos y, por el rendimiento que ha obtenido k-NN en comparación con Regresión Logística por lo general, vemos que en estos problemas en concreto parece que es más probable que las variables guarden relaciones no lineales entre ellas, y podemos entender por qué uno ha ejercido tan bien y el otro algo peor. Además, k-NN es adaptable al tamaño del conjunto de datos, es decir, no necesita una gran cantidad de datos para funcionar bien, lo que hace que tuviera un buen rendimiento tanto antes como después de aplicar SMOTE. Pero k-NN también ha presentado sus limitaciones. Como en Bank Marketing antes de procesar: un problema muy desbalanceado en el que parece que k-NN no ha encontrado suficientes 'vecinos' cercanos a la clase minoritaria.
- El tercer puesto es algo complicado elegirlo debido a que, al utilizar en uno de los problemas, algoritmos diferentes, es más difícil medir su rendimiento con respecto al resto. No obstante, he decidido darle el bronce al **Árbol de Decisión**. Creo que se merece estar en el podio puesto que, a pesar de ser uno de los modelos más simples, también demuestra ser uno de los más eficaces, con unos índices y resultados que

sorprenden. En el problema de la Identificación de Signos ha sido un buen competidos frente a algoritmos más complejos como Random Forest o k-NN. Gracias al criterio Information Gain Ratio, sabe decidir cuáles son aquellos atributos que conviene elegir para los nodos de decisión. Además, es muy fácil interpretar las decisiones que ha ido tomando el modelo en cada nodo.



La cuchara de palo se la han llevado Naïve Bayes, Regresión Logística y las Redes Neuronales.

Como hemos ido comentando a lo largo de los análisis de resultados de cada problema, parece que **Naïve Bayes**, al suponer independencia entre variables, y **Regresión Logística**, al suponer una relación lineal entre las variables y el objetivo, no han sabido captar los patrones de esta base de datos y no han acertado del todo. Tal vez ajustando mejor la base de datos y sus hiperparámetros nos darían mejores resultados de los que hemos obtenido. No obstante, dudo que obtengan, en estos problemas en concreto, mejores resultados que los algoritmos en el podio.

En cuando a las **Redes Neuronales**, es posible que la base de datos no haya sido lo suficientemente grande como para capturar patrones determinados para una mejor predicción, lo que le ha llevado a un importante sobreajuste. Sin embargo, considero que haciendo un SMOTE aún mayor, la base de datos se podría ver corrompida por demasiadas instancias artificiales y esto podría añadirle mucho ruido. Pensando que podía ser por los hiperparámetros, cambié la configuración y especifiqué cinco capas ocultas en vez de una. No obstante, los resultados fueron aún peores. Es de suponer que, a lo mejor sería aumentando el número de iteraciones la forma de obtener mejores resultados, pero esto haría el proceso de ejecución mucho más lento y al algoritmo menos eficiente.

6 INTERPRETACIÓN DE LOS DATOS

Desde el principio hemos reconocido que Hotel Registration y Bank Marketing son dos problemas del mismo tipo: variables nominales y numéricas, clasificación binaria, desbalanceo de clases… Y durante la ejecución de los algoritmos de clasificación en ambas, también hemos encontrado varias similitudes más, debido a que los algoritmos utilizados han reaccionado prácticamente de la misma manera para ambas clases.

Mis suposiciones son que, en ambos casos, la clase objetivo no sigue una relación lineal con respecto a las variables. Es decir, ni todos los clientes que han cancelado una reserva, ni todos los clientes que han contratado el nuevo producto del banco siguen un patrón lineal. Por ello, k-NN y Random Forest son mejores para captar esas relaciones no lineales.

Aunque no hemos podido eliminar características redundantes en ninguno de los dos problemas, sí que podemos hacernos una idea de cuáles son las más importantes. Pues serán las que Random Forest ha tenido más veces en consideración en la raíz a la hora de construir sus árboles de decisión.

