UNIVERSIDAD DE GRANADA

Inteligencia de Negocio

PRÁCTICA 1

Análisis Predictivo Mediante Clasificación.

José Antonio Fernández Aranda

Subgrupo 1

jantoniofer@correo.ugr.es

Índice

Problema 1: Predicción de aprobación de créditos

1. Introducción

El objetivo de este proyecto es desarrollar modelos de aprendizaje automático para predecir el estado de aprobación de préstamos bancarios. Este problema de clasificación se enfoca en determinar si un préstamo será aprobado o no, basándonos en diferentes características demográficas y financieras del solicitante. Para ello se empleará un análisis de los 5 algoritmos detallados a continuación, pero antes se hará un análisis previo de estudio de los datos. Algunas de las características generales de los datos son:

El conjunto de datos cuenta con 32,581 instancias y 12 variables que representan características relevantes para la evaluación de un crédito. A continuación, se detallan las principales características de estas variables:

- Person_age: Edad del solicitante en años, con un rango típico de 20 a 30 años.
- Person_income: Ingresos anuales del solicitante en dólares estadounidenses (USD).
- Person_home_ownership: Situación de la propiedad de la vivienda, una variable categórica que puede tomar los valores "Alquiler", "Propiedad" o "Hipoteca".
- Person_emp_length: Duración del empleo en años. Esta variable tiene 895 valores faltantes.
- Loan_intent: Finalidad del préstamo, una variable categórica con valores como "Educación", "Médico" o "Personal".
- Loan_grade: Grado de riesgo asignado al préstamo, que indica la solvencia del solicitante, categorizado en niveles de "A" a "G".
- Loan_amnt: Monto total del préstamo solicitado por el solicitante en USD.
- Loan_int_rate: Tasa de interés asociada al préstamo. Esta variable presenta 3,116 valores faltantes.
- Loan_status: Estado de aprobación del préstamo (aprobado o no aprobado). Esta es nuestra variable objetivo, de tipo categórica.
- Loan_percent_income: Porcentaje de los ingresos del solicitante destinado al pago del préstamo.
- Cb_person_default_on_file: Indica si el solicitante tiene antecedentes de impago ("Y" para sí, "N" para no).
- Cb_person_cred_hist_length: Duración del historial crediticio del solicitante en años.

Tipos de Variables

Las 12 variables del conjunto de datos se dividen en los siguientes tipos:

- 5 variables enteras.
- 3 variables de tipo flotante.
- 4 variables categóricas.

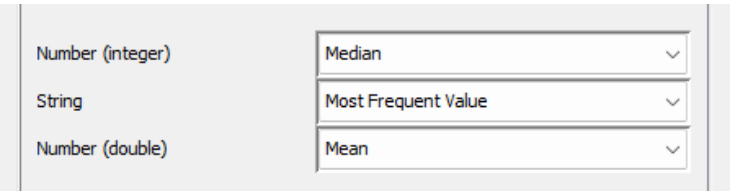
Particularidades del Conjunto de Datos

Valores Faltantes: Las variables Person_emp_length y Loan_int_rate contienen 895 y 3116 valores faltantes, respectivamente. Estos valores faltantes pueden afectar la precisión de los

modelos y, por lo tanto, deben ser tratados adecuadamente durante el procesamiento de datos.

Para ello se ha usado el nodo de Knime *Missing Value* con los siguientes valores:

- Para el caso de disponer de número enteros se realiza la mediana del conjunto de datos para establecer el valor de esta tupla.
- Para el caso de disponer de cadenas de String se ha seguido el criterio de el valor más frecuente.
- Y para el caso de disponer de números dobles se ha realizado la media aritmética de los valores del conjunto de datos para establecer el valor asignado a ese caso.



Desbalanceo de Clases: Dado que la variable objetivo *Loan_status* es de tipo categórica, se debe verificar la proporción de las clases (aprobado y no aprobado) para identificar si existe un desbalance significativo. En este caso podemos observar aplicando el nodo bar chart de la clase loan_status lo siguiente:

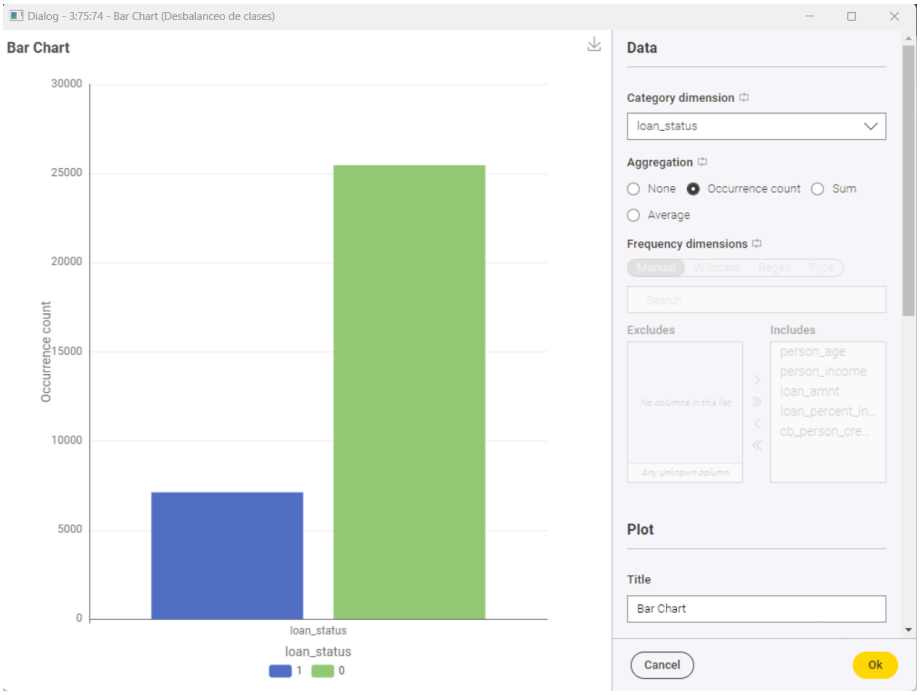


Figura 1 podemos ver Distribución de clases en la variable Loan_status. En dicha figura observamos el desbalance de clases, donde la mayoría de los préstamos se encuentran en la categoría '0' (no aprobado), mientras que una menor proporción corresponde a la categoría '1' (aprobado).

Relevancia de las Variables: Se espera que variables como Person_income, Loan_grade, Loan_int_rate y Cb_person_default_on_file tengan un impacto significativo en la predicción del estado del préstamo, ya que están directamente relacionadas con la solvencia del solicitante.

Eliminación de Tuplas con Valores Anómalos o Fuera de Rango: Debido a que la descripción del dataset, tiene una explicación diferente a la realidad de los datos, voy a realizar un filtrado de diferentes variables, para tratar menos datos, consiguiendo así un modelo más ligero y con menos carga computacional siempre y cuando no pierda predicción ni calidad el algoritmo. Algunas de las variables que se han modificado son las siguientes:

- Para la variable person_age, se ha establecido una cota inferior de 18 años y una cota superior de 100 años.

