Inteligencia de Negocio

Práctica 1:
Análisis Predictivo Mediante Clasificación

UNIVERSIDAD
DE GRANADA

Germán José Padua Pleguezuelo

germanpadua@correo.ugr.es

Grupo 1 – 5º DGIIM

Índice:

1. Hotel Registration

1.1. Introducción

El conjunto de datos *Hotel Registration* representa los detalles sobre las reservas de clientes en un hotel. Es cada vez más común cancelar las reservas debido a múltiples factores. Nuestro objetivo en este problema será predecir si un cierto cliente cancelará su reserva o no. La BD consiste en 36275 clientes con 19 atributos cada uno:

- Booking_ID (Nominal)
- no_of_adults (Numérica discreta)
- no_of_children (Numérica discreta)
- no_of_weekend_nights (Numérica discreta)
- no_of_week_nights (Numérica discreta)
- type_of_meal_plan (Nominal)
- required_car_parking_space (Categórica booleana)
- room_type_reserved (Nominal)
- lead_time (Numérica discreta)
- arrival_year (Numérica discreta)
- arrival_month (Numérica discreta)
- arrival_date (Numérica discreta)
- market_segment_type (Nominal)
- repeated_guest (Categórica booleana)
- no_of_previous_cancellations (Numérica discreta)
- no_of_previous_bookings_not_cancelled (Numérica discreta)
- avg_price_per_room (Numérica continua)
- no_of_special_requests (Numérica discreta)
- booking_status (Nominal)

Observamos que es un problema de clasificación binario, sólo hay 2 clases (*Canceled* / *Not_Canceled*), en el que hay atributos de todo tipo. No hay valores perdidos ni hay presente un gran desbalance entre clases, 67% de los datos pertenecen a la clase *Not_Canceled* y el 33% restante a la clase *Canceled*.

1.2. Procesado de datos

El preprocesamiento empleado en este caso es muy simple. En primer lugar, eliminamos la columna *Booking_ID*, ya que solamente indica el identificador de la reserva, por lo que no será útil para los clasificadores.

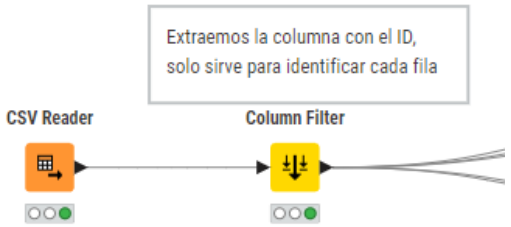

Figura 1.1: Captura de KNIME de un filtro de columna que elimina Booking_ID

Por otro lado, debemos realizar un preprocesamiento específico para dos algoritmos: XGBoost y KNN. En el caso de KNN, es esencial normalizar los atributos para que todos tengan el mismo peso en el algoritmo. Además, en KNIME no he hallado la forma de poder definir una distancia que use la distancia euclídea para los atributos numéricos y la de Hamming para las nominales. Por lo que haremos uso del one-hot encoding (nodo one to many en KNIME) para tener solamente variables numéricas.

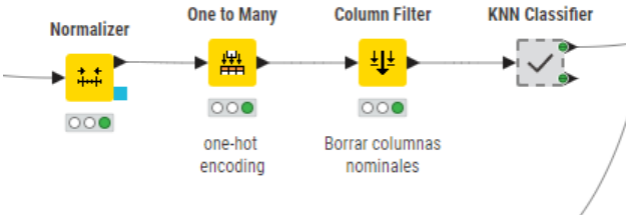

Figura 1.2: Captura de KNIME del preprocesamiento para KNN

El algoritmo XGBoost también requiere que las características sean numéricas, así que aplicaremos lo mismo que para KNN, pero el normalizado no es necesario.

Figura 1.3: Captura de KNIME del preprocesamiento para XGBoost

Por último, también realizaremos un submuestreo y un sobremuestreo para intentar mejorar los resultados de Naive Bayes.

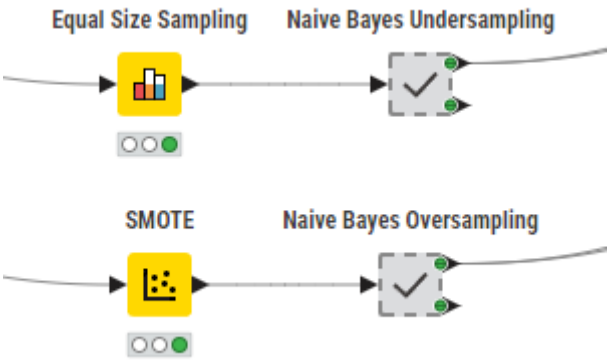

Figura 1.4: Captura de KNIME del preprocesamiento para submuestreo y sobremuestreo

1.3. Resultados obtenidos

1.3.1. Decision Tree

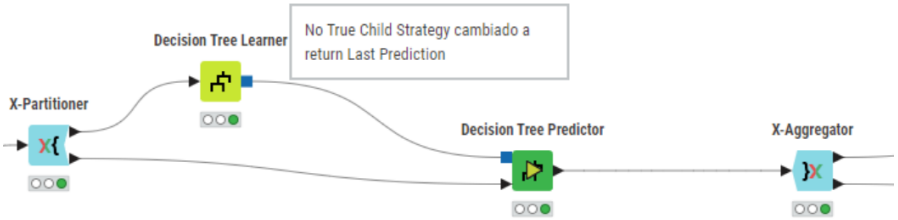

Figura 1.5: Captura de KNIME del flujo de Decision Tree

En el algoritmo de *Decision Tree*, el único ajuste modificado ha sido que se devuelva la última predicción cuando no haya ningún hijo verdadero.

real \ predicho	Canceled	Not_Canceled
Canceled	9424	2461
Not_Canceled	2281	22109

Figura 1.6: Matriz de confusión de Decision Tree

TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
9424	2281	22109	2461	0,805	0,793	0,906	0,799	0,848	0,869	0,850

Figura 1.7: Medidas de Decision Tree

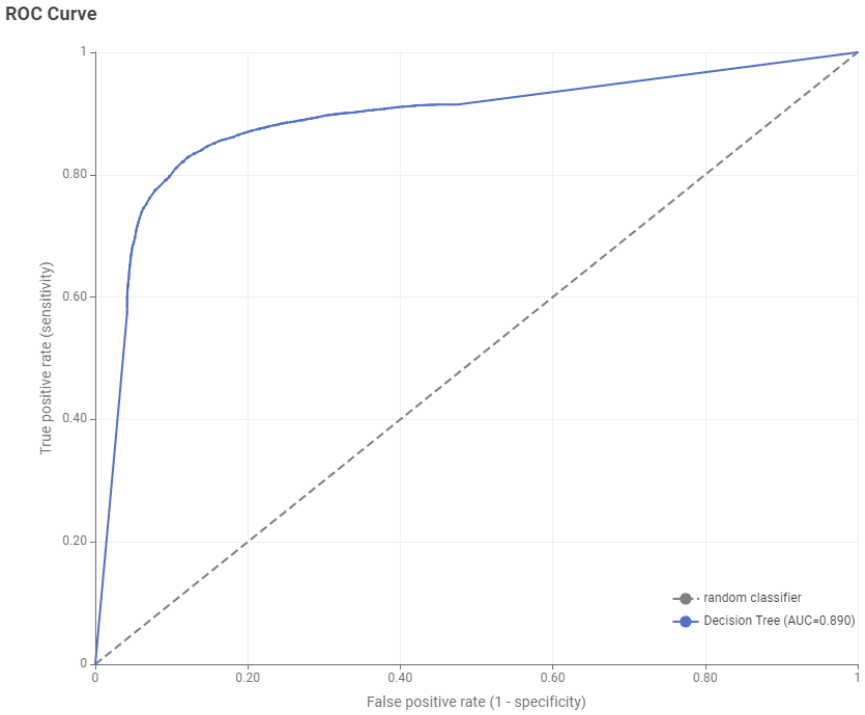

Figura 1.8: Curva ROC de Decision Tree

fold #	Error in %	Size of Test Set	Error Count	Model size
0	12,998	7255	943	1976
1	13,191	7255	957	1920
2	12,695	7255	921	1957
3	13,329	7255	967	1945
4	13,15	7255	954	1940

Figura 1.9: Tabla de errores y tamaño del modelo Decision Tree

1.3.2. Random Forest

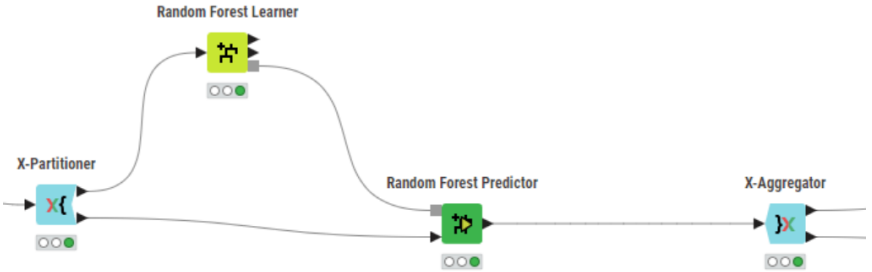

Figura 1.10: Captura de KNIME del flujo de Random Forest

real \ predicho	Canceled	Not_Canceled
Canceled	9634	2251
Not_Canceled	1268	23122

Figura 1.11: Matriz de confusión de Random Forest

TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
9634	1268	23122	2251	0,884	0,811	0,948	0,846	0,877	0,903	0,879

Figura 1.12: Medidas de Random Forest

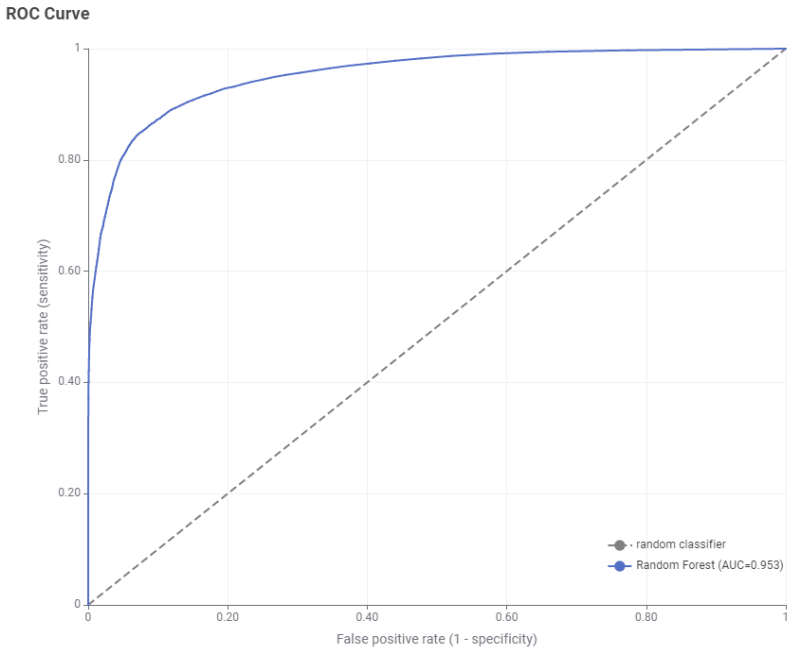

Figura 1.13: Curva ROC de Random Forest

En Random Forest se entrenan 100 árboles de decisión por partición.

fold #	Error in %	Size of Test Set	Error Count
0	9,869	7255	716
1	9,704	7255	704
2	9,773	7255	709
3	9,538	7255	692
4	9,621	7255	698

Figura 1.14: Tabla de errores de Random Forest

1.3.3. XGBoost

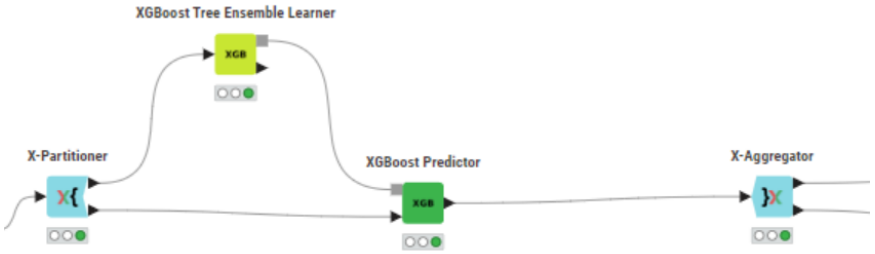

Figura 1.15: Captura de KNIME del flujo de XGBoost

real \ predicho	Canceled	Not_Canceled
Canceled	9535	2350
Not_Canceled	1504	22886

Figura 1.16: Matriz de confusión de XGBoost

TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
9535	1504	22886	2350	0,864	0,802	0,938	0,832	0,868	0,894	0,870

Figura 1.17: Medidas de XGBoost

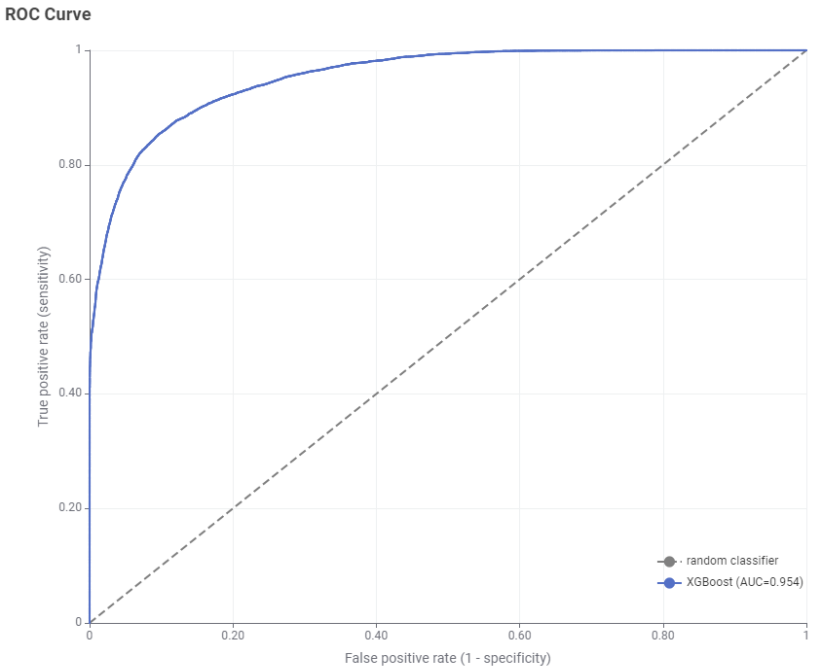

Figura 1.18: Curva ROC de XGBoost

En XGBoost se entrenan 100 árboles de decisión en el conjunto de boosting (por partición).

fold #	Error in %	Size of Test Set	Error Count
0	11,041	7255	801
1	10,365	7255	752
2	10,42	7255	756
3	10,572	7255	767
4	10,724	7255	778

Figura 1.19: Tabla de errores de XGBoost

1.3.4. K Nearest Neighbor

Figura 1.20: Captura de KNIME del flujo de KNN

real \ predicho	Canceled	Not_Canceled
Canceled	8656	3229
Not_Canceled	2267	22123

Figura 1.21: Matriz de confusión de KNN

TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
8656	2267	22123	3229	0,792	0,728	0,907	0,759	0,813	0,848	0,818

Figura 1.22: Medidas de KNN

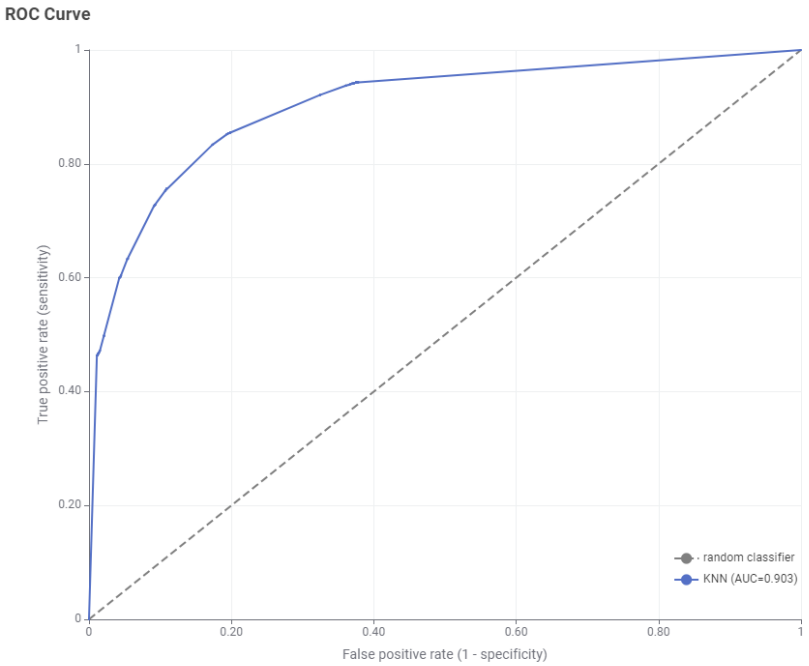

Figura 1.23: Curva ROC de KNN

fold #	Error in %	Size of Test Set	Error Count
0	14,941	7255	1084
1	14,748	7255	1070
2	15,438	7255	1120
3	15,837	7255	1149
4	14,79	7255	1073

Figura 1.24: Tabla de errores de KNN

1.3.5. Naive-Bayes

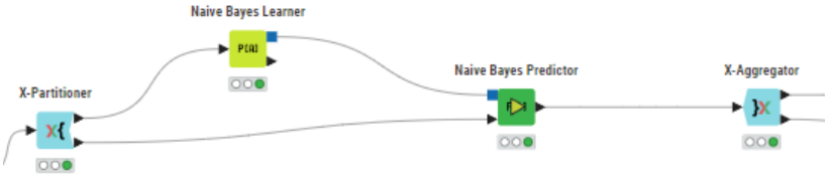

Figura 1.25: Captura de KNIME del flujo de Naive-Bayes

real \ predicho	Canceled	Not_Canceled
Canceled	10767	118
Not_Canceled	21123	12605

Figura 1.26: Matriz de confusión de Naive-Bayes

TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
11767	21123	3267	118	0,358	0,990	0,134	0,526	0,364	0,414	0,562

Figura 1.27: Medidas de Naive-Bayes

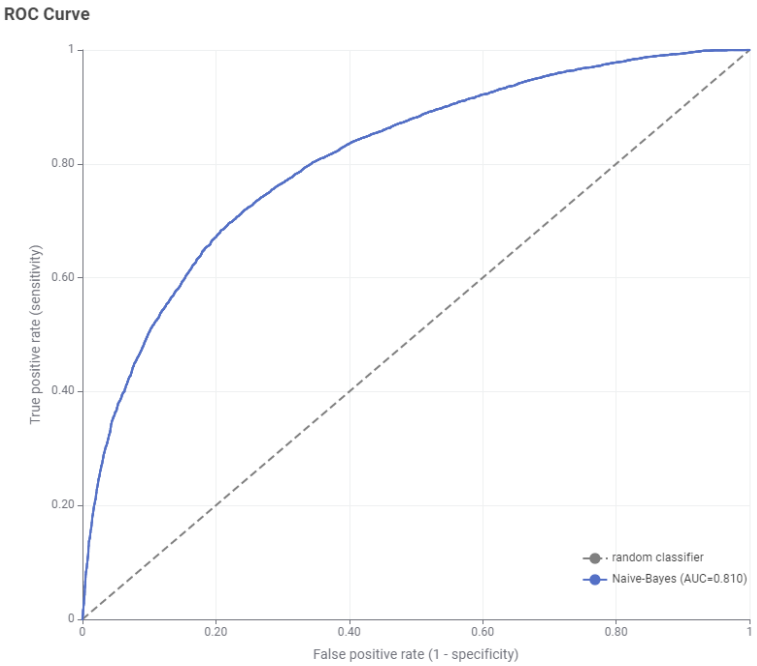

Figura 1.28: Curva ROC de Naive-Bayes

fold #	Error in %	Size of Test Set	Error Count
0	57,464	7255	4169
1	58,291	7255	4229
2	60,041	7255	4356
3	58,732	7255	4261
4	58,249	7255	4226

Figura 1.29: Tabla de errores de Naive-Bayes

1.4. Configuración de algoritmos

1.4.1. Decision Tree

Estudiaremos dos variaciones en los parámetros por defecto: cambiaremos el criterio de selección de variables de Gini a GainRatio y también probaremos a podar el árbol.

TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
9424	2281	22109	2461	0,805	0,793	0,906	0,799	0,848	0,869	0,850

Figura 1.30: Medidas de Decision Tree por defecto

TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
9537	2221	22169	2348	0,811	0,802	0,909	0,807	0,854	0,874	0,856

Figura 1.31: Medidas de Decision Tree usando GainRatio

TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
9065	1606	22784	2820	0,849	0,763	0,934	0,804	0,844	0,878	0,848

Figura 1.32: Medidas de Decision Tree con poda

Observamos que cambiar el criterio de selección a GainRatio parece haber aumentado levemente la capacidad predictiva del modelo, todas las medidas han aumentado un poco.

Por otro lado, la poda del árbol no ha resultado en una mejora considerable del rendimiento. El accuracy aumenta muy poco y el AUC disminuye. La tasa de acierto de la clase negativa aumenta mientras que la de la clase positiva se ve reducida. Así que no se ha solucionado un posible problema de sobreaprendizaje. Tendríamos que probar una poda mayor para ver si se mejoran más los resultados.

1.4.2. Random Forest

Estudiaremos tres variaciones en los parámetros por defecto: cambiaremos el criterio de selección de variables de Gini a GainRatio y también probaremos con diferentes números de árboles.

TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
9634	1268	23122	2251	0,884	0,811	0,948	0,846	0,877	0,903	0,879

Figura 1.33: Medidas de Random Forest por defecto

TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
9638	1376	23014	2247	0,875	0,811	0,944	0,842	0,875	0,900	0,877

Figura 1.34: Medidas de Random Forest usando GainRatio

TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
9666	1277	23113	2219	0,883	0,813	0,948	0,847	0,878	0,904	0,880

Figura 1.35: Medidas de Random Forest con 125 árboles por partición

TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
9653	1290	23100	2232	0,882	0,812	0,947	0,846	0,877	0,903	0,880

Figura 1.36: Medidas de Random Forest con 75 árboles por partición

En general, los resultados sugieren que las diferentes configuraciones de Random Forest que se probaron (cambiar el criterio de selección de variables y el número de árboles) no tienen un impacto sustancial en el rendimiento del modelo. El modelo por defecto y las variantes probadas tienen un rendimiento muy similar en términos de las métricas de evaluación.

Es significativo que el rendimiento con un conjunto de 75 árboles es igual que con los otros tamaños. Por lo que será recomendable usar este modelo por ser más computacionalmente eficiente. Usar un número menor de árboles permite un menor tiempo de entrenamiento, menor utilización de memoria, además de aumentar la simplicidad e interpretabilidad.

1.4.3. XGBoost

Estudiaremos dos variaciones de XGBoost con diferente número de árboles.

TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
9535	1504	22886	2350	0,864	0,802	0,938	0,832	0,868	0,894	0,870

Figura 1.37: Medidas de XGBoost por defecto

TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
9583	1476	22914	2302	0,867	0,806	0,939	0,835	0,870	0,896	0,873

Figura 1.38: Medidas de XGBoost usando 125 árboles

TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
9320	1559	22831	2565	0,857	0,784	0,936	0,819	0,857	0,886	0,860

Figura 1.39: Medidas de XGBoost usando 50 árboles

A pesar de la variación en el número de árboles, todas las configuraciones de XGBoost logran un alto rendimiento, con valores de accuracy y AUC superiores al 88% y 86%, respectivamente. Al aumentar el número de árboles a 125, observamos una ligera mejora en el rendimiento. Y al reducir el número a 50, el accuracy y el AUC disminuyen levemente.

XGBoost es un algoritmo muy poderoso para problemas de clasificación y muestra un rendimiento excepcional en todas las configuraciones probadas. Usar menos árboles podría ser una opción válida si se busca un buen rendimiento con menos recursos computacionales.

1.4.4. K Nearest Neighbor

Estudiaremos tres variantes del algoritmo de K vecinos más cercanos: con K=10, con distancias con pesos y con K=10 y distancias con pesos.

TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
8656	2267	22123	3229	0,792	0,728	0,907	0,759	0,813	0,848	0,818

Figura 1.40: Medidas de KNN por defecto

TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
8318	2116	22274	3567	0,797	0,700	0,913	0,745	0,799	0,843	0,807

Figura 1.41: Medidas de KNN con K=10

TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
9162	2099	22291	2723	0,814	0,771	0,914	0,792	0,839	0,867	0,842

Figura 1.42: Medidas de KNN con distancias con pesos

TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
9150	1958	22432	2735	0,824	0,770	0,920	0,796	0,841	0,871	0,845

Figura 1.43: Medidas de KNN con K=10 y distancias con pesos

Al aumentar el número de vecinos a 10, observamos que el accuracy baja, sin embargo, la capacidad de predecir ejemplos negativos aumenta.

La introducción de distancias con pesos mejora el algoritmo por defecto. Tiene sentido que los vecinos más cercanos influyan más en la predicción que los más lejanos. Por lo tanto, con K=10 y distancias con pesos también obtenemos un accuracy más alto.

Vamos a extender un poco la comparación con esta última variante. Se obtiene un valor de G-mean mayor, 0,841 frente a 0,813, lo que sugiere que se ha mejorado la tasa de acierto en las clases positiva y negativa. Además, la medida F1-score mejora, lo que indica que es mejor al identificar verdaderos positivos y, al mismo tiempo, limitar falsos positivos.

1.4.5. Naive Bayes

Hemos obtenido un resultado muy malo con Naive Bayes, una tasa de acierto por debajo del 50%, por lo que para intentar mejorar estos resultados vamos a balancear completamente las clases positiva y negativa por si esta fuera la causa de ese mal rendimiento. Realizaremos por tanto un submuestreo y sobremuestreo para comparar los resultados.

TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
11767	21123	3267	118	0,358	0,990	0,134	0,526	0,364	0,414	0,562

Figura 1.44: Medidas de Naive-Bayes por defecto

La tasa de aciertos es inferior al 50%, lo que indica una clasificación deficiente. Sin embargo, el AUC es relativamente alto, lo que podría deberse a una alta TPR, pero no compensa el bajo TNR. Esto se debe a que en la gran mayoría de los casos, se predice la clase positiva. El modelo por defecto no es nada recomendable para este problema, lo que sugiere que la hipótesis necesaria para poder aplicar este algoritmo, todos los atributos son independientes conocida la variable clase, es errónea.

TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
11813	21964	2426	72	0,35	0,994	0,099	0,517	0,314	0,393	0,547

Figura 1.45: Medidas de Naive-Bayes con submuestreo

TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
11816	22090	2300	69	0,348	0,994	0,094	0,516	0,306	0,389	0,544

Figura 1.46: Medidas de Naive-Bayes con sobremuestreo

El balanceo de clases no ha mejorado al algoritmo inicial. Por lo tanto, concluimos que no es un problema en este caso de desbalanceo entre las clases sino del propio algoritmo Naive-Bayes y la hipótesis necesaria.

1.5. Análisis de resultados

	TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
Decision Tree	9424	2281	22109	2461	0,805	0,793	0,906	0,799	0,848	0,869	0,850
Random Forest	9634	1268	23122	2251	0,884	0,811	0,948	0,846	0,877	0,903	0,879
XGBoost	9535	1504	22886	2350	0,864	0,802	0,938	0,832	0,868	0,894	0,870
K Nearest Neighbor	8656	2267	22123	3229	0,792	0,728	0,907	0,759	0,813	0,848	0,818
Naive Bayes	11767	21123	3267	118	0,358	0,990	0,134	0,526	0,364	0,414	0,562
Decision Tree GainRatio	9537	2221	22169	2348	0,811	0,802	0,909	0,807	0,854	0,874	0,856
Decision Tree Pruning	9065	1606	22784	2820	0,849	0,763	0,934	0,804	0,844	0,878	0,848
Random Forest GainRatio	9638	1376	23014	2247	0,875	0,811	0,944	0,842	0,875	0,900	0,877
Random Forest 125	9666	1277	23113	2219	0,883	0,813	0,948	0,847	0,878	0,904	0,880
Random Forest 75	9653	1290	23100	2232	0,882	0,812	0,947	0,846	0,877	0,903	0,880
XGBoost 125	9583	1476	22914	2302	0,867	0,806	0,939	0,835	0,870	0,896	0,873
XGBoost 50	9320	1559	22831	2565	0,857	0,784	0,936	0,819	0,857	0,886	0,860
KNN 10	8318	2116	22274	3567	0,797	0,700	0,913	0,745	0,799	0,843	0,807
KNN w	9162	2099	22291	2723	0,814	0,771	0,914	0,792	0,839	0,867	0,842
KNN 10 w	9150	1958	22432	2735	0,824	0,770	0,920	0,796	0,841	0,393	0,845
Naive Bayes undersamp	11813	21964	2426	72	0,35	0,994	0,099	0,517	0,314	0,393	0,547
Naive Bayes oversamp	11816	22090	2300	69	0,348	0,994	0,094	0,516	0,306	0,389	0,544

Figura 1.47: Tabla conjunto de los resultados de todos los modelos

Decision Tree:

- Pros: Buen accuracy (0,869) y AUC (0,85).
- Contras: PPV y F1-score podrían mejorarse.

En general, obtenemos unos resultados sólidos con árboles de decisión. La poda del árbol parece mejorar la generalización, pero reduce el rendimiento en términos de precisión.

Random Forest:

- Pros: Valores muy altos de TNR y AUC. Es el algoritmo con mayor tasa de acierto.

- Contras: Resultados similares con distintos tamaños, por lo que probablemente podamos reducir aún más el modelo sin empeorarlo.

XGBoost:

- Pros: AUC y accuracy muy altos.
- Contras: Rendimiento muy similar con diferente número de árboles, por lo que se podría reducir aún más el modelo.

KNN:

- Pros: Modelo sencillo con el que el accuracy es alto.
- Contras: No se construye un modelo, el modelo es la propia BD.

Naive Bayes:

- Contras: Son los peores resultados de todos con diferencia.

Que se obtengan tan malas medidas con Naive Bayes significa que las variables tienen una fuerte dependencia condicional respecto a la clase.

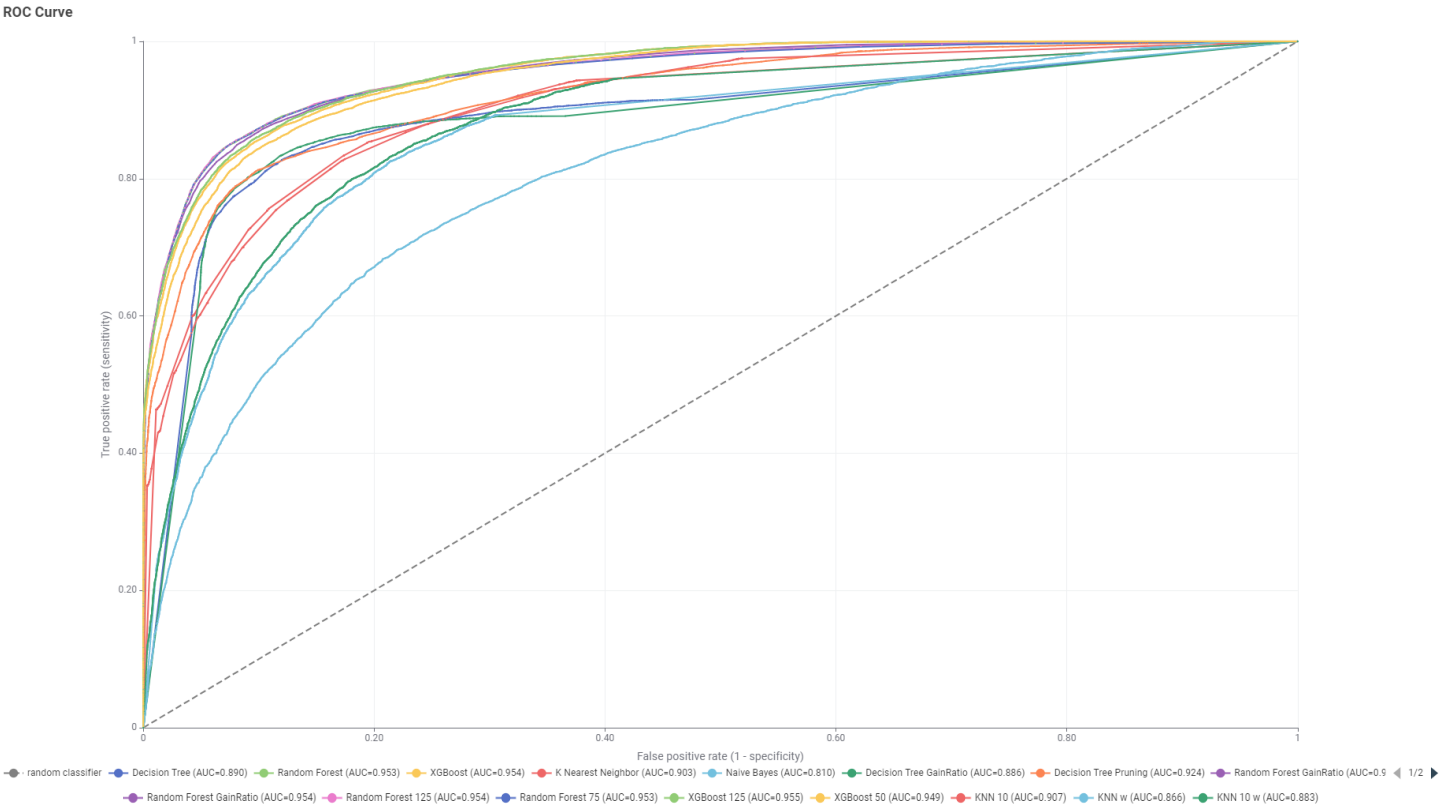

Figura 1.48: Curvas ROC de todos los modelos

La curva ROC representa la relación entre la tasa de verdaderos positivos (TPR) y la tasa de falsos positivos (FPR). Observamos que las curvas más altas son las de Random Forest y XGBoost. Esto puede significar dos cosas, que tienen mayor TPR en comparación con los demás modelos en todos los umbrales de clasificación, es decir, son más efectivos a la hora de identificar TP en el conjunto de datos. Por otro lado, también puede significar que tienen una FPR en comparación con los demás en todos los umbrales, lo que indica que son más precisos al evitar clasificar incorrectamente instancias negativas como positivas.

Podemos distinguir unos 5 grupos de curvas en el gráfico. En orden ascendente:

- Naive Bayes. Destacamos su bajo rendimiento para la mayoría de valores del umbral, salvo para valores extremos para los cuales la curva supera a los Decision Tree.
- KNN con pesos (para ambos valores de K. A pesar de tener mejor precisión que sin pesos, tienen un AUC inferior, lo que implica que su capacidad de discrimación entre clases positiva y negativa es inferior.

- KNN con K=5 y K=10, El pico de la curva no es exagerado, cambia lentamente de vertical a horizontal, lo que sugiere que tienen un buen rendimiento pero la capacidad de ajuste a diferentes umbrales de clasificación no es tan pronunciada como otros modelos.
- Decision Trees. El rendimiento al principio es muy alto pero finalmente se estabiliza. Indicando que los Decision Trees tienen una buena capacidad de ajuste inicial, pero su rendimiento puede que no sea tan consistente en umbrales extremos.
- XGBoost y Random Forests.