En el caso de Hotel Registration, se trata de *lead_time*, que es el número de días desde que se hace la reserva hasta el día de entrada al hotel. No es de extrañar que sea un factor determinante pues, aquellos que han hecho la reserva con más tiempo, pueden encontrarse con más contratiempos hasta la fecha y, por tanto, tener más razones para cancelar. El otro atributo también muy importante es el número de reservas anteriores no canceladas. El algoritmo detecta que aquellos clientes poco propensos a cancelar en el pasado, tampoco lo harán en el futuro.

Row ID	#splits (level 0)	#splits (level 1)	#splits (level 2)	#candidates (level 0)	#candidates (level 1)	#candidates (level 2)
no_of_adults	0	5	5	27	49	89
no_of_children	0	0	0	19	34	80
no_of_weeke...	0	2	6	27	47	73
no_of_week_...	0	2	14	19	52	85
type_of_meal...	0	4	10	29	43	84
required_car_...	9	15	21	20	46	93
room_type_r...	0	0	11	22	48	93
lead_time	20	36	38	20	51	89
arrival_year	2	10	14	21	53	82
arrival_month	6	6	33	26	50	104
arrival_date	0	0	12	23	31	86
market_segm...	15	27	27	26	57	82
repeated_guest	16	22	24	28	47	83
no_of_previo...	9	12	22	27	44	83
no_of_previo...	20	17	23	24	47	94
avg_price_pe...	1	17	28	23	55	107
no_of_special...	2	10	22	19	46	73

En el caso de Bank Marketing, el atributo que con diferencia más se tiene en cuenta es el mes en el que se le ofrece el producto al cliente. Esto puede resultarle muy interesante al banco pues, estudiando cuál es el mes en el que dicen más que sí, pueden reforzar la campaña en esas fechas específicas.

Row ID	#splits (level 0)	#splits (level 1)	#splits (level 2)	#candidates (level 0)	#candidates (level 1)	#candidates (level 2)
age	5	6	34	23	36	110
job	10	13	22	32	57	106
marital	0	2	20	27	62	119
education	1	7	10	30	62	109
default	0	0	0	23	43	101
balance	9	29	42	22	61	99
housing	10	24	69	28	45	115
loan	3	7	13	25	51	114
day	3	19	37	28	59	109
month	32	29	62	32	36	86
campaign	19	38	56	26	59	93
pdays	15	33	57	28	57	119
previous	18	33	47	26	64	109
poutcome	0	9	19	25	58	105

Como también hemos utilizado Random Forest para Identificación de Gestos, y también ha sido el que mejor ha rendido. Vamos a echarle un vistazo a sus estadísticas para ver qué variables ha considerado más relevantes.

La que más se ha tenido en cuenta ha sido X26 que corresponde con la velocidad escalar de la mano derecha. Tiene sentido pues, lo más probable es que las personas en los vídeos representando los gestos sean diestros y, sea con su mano dominante con la que hagan los gestos más diferenciables.

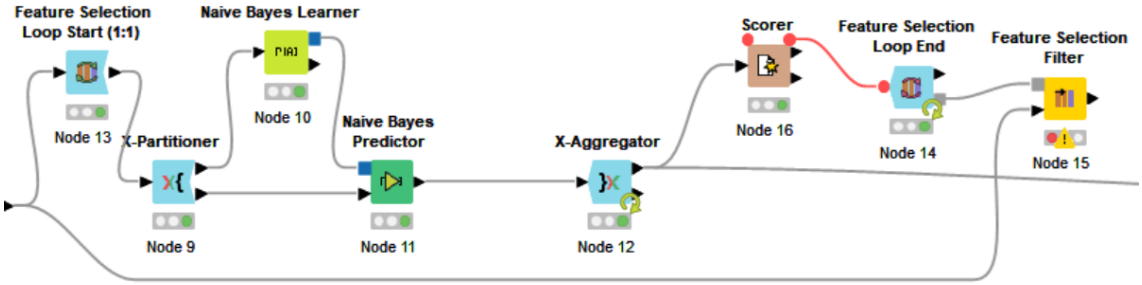
Row ID	#splits (level 0)	#splits (level 1)	#splits (level 2)	#candidates (level 0)	#candidates (level 1)	#candidates (level 2)
X1	2	1	3	20	32	57
X2	12	27	41	21	37	63
X3	0	4	13	18	36	55
X4	1	3	13	22	23	70
X5	12	21	49	14	26	69
X6	2	1	17	18	22	69
X13	0	15	21	13	39	70
X14	0	9	11	20	39	65
X15	0	1	7	22	38	71
X16	0	1	8	16	40	68
X17	6	13	14	19	35	61
X18	0	0	6	15	51	64
X19	0	4	9	19	36	63
X20	0	3	10	16	33	66
X21	0	1	6	8	31	68
X22	2	1	2	15	43	73
X23	8	11	25	18	33	66
X24	1	2	9	17	34	70
X25	7	19	27	10	30	68
X26	19	23	41	19	26	73
X29	7	5	14	17	30	79
X30	5	14	15	10	36	53
X31	2	8	17	14	26	68
X32	14	13	22	19	24	71