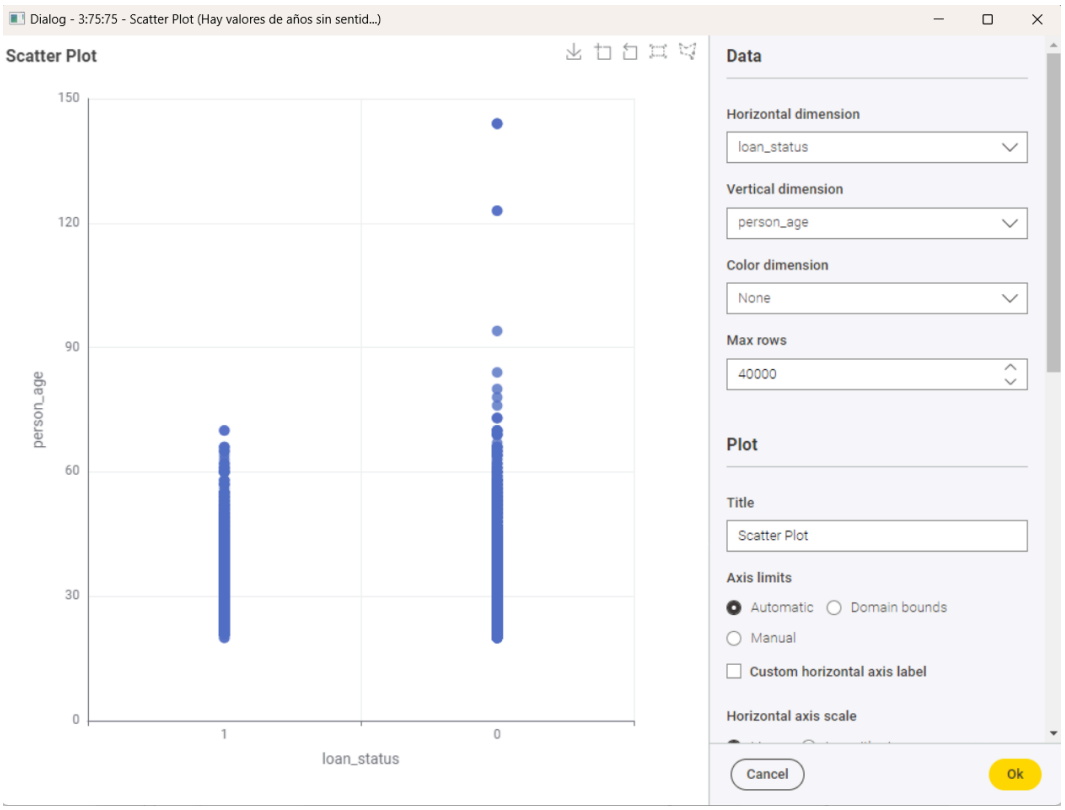


Figura 2 podemos ver la relación entre edad del solicitante y estado de aprobación del préstamo. En dicha figura observamos la distribución de la variable "person_age" en función del "loan_status", donde se aprecia que la edad de los solicitantes no parece tener una relación visible con la aprobación o rechazo del préstamo, ya que los valores de edad se concentran de forma similar en ambas clases (aprobado y no aprobado).

- Para la variable person_emp_length se ha establecido una copa superior de 100 años ya que la vida de una persona no suele superar ese tamaño.

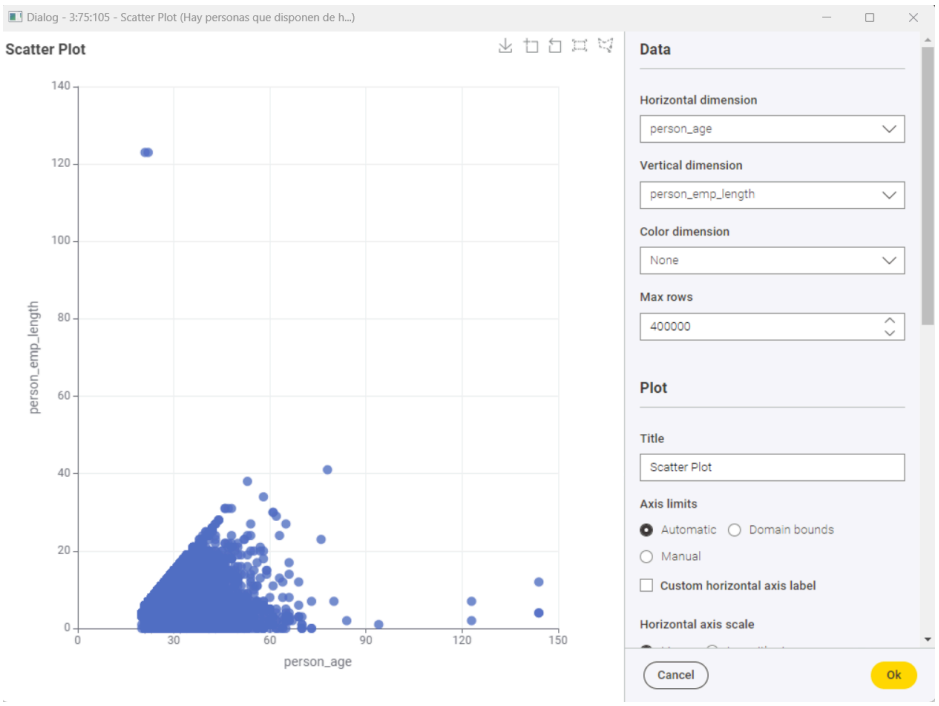
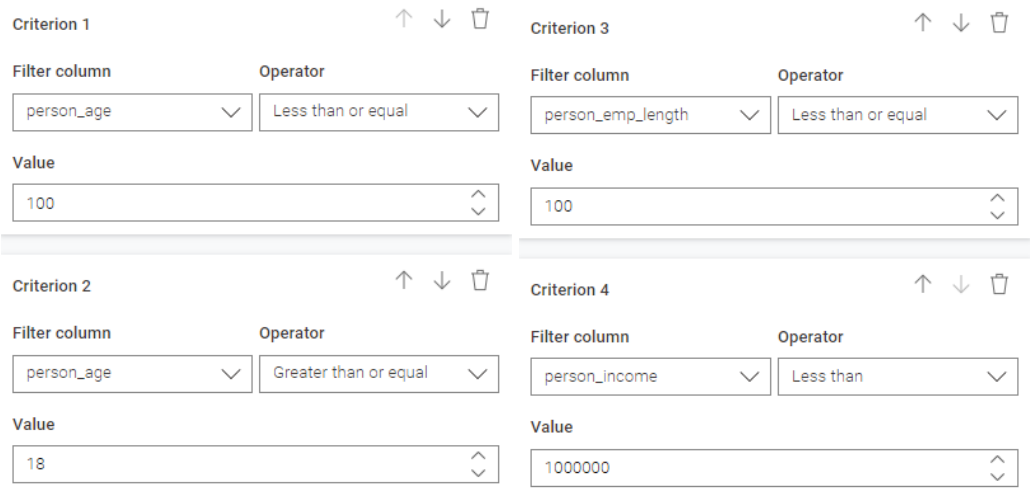


Figura 3 podemos ver la relación entre la edad del solicitante y la duración del empleo. En dicha figura observamos que la duración del empleo ("person_emp_length") tiende a aumentar con la edad, mostrando mayor variabilidad en solicitantes jóvenes (especialmente entre 20 y 40 años) y estabilizándose en valores más bajos para edades mayores, lo cual podría indicar un patrón de antigüedad laboral en personas de menor edad.