1.6. Interpretación de los datos

En primer lugar, comenzamos calculando la matriz de correlaciones de todos los atributos:

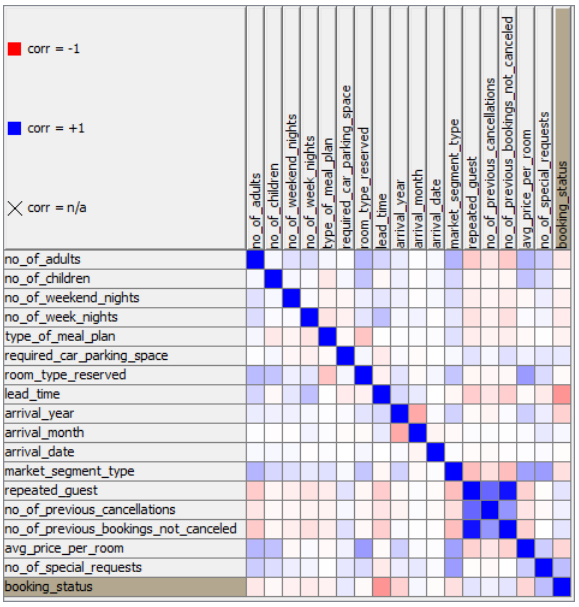

Figura 1.49: Matriz de correlaciones entre los atributos

Podemos sacar algunas conclusiones sencillas como por ejemplo que si el cliente ha repetido influye en el número de cancelaciones o no cancelaciones previo, que el precio medio por habitación está correlacionado positivamente con el tipo de habitación y también con el tipo de segmento de mercado.

La única variable que está correlacionada significativamente con el estado de la reserva (clase) es *lead_time*, que es el número de días entre la fecha de reserva y la fecha de inicio de la estancia.

A continuación, vamos a visualizar los primeros niveles de algún árbol de decisión. En KNIME generamos 5 árboles distintos, uno para cada partición de entrenamiento, pero he observado que los primeros niveles son siempre muy similares.

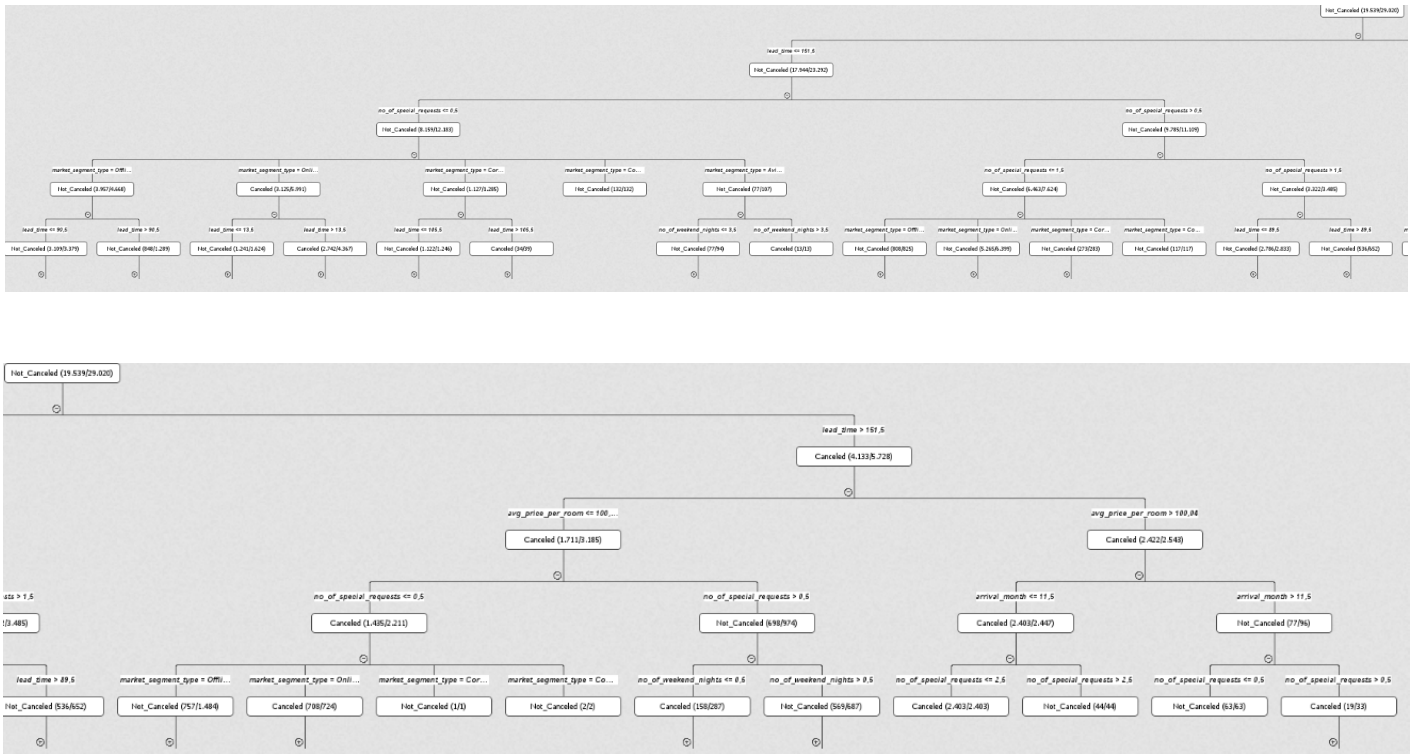

Figura 1.50: Primeros niveles de un árbol de decisión

Observamos que la primera variable por la que se divide el árbol es *lead_time*, tendrá el índice Gini menor. Luego se divide por el número de peticiones especiales y por el precio medio en la otra rama. Más tarde aparecen atributos como *market_segment_type*, *arrival_month* y *no_of_weekend_nights.*

Vamos a ver qué ocurre con algún árbol que use GainRatio

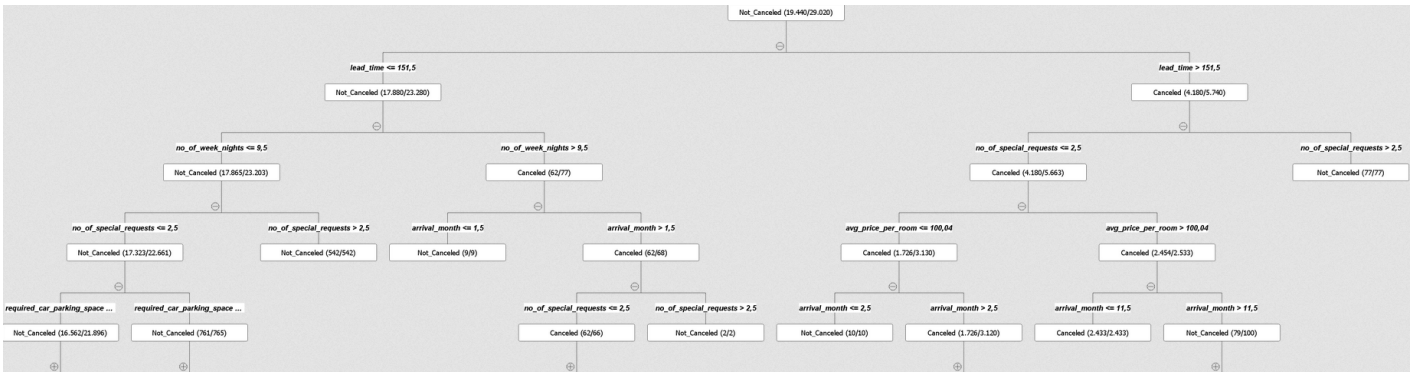

Figura 1.51: Primeros niveles de un árbol de decisión que usa GainRatio

En este caso las divisiones del árbol siguen:

- 1. *lead_time*
- 2. *no_of_week_nights / no_of_special_requests*
- 3. *no_of_special_requests / arrival_month / avg_price_per_room*
- 4. *required_car_parking_space / no_of_special_requests / arrival_month*

Ambas versiones de árbol de decisión muestran una preferencia por la variable *lead_time* como la primera variable de división. Esto sugiere que es un atributo importante para la toma de decisiones de los modelos.

Además, las dos variantes también consideran atributos comunes, como *no_of_special_requests* y *arrival_month, en niveles posteriores,* por lo que estos atributos tendrán su importancia.

variables	#splits (level 0)	#splits (level 1)	#splits (level 2)	#candidates (level 0)	#candidates (level 1)	#candidates (level 2)	importance (all levels)
lead_time	20	40	76	20	46	95	2,6697
no_of_special_requests	25	22	37	29	35	86	1,92
avg_price_per_room	15	23	38	29	43	83	1,509
arrival_year	12	26	28	22	54	97	1,315
market_segment_type	8	19	42	20	51	94	1,219
arrival_month	5	15	38	20	49	98	0,9431
no_of_adults	2	12	20	24	48	93	0,548
type_of_meal_plan	1	8	27	20	43	94	0,523
repeated_guest	4	6	9	18	46	90	0,452
no_of_week_nights	1	7	23	26	54	102	0,393
no_of_weekend_nights	0	8	19	21	44	101	0,369
no_of_previous_bookings_not_canceled	4	5	6	28	44	98	0,317
required_car_parking_space	3	3	5	30	55	85	0,213
arrival_date	0	2	8	30	48	98	0,123
room_type_reserved	0	1	8	24	50	101	0,099
no_of_children	0	1	6	20	42	96	0,086
no_of_previous_cancellations	0	2	1	19	48	89	0,0529

Figura 1.52: Tabla con la importancia de los atributos en Random Forest

Con los datos de la importancia de las variables en Random Forest obtenemos unas conclusiones similares, *lead_time* sigue siendo la característica más importante, seguida por *avg_price_per_room, arrival_year* y *market_segment_type.*

Obtenemos también las variables más importantes del modelo XGBoost ordenadas por la columna Total Gain, la cual indica la ganancia a lo largo de todas las divisiones en las que la variable se ha usado

Feature Name	Weight	Gain	Cover	Total Gain	Total Cover
lead_time	1700	10,58	968,0111158	17986,14971	1645618,897
avg_price_per_room	1676	4,45501	1040,861958	7468,007786	1744484,642
no_of_special_requests	418	12,505	1644,149713	5227,248666	687254,58
arrival_month	774	4,3531	921,2866843	3369,353953	713075,8936
arrival_date	922	1,4989	476,141445	1381,165949	439002,4123
no_of_weekend_nights	356	3,5557	280,1163539	1265,83043	99721,422
no_of_adults	176	6,8638	669,3716345	1207,922464	117809,4077
no_of_week_nights	468	2,32848	912,0184797	1089,706751	426824,6485
arrival_year	110	7,722299	921,1738153	849,4527766	101329,1197
required_car_parking_space	50	11,3687	4601,240093	568,4389337	230062,0046
no_of_children	60	1,1212	696,6492175	67,27268085	41798,95305
repeated_guest	10	4,937	3760,979512	49,37597683	37609,79512
no_of_previous_bookings_not_canceled	26	1,332	3694,112458	34,65278002	96046,9239
no_of_previous_cancellations	16	0,629	9,264119311	10,06764351	148,225909

Figura 1.53: Tabla con la importancia de los atributos en XGBoost

Por último, hacemos uso de los nodos *AutoML* y *Global Feature Importance,* los cuales usan diferentes modelos como Redes Neuronales, Random Forestm, Deep Learning (Keras) ..., para obtener las características más importantes del problema. [1]

Surrogate Random Forest:

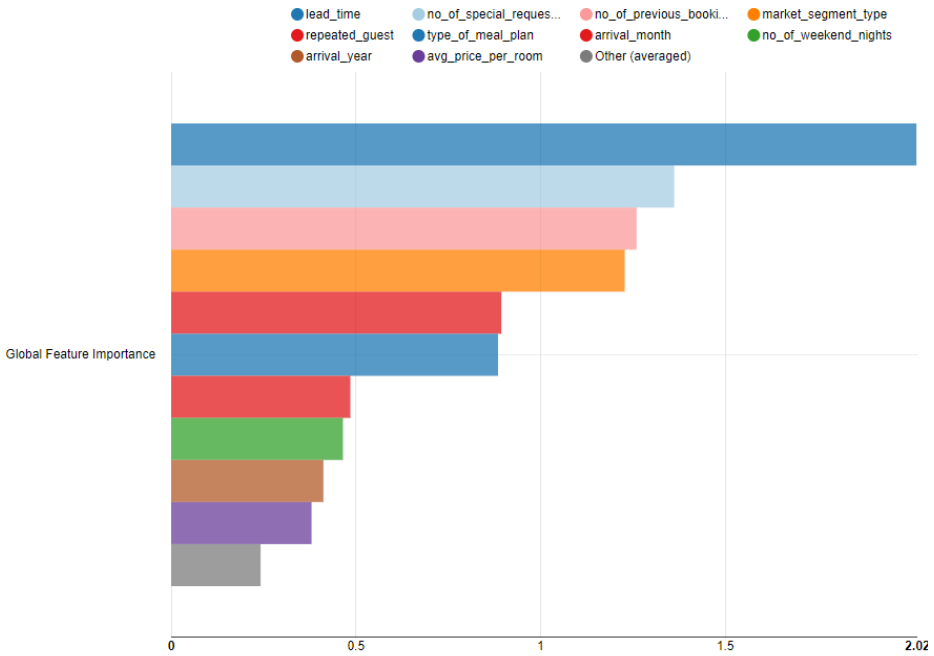

Figura 1.54: Importancia Global de las Características usando Surrogate Random Forest

Es un modelo de Random Forest entrenado para aproximar las predicciones del modelo original. El Random Forest se entrena con los datos de entrada preprocesados de manera estándar y con los parámetros optimizados de "Profundidad del árbol", "Número de modelos" y "Tamaño mínimo del nodo hijo".

La importancia de las características se calcula contando cuántas veces se ha seleccionado una característica para una división y en qué posición (nivel) entre todas las características disponibles (candidatas) en los árboles del bosque aleatorio. Un valor más alto indica una mayor importancia de la característica.

Permutation Feature Importance:

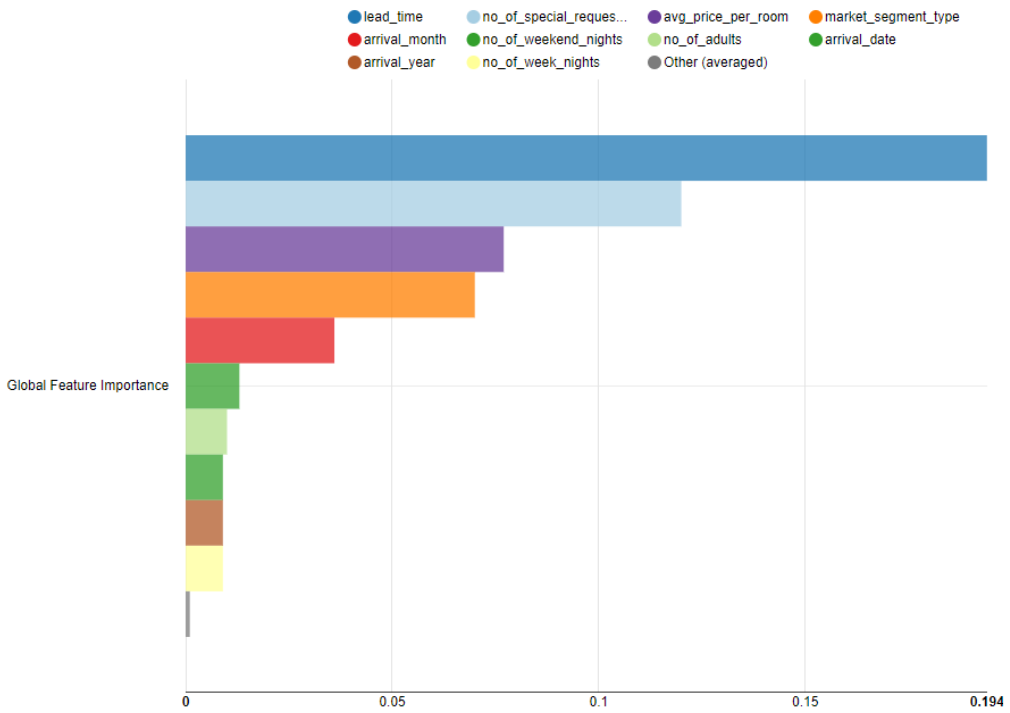

Figura 1.55: Importancia Global de las Características usando Permutation Feature Importance

Esta medida indica la diferencia entre la puntuación de rendimiento del modelo estimado en predicciones utilizando todas las características originales y la puntuación de rendimiento del modelo estimado en predicciones utilizando todas las características originales excepto una que se permuta de manera aleatoria. Si una característica se permuta varias veces, se calcula la diferencia promedio.

Una diferencia de puntuación grande indica que la característica era importante para la predicción, ya que romper la relación entre esta característica y el objetivo disminuyó el rendimiento del modelo. Una diferencia cercana a 0 significa que permutar la característica no disminuye el rendimiento del modelo. La diferencia negativa indica que, por alguna razón, permutar la característica aumenta el rendimiento del modelo.

Como conclusión, los atributos que identifican más a la clase son *lead_time, no_of_special_requests, avg_price_per_room, market_segment_type, no_of_previous_bookings_not_canceled*

1.7. Contenido adicional

1.8. Bibliografía

https://machinelearningmastery.com/classification-accuracy-is-not-enough-more-performance-measures-you-can-use/
Fawcett, T. (2006). An introduction to ROC analysis. Pattern recognition letters, 27(8), 861-874
[1] Understand Your ML Model: Global Feature Importance | KNIME

2. Identificación de Gestos

2.1. Introducción

El conjunto de datos gestos tiene como objetivo estudiar la segmentación de las fases de gestos. El dataset está compuesto por 32 atributos numéricos (double), que representan las velocidades y aceleraciones vectoriales de cada muñeca y mano, y un atributo de clase que indica en qué fase del gesto se encuentra esa persona. Estas características fueron extraídas de 7 vídeos de gente gesticulando.

Es un problema de clasificación multiclase, en concreto tiene 5 clases distintas:

- D (rest position).
- P (preparation).
- S (stroke).
- H (hold).
- R (retraction).