7 CONTENIDO ADICIONAL

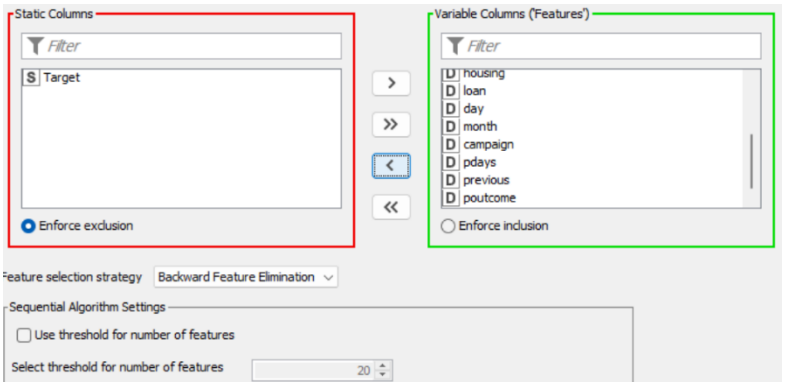
7.1 BANK MARKETING

Como aportación adicional he decidido tratar alguna técnica más con Bank Marketing para intentar mejorar aún más las estadísticas de los algoritmos.

Investigando por el fórum de KNIME, donde todos los usuarios pueden preguntar y resolverse dudas mutuamente, he encontrado estos nodos para la selección de características. Se trata de los nodos Feature Selection Loop Start, Loop End y Filter. Poniéndolos al principio y al final de un algoritmo de clasificación como se ve en la imagen, es capaz de mostrar qué conjunto de variables obtendrán un mejor índice de precisión.



En la configuración de los nodos, he elegido la técnica Backward Feature Elimination, vista en clase, puesto que es adecuada en casos con muchas variables de las cuales se quieren seleccionar sólo las que mejores resultados dan.



Vemos que poniendo a prueba estos nodos para Naïve Bayes, obtenemos una precisión un poco mejor, 0.706, que la anterior, 0.697, quitando los atributos *poutcome*, *job* y *default*.

Accuracy	Nr. of features
0,706	11
0,705	12
0,705	9
0,705	13
0,704	10
0,703	14
0,702	8
0,702	7
0,699	6
0,69	5
0,685	4
0,672	3
0,661	2
0,615	1

S age
S job
S marital
S education
S default
S balance
S housing
S loan
S day
S month
S campaign
S pdays
D previous
S poutcome
S Target

Le he aplicado esta técnica a cada algoritmo para, posteriormente, aplicarle un filtro de columnas a cada uno y poder sacar la mejor precisión posible de todos.

Accuracy	Nr. of features
0,893	8
0,893	2
0,893	6
0,892	11
0,891	14
0,891	9
0,891	10
0,891	12
0,886	5
0,886	11
0,885	10
0,883	3
0,883	3
0,883	2

I	age
S	job
S	marital
S	education
S	default
I	balance
S	housing
S	loan
I	day
S	month
I	campaign
I	pdays
I	previous
S	poutcome
S	Target

Para Random Forest me predecía que, si eliminaba los atributos age, job, education, balance, housing y pdays obtendría una precisión dos centésimas mejor.