- Siguiendo con la descripción del conjunto de datos, se establece el rango de ingresos de los solicitantes del crédito, es cierto que aquí se han encontrado diferencias con las descripciones del caso con la realidad, aún así he decidido limpiar aquellos valores que se van demasiado del rango para poder aplicar en un rango y obtener mejores resultados en la normalización.

Todas estas reglas han sido recogidas en el nodo *Row Filter* de la siguiente forma:



Con relación al conjunto de datos esto ha provocado que se reduzcan las instancias de 32581 a 31671 filas. El filtrado por columnas después de numerosas pruebas en distintos algoritmos se ha establecido a elección de cada caso concreto.

Por último, para abordar el problema de la predicción del estado de aprobación de préstamos en este conjunto de datos, se seleccionaron los siguientes algoritmos de aprendizaje automático. Cada uno aporta características únicas que permiten explorar diferentes enfoques y mejorar la precisión de la predicción:

- Árbol de Decisión (Decision Tree): Los árboles de decisión son especialmente útiles para interpretar y visualizar las decisiones de clasificación. Este algoritmo permite desglosar los factores clave que afectan la aprobación o rechazo de un préstamo, proporcionando una visión clara de las variables más influyentes.

- K-Vecinos más Cercanos (K-NN): El algoritmo de K-vecinos más cercanos es útil para identificar patrones de similitud entre los solicitantes de crédito. Al clasificar una solicitud en función de las solicitudes anteriores similares, este algoritmo permite hacer predicciones precisas, especialmente cuando las características importantes tienen una influencia clara y directa en la decisión.

- SGD (Stochastic Gradient Descent): El gradiente descendente estocástico es un método de optimización ampliamente usado en problemas de clasificación binaria. Es eficiente para manejar grandes volúmenes de datos y permite la actualización rápida de los parámetros, lo cual es ventajoso en este caso, ya que el conjunto de datos cuenta con una gran cantidad de instancias.

- Random Forest: Los bosques aleatorios combinan múltiples árboles de decisión para mejorar la precisión y reducir el sobreajuste. Este enfoque es robusto y eficaz para capturar relaciones no lineales en conjuntos de datos mixtos, como el de este problema, que incluye variables numéricas y categóricas.

- Naive Bayes: Naive Bayes es un algoritmo basado en la probabilidad que es eficaz para problemas de clasificación, especialmente cuando las variables son independientes entre sí. A pesar de su simplicidad, es un modelo potente para problemas de clasificación binaria y permite obtener predicciones rápidas y precisas en un conjunto de datos con características mixtas.

Cada uno de estos algoritmos fue seleccionado para proporcionar un enfoque diverso en la predicción de aprobaciones de préstamos, permitiendo evaluar diferentes técnicas y obtener una perspectiva completa sobre los factores determinantes en la toma de decisiones.

2. Procesado de Datos

Para asegurar la calidad y precisión de los modelos predictivos, fue necesario realizar un preprocesamiento exhaustivo de los datos. Este proceso incluyó el tratamiento de valores faltantes, la eliminación de valores anómalos y el ajuste de desbalanceo de clases. A continuación, se detallan los pasos de procesamiento realizados:

Tratamiento de Valores Faltantes

Anteriormente mencionamos, que las variables person_emp_length y loan_int_rate, se encontraban con valores huecos o vacíos, para ellos se ha empleado las siguientes estimaciones:

- person_emp_length: Dado que la duración del empleo es una variable numérica, los valores faltantes se imputaron utilizando la mediana de los valores conocidos. La mediana se eligió en lugar de la media para reducir el efecto de valores atípicos.
- loan_int_rate: La tasa de interés es una variable crucial en el análisis de préstamos. Dado que el interés puede estar relacionado con el grado de riesgo (Loan_grade), los valores faltantes de Loan_int_rate se imputaron utilizando la media de las tasas de interés dentro de cada grupo de Loan_grade. Esto permite una imputación más precisa que refleja la relación entre estas dos variables.

Tratamiento del Desbalanceo de Clases

Para el tratamiento de balanceo de clases se ha creado en todos los algoritmos, concretamente en el metanodo que gestiona cada algoritmo un X particioner, para evaluar el rendimiento de los modelos de manera más confiable.

Para combatir el desbalanceo de la variable clase loan_status dispone de los siguientes atributos:

- La validación cruzada con 10 pliegues, que permite una evaluación robusta.
- El muestreo estratificado, que asegura la representación de ambas clases en cada pliegue.
- La especificación de la columna de clase, loan_status, como referencia para mantener la proporción de clases, ya que es nuestra variable objetivo.
- La semilla aleatoria para garantizar la reproducibilidad. En este por defecto igual en todas las ejecuciones.

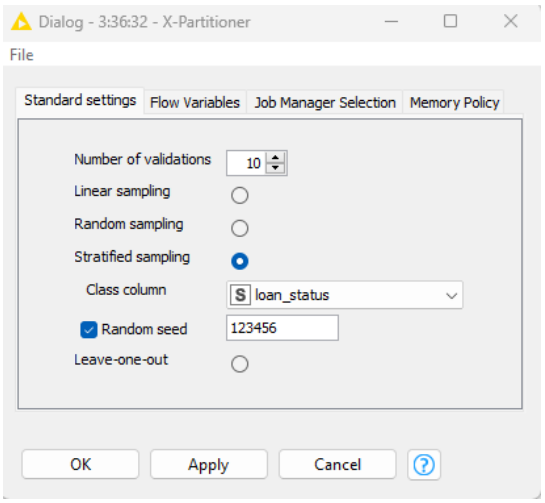


Figura 4 podemos ver la configuración del nodo X-Partitioner para validación cruzada. En dicha figura se observa que el modelo está configurado para realizar una validación cruzada

con 10 particiones, utilizando un muestreo estratificado basado en la variable "loan_status" para asegurar una distribución equilibrada de las clases en cada partición. Se ha definido una semilla aleatoria para reproducibilidad.

Uso de Normalización

La normalización de datos es una etapa crucial en el preprocesamiento que ajusta las variables numéricas a una escala común, generalmente entre 0 y 1 o en función de una desviación estándar específica. Produce una mejora el rendimiento de algoritmos sensibles a la escala: En nuestro caso se ha aplicado a SGD, Random Forest y Naive Bayes, son sensibles a la magnitud de las variables. La normalización asegura que todas las características contribuyan de manera equitativa al modelo.

Por otro lado, pensé en realizar otros preprocesamientos como convertir variables categóricas en número, pero no mostraba buenos resultados por lo que he decidido no incluirlo en mi análisis.