Todos los atributos son numéricos, por lo que deberemos tener cuidado y discretizar los datos si queremos aplicar algún algoritmo que solamente admita variables nominales.

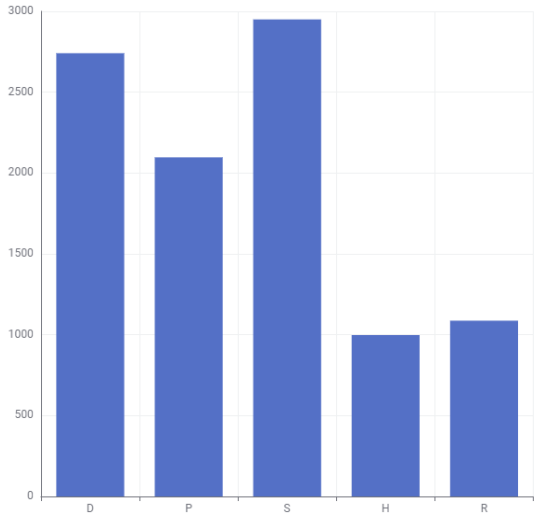

Figura 2.1: Frecuencia de los valores de la clase

Observamos que las clases menos frecuentes en los datos son H ý R, por lo que, en un estudio futuro para mejorar los resultados es posible que tengamos que hacer uso de técnicas de muestreo.

2.2. Procesado de datos

En este problema no realizaremos ningún preprocesamiento general en primer lugar. El único algoritmo en el que es necesario hacer un preprocesamiento en específico es KNN. Es esencial normalizar los atributos para que todos tengan el mismo peso en el algoritmo. Además, en este caso como todas las variables son numéricas podemos usar la distancia euclídea que viene implementada en KNIME.

El único dilema es qué normalización usar: min-max o zero-mean (Z-score). Para ver cuál de ellas será más adecuada lo que haremos será mostrar las distribuciones de los algoritmos y si se asemejan a una gaussiana usaremos la normalización Z-score.

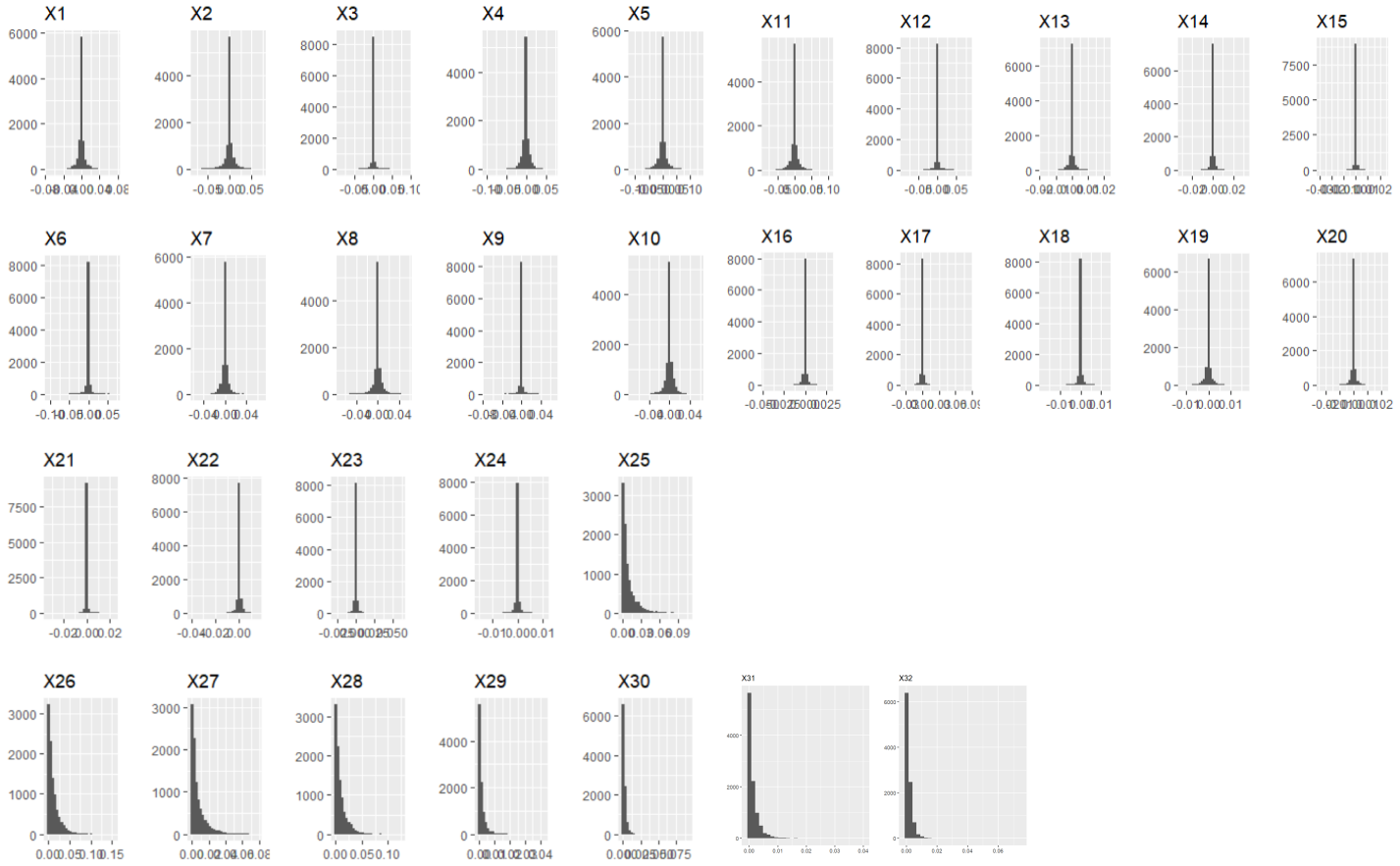

Figura 2.2: Distribución de las 32 variables numéricas

Y observamos que la mayoría de los atributos se pueden ajustar a una función gaussiana. Por lo que en el nodo *Normalize* usaremos el ajuste Z-score normalization.

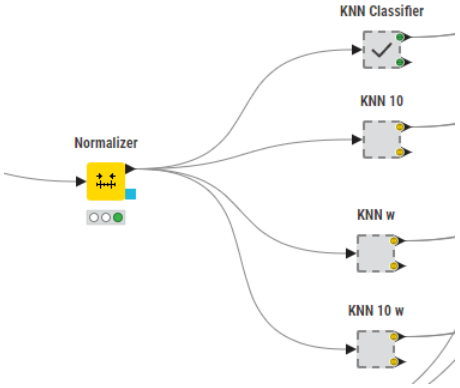

Figura 2.3: Captura de KNIME del normalizado de datos

2.3. Resultados obtenidos

2.3.1. Decision Tree

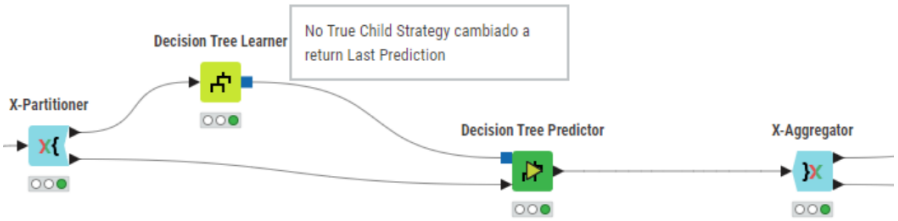

Figura 2.4: Captura de KNIME del flujo de Decision Tree

En el algoritmo de *Decision Tree*, el único ajuste que he cambiado de lo que venía por defecto ha sido en los ajustes del nodo *Decision Tree Learner* en PMMLSettings. Así que, lo que he cambiado es que cuando no haya ningún hijo verdadero se devuelva la última predicción en lugar de nulo.

Clase	TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
D	1768	1090	6042	973	0,618	0,64	0,847	0,6315	0,739	0,791	0,746
H	328	591	8284	670	0,356	0,328	0,933	0,342	0,5538	0,872	0,631
P	968	1196	6580	1129	0,447	0,4616	0,841	0,4543	0,6249	0,764	0,653
R	392	566	8220	695	0,409	0,360	0,935	0,383	0,581	0,8722	0,648
S	1661	1313	5610	1289	0,559	0,563	0,810	0,560	0,675	0,73642	0,686

Figura 2.5: Medidas de Decision Tree

Accuracy del algoritmo: 0,518
Balanced Accuracy: 0,42848
Balanced Accuracy Weighted: 0,104

fold #	Error in %	Size of Test Set	Error Count	Model size
0	48,456	1975	957	1194
1	48,253	1975	953	1205
2	48,126	1974	950	1245
3	47,190	1975	932	1201
4	48,835	1974	964	1181

Figura 2.6: Tabla de errores y tamaño del modelo Decision Tree

2.3.2. Random Forest

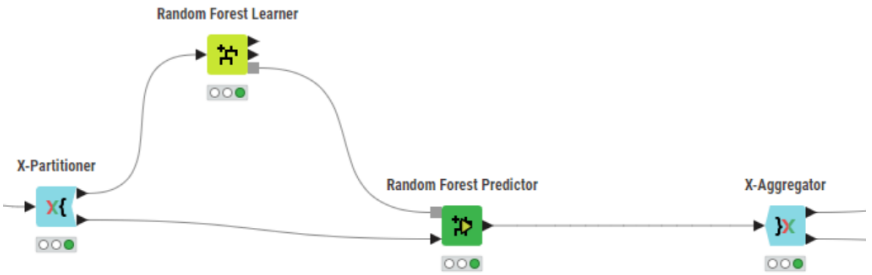

Figura 2.7: Captura de KNIME del flujo de Random Forest

Clase	TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
D	2369	1106	6026	372	0,681	0,864	0,844	0,762	0,854	0,850	0,854
H	384	100	8775	614	0,793	0,384	0,988	0,518	0,616	0,927	0,686
P	1127	539	7237	970	0,676	0,537	0,930	0,598	0,707	0,847	0,734
R	437	255	8531	650	0,631	0,402	0,970	0,491	0,624	0,908	0,686
S	2276	1280	5643	674	0,640	0,771	0,815	0,699	0,793	0,802	0,793

Figura 2.8: Medidas de Random Forest

Accuracy del algoritmo: 0,668
Balanced Accuracy: 0,592
Balanced Accuracy Weighted: 0,134

fold #	Error in %	Size of Test Set	Error Count
0	33,0127	1975	652
1	35,3418	1975	698
2	30,1418	1974	595
3	34,5316	1975	682
4	33,0800	1974	653

Figura 2.9: Tabla de errores de Random Forest

2.3.3. XGBoost

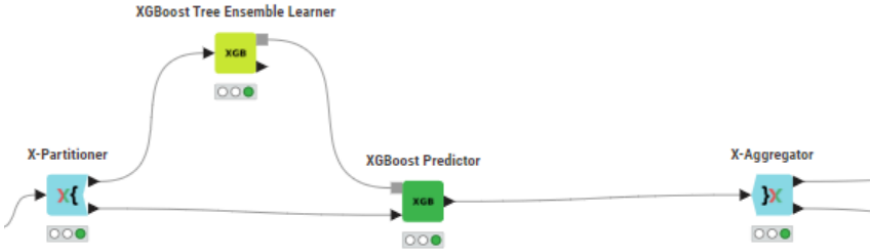

Figura 2.10: Captura de KNIME del flujo de XGBoost

Clase	TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
D	2298	909	6223	443	0,716	0,838	0,872	0,772	0,855	0,863	0,855
H	474	187	8688	524	0,717	0,474	0,978	0,571	0,681	0,927	0,726
P	1174	683	7093	923	0,632	0,559	0,912	0,593	0,714	0,837	0,736
R	496	323	8463	591	0,605	0,456	0,963	0,520	0,662	0,907	0,709
S	2180	1149	5774	770	0,654	0,738	0,834	0,694	0,785	0,805	0,786

Figura 2.11: Medidas de XGBoost

Accuracy del algoritmo: 0,671
Balanced Accuracy: 0,592
Balanced Accuracy Weighted: 0,134

fold #	Error in %	Size of Test Set	Error Count
0	33,620	1975	664
1	33,418	1975	660
2	30,851	1974	609
3	34,582	1975	683
4	32,168	1974	635

Figura 2.12: Tabla de errores de XGBoost

2.3.4. KNN

Figura 2.13: Captura de KNIME del flujo de KNN

Clase	TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
D	2223	1457	5675	518	0,604	0,811	0,795	0,692	0,803	0,799	0,803
H	340	302	8573	658	0,529	0,340	0,965	0,414	0,573	0,902	0,653
P	940	710	7066	1157	0,569	0,448	0,908	0,501	0,638	0,810	0,678
R	340	264	8522	747	0,562	0,312	0,969	0,402	0,550	0,897	0,641
S	1991	1306	5617	959	0,603	0,674	0,811	0,637	0,739	0,770	0,743

Figura 2.14: Medidas de KNN

Accuracy del algoritmo: 0,591
Balanced Accuracy: 0,5175
Balanced Accuracy Weighted: 0,1182

fold #	Error in %	Size of Test Set	Error Count
0	40,911	1975	808
1	41,519	1975	820
2	39,514	1974	780
3	41,873	1975	827
4	40,729	1974	804

Figura 2.15: Tabla de errores de KNN

2.3.5. Naive-Bayes

Figura 2.16: Captura de KNIME del flujo de Naive-Bayes

Clase	TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
D	2340	2744	4388	401	0,460	0,853	0,615	0,598	0,724	0,681	0,734
H	128	931	7944	870	0,120	0,128	0,895	0,124	0,338	0,817	0,511
P	460	753	7023	1637	0,379	0,219	0,903	0,277	0,445	0,757	0,561
R	179	494	8292	908	0,265	0,164	0,9437	0,203	0,394	0,857	0,554
S	1050	794	6129	1900	0,569	0,355	0,885	0,438	0,561	0,727	0,620

Figura 2.17: Medidas de Naive-Bayes

Accuracy del algoritmo: 0,421
Balanced Accuracy: 0,344
Balanced Accuracy Weighted: 0,084

fold #	Error in %	Size of Test Set	Error Count
0	57,266	1975	1131
1	57,823	1975	1142
2	55,471	1974	1095
3	59,899	1975	1183
4	59,017	1974	1165

Figura 2.18: Tabla de errores de Naive-Bayes

2.4. Configuración de algoritmos

2.4.1. Decision Tree

Estudiaremos dos variaciones de los parámetros por defecto: cambiaremos el criterio de selección de variables de Gini a GainRatio y también probaremos a podar el árbol.

Clase	TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
D	1768	1090	6042	973	0,618	0,64	0,847	0,6315	0,739	0,791	0,746
H	328	591	8284	670	0,356	0,328	0,933	0,342	0,5538	0,872	0,631
P	968	1196	6580	1129	0,447	0,4616	0,841	0,4543	0,6249	0,764	0,653
R	392	566	8220	695	0,409	0,360	0,935	0,383	0,581	0,8722	0,648
S	1661	1313	5610	1289	0,559	0,563	0,810	0,560	0,675	0,73642	0,686

Figura 2.19: Medidas de Decision Tree por defecto

Accuracy del algoritmo: 0,448
Balanced Accuracy: 0,4284
Balanced Accuracy Weighted: 0,104

Clase	TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
D	1706	1311	5821	1035	0,565	0,622	0,816	0,593	0,713	0,762	0,719
H	255	493	8382	743	0,341	0,256	0,944	0,292	0,491	0,875	0,600
P	827	1433	6343	1270	0,366	0,394	0,816	0,380	0,567	0,726	0,605
R	256	613	8173	831	0,295	0,236	0,930	0,262	0,468	0,854	0,583
S	1375	1604	5319	1575	0,462	0,466	0,768	0,464	0,598	0,678	0,617

Figura 2.20: Medidas de Decision Tree usando GainRatio

Accuracy del algoritmo: 0,4476
Balanced Accuracy: 0,3948
Balanced Accuracy Weighted: 0,0895

Clase	TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
D	1953	1160	5972	788	0,627	0,713	0,837	0,667	0,772	0,803	0,775
H	262	446	8429	736	0,370	0,263	0,950	0,307	0,499	0,880	0,606
P	924	936	6840	1173	0,497	0,441	0,880	0,467	0,623	0,786	0,660
R	364	469	8317	723	0,437	0,335	0,947	0,379	0,563	0,879	0,641
S	1849	1510	5413	1101	0,550	0,627	0,782	0,586	0,700	0,736	0,704

Figura 2.21: Medidas de Decision Tree con poda

Accuracy del algoritmo: 0,542
Balanced Accuracy: 0,4755
Balanced Accuracy Weighted: 0,1084

Observamos que cambiar el criterio de selección a GainRatio parece haber aumentado levemente la capacidad predictiva del modelo, todas las medidas han aumentado un poco.

Por otro lado, la poda del árbol ha resultado en un mejor rendimiento en términos generales. El accuracy aumenta a 0,878 y el AUC a 0,92, lo que indica que la poda del árbol ha ayudado a reducir el sobreajuste y a mejorar la generalización del modelo.

2.4.2. Random Forest

Estudiaremos cuatro variaciones en los parámetros por defecto: cambiaremos el criterio de selección de variables de Gini a GainRatio y también probaremos con diferentes números de árboles.