Accuracy	Nr. of features
0,893	13
0,893	16
0,893	19
0,893	12
0,893	3
0,893	6
0,893	5
0,893	7
0,893	20
0,893	22
0,893	21
0,893	21
0,893	20

D	age
S	job
S	marital
S	education
S	default
D	balance
S	housing
S	loan
D	day
S	month
D	campaign
D	pdays
D	previous
S	poutcome
S	Target

Para Regresión Logística, filtrando age, job, marital, education, balance y previous llegaría a mejorar la precisión 2 décimas.

Accuracy	Nr. of features
0,893	4
0,893	1
0,893	3
0,893	2
0,892	5
0,892	6
0,891	7
0,891	8
0,891	9
0,889	10
0,888	11
0,887	12
0,887	13
0,885	14

D	age
D	job
D	marital
D	education
D	default
D	balance
D	housing
D	loan
D	day
D	month
D	campaign
D	pdays
D	previous
D	poutcome
S	Target

Y para kNN sólo con las columnas day, month, pdays y poutcome bastaría para obtener un mejor rendimiento y una precisión que ascendería de un 0.806 a 0.893.

Tras haber añadido los Column Filter correspondientes, he ejecutado mi *worfkflow* para comprobar si verdaderamente obtendría mejores resultados, con la idea en mente de que, probablemente no encuentre medidas de precisión tan buenas como me aseguran los nodos Feature Selection, pues son sólo un cálculo aproximado y, probablemente, optimista.

Row ID	I TP	I FP	I TN	I FN	D TPR	D TNR	D PPV	D Accuracy	D F1-score	D G-mean
ZeroR	7984	7986	31936	31938	0.2	0.8	0.5	0.5	0.286	0.4
Random Forest	27101	3471	36451	12821	0.679	0.913	0.886	0.796	0.769	0.787
Naïve Bayes	26541	10338	29584	13381	0.665	0.741	0.72	0.703	0.691	0.702
Logistic Regre...	26680	13336	26586	13242	0.668	0.666	0.667	0.667	0.668	0.667
kNN	31153	2865	37057	8769	0.78	0.928	0.916	0.854	0.843	0.851

Observando nuestra tabla de resultados parece que Naïve Bayes ha cumplido y, aunque no tiene el accuracy que nos prometía el Feature Selection Loop, sí que es mejor que el anterior.

kNN también ha mejorado considerablemente su precisión, teniendo en cuenta que sólo ha utilizado cuatro atributos para esta predicción. Esta medida le ha venido muy bien pues no sólo ha mejorado sino que ahora es incluso más rápido.

Sin embargo, nos sorprende ver que tanto Random Forest como Regresión Logística han empeorado sus índices de precisión. Es posible que el Feature Selection Loop a lo mejor necesitase más iteraciones para hacer una mejor selección de características en estos algoritmos. Sin embargo, es una técnica que le ha llevado bastante tiempo de ejecución a mi portátil y que, a pesar de que ha tenido buenos resultados para Naïve Bayes y kNN, no considero que merezca la pena aplicarlo a Random Forest, sabiendo los buenos resultados que ya obtiene.

Por tanto, dejaremos los nuevos filtros para kNN y Naïve Bayes pero prescindiremos de ellos para Random Forest y Regresión Logística.

Para tratar de mejorar el Random Forest aún más he decidido hacer pruebas cambiando la configuración y viendo los resultados obtenidos para evaluar el rendimiento óptimo:

- 100 árboles, Information Gain Ratio

Row ID	TP	FP	TN	FN	TPR	TNR	PPV	Accuracy	F1-score	G-mean
ZeroR	7984	7986	31936	31938	0.2	0.8	0.5	0.5	0.286	0.4
Random Forest	33531	2832	37090	6391	0.84	0.929	0.922	0.884	0.879	0.883
Naïve Bayes	26067	10301	29621	13855	0.653	0.742	0.717	0.697	0.683	0.696
Logistic Regre...	25848	10695	29227	14074	0.647	0.732	0.707	0.69	0.676	0.688
kNN	34953	10501	29421	4969	0.876	0.737	0.769	0.806	0.819	0.803