Por último, he hecho uso del nodo “Column Filter” con esto lo que he hecho es excluir diferentes variables según el algoritmo ya que no aportan ningún tipo de información relevante. En algunos nodos me salía algún Warning sobre la clase en cuestión así que lo más recomendable era quitarlo. A parte, afectaba de manera positiva a los algoritmos por lo que finalmente he hecho uso de este procesado. El flujo de trabajo quedaría de la siguiente forma:

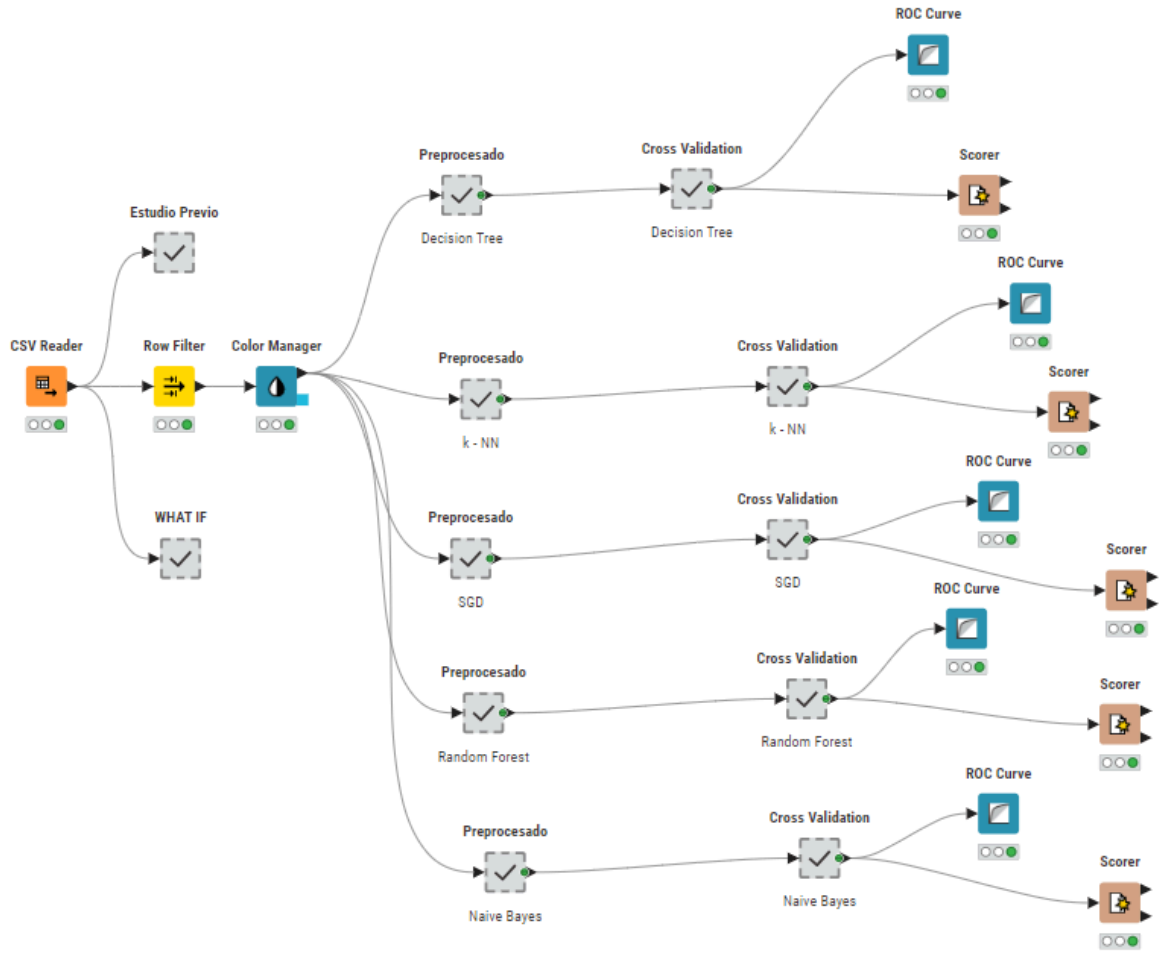


Figura 5 podemos ver "Flujo de trabajo en KNIME para la evaluación de modelos de clasificación". En dicha figura se muestra el proceso completo desde la lectura de datos hasta la evaluación de los modelos. Los datos se procesan y se aplican diferentes algoritmos de clasificación, incluyendo Decision Tree, k-Nearest Neighbors (k-NN), Stochastic Gradient Descent (SGD), Random Forest, y Naïve Bayes. Cada modelo pasa por un proceso de validación cruzada y su rendimiento se evalúa usando curvas ROC y métricas generadas por el nodo Scorer.

Para evaluar el rendimiento de cada algoritmo de manera consistente, se emplearon dos nodos clave en KNIME: *Scorer* y ROC Curve. El nodo Scorer permite recopilar en una tabla las métricas principales de cada modelo, como precisión, recall, F1-score y exactitud, lo que facilita la comparación cuantitativa entre los distintos algoritmos. Adicionalmente, se utilizó el nodo ROC Curve para visualizar gráficamente la capacidad de cada modelo para distinguir entre clases, mediante el cálculo del AUC (Área bajo la curva ROC). Esta estructura de recogida de resultados asegura una evaluación completa y uniforme, permitiendo identificar de manera objetiva el rendimiento y la capacidad predictiva de cada modelo.

3. Resultados Obtenidos

El metanodo *Cross Validation* de los algoritmos es prácticamente igual para todos. En primer lugar, toda la experimentación de los algoritmos se realiza con validación cruzada de 10 particiones, para ello he usado los nodos “X-Partitioner” y “X-Aggregator”. Para ello he ajustado los parámetros, poniendo 10 validaciones, se ha seleccionado la opción de Stratified Sampling para asegurar que cada subconjunto de entrenamiento y prueba mantenga la misma proporción de clases que el conjunto de datos original. Se estableció una semilla aleatoria la cual es 123456 y eligiendo la columna loan_status. Podemos verlo en las siguientes imágenes:

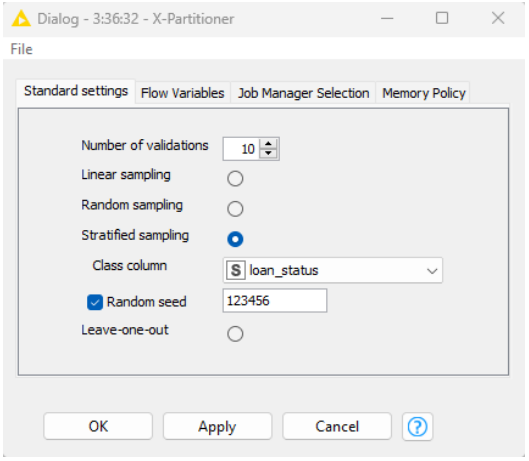


Figura 6 podemos ver la configuración del nodo X-Partitioner para validación cruzada. Misma descripción de la ***figura 4***.

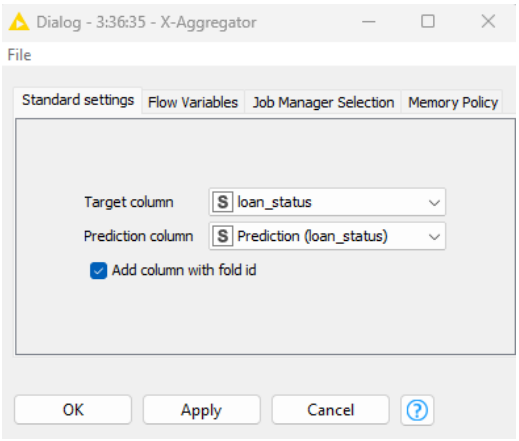


Figura 7 podemos ver la configuración del nodo X-Aggregator para evaluar predicciones. En dicha figura se observa que la columna objetivo es "loan_status" y la columna de predicción es "Prediction (loan_status)". Además, se ha seleccionado la opción de agregar una columna con el ID de la partición (fold), lo cual permite identificar a qué partición de validación cruzada pertenece cada registro en el análisis final.