Clase	TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
D	2369	1106	6026	372	0,681	0,864	0,844	0,762	0,854	0,850	0,854
H	384	100	8775	614	0,793	0,384	0,988	0,518	0,616	0,927	0,686
P	1127	539	7237	970	0,676	0,537	0,930	0,598	0,707	0,847	0,734
R	437	255	8531	650	0,631	0,402	0,970	0,491	0,624	0,908	0,686
S	2276	1280	5643	674	0,640	0,771	0,815	0,699	0,793	0,802	0,793

Figura 2.22: Medidas de Random Forest por defecto

Accuracy del algoritmo: 0,668
Balanced Accuracy: 0,592
Balanced Accuracy Weighted: 0,134

Clase	TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
D	2366	1099	6033	375	0,682	0,863	0,845	0,762	0,854	0,850	0,854
H	399	109	8766	599	0,785	0,399	0,987	0,529	0,628	0,928	0,693
P	1132	545	7231	965	0,675	0,539	0,929	0,599	0,708	0,847	0,734
R	439	229	8557	648	0,657	0,403	0,973	0,500	0,627	0,911	0,688
S	2264	1291	5632	686	0,636	0,767	0,813	0,696	0,790	0,799	0,790

Figura 2.23: Medidas de Random Forest usando GainRatio

Accuracy del algoritmo: 0,668
Balanced Accuracy: 0,595
Balanced Accuracy Weighted: 0,134

Clase	TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
D	2371	1115	6017	370	0,680	0,865	0,843	0,761	0,854	0,849	0,854
H	384	88	8787	614	0,814	0,385	0,990	0,522	0,617	0,929	0,687
P	1130	527	7249	967	0,682	0,539	0,932	0,602	0,709	0,849	0,736
R	430	246	8540	657	0,636	0,396	0,972	0,488	0,620	0,909	0,684
S	2281	1301	5622	669	0,637	0,773	0,812	0,698	0,792	0,800	0,793

Figura 2.24: Medidas de Random Forest con 125 árboles por partición

Accuracy del algoritmo: 0,668
Balanced Accuracy: 0,591
Balanced Accuracy Weighted: 0,1336

Clase	TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
D	2369	1106	6026	372	0,681	0,864	0,844	0,762	0,854	0,850	0,854
H	384	100	8775	614	0,793	0,384	0,988	0,518	0,616	0,927	0,686
P	1127	539	7237	970	0,676	0,537	0,930	0,598	0,707	0,847	0,734
R	437	255	8531	650	0,631	0,402	0,970	0,491	0,624	0,908	0,686
S	2276	1280	5643	674	0,640	0,771	0,815	0,699	0,793	0,802	0,793

Figura 2.25: Medidas de Random Forest con 75 árboles por partición

Accuracy del algoritmo: 0,664
Balanced Accuracy: 0,59
Balanced Accuracy Weighted: 0,1328

Clase	TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
D	2240	786	6346	501	0,740	0,817	0,890	0,777	0,853	0,870	0,854
H	612	475	8400	386	0,563	0,613	0,946	0,587	0,762	0,913	0,780
P	1201	675	7101	896	0,640	0,573	0,913	0,605	0,723	0,841	0,743
R	654	602	8184	433	0,521	0,602	0,931	0,558	0,749	0,895	0,767
S	1885	743	6180	1065	0,717	0,639	0,893	0,676	0,755	0,817	0,766

Figura 2.26: Medidas de Random Forest con Smote

Accuracy del algoritmo: 0,66677
Balanced Accuracy: 0,6488
Balanced Accuracy Weighted: 0,1335

En general, los resultados sugieren que las diferentes configuraciones de Random Forest que se probaron (cambiar el criterio de selección de variables y el número de árboles) no tienen un impacto sustancial en el rendimiento del modelo. El modelo por defecto y las variantes probadas tienen un rendimiento muy similar en términos de las métricas de evaluación.

Es significativo que el rendimiento con un conjunto de 75 árboles es igual que con los otros tamaños. Por lo que será recomendable usar este modelo por ser más computacionalmente eficiente. Usar un número menor de árboles permite un menor tiempo de entrenamiento, menor utilización de memoria, además de aumentar la simplicidad e interpretabilidad.

2.4.3. XGBoost

Estudiaremos dos variaciones de XGBoost con diferente número de árboles.

Clase	TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
D	2298	909	6223	443	0,716	0,838	0,872	0,772	0,855	0,863	0,855
H	474	187	8688	524	0,717	0,474	0,978	0,571	0,681	0,927	0,726
P	1174	683	7093	923	0,632	0,559	0,912	0,593	0,714	0,837	0,736
R	496	323	8463	591	0,605	0,456	0,963	0,520	0,662	0,907	0,709
S	2180	1149	5774	770	0,654	0,738	0,834	0,694	0,785	0,805	0,786

Figura 2.27: Medidas de XGBoost por defecto

Accuracy del algoritmo: 0,671
Balanced Accuracy: 0,592
Balanced Accuracy Weighted: 0,134

Clase	TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
D	2314	862	6270	427	0,729	0,844	0,879	0,782	0,861	0,869	0,862
H	496	189	8686	502	0,724	0,497	0,979	0,589	0,697	0,930	0,738
P	1186	668	7108	911	0,640	0,566	0,914	0,600	0,719	0,840	0,740
R	496	325	8461	591	0,604	0,456	0,963	0,520	0,663	0,907	0,710
S	2199	1138	5785	751	0,659	0,745	0,836	0,700	0,789	0,809	0,791

Figura 2.28: Medidas de XGBoost con 125 árboles

Accuracy del algoritmo: 0,6777
Balanced Accuracy: 0,622
Balanced Accuracy Weighted: 0,1355

Clase	TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
D	2286	1023	6109	455	0,691	0,834	0,857	0,756	0,845	0,850	0,845
H	415	192	8683	583	0,684	0,416	0,978	0,517	0,638	0,922	0,697
P	1122	687	7089	975	0,620	0,535	0,912	0,575	0,698	0,832	0,723
R	475	354	8432	612	0,573	0,437	0,960	0,496	0,648	0,902	0,698
S	2128	1191	5732	822	0,641	0,721	0,828	0,679	0,773	0,796	0,775

Figura 2.29: Medidas de XGBoost con 50 árboles

Accuracy del algoritmo: 0,6509
Balanced Accuracy: 0,5886
Balanced Accuracy Weighted: 0,1301

Clase	TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
D	0,758	0,803	0,902	0,780	0,851	0,874	0,852	0,758	0,803	0,902	0,780
H	0,599	0,597	0,955	0,598	0,755	0,919	0,776	0,599	0,597	0,955	0,598
P	0,620	0,591	0,902	0,605	0,730	0,836	0,747	0,620	0,591	0,902	0,605
R	0,531	0,580	0,937	0,554	0,737	0,897	0,758	0,531	0,580	0,937	0,554
S	0,706	0,668	0,882	0,686	0,767	0,818	0,775	0,706	0,668	0,882	0,686

Figura 2.30: Medidas de XGBoost usando SMOTE

Accuracy del algoritmo: 0,6721
Balanced Accuracy: 0,6477
Balanced Accuracy Weighted: 0,1344

A pesar de la variación en el número de árboles, todas las configuraciones de XGBoost logran un alto rendimiento, con valores de accuracy y AUC superiores al 89% y 94%, respectivamente. Al aumentar el número de árboles a 125, observamos una ligera mejora en el rendimiento. Y al reducir el número a 50, el accuracy alcanza el 90%, aunque el AUC se ve reducido levemente.

XGBoost es un algoritmo muy poderoso para problemas de clasificación y muestra un rendimiento excepcional en todas las configuraciones probadas. Usar menos árboles podría ser una opción

válida si se busca un buen rendimiento con menos recursos computacionales.

2.4.4. K Nearest Neighbor

Estudiaremos tres variantes del algoritmo de K vecinos más cercanos: con K=10, con distancias con pesos y con K=10 y distancias con pesos.

Clase	TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
D	2223	1457	5675	518	0,604	0,811	0,795	0,692	0,803	0,799	0,803
H	340	302	8573	658	0,529	0,340	0,965	0,414	0,573	0,902	0,653
P	940	710	7066	1157	0,569	0,448	0,908	0,501	0,638	0,810	0,678
R	340	264	8522	747	0,562	0,312	0,969	0,402	0,550	0,897	0,641
S	1991	1306	5617	959	0,603	0,674	0,811	0,637	0,739	0,770	0,743

Figura 2.31: Medidas de KNN por defecto

Accuracy del algoritmo: 0,591
Balanced Accuracy: 0,5175
Balanced Accuracy Weighted: 0,1182

Clase	TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
D	2247	1647	5485	494	0,577	0,820	0,769	0,677	0,794	0,783	0,794
H	278	306	8569	720	0,476	0,279	0,966	0,351	0,519	0,896	0,622
P	900	721	7055	1197	0,555	0,429	0,907	0,484	0,624	0,806	0,668
R	317	270	8516	770	0,540	0,292	0,969	0,379	0,532	0,895	0,630
S	1877	1310	5613	1073	0,589	0,636	0,811	0,612	0,718	0,759	0,724

Figura 2.32: Medidas de KNN con K=10

Accuracy del algoritmo: 0,59692
Balanced Accuracy: 0,491
Balanced Accuracy Weighted: 0,1138

Clase	TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
D	2236	1328	5804	505	0,627	0,816	0,814	0,709	0,815	0,814	0,815
H	443	409	8466	555	0,520	0,444	0,954	0,479	0,651	0,902	0,699
P	1027	807	6969	1070	0,560	0,490	0,896	0,523	0,663	0,810	0,693
R	404	365	8421	683	0,525	0,372	0,958	0,435	0,597	0,894	0,665
S	1848	1006	5917	1102	0,648	0,626	0,855	0,637	0,732	0,786	0,741

Figura 2.33: Medidas de KNN usando distancia con pesos

Accuracy del algoritmo: 0,591
Balanced Accuracy: 0,55
Balanced Accuracy Weighted: 0,1207

Clase	TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
D	2278	1565	5567	463	0,593	0,831	0,781	0,692	0,805	0,795	0,806
H	369	364	8511	629	0,503	0,370	0,959	0,426	0,595	0,899	0,664
P	992	792	6984	1105	0,556	0,473	0,898	0,511	0,652	0,808	0,686
R	376	300	8486	711	0,556	0,346	0,966	0,427	0,578	0,898	0,656
S	1801	1036	5887	1149	0,635	0,611	0,850	0,622	0,721	0,779	0,730

Figura 2.34: Medidas de KNN con K=10 usando distancia con pesos

Accuracy del algoritmo: 0,589
Balanced Accuracy: 0,526
Balanced Accuracy Weighted: 0,1182

Al aumentar el número de vecinos a 10, observamos que el accuracy disminuye a 82,9%, sin embargo, la capacidad de predecir ejemplos negativos aumenta.

La introducción de distancias con pesos mejora el algoritmo por defecto. Tiene sentido que los vecinos más cercanos influyan más en la predicción que los más lejanos. Por lo tanto, con K=10 y distancias con pesos también obtenemos un accuracy más alto.

Vamos a extender un poco la comparación con esta última variante. Se obtiene un valor de G-mean mayor, 0,857 frente a 0,832, lo que sugiere que se ha mejorado la tasa de acierto en las clases positiva y negativa. Además, la medida F1-score mejora, lo que indica que es mejor al identificar verdaderos positivos y, al mismo tiempo, limitar falsos positivos.

2.5. Análisis de resultados

Para comparar los distintos algoritmos, lo que haremos será crear 5 tablas, una por cada valor del atributo clase, y visualizaremos los distintos resultados. Es decir, en primer lugar realizaremos un análisis de OVA (One vs All), tomando como uno de los valores como clase positiva y los 4 restantes como clase negativa.

Tomando D como clase positiva:

	TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
Decision Tree	1768	1090	6042	973	0,618	0,640	0,847	0,632	0,739	0,791	0,746
Decision Tree GainRatio	1706	1311	5821	1035	0,565	0,622	0,816	0,593	0,713	0,762	0,719
Decision Tree Pruning	1953	1160	5972	788	0,627	0,713	0,837	0,667	0,772	0,803	0,775
Random Forest	2369	1106	6026	372	0,681	0,864	0,844	0,762	0,854	0,850	0,854
Random Forest GainRatio	2366	1099	6033	375	0,682	0,863	0,845	0,762	0,854	0,850	0,854
Random Forest 125	2371	1115	6017	370	0,680	0,865	0,843	0,761	0,854	0,849	0,854
Random Forest 75	2369	1106	6026	372	0,681	0,864	0,844	0,762	0,854	0,850	0,854
Random Forest Oversamp	1945	692	6440	796	0,738	0,710	0,903	0,723	0,800	0,849	0,806
XGBoost	2298	909	6223	443	0,716	0,838	0,872	0,772	0,855	0,863	0,855
XGBoost 125	2314	862	6270	427	0,729	0,844	0,879	0,782	0,861	0,869	0,862
XGBoost 50	2286	1023	6109	455	0,691	0,834	0,857	0,756	0,845	0,850	0,845
XGBoost Oversamp	1921	652	6480	820	0,747	0,701	0,909	0,723	0,798	0,851	0,805
K Nearest Neighbor	2223	1457	5675	518	0,604	0,811	0,795	0,692	0,803	0,799	0,803
KNN 10	2247	1647	5485	494	0,577	0,820	0,769	0,677	0,794	0,783	0,794
KNN w	2236	1328	5804	505	0,627	0,816	0,814	0,709	0,815	0,814	0,815
KNN 10 w	2278	1565	5567	463	0,593	0,831	0,781	0,692	0,805	0,795	0,806
Naive Bayes	2340	2744	4388	401	0,460	0,853	0,615	0,598	0,724	0,681	0,734
Decision Tree	1768	1090	6042	973	0,618	0,640	0,847	0,632	0,739	0,791	0,746
Decision Tree GainRatio	1706	1311	5821	1035	0,565	0,622	0,816	0,593	0,713	0,762	0,719

Figura 2.35: Tabla con todos los resultados tomando C como clase positiva

Tomando H como clase positiva:

	TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
Decision Tree	328	591	8284	670	0,356	0,328	0,933	0,342	0,554	0,872	0,631
Decision Tree GainRatio	255	493	8382	743	0,341	0,256	0,944	0,292	0,491	0,875	0,600
Decision Tree Pruning	262	446	8429	736	0,370	0,263	0,950	0,307	0,499	0,880	0,606
Random Forest	384	100	8775	614	0,793	0,384	0,988	0,518	0,616	0,927	0,686
Random Forest GainRatio	399	109	8766	599	0,785	0,399	0,987	0,529	0,628	0,928	0,693
Random Forest 125	384	88	8787	614	0,814	0,385	0,990	0,522	0,617	0,929	0,687
Random Forest 75	384	100	8775	614	0,793	0,384	0,988	0,518	0,616	0,927	0,686
Random Forest Oversamp	672	889	7986	326	0,430	0,673	0,900	0,525	0,778	0,877	0,787
XGBoost	474	187	8688	524	0,717	0,474	0,978	0,571	0,681	0,927	0,726
XGBoost 125	496	189	8686	502	0,724	0,497	0,979	0,589	0,697	0,930	0,738
XGBoost 50	415	192	8683	583	0,684	0,416	0,978	0,517	0,638	0,922	0,697
XGBoost Oversamp	673	810	8065	325	0,454	0,674	0,909	0,543	0,783	0,885	0,792
K Nearest Neighbor	340	302	8573	658	0,529	0,340	0,965	0,414	0,573	0,902	0,653
KNN 10	278	306	8569	720	0,476	0,279	0,966	0,351	0,519	0,896	0,622
KNN w	443	409	8466	555	0,520	0,444	0,954	0,479	0,651	0,902	0,699
KNN 10 w	369	364	8511	629	0,503	0,370	0,959	0,426	0,595	0,899	0,664
Naive Bayes	128	931	7944	870	0,120	0,128	0,895	0,124	0,338	0,817	0,511
Decision Tree	328	591	8284	670	0,356	0,328	0,933	0,342	0,554	0,872	0,631
Decision Tree GainRatio	255	493	8382	743	0,341	0,256	0,944	0,292	0,491	0,875	0,600

Figura 2.36: Tabla con todos los resultados tomando H como clase positiva

Tomando P como clase positiva:

	TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
Decision Tree	968	1196	6580	1129	0,447	0,462	0,841	0,454	0,625	0,764	0,653
Decision Tree GainRatio	827	1433	6343	1270	0,366	0,394	0,816	0,380	0,567	0,726	0,605
Decision Tree Pruning	924	936	6840	1173	0,497	0,441	0,880	0,467	0,623	0,786	0,660
Random Forest	1127	539	7237	970	0,676	0,537	0,930	0,598	0,707	0,847	0,734
Random Forest GainRatio	1132	545	7231	965	0,675	0,539	0,929	0,599	0,708	0,847	0,734
Random Forest 125	1130	527	7249	967	0,682	0,539	0,932	0,602	0,709	0,849	0,736
Random Forest 75	1127	539	7237	970	0,676	0,537	0,930	0,598	0,707	0,847	0,734
Random Forest Oversamp	1128	920	6856	969	0,551	0,538	0,882	0,544	0,689	0,809	0,710
XGBoost	1174	683	7093	923	0,632	0,559	0,912	0,593	0,714	0,837	0,736
XGBoost 125	1186	668	7108	911	0,640	0,566	0,914	0,600	0,719	0,840	0,740
XGBoost 50	1122	687	7089	975	0,620	0,535	0,912	0,575	0,698	0,832	0,723
XGBoost Oversamp	1143	912	6864	954	0,556	0,545	0,883	0,551	0,694	0,811	0,714
K Nearest Neighbor	940	710	7066	1157	0,569	0,448	0,908	0,501	0,638	0,810	0,678
KNN 10	900	721	7055	1197	0,555	0,429	0,907	0,484	0,624	0,806	0,668
KNN w	1027	807	6969	1070	0,560	0,490	0,896	0,523	0,663	0,810	0,693
KNN 10 w	992	792	6984	1105	0,556	0,473	0,898	0,511	0,652	0,808	0,686
Naive Bayes	460	753	7023	1637	0,379	0,219	0,903	0,277	0,445	0,757	0,561
Decision Tree	968	1196	6580	1129	0,447	0,462	0,841	0,454	0,625	0,764	0,653
Decision Tree GainRatio	827	1433	6343	1270	0,366	0,394	0,816	0,380	0,567	0,726	0,605

Figura 2.37: Tabla con todos los resultados tomando P como clase positiva

Tomando R como clase positiva:

	TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
Decision Tree	392	566	8220	695	0,409	0,360	0,935	0,383	0,581	0,872	0,648
Decision Tree GainRatio	256	613	8173	831	0,295	0,236	0,930	0,262	0,468	0,854	0,583
Decision Tree Pruning	364	469	8317	723	0,437	0,335	0,947	0,379	0,563	0,879	0,641
Random Forest	437	255	8531	650	0,631	0,402	0,970	0,491	0,624	0,908	0,686
Random Forest GainRatio	439	229	8557	648	0,657	0,403	0,973	0,500	0,627	0,911	0,688
Random Forest 125	430	246	8540	657	0,636	0,396	0,972	0,488	0,620	0,909	0,684
Random Forest 75	437	255	8531	650	0,631	0,402	0,970	0,491	0,624	0,908	0,686
Random Forest Oversamp	745	1058	7728	342	0,413	0,685	0,880	0,516	0,776	0,858	0,782
XGBoost	496	323	8463	591	0,605	0,456	0,963	0,520	0,662	0,907	0,709
XGBoost 125	496	325	8461	591	0,604	0,456	0,963	0,520	0,663	0,907	0,710
XGBoost 50	475	354	8432	612	0,573	0,437	0,960	0,496	0,648	0,902	0,698
XGBoost Oversamp	738	977	7809	349	0,430	0,679	0,889	0,527	0,777	0,866	0,784
K Nearest Neighbor	340	264	8522	747	0,562	0,312	0,969	0,402	0,550	0,897	0,641
KNN 10	317	270	8516	770	0,540	0,292	0,969	0,379	0,532	0,895	0,630
KNN w	404	365	8421	683	0,525	0,372	0,958	0,435	0,597	0,894	0,665
KNN 10 w	376	300	8486	711	0,556	0,346	0,966	0,427	0,578	0,898	0,656
Naive Bayes	179	494	8292	908	0,265	0,164	0,944	0,203	0,394	0,857	0,554
Decision Tree	392	566	8220	695	0,409	0,360	0,935	0,383	0,581	0,872	0,648
Decision Tree GainRatio	256	613	8173	831	0,295	0,236	0,930	0,262	0,468	0,854	0,583

Figura 2.38: Tabla con todos los resultados tomando R como clase positiva

Tomando S como clase positiva:

	TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
Decision Tree	1661	1313	5610	1289	0,559	0,563	0,810	0,560	0,675	0,736	0,686
Decision Tree GainRatio	1375	1604	5319	1575	0,462	0,466	0,768	0,464	0,598	0,678	0,617
Decision Tree Pruning	1849	1510	5413	1101	0,550	0,627	0,782	0,586	0,700	0,736	0,704
Random Forest	2276	1280	5643	674	0,640	0,771	0,815	0,699	0,793	0,802	0,793
Random Forest GainRatio	2264	1291	5632	686	0,636	0,767	0,813	0,696	0,790	0,799	0,790
Random Forest 125	2281	1301	5622	669	0,637	0,773	0,812	0,698	0,792	0,800	0,793
Random Forest 75	2276	1280	5643	674	0,640	0,771	0,815	0,699	0,793	0,802	0,793
Random Forest Oversamp	1329	495	6428	1621	0,729	0,451	0,928	0,557	0,647	0,786	0,690
XGBoost	2180	1149	5774	770	0,654	0,738	0,834	0,694	0,785	0,805	0,786
XGBoost 125	2199	1138	5785	751	0,659	0,745	0,836	0,700	0,789	0,809	0,791
XGBoost 50	2128	1191	5732	822	0,641	0,721	0,828	0,679	0,773	0,796	0,775
XGBoost Oversamp	1480	567	6356	1470	0,723	0,502	0,918	0,592	0,679	0,794	0,710
K Nearest Neighbor	1991	1306	5617	959	0,603	0,674	0,811	0,637	0,739	0,770	0,743
KNN 10	1877	1310	5613	1073	0,589	0,636	0,811	0,612	0,718	0,759	0,724
KNN w	1848	1006	5917	1102	0,648	0,626	0,855	0,637	0,732	0,786	0,741
KNN 10 w	1801	1036	5887	1149	0,635	0,611	0,850	0,622	0,721	0,779	0,730
Naive Bayes	1050	794	6129	1900	0,569	0,355	0,885	0,438	0,561	0,727	0,620
Decision Tree	1661	1313	5610	1289	0,559	0,563	0,810	0,560	0,675	0,736	0,686
Decision Tree GainRatio	1375	1604	5319	1575	0,462	0,466	0,768	0,464	0,598	0,678	0,617

Figura 2.39: Tabla con todos los resultados tomando S como clase positiva

A la hora de comparar los distintos modelos podemos también usar otras medidas que miden su rendimiento general como accuracy, balanced accuracy y balanced accuracy weighted. [1]
Accuracy medirá el porcentaje de acierto del algoritmo. Esta medida puede suponer un problema en datos desbalanceados, ya que tiende a ocultar errores de clasificación en clases con pocos valores.

Balanced Accuracy es esencialmente la media de TPR. El valor de TPR para cada clase se puede interpretar como: ¿cómo de probable un individuo de esa clase será clasificado correctamente? Por tanto, Balanced Accuracy nos da una media de esta medida. Como consecuencia tenemos que las clases más pequeñas tendrán más peso que el proporcional en la fórmula. Sin embargo, si el dataset está balanceado, Accuracy y Balanced Accuracy convergen al mismo valor. Esta medida nos puede ayudar a encontrar posibles problemas en la clasificación de clases minoritarias.

Por último, Balanced Accuracy Weighted es como la medida anterior pero manteniendo la importancia de cada clase gracias a su frecuencia. En este caso, las clases de distintos tamaños tienen un efecto proporcional en el resultado.

	Accuracy	Balanced Accuracy	Balanced Accuracy Weighted
Decision Tree	0,448	0,4284	0,104
Decision Tree GainRatio	0,4476	0,3948	0,0895
Decision Tree Pruning	0,542	0,4755	0,1084
Random Forest	0,668	0,592	0,134
Random Forest GainRatio	0,668	0,595	0,134
Random Forest 125	0,668	0,591	0,1336
Random Forest 75	0,664	0,59	0,1328
Random Forest Oversamp	0,589	0,611	0,179
XGBoost	0,671	0,592	0,134
XGBoost 125	0,6777	0,622	0,1355
XGBoost 50	0,6509	0,5886	0,1301
XGBoost 125 Oversamp.	0,603	0,620	0,121
K Nearest Neighbor	0,591	0,5175	0,1182
KNN 10	0,5969	0,491	0,1138
KNN w	0,591	0,55	0,1207
KNN 10 w	0,589	0,526	0,1182
Naive Bayes	0,421	0,344	0,084

Figura 2.40: Medidas para todos los modelos

Así, podemos ver que Random Forest con sobremuestreo es el modelo que mejor clasifica dando una importancia a todas las variables en función de su frecuencia. Sin embargo, no es el algoritmo que mejor porcentaje de acierto tiene, no es el accuracy más alto. En problemas como este que son multiclase, depende de qué esperemos del algoritmo será mejor una medida o la otra. Si todas las clases son igual de importantes en el problema entonces Balanced Accuracy Weighted será una medida adecuada, en cambio si solamente queremos maximizar el número de aciertos, nos quedaremos con el accuracy.

2.6. Interpretación de los datos

Actuaremos de manera similar a como lo hicimos en el primer problema. Comenzaremos calculando la matriz de correlaciones y luego usaremos los nodos *AutoML* y *Global Feature* para extraer los atributos más relevantes para el problema.

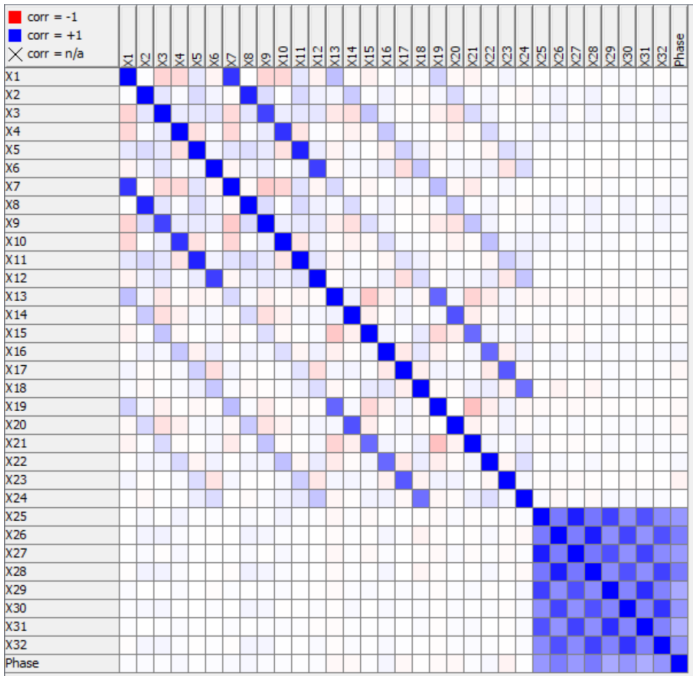

Figura 2.41:

Surrogate Generalized Linear Model:

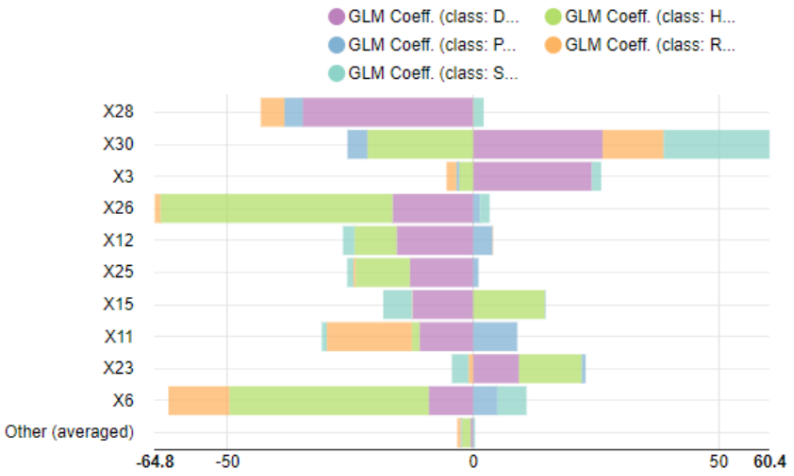

Figura 2.42

La magnitud del coeficiente del GLM indica la importancia de la característica. Un coeficiente positivo (negativo) significa que un valor de característica más alto conduce a una probabilidad más alta (más baja) del evento

Surrogate Random Forest:

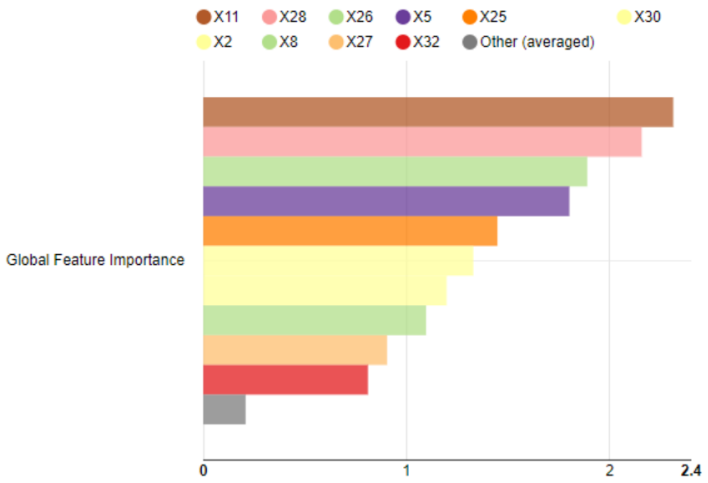

Figura 2.43

Es un modelo de Random Forest entrenado para aproximar las predicciones del modelo original. El Random Forest se entrena con los datos de entrada preprocesados de manera estándar y con los

parámetros optimizados de "Profundidad del árbol", "Número de modelos" y "Tamaño mínimo del nodo hijo".

La importancia de las características se calcula contando cuántas veces se ha seleccionado una característica para una división y en qué posición (nivel) entre todas las características disponibles (candidatas) en los árboles del bosque aleatorio. Un valor más alto indica una mayor importancia de la característica.

Con estas tres tablas podemos sacar algunas conclusiones importantes sobre los datos. Las características desde X25 hasta X32 están fuertemente correlacionadas entre sí y también con la clase. Es normal que algunas de las variables estén relacionadas entre sí, ya que son aceleraciones de distintas partes del cuerpo.

Sin embargo, gracias a los nodos *AutoML* y *Global Feature*, descubrimos que no son los atributos más importantes para determinar la clase. Existen algunos atributos como X3 que influye mucho en la clase D, al igual que le pasa a X6 con la clase H.

A partir de la figura 2.42 podemos deducir que ciertos atributos son más influyentes en una clase que en otra. Por ejemplo para la clase D, X28, X30 ý X3 son las variables que determinan más esta clase.

2.7. Contenido adicional

2.8. Bibliografía

https://arxiv.org/pdf/2008.05756.pdf
https://www.knime.com/blog/from-modeling-to-scoring-confusion-matrix-and-class-statistics

3. Bank Marketing

3.1. Introducción

El conjunto de datos *bank-full* desea predecir si un cierto cliente va a comprar un nuevo producto o no. La BD consiste en 45211 clientes con 17 atributos cada uno, como por ejemplo: edad, trabajo, educación, balance, duración...
Observamos que es un problema de clasificación binario, sólo hay 2 clases (*yes* / *no*), en el que hay atributos de todo tipo. No hay valores perdidos pero sí que hay un gran desbalance entre clases. no (39922) frente a yes (5289). Por lo que veremos cómo responden los distintos modelos a este problema.

3.2. Procesado de datos

En este problema no aplicaremos ningún preprocesamiento en general de los datos. Solamente debemos preocuparnos por los algoritmos que tengan ciertas restricciones para las variables, ya sean numéricas o nominales.

Como ya hicimos para el primer problema, debemos realizar un preprocesamiento específico para dos algoritmos: XGBoost y KNN. En el caso de KNN, es esencial normalizar los atributos para que todos tengan el mismo peso en el algoritmo. Además, haremos uso del one-hot encoding (nodo one to many en KNIME) para tener solamente variables numéricas, ya que KNIME no permite atributos de ambos tipos en KNN.

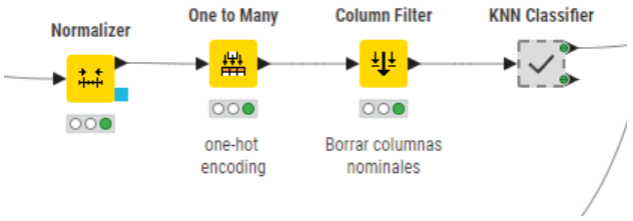

Figura 3.1: Captura de KNIME del preprocesamiento para KNN

El algoritmo XGBoost también requiere que las características sean numéricas, así que aplicaremos lo mismo que para KNN, pero el normalizado no es necesario.

Figura 3.2: Captura de KNIME del preprocesamiento para XGBoost

3.3. Resultados obtenidos

3.3.1. Decision Tree

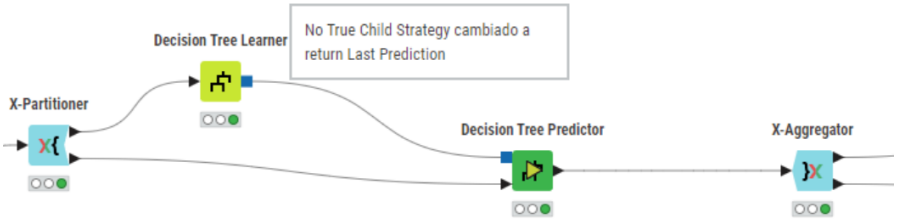

Figura 3.3: Captura de KNIME del flujo de Decision Tree

real \ predicho	Canceled	Not_Canceled
Canceled	2523	2766
Not_Canceled	2413	37509

Figura 3.4: Matriz de confusión de Decision Tree

TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
2523	2413	37509	2766	0,511	0,477	0,940	0,493	0,669	0,885	0,708

Figura 3.5: Medidas de Decision Tree

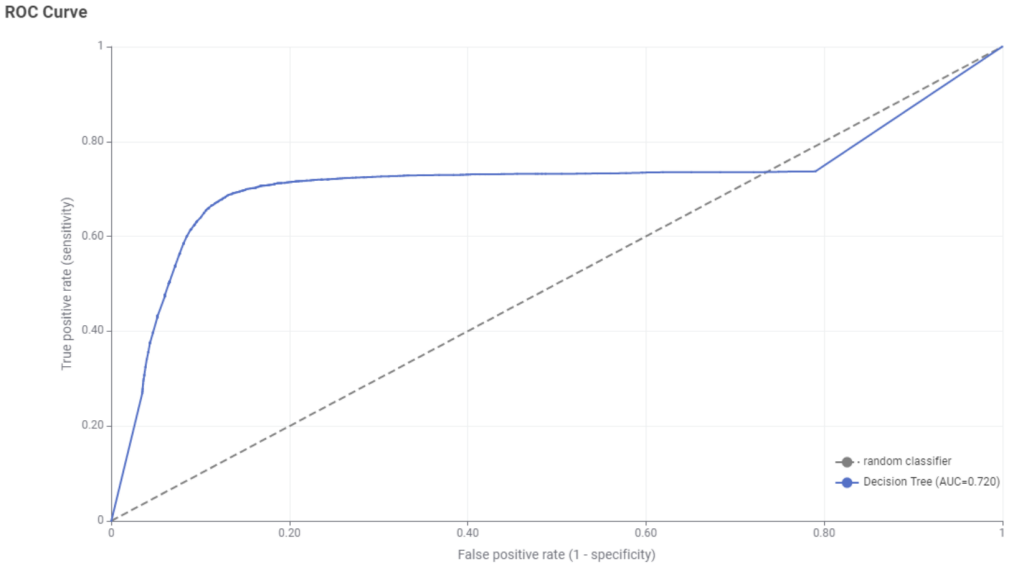

Figura 3.6: Curva ROC de Decision Tree

fold #	Error in %	Size of Test Set	Error Count	Model size
0	11,335	9043	1025	2639
1	10,949	9042	990	2796
2	11,369	9042	1028	2778
3	11,303	9042	1022	2757
4	12,320	9042	1114	2714