- 75 árboles, Information Gain Ratio

Row ID	TP	FP	TN	FN	TPR	TNR	PPV	Accuracy	F1-score	G-mean
ZeroR	7984	7986	31936	31938	0.2	0.8	0.5	0.5	0.286	0.4
Random Forest	33575	2920	37002	6347	0.841	0.927	0.92	0.884	0.879	0.883
Naïve Bayes	26541	10338	29584	13381	0.665	0.741	0.72	0.703	0.691	0.702
Logistic Regre...	26683	13344	26578	13239	0.668	0.666	0.667	0.667	0.668	0.667
kNN	31153	2865	37057	8769	0.78	0.928	0.916	0.854	0.843	0.851

- 125 árboles, Information Gain Ratio

Row ID	TP	FP	TN	FN	TPR	TNR	PPV	Accuracy	F1-score	G-mean
ZeroR	7984	7986	31936	31938	0.2	0.8	0.5	0.5	0.286	0.4
Random Forest	35586	2833	37089	4336	0.891	0.929	0.926	0.91	0.908	0.91
Naïve Bayes	26541	10338	29584	13381	0.665	0.741	0.72	0.703	0.691	0.702
Logistic Regre...	26683	13344	26578	13239	0.668	0.666	0.667	0.667	0.668	0.667
kNN	31153	2865	37057	8769	0.78	0.928	0.916	0.854	0.843	0.851

- 75 árboles, GINI

Row ID	TP	FP	TN	FN	TPR	TNR	PPV	Accuracy	F1-score	G-mean
ZeroR	7984	7986	31936	31938	0.2	0.8	0.5	0.5	0.286	0.4
Random Forest	33541	2851	37071	6381	0.84	0.929	0.922	0.884	0.879	0.883
Naïve Bayes	26541	10338	29584	13381	0.665	0.741	0.72	0.703	0.691	0.702
Logistic Regre...	26683	13344	26578	13239	0.668	0.666	0.667	0.667	0.668	0.667
kNN	31153	2865	37057	8769	0.78	0.928	0.916	0.854	0.843	0.851

- 100 árboles, GINI

Row ID	TP	FP	TN	FN	TPR	TNR	PPV	Accuracy	F1-score	G-mean
ZeroR	7984	7986	31936	31938	0.2	0.8	0.5	0.5	0.286	0.4
Random Forest	35499	2747	37175	4423	0.889	0.931	0.928	0.91	0.908	0.91
Naïve Bayes	26541	10338	29584	13381	0.665	0.741	0.72	0.703	0.691	0.702
Logistic Regre...	26683	13344	26578	13239	0.668	0.666	0.667	0.667	0.668	0.667
kNN	31153	2865	37057	8769	0.78	0.928	0.916	0.854	0.843	0.851

- 125 árboles, GINI

Row ID	TP	FP	TN	FN	TPR	TNR	PPV	Accuracy	F1-score	G-mean
ZeroR	7984	7986	31936	31938	0.2	0.8	0.5	0.5	0.286	0.4
Random Forest	35603	2775	37147	4319	0.892	0.93	0.928	0.911	0.909	0.911
Naïve Bayes	26541	10338	29584	13381	0.665	0.741	0.72	0.703	0.691	0.702
Logistic Regre...	26333	14260	25662	13589	0.66	0.643	0.649	0.651	0.654	0.651
kNN	31153	2865	37057	8769	0.78	0.928	0.916	0.854	0.843	0.851

En la siguiente tabla resumiré mejor los datos que nos interesan para que se vean más claros:

Nº árboles	Criterio	Accuracy	F1-Score	G-mean

100	InfoGain Ratio	0.884	0.879	0.883
75	InfoGain Ratio	0.884	0.879	0.883
125	InfoGain Ratio	0.91	0.908	0.91
100	GINI	0.91	0.908	0.91
75	GINI	0.884	0.879	0.883
125	GINI	0.911	0.909	0.911