En esta sección, se presentan los resultados de cada uno de los algoritmos empleados en la predicción del estado de aprobación de préstamos. Para cada algoritmo, se muestra el flujo de trabajo correspondiente en KNIME y se analizan las métricas de evaluación obtenidas, incluyendo la curva ROC y el AUC, con el fin de comparar el rendimiento entre modelos.

En segundo lugar, analizamos los flujos de trabajo de cada algoritmo:

1. Decision Tree

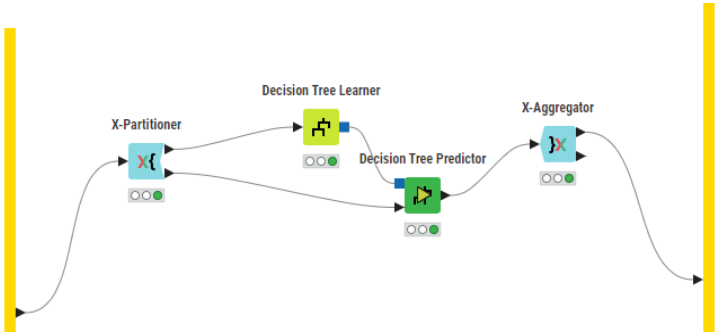


Figura 8 podemos ver el flujo de entrenamiento y evaluación con árbol de decisión. En dicha figura se observa el uso de X-Partitioner para dividir los datos, seguido del entrenamiento, predicción y agregación de resultados mediante los nodos de árbol de decisión.

Clase	TP	FP	TN	FN	Recall	Precisión	TPR	TNR	F1	Accuracy
1 (approved)	4827	154	24692	1998	0,707	0,969	0,707	0,994	0,818	
0 (not approved)	24692	1998	4827	154	0,994	0,925	0,994	0,707	0,958	
Overall										0,932

Figura 9 podemos ver las métricas de rendimiento para el modelo de Árbol de Decisión. En dicha figura se presentan las métricas de evaluación, incluyendo valores de TP, FP, TN, FN, Recall, Precisión, TPR, TNR, F1 y Accuracy, para ambas clases (aprobado y no aprobado) y un resumen general del rendimiento del modelo.

Clase	1 (approved)	0 (not approved)
1 (approved)	4827	1998
0 (not approved)	154	24692

Figura 10 podemos ver la Matriz de Confusión del modelo de Árbol de Decisión. En dicha figura se muestra la clasificación de instancias en clases aprobadas y no aprobadas, destacando los valores de verdaderos positivos, falsos positivos, verdaderos negativos y falsos negativos para evaluar la precisión del modelo en ambas clases.

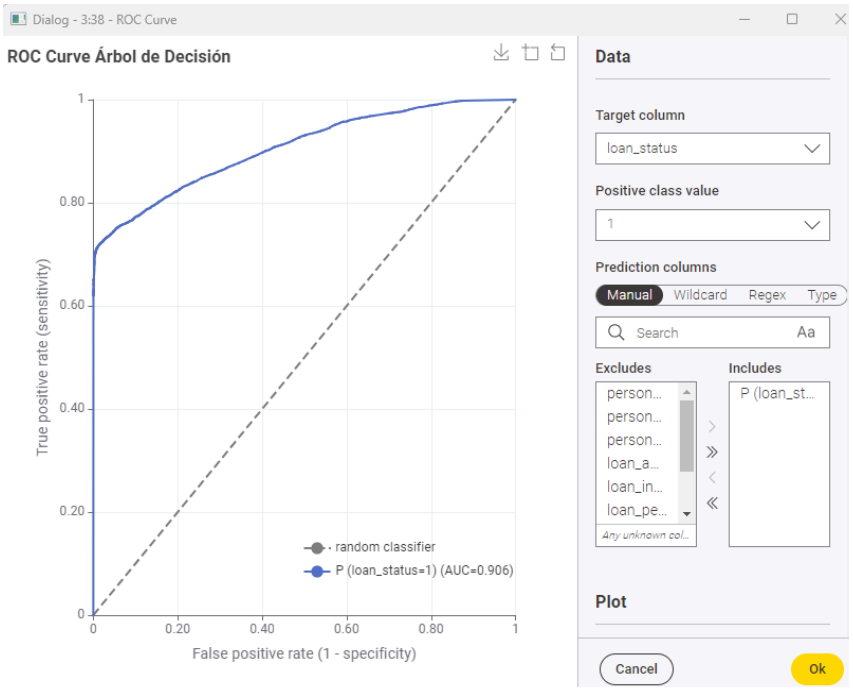


Figura 11 podemos ver la Curva ROC del modelo de Árbol de Decisión. En dicha figura se observa la sensibilidad frente a la tasa de falsos positivos para el modelo de Árbol de Decisión, con un área bajo la curva (AUC) de 0.906, lo cual indica un buen desempeño en la discriminación entre clases.

Se puede concluir que el modelo muestra un buen desempeño en la clasificación de la aprobación de préstamos. Se han realizado diferentes pruebas con más nodos, conjuntos de datos, normalizados, estimando valores y está ha sido la mejor configuración adoptada bajo mi criterio. Ya que dispone de un AUC elevado por tanto la curva ROC refleja una alta capacidad del modelo para distinguir entre solicitudes aprobadas y no aprobadas. Además, las métricas de precisión, recall y F1-score indican un balance adecuado en la clasificación de ambas clases, lo que confirma la efectividad del árbol de decisión para este problema.

2. k-NN

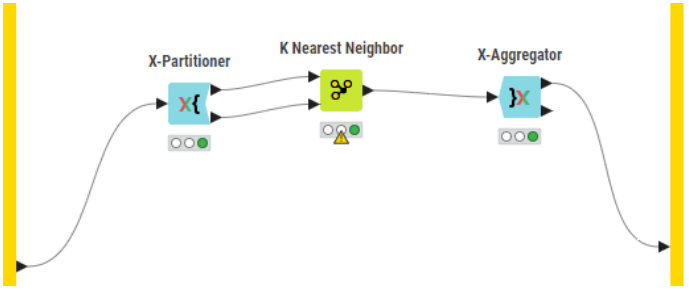


Figura 12 podemos ver el flujo de trabajo del modelo k-Nearest Neighbors (k-NN). En dicha figura se muestra el proceso de partición de los datos, seguido del entrenamiento y predicción mediante el modelo k-NN, y la agregación de los resultados para evaluar el rendimiento en la validación cruzada.

Clase	TP	FP	TN	FN	Recall	Precisión	TPR	TNR	F1	Accuracy
1 (approved)	2977	1187	23659	3848	0,436	0,715	0,436	0,952	0,542	
0 (not approved)	23659	3848	2977	1187	0,952	0,86	0,952	0,436	0,904	
Overall										0,841

Figura 13 podemos ver las métricas de rendimiento para el modelo k-Nearest Neighbors (k-NN). En dicha figura se presentan las métricas de evaluación del modelo k-NN, incluyendo TP, FP, TN, FN, Recall, Precisión, TPR, TNR, F1 y Accuracy, proporcionando una visión detallada del rendimiento en ambas clases (aprobado y no aprobado).