Figura 3.7: Tabla de errores y tamaño del modelo Decision Tree

3.3.2. Random Forest

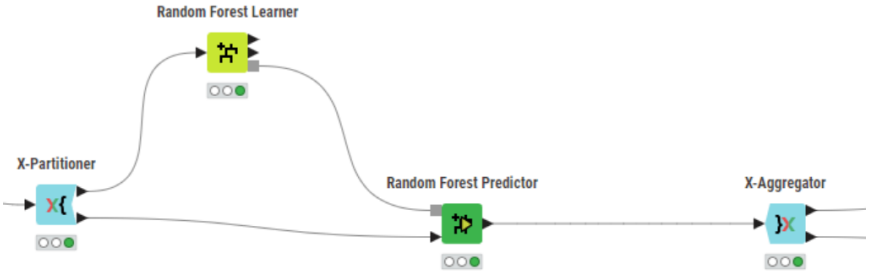

Figura 3.8: Captura de KNIME del flujo de Random Forest

real \ predicho	Canceled	Not_Canceled
Canceled	2134	3155
Not_Canceled	1064	38858

Figura 3.9: Matriz de confusión de Random Forest

TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
2134	1064	38858	3155	0,667	0,403	0,973	0,503	0,627	0,907	0,688

Figura 3.10: Medidas de Random Forest

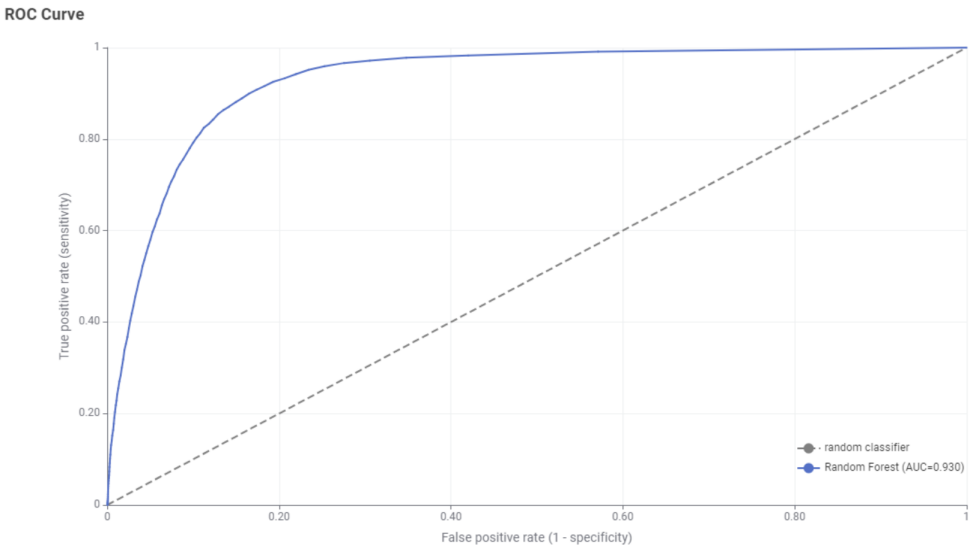

Figura 3.11: Curva ROC de Random Forest

En Random Forest se entrenan 100 árboles de decisión por partición.

fold #	Error in %	Size of Test Set	Error Count
0	9,488	9043	858
1	8,704	9042	787
2	9,113	9042	824
3	9,710	9042	878
4	9,644	9042	872

Figura 3.12: Tabla de errores de Random Forest

3.3.3. XGBoost

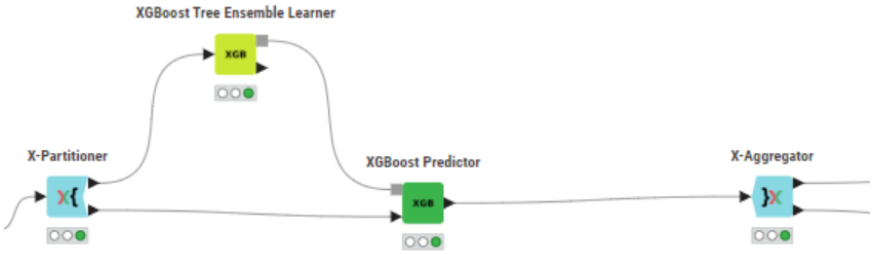

Figura 3.13: Captura de KNIME del flujo de XGBoost

real \ predicho	Canceled	Not_Canceled
Canceled	2618	2671
Not_Canceled	1499	38423

Figura 3.14: Matriz de confusión de XGBoost

TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
2618	1499	38423	2671	0,636	0,495	0,962	0,557	0,690	0,908	0,729

Figura 3.15: Medidas de XGBoost

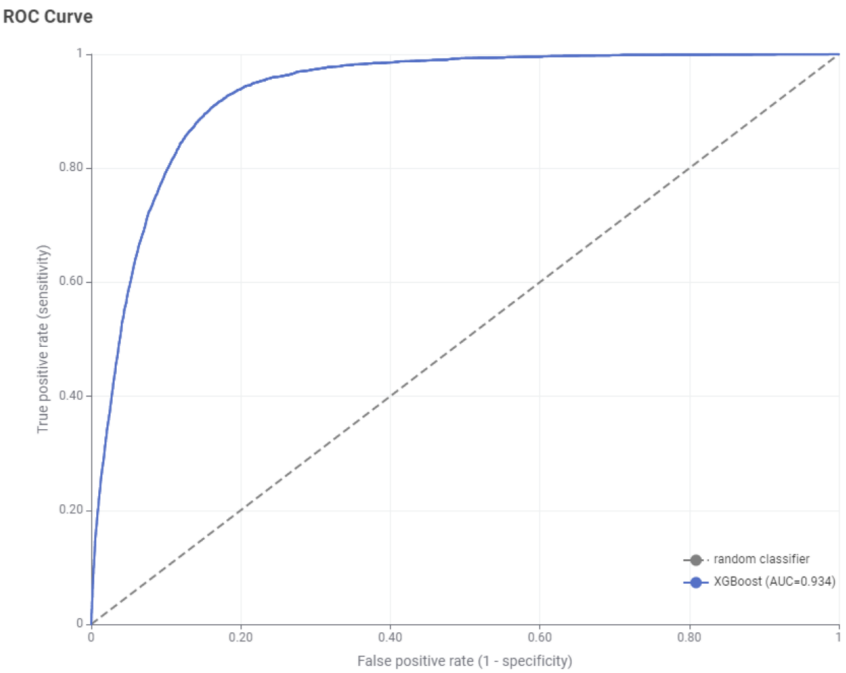

Figura 3.16: Curva ROC de XGBoost

En XGBoost se entrenan 100 árboles de decisión en el conjunto de boosting (por partición).

fold #	Error in %	Size of Test Set	Error Count
0	9,0457	9043	818
1	8,8255	9042	798
2	9,0909	9042	822
3	9,3232	9042	843
4	9,8319	9042	889

Figura 3.17: Tabla de errores de XGBoost

3.3.4. K Nearest Neighbor

Figura 3.18: Captura de KNIME del flujo de KNN

real \ predicho	Canceled	Not_Canceled
Canceled	1314	3975
Not_Canceled	940	38982

Figura 3.19: Matriz de confusión de KNN

TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
1314	940	38982	3975	0,583	0,248	0,976	0,348	0,493	0,891	0,612

Figura 3.20: Medidas de KNN

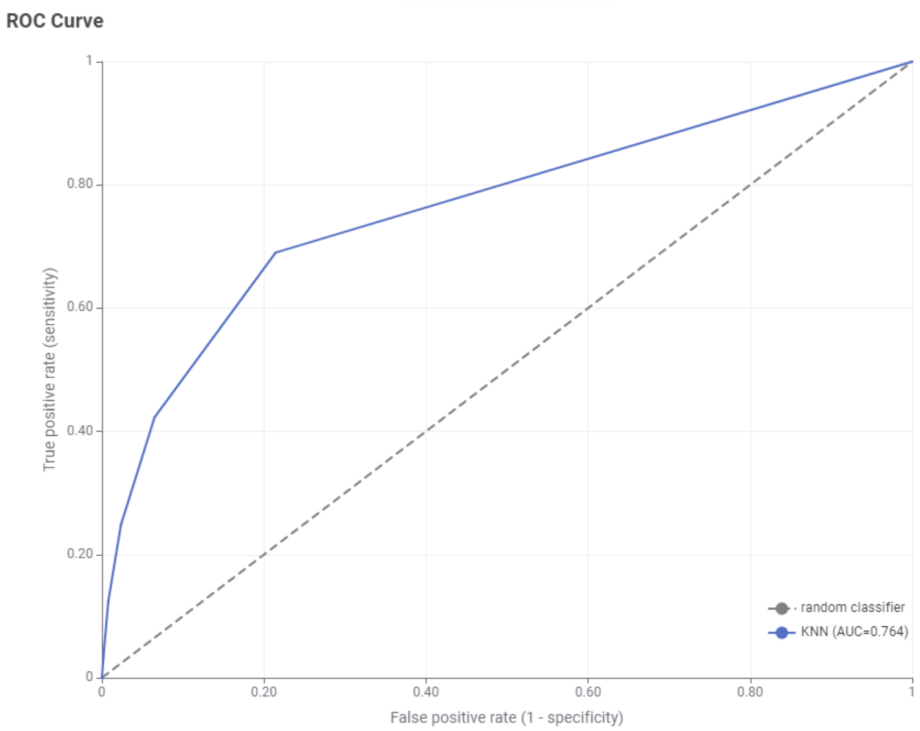

Figura 3.21: Curva ROC de KNN

fold #	Error in %	Size of Test Set	Error Count
0	10,505	9043	950
1	10,263	9042	928
2	10,683	9042	966
3	11,104	9042	1004
4	11,800	9042	1067

Figura 3.22: Tabla de errores de KNN

3.3.5. Naive-Bayes

Figura 3.23: Captura de KNIME del flujo de Naive-Bayes

real \ predicho	Canceled	Not_Canceled
Canceled	1832	3457
Not_Canceled	1622	38300

Figura 3.24: Matriz de confusión de Naive-Bayes

TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
1832	1622	38300	3457	0,530	0,346	0,959	0,419	0,576	0,888	0,653

Figura 3.25: Medidas de Naive-Bayes

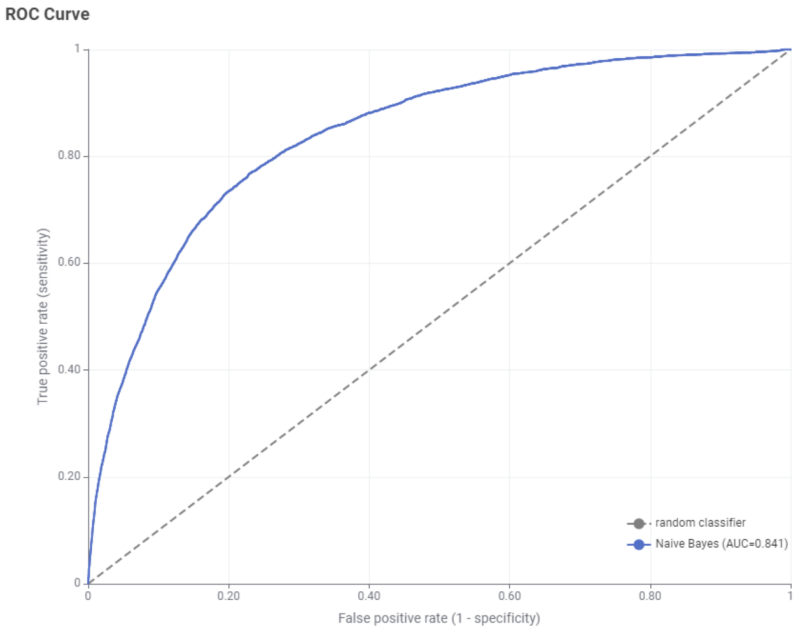

Figura 3.26: Curva ROC de Naive-Bayes

fold #	Error in %	Size of Test Set	Error Count
0	10,483	9043	948
1	10,938	9042	989
2	10,794	9042	976
3	11,159	9042	1009
4	12,796	9042	1157

Figura 3.27: Tabla de errores de Naive-Bayes

3.4. Configuración de algoritmos

3.5. Análisis de resultados

	TP	FP	TN	FN	PPV	TPR	TNR	F1-score	G-mean	Accuracy	AUC
Decision Tree	9424	2281	22109	2461	0,805	0,793	0,906	0,799	0,848	0,869	0,850
Random Forest	9634	1268	23122	2251	0,884	0,811	0,948	0,846	0,877	0,903	0,879
XGBoost	9535	1504	22886	2350	0,864	0,802	0,938	0,832	0,868	0,894	0,870
K Nearest Neighbor	8656	2267	22123	3229	0,792	0,728	0,907	0,759	0,813	0,848	0,818
Naive Bayes	11767	21123	3267	118	0,358	0,990	0,134	0,526	0,364	0,414	0,562

Figura 1.47: Tabla conjunto de los resultados de todos los modelos

Decision Tree:

- Pros: Buen accuracy (0,869) y AUC (0,85).
- Contras: PPV y F1-score podrían mejorarse.

En general, obtenemos unos resultados sólidos con árboles de decisión. La poda del árbol parece mejorar la generalización, pero reduce el rendimiento en términos de precisión.

Random Forest:

- Pros: Valores muy altos de TNR y AUC. Es el algoritmo con mayor tasa de acierto.
- Contras: Resultados similares con distintos tamaños, por lo que probablemente podamos reducir aún más el modelo sin empeorarlo.

XGBoost:

- Pros: AUC y accuracy muy altos.
- Contras: Rendimiento muy similar con diferente número de árboles, por lo que se podría reducir aún más el modelo.

KNN:

- Pros: Modelo sencillo con el que el accuracy es alto.
- Contras: No se construye un modelo, el modelo es la propia BD.

Naive Bayes:

- Contras: Son los peores resultados de todos con diferencia.

Que se obtengan tan malas medidas con Naive Bayes significa que las variables tienen una fuerte dependencia condicional respecto a la clase.

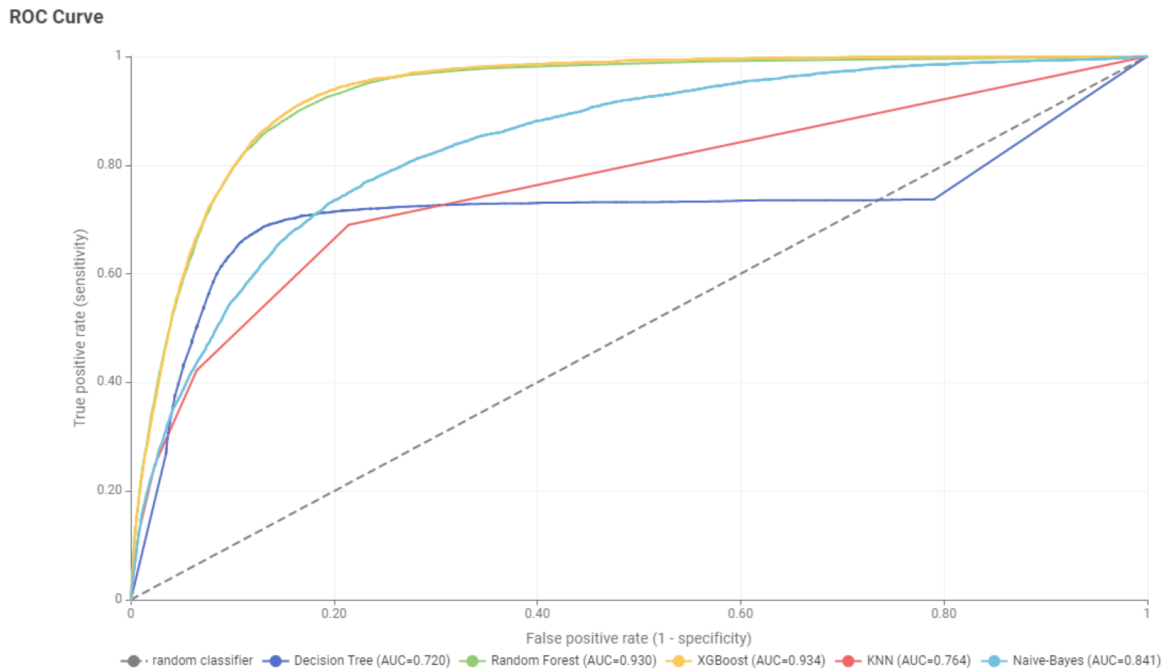

Figura 1.48: Curvas ROC de todos los modelos

La curva ROC representa la relación entre la tasa de verdaderos positivos (TPR) y la tasa de falsos positivos (FPR). Observamos que las curvas más altas son las de Random Forest y XGBoost. Esto puede significar dos cosas, que tienen mayor TPR en comparación con los demás modelos en todos los umbrales de clasificación, es decir, son más efectivos a la hora de identificar TP en el conjunto de datos. Por otro lado, también puede significar que tienen una FPR en comparación con los demás en todos los umbrales, lo que indica que son más precisos al evitar clasificar incorrectamente instancias negativas como positivas.

Podemos distinguir unos 5 grupos de curvas en el gráfico. En orden ascendente:

- Naive Bayes. Destacamos su bajo rendimiento para la mayoría de valores del umbral, salvo para valores extremos para los cuales la curva supera a los Decision Tree.
- KNN con pesos (para ambos valores de K. A pesar de tener mejor precisión que sin pesos, tienen un AUC inferior, lo que implica que su capacidad de discrimación entre clases positiva y negativa es inferior.
- KNN con K=5 y K=10, El pico de la curva no es exagerado, cambia lentamente de vertical a horizontal, lo que sugiere que tienen un buen rendimiento pero la capacidad de ajuste a diferentes

umbrales de clasificación no es tan pronunciada como otros modelos.