Como se puede apreciar, el criterio GINI ha rendido mucho mejor que el que KNIME decide por defecto, Information Gain Ratio. El criterio GINI evalúa qué tan bien una variable divide los datos en clases, o, categorías en un nodo del árbol. Además, no sorprende que, cuantos más árboles tomamos, mejores medidas de precisión obtenemos. Sin embargo, debemos ajustar bien el parámetro del número de árboles pues si construimos muchos, la ejecución puede tomar demasiado tiempo

Al igual que he tratado de mejorar los resultados de Random Forest, viendo cómo ha mejorado el rendimiento de kNN tras la Selección de Características he decidido evaluar también su comportamiento cuando cambiamos el hiperparámetro k a uno más alto, como 10, y a uno más bajo, como 1, lo que sería comúnmente el 1-NN.

- k=5

Row ID	TP	FP	TN	FN	TPR	TNR	PPV	Accuracy	F1-score	G-mean
ZeroR	7984	7986	31936	31938	0.2	0.8	0.5	0.5	0.286	0.4
Random Forest	27101	3471	36451	12821	0.679	0.913	0.886	0.796	0.769	0.787
Naïve Bayes	26541	10338	29584	13381	0.665	0.741	0.72	0.703	0.691	0.702
Logistic Regre...	26333	14260	25662	13589	0.66	0.643	0.649	0.651	0.654	0.651
kNN	31153	2865	37057	8769	0.78	0.928	0.916	0.854	0.843	0.851

- k=10

Row ID	TP	FP	TN	FN	TPR	TNR	PPV	Accuracy	F1-score	G-mean
ZeroR	7984	7986	31936	31938	0.2	0.8	0.5	0.5	0.286	0.4
Random Forest	35499	2747	37175	4423	0.889	0.931	0.928	0.91	0.908	0.91
Naïve Bayes	26541	10338	29584	13381	0.665	0.741	0.72	0.703	0.691	0.702
Logistic Regre...	26683	13344	26578	13239	0.668	0.666	0.667	0.667	0.668	0.667
kNN	30474	2885	37037	9448	0.763	0.928	0.914	0.846	0.832	0.842

Row ID	TP	FP	TN	FN	TPR	TNR	PPV	Accuracy	F1-score	G-mean
ZeroR	7984	7986	31936	31938	0.2	0.8	0.5	0.5	0.286	0.4
Random Forest	35499	2747	37175	4423	0.889	0.931	0.928	0.91	0.908	0.91
Naïve Bayes	26541	10338	29584	13381	0.665	0.741	0.72	0.703	0.691	0.702
Logistic Regre...	26683	13344	26578	13239	0.668	0.666	0.667	0.667	0.668	0.667
kNN	31559	2495	37427	8363	0.791	0.938	0.927	0.864	0.853	0.861

- k=1

En esta tabla más resumida:

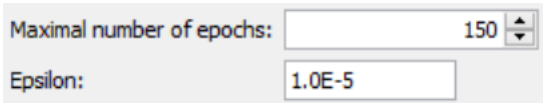
k	**Accuracy**	**F1-Score**	**G-mean**
5	0.854	0.843	0.851
10	0.846	0.832	0.842
1	0.864	0.853	0.861

Sacamos la conclusión de que, no sólo perjudica al algoritmo tener en consideración más vecinos, sino que, cuando mejor rinde, es cuando sólo tiene en cuenta al más cercano. Lo cual también conviene mucho más pues, lo hace mucho más rápido y eficiente.