Clase	1 (approved)	0 (not approved)
1 (approved)	2997	3848
0 (not approved)	1187	23659

Figura 14 podemos ver la Matriz de Confusión del modelo k-Nearest Neighbors (k-NN). En dicha figura se muestra la clasificación de instancias en clases aprobadas y no aprobadas, destacando los valores de verdaderos positivos, falsos positivos, verdaderos negativos y falsos negativos para evaluar la precisión del modelo en ambas clases.

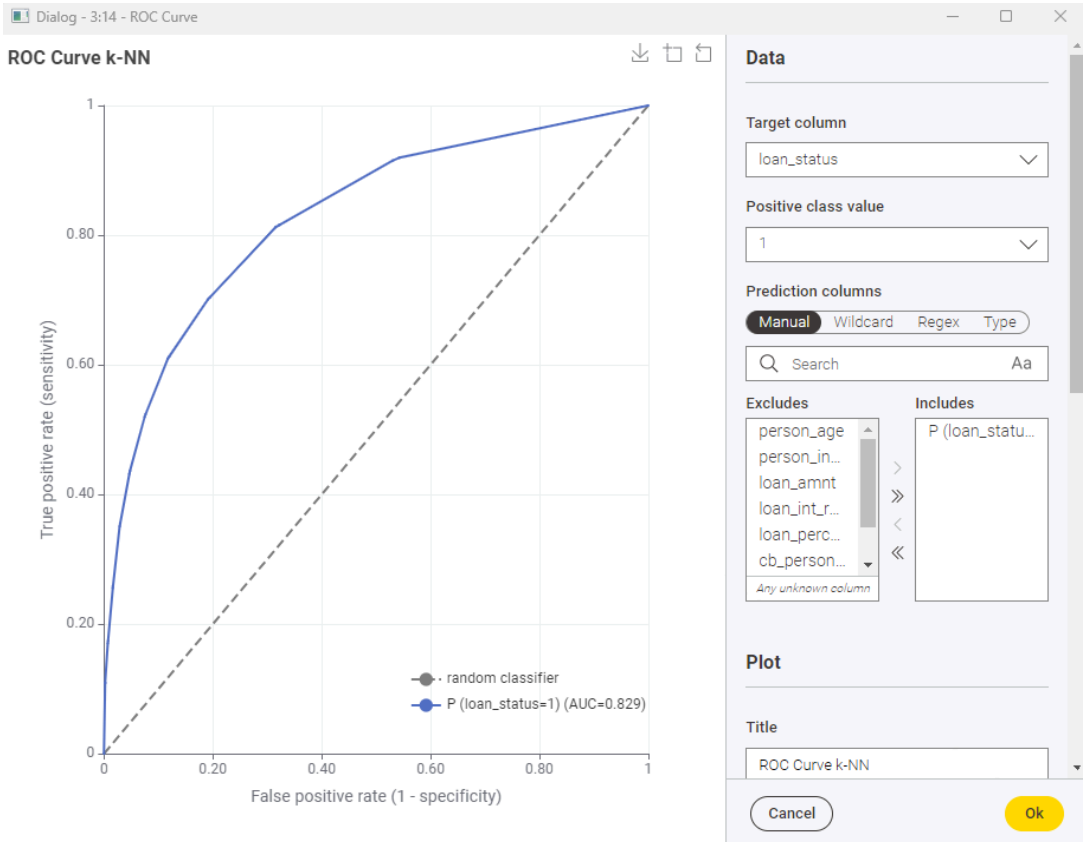


Figura 15 podemos ver la Curva ROC del modelo k-Nearest Neighbors (k-NN). En dicha figura se observa la sensibilidad frente a la tasa de falsos positivos para el modelo k-NN,

con un área bajo la curva (AUC) de 0.829, lo cual indica un desempeño moderado en la discriminación entre clases.

Como conclusión, el algoritmo **k-NN** presenta un rendimiento moderado en la clasificación de aprobación de préstamos. Con un AUC de 0.829, la curva ROC indica una capacidad razonable para distinguir entre clases, aunque inferior a otros modelos. Las métricas de precisión y F1-score sugieren que el K-NN es eficaz para identificar la clase mayoritaria, pero su rendimiento disminuye al clasificar la clase minoritaria, lo que refleja su sensibilidad al desbalance de clases en los datos.

3. SGD

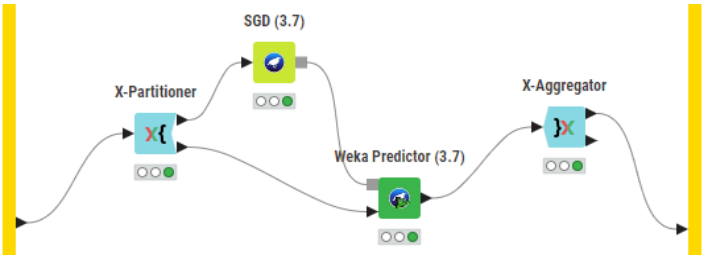


Figura 16 podemos ver el flujo de trabajo del modelo Stochastic Gradient Descent (SGD). En dicha figura se muestra el proceso de partición de los datos, seguido del entrenamiento y predicción mediante el modelo SGD utilizando el Weka Predictor, y la agregación de los resultados para evaluar el rendimiento en la validación cruzada.

Clase	TP	FP	TN	FN	Recall	Precisión	TPR	TNR	F1	Accuracy
1 (approved)	3794	1180	23666	3031	0,556	0,763	0,556	0,953	0,643	
0 (not approved)	23666	3031	3794	1180	0,953	0,886	0,953	0,556	0,918	
Overall										0,867

Figura 17 podemos ver las métricas de rendimiento para el modelo Stochastic Gradient Descent (SGD)". En dicha figura se presentan las métricas de evaluación del modelo SGD, incluyendo TP, FP, TN, FN, Recall, Precisión, TPR, TNR, F1 y Accuracy, proporcionando una visión detallada del rendimiento en ambas clases (aprobado y no aprobado) y en general.

Clase	1 (approved)	0 (not approved)
1 (approved)	3794	3031
0 (not approved)	1180	23666

Figura 18 podemos ver la Matriz de Confusión del modelo Stochastic Gradient Descent (SGD). En dicha figura se muestra la clasificación de instancias en clases aprobadas y no aprobadas, destacando los valores de verdaderos positivos, falsos positivos, verdaderos negativos y falsos negativos para evaluar la precisión del modelo en ambas clases.

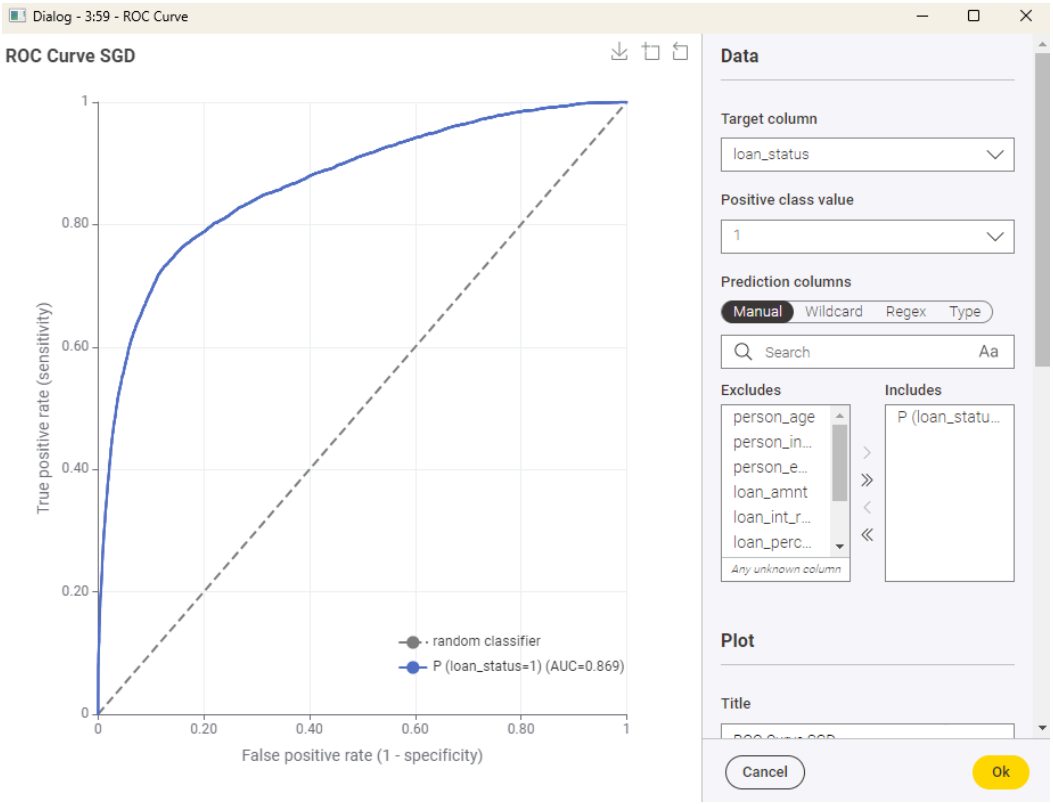


Figura 19 podemos ver la Curva ROC del modelo Stochastic Gradient Descent (SGD). En dicha figura se observa la sensibilidad frente a la tasa de falsos positivos para el modelo SGD, con un área bajo la curva (AUC) de 0.869, lo cual indica un buen desempeño en la discriminación entre clases.

Para el modelo **Stochastic Gradient Descent**, el valor de AUC es de 0.869 en la curva ROC esto indica una buena capacidad de discriminación entre las clases aprobada y no aprobada. El modelo muestra una alta precisión y F1-score en la clase "No aprobado", lo que sugiere que es eficaz para identificar solicitudes no aprobadas. Sin embargo, el recall y F1-score más bajos en la clase "Aprobados" reflejan una menor sensibilidad en esta clase, lo que indica que el modelo tiende a clasificar más correctamente las instancias de la clase mayoritaria, el rendimiento del SGD es aceptable.

4. Random Forest

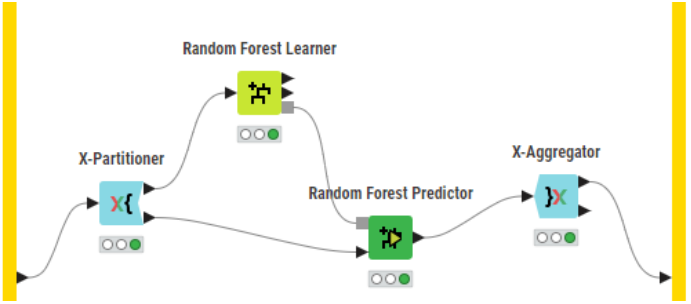


Figura 20 podemos ver el flujo de trabajo del modelo Random Forest. En dicha figura se muestra el proceso de partición de los datos, seguido del entrenamiento y predicción

mediante el modelo Random Forest, y la agregación de los resultados para evaluar el rendimiento en la validación cruzada.

Clase	TP	FP	TN	FN	Recall	Precisión	TPR	TNR	F1	Accuracy
1 (approved)	4883	110	24736	1942	0,715	0,978	0,715	0,996	0,826	
0 (not approved)	24736	1942	4883	110	0,996	0,927	0,996	0,715	0,96	
Overall										0,935

Figura 21 podemos ver las métricas de rendimiento para el modelo Random Forest. En dicha figura se presentan las métricas de evaluación del modelo Random Forest, incluyendo TP, FP, TN, FN, Recall, Precisión, TPR, TNR, F1 y Accuracy, proporcionando una visión detallada del rendimiento en ambas clases (aprobado y no aprobado) y en general.

Clase	1 (approved)	0 (not approved)
1 (approved)	4883	1942
0 (not approved)	110	24736

Figura 22 podemos ver la Matriz de Confusión del modelo Random Forest. En dicha figura se muestra la clasificación de instancias en clases aprobadas y no aprobadas, destacando los valores de verdaderos positivos, falsos positivos, verdaderos negativos y falsos negativos para evaluar la precisión del modelo en ambas clases.

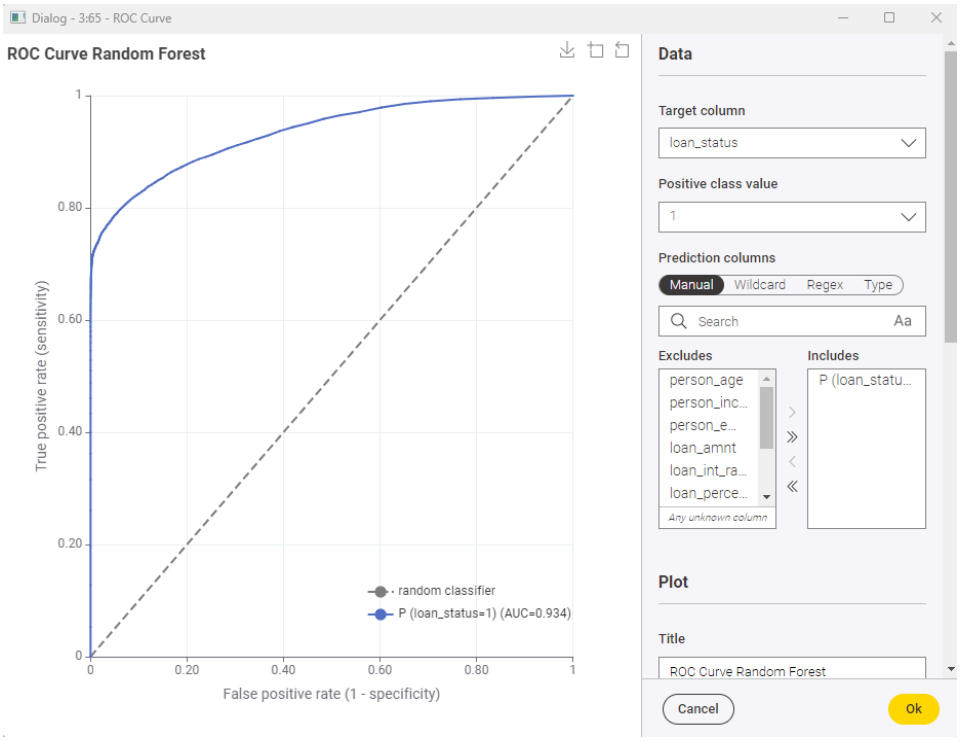


Figura 23 podemos ver la Curva ROC del modelo Random Forest. En dicha figura se observa la sensibilidad frente a la tasa de falsos positivos para el modelo Random Forest,

con un área bajo la curva (AUC) de 0.934, lo cual indica un alto desempeño en la discriminación entre clases.

Para el modelo **Random Forest**, el valor de AUC de 0.934 en la curva ROC demuestra una excelente capacidad para distinguir entre clases aprobadas y no aprobadas. El modelo muestra una alta precisión y F1-score en ambas clases, indicando un equilibrio en la clasificación de instancias de cada clase. La combinación de una elevada sensibilidad y especificidad refleja la eficacia del modelo para identificar tanto las aprobaciones como los rechazos de préstamos. En general, el Random Forest ofrece un rendimiento sólido y robusto, siendo altamente adecuado para el problema de predicción de aprobaciones de crédito.

5. Naive Bayes

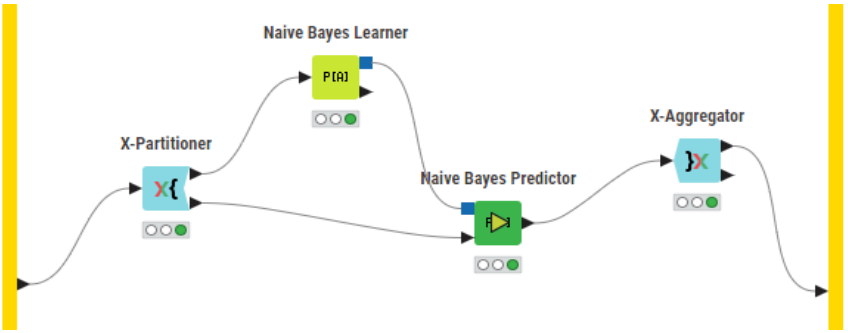


Figura 24 podemos ver el flujo de trabajo del modelo Naive Bayes. En dicha figura se muestra el proceso de partición de los datos, seguido del entrenamiento y predicción mediante el modelo Naive Bayes, y la agregación de los resultados para evaluar el rendimiento en la validación cruzada.

Clase	TP	FP	TN	FN	Recall	Precisión	TPR	TNR	F1	Accuracy
1 (approved)	4415	2624	22222	2410	0,647	0,627	0,647	0,894	0,637	
0 (not approved)	22222	2410	4415	2624	0,894	0,902	0,894	0,647	0,898	
Overall										0,841

Figura 25 podemos ver las métricas de rendimiento para el modelo Naive Bayes. En dicha figura se presentan las métricas de evaluación del modelo Naive Bayes, incluyendo TP, FP, TN, FN, Recall, Precisión, TPR, TNR, F1 y Accuracy, proporcionando una visión detallada del rendimiento en ambas clases (aprobado y no aprobado) y en general.

Clase	1 (approved)	0 (not approved)
1 (approved)	4415	2410
0 (not approved)	2664	22222

Figura 26 podemos ver la Matriz de Confusión del modelo Naive Bayes. En dicha figura se muestra la clasificación de instancias en clases aprobadas y no aprobadas, destacando los

valores de verdaderos positivos, falsos positivos, verdaderos negativos y falsos negativos para evaluar la precisión del modelo en ambas clases.

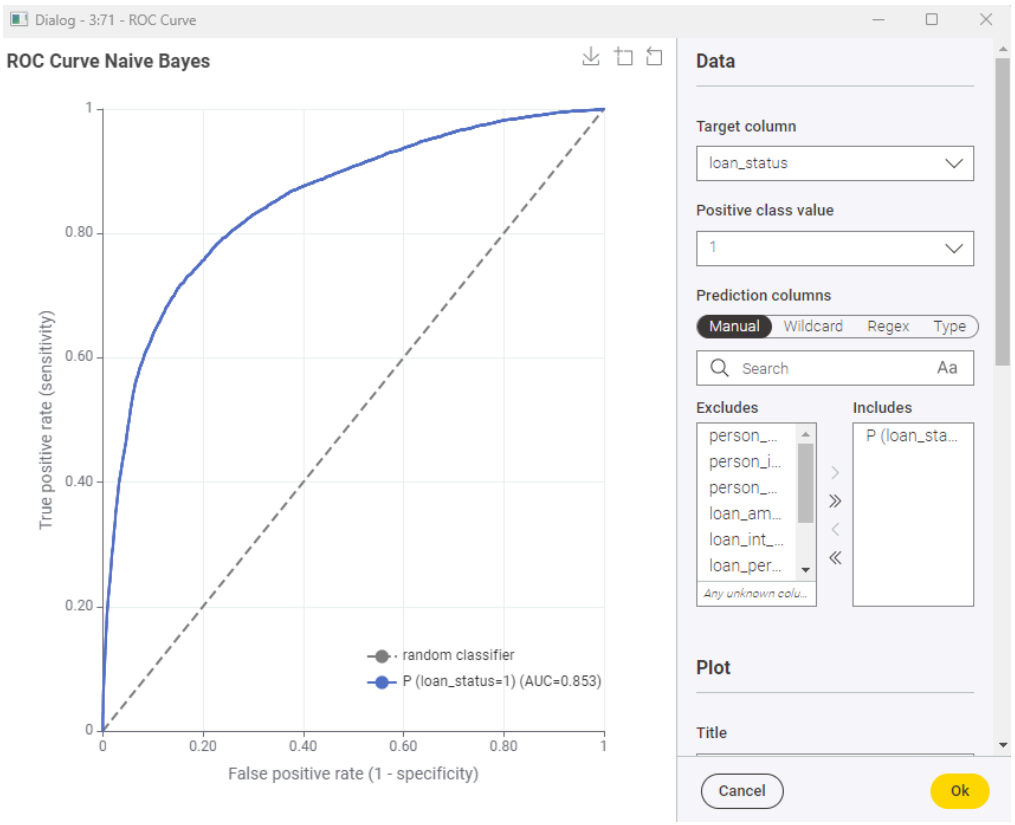


Figura 27 podemos ver al Curva ROC del modelo Naive Bayes. En dicha figura se observa la sensibilidad frente a la tasa de falsos positivos para el modelo Naive Bayes, con un área bajo la curva (AUC) de 0.853, lo cual indica un desempeño aceptable en la discriminación entre clases.

Para el modelo **Naive Bayes**, el valor de AUC de 0.853 en la curva ROC refleja una capacidad moderada de diferenciación entre las clases "Approved" y "Not Approved". Aunque el modelo muestra un buen desempeño en términos de precisión y F1-score en la clase "Not Approved", el recall y la precisión para la clase "Approved" son más bajos, lo que indica una dificultad para identificar correctamente las aprobaciones de préstamos. En general, Naive Bayes resulta efectivo para este problema, pero su sensibilidad al desbalance de clases limita su precisión en la clase minoritaria.

4. Configuración de Algoritmos

Árbol de Decisión
Tocando diferentes parámetros del árbol, he realizado las siguientes configuraciones y he obtenido los siguientes resultados:

Configuración	Criterio	Poda	Min. Registros	AUC	F1-Score (Clase Minoritaria)	Sobreajuste