- Decision Trees. El rendimiento al principio es muy alto pero finalmente se estabiliza. Indicando que los Decision Trees tienen una buena capacidad de ajuste inicial, pero su rendimiento puede que no sea tan consistente en umbrales extremos.
- XGBoost y Random Forests.

3.6. Interpretación de los datos

En primer lugar, comenzamos calculando la matriz de correlaciones de todos los atributos:

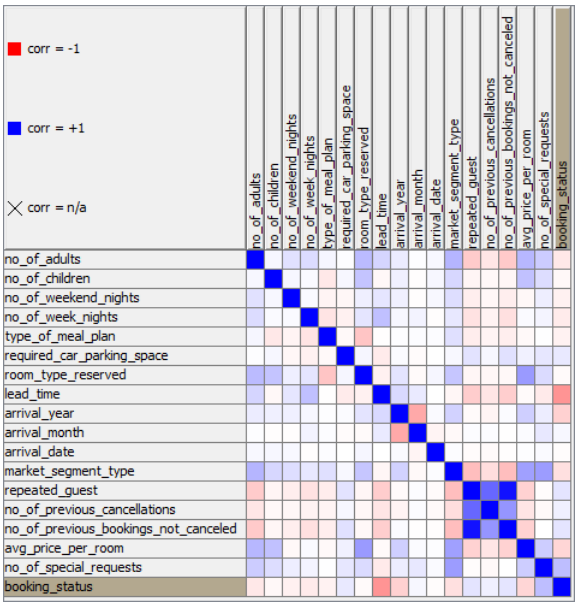

Figura 1.49: Matriz de correlaciones entre los atributos

Podemos sacar algunas conclusiones sencillas como por ejemplo que si el cliente ha repetido influye en el número de cancelaciones o no cancelaciones previo, que el precio medio por habitación está correlacionado positivamente con el tipo de habitación y también con el tipo de segmento de mercado.

La única variable que está correlacionada significativamente con el estado de la reserva (clase) es *lead_time*, que es el número de días entre la fecha de reserva y la fecha de inicio de la estancia.

A continuación, vamos a visualizar los primeros niveles de algún árbol de decisión. En KNIME generamos 5 árboles distintos, uno para cada partición de entrenamiento, pero he observado que los primeros niveles son siempre muy similares.

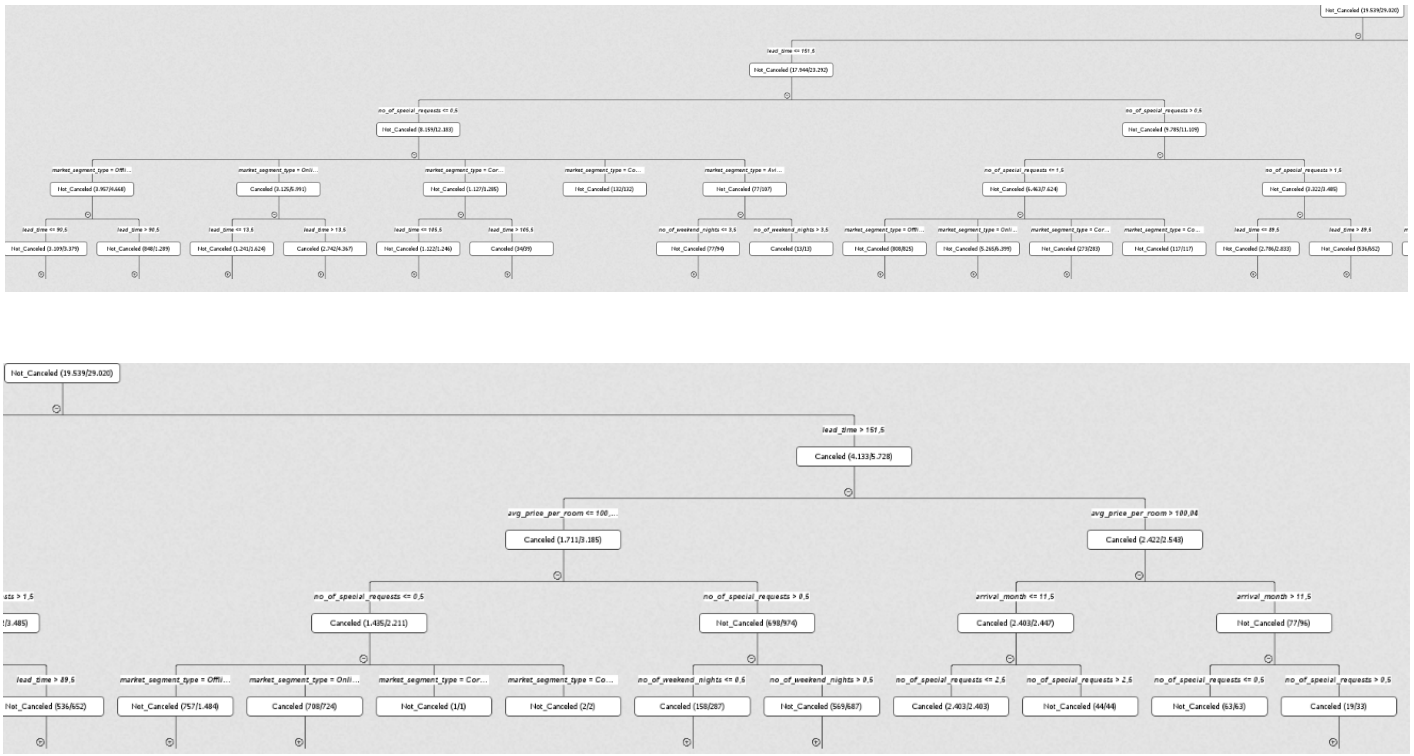

Figura 1.50: Primeros niveles de un árbol de decisión

Observamos que la primera variable por la que se divide el árbol es *lead_time*, tendrá el índice Gini menor. Luego se divide por el número de peticiones especiales y por el precio medio en la otra rama. Más tarde aparecen atributos como *market_segment_type*, *arrival_month* y *no_of_weekend_nights.*

Vamos a ver qué ocurre con algún árbol que use GainRatio

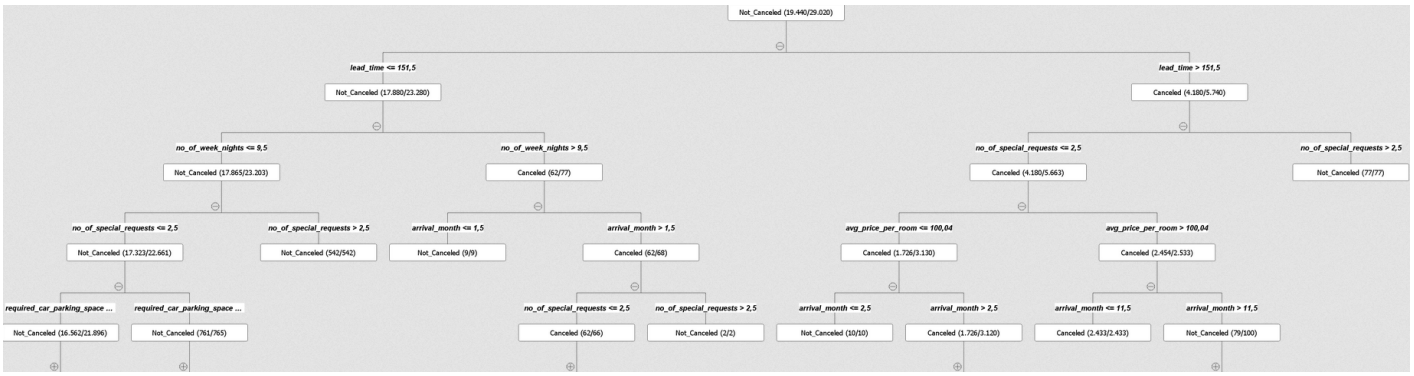

Figura 1.51: Primeros niveles de un árbol de decisión que usa GainRatio

En este caso las divisiones del árbol siguen:

- 1. *lead_time*
- 2. *no_of_week_nights / no_of_special_requests*
- 3. *no_of_special_requests / arrival_month / avg_price_per_room*
- 4. *required_car_parking_space / no_of_special_requests / arrival_month*

Ambas versiones de árbol de decisión muestran una preferencia por la variable *lead_time* como la primera variable de división. Esto sugiere que es un atributo importante para la toma de decisiones de los modelos.

Además, las dos variantes también consideran atributos comunes, como *no_of_special_requests* y *arrival_month, en niveles posteriores,* por lo que estos atributos tendrán su importancia.

variables	#splits (level 0)	#splits (level 1)	#splits (level 2)	#candidates (level 0)	#candidates (level 1)	#candidates (level 2)	importance (all levels)
lead_time	20	40	76	20	46	95	2,6697
no_of_special_requests	25	22	37	29	35	86	1,92
avg_price_per_room	15	23	38	29	43	83	1,509
arrival_year	12	26	28	22	54	97	1,315
market_segment_type	8	19	42	20	51	94	1,219
arrival_month	5	15	38	20	49	98	0,9431
no_of_adults	2	12	20	24	48	93	0,548
type_of_meal_plan	1	8	27	20	43	94	0,523
repeated_guest	4	6	9	18	46	90	0,452
no_of_week_nights	1	7	23	26	54	102	0,393
no_of_weekend_nights	0	8	19	21	44	101	0,369
no_of_previous_bookings_not_canceled	4	5	6	28	44	98	0,317
required_car_parking_space	3	3	5	30	55	85	0,213
arrival_date	0	2	8	30	48	98	0,123
room_type_reserved	0	1	8	24	50	101	0,099
no_of_children	0	1	6	20	42	96	0,086
no_of_previous_cancellations	0	2	1	19	48	89	0,0529

Figura 1.52: Tabla con la importancia de los atributos en Random Forest

Con los datos de la importancia de las variables en Random Forest obtenemos unas conclusiones similares, *lead_time* sigue siendo la característica más importante, seguida por *avg_price_per_room, arrival_year* y *market_segment_type.*

Obtenemos también las variables más importantes del modelo XGBoost ordenadas por la columna Total Gain, la cual indica la ganancia a lo largo de todas las divisiones en las que la variable se ha usado

Feature Name	Weight	Gain	Cover	Total Gain	Total Cover
lead_time	1700	10,58	968,0111158	17986,14971	1645618,897
avg_price_per_room	1676	4,45501	1040,861958	7468,007786	1744484,642
no_of_special_requests	418	12,505	1644,149713	5227,248666	687254,58
arrival_month	774	4,3531	921,2866843	3369,353953	713075,8936
arrival_date	922	1,4989	476,141445	1381,165949	439002,4123
no_of_weekend_nights	356	3,5557	280,1163539	1265,83043	99721,422
no_of_adults	176	6,8638	669,3716345	1207,922464	117809,4077
no_of_week_nights	468	2,32848	912,0184797	1089,706751	426824,6485
arrival_year	110	7,722299	921,1738153	849,4527766	101329,1197
required_car_parking_space	50	11,3687	4601,240093	568,4389337	230062,0046
no_of_children	60	1,1212	696,6492175	67,27268085	41798,95305
repeated_guest	10	4,937	3760,979512	49,37597683	37609,79512
no_of_previous_bookings_not_canceled	26	1,332	3694,112458	34,65278002	96046,9239
no_of_previous_cancellations	16	0,629	9,264119311	10,06764351	148,225909

Figura 1.53: Tabla con la importancia de los atributos en XGBoost

Por último, hacemos uso de los nodos *AutoML* y *Global Feature Importance,* los cuales usan diferentes modelos como Redes Neuronales, Random Forestm, Deep Learning (Keras) ..., para obtener las características más importantes del problema. [1]

Surrogate Random Forest:

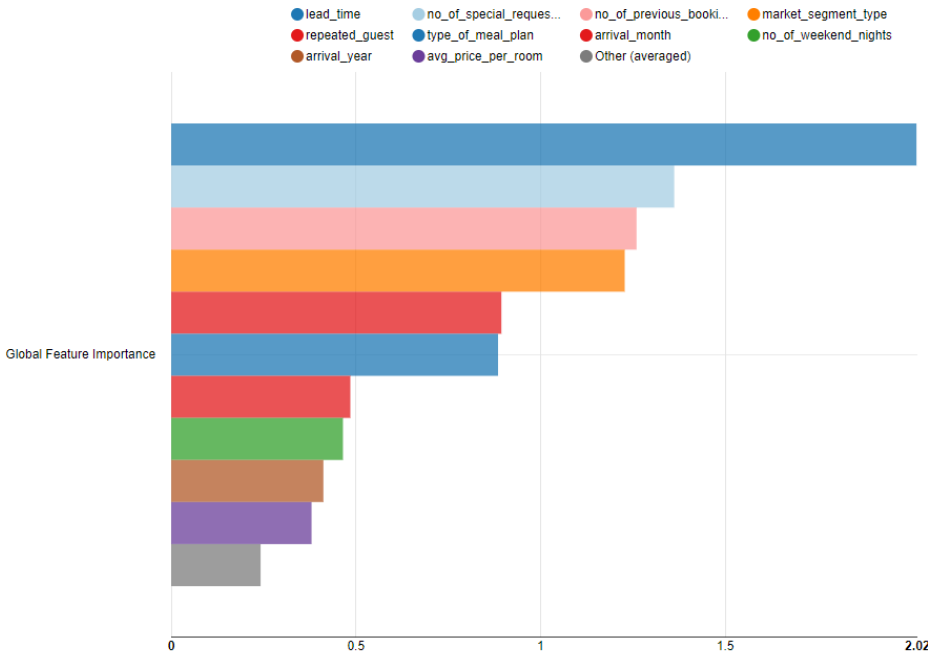

Figura 1.54: Importancia Global de las Características usando Surrogate Random Forest

Es un modelo de Random Forest entrenado para aproximar las predicciones del modelo original. El Random Forest se entrena con los datos de entrada preprocesados de manera estándar y con los parámetros optimizados de "Profundidad del árbol", "Número de modelos" y "Tamaño mínimo del nodo hijo".

La importancia de las características se calcula contando cuántas veces se ha seleccionado una característica para una división y en qué posición (nivel) entre todas las características disponibles (candidatas) en los árboles del bosque aleatorio. Un valor más alto indica una mayor importancia de la característica.

Permutation Feature Importance:

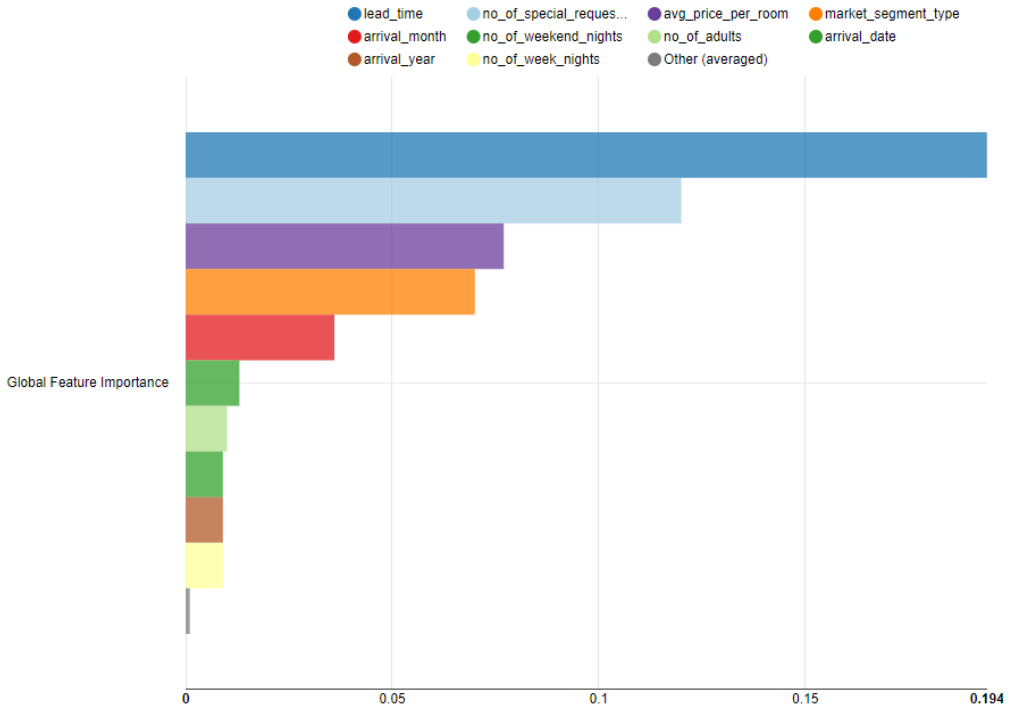

Figura 1.55: Importancia Global de las Características usando Permutation Feature Importance

Esta medida indica la diferencia entre la puntuación de rendimiento del modelo estimado en predicciones utilizando todas las características originales y la puntuación de rendimiento del modelo estimado en predicciones utilizando todas las características originales excepto una que se permuta de manera aleatoria. Si una característica se permuta varias veces, se calcula la diferencia promedio.

Una diferencia de puntuación grande indica que la característica era importante para la predicción, ya que romper la relación entre esta característica y el objetivo disminuyó el rendimiento del modelo. Una diferencia cercana a 0 significa que permutar la característica no disminuye el rendimiento del modelo. La diferencia negativa indica que, por alguna razón, permutar la característica aumenta el rendimiento del modelo.

Como conclusión, los atributos que identifican más a la clase son *lead_time, no_of_special_requests, avg_price_per_room, market_segment_type, no_of_previous_bookings_not_canceled*

3.7. Contenido adicional

3.8. Bibliografía

https://machinelearningmastery.com/classification-accuracy-is-not-enough-more-performance-measures-you-can-use/
Fawcett, T. (2006). An introduction to ROC analysis. Pattern recognition letters, 27(8), 861-874
[1] Understand Your ML Model: Global Feature Importance | KNIME