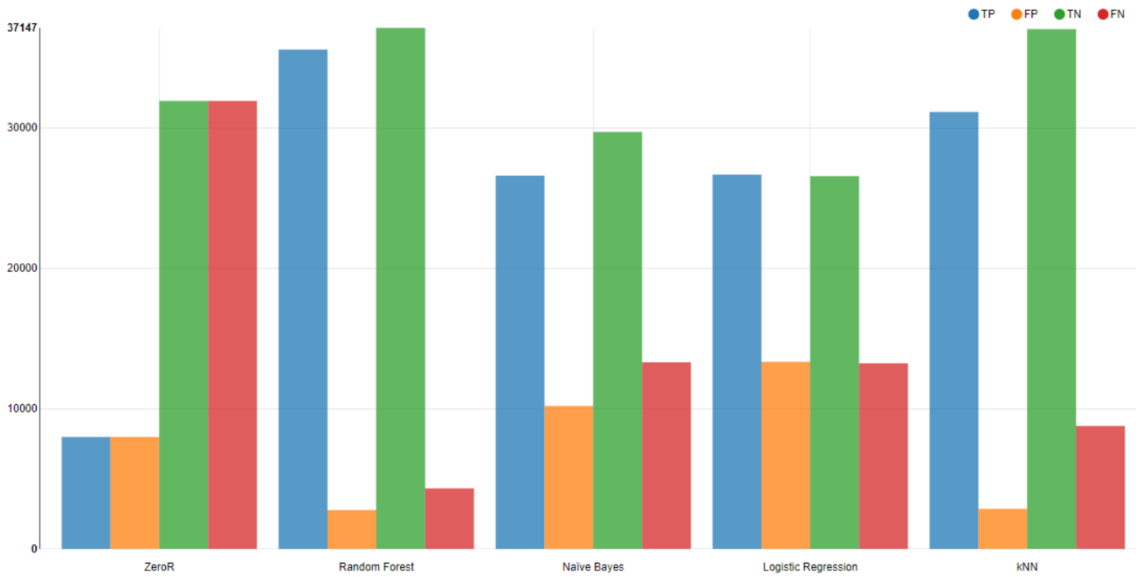
En cuanto a Regresión Logística, he intentado cambiar la medida *epoch* a una más alto. Una *epoch* representa una pasada completa a través de todo el conjunto de entrenamiento durante el proceso de ajuste de los parámetros del modelo.

Row ID	I TP	I FP	I TN	I FN	D TPR	D TNR	D PPV	D Accuracy	D F1-score	D G-mean
ZeroR	7984	7986	31936	31938	0.2	0.8	0.5	0.5	0.286	0.4
Random Forest	35603	2775	37147	4319	0.892	0.93	0.928	0.911	0.909	0.911
Naive Bayes	26614	10194	29728	13308	0.667	0.745	0.723	0.706	0.694	0.705
Logistic Regre...	26680	13336	26586	13242	0.668	0.666	0.667	0.667	0.668	0.667
kNN	31153	2865	37057	8769	0.78	0.928	0.916	0.854	0.843	0.851

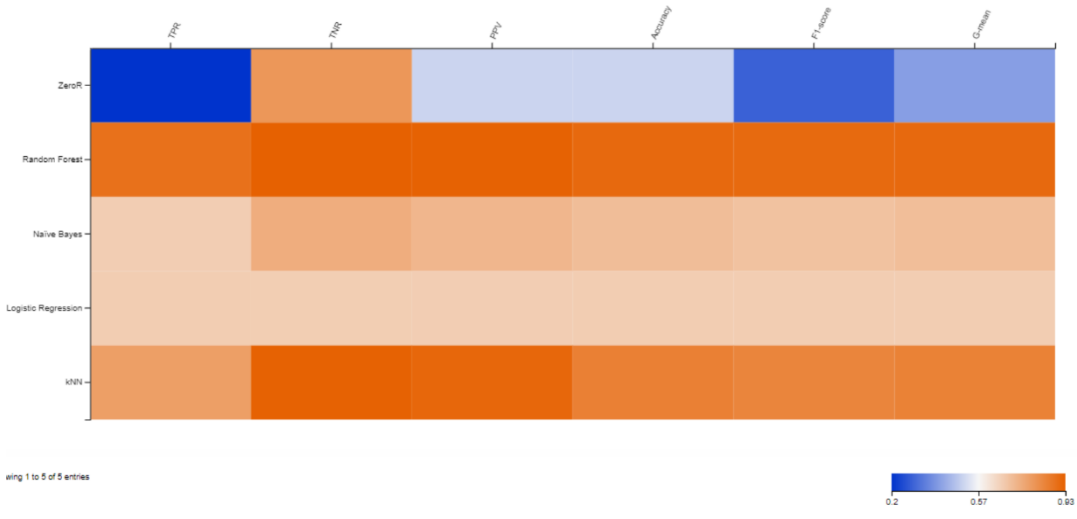
En vez de 100, he configurado 150. Pensaba que así se obtendrían mejores resultados, pero al contrario, he obtenido estadísticas peores. Por tanto, dejaremos la configuración como nos la encontramos por defecto.



Después de todas estas mejoras, Bank Marketing tendría las siguientes estadísticas vistas en el mismo gráfico de barras con el que hemos analizado durante todo el informe.

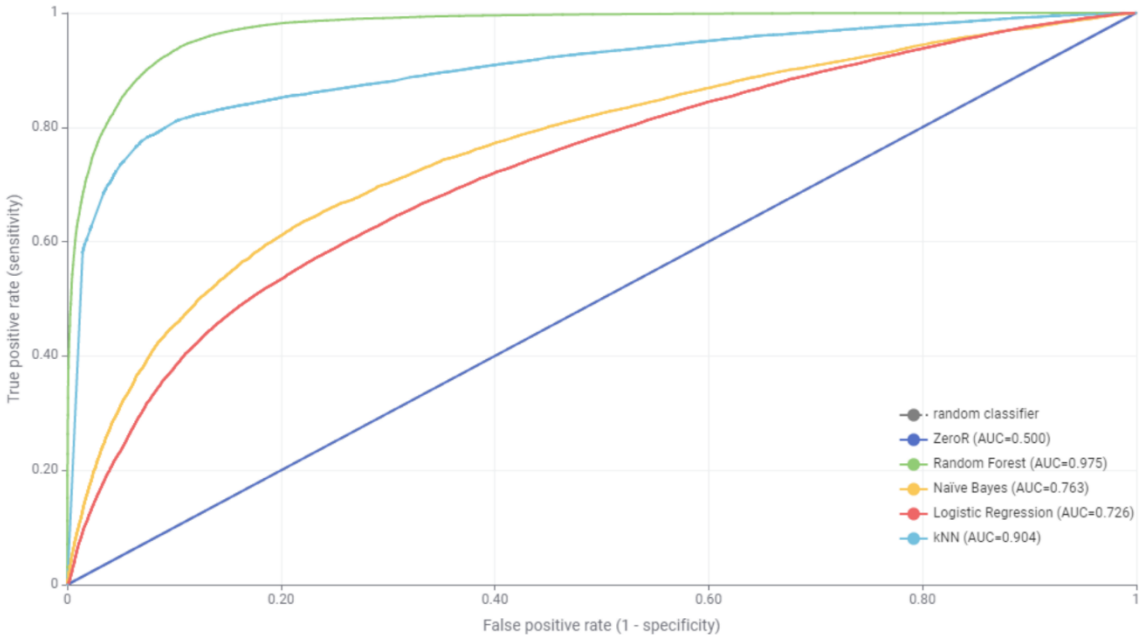


Donde destacan unas columnas azules y verdes bien altas. Otra forma de visualizar estos datos es mediante un *Heat map*.



Se trata de un “mapa de calor” en el que cuánto más naranja sean los datos, más nos acercamos al valor deseado. Se puede apreciar perfectamente que Random Forest es el algoritmo más ‘caluroso’ y que Regresión Logística es el que peor rendimiento ofrece.

Por último, echaremos un vistazo a cómo han quedado nuestras curvas ROC.



La posición de cada curva sigue en el orden analizado anteriormente, pero se puede apreciar como cada una ha mejorado dentro de sus capacidades.

8 BIBLIOGRAFÍA

- https://forum.knime.com/
- https://hub.knime.com/
- https://ccia.ugr.es/~casillas/knime.html
- https://www.kaggle.com/datasets/ahsan81/hotel-reservations-classification-dataset
- https://openml.org/search?type=data&id=4538&sort=runs&status=active
- https://archive-beta.ics.uci.edu/dataset/222/ba%20nk+marketing
- https://nodepit.com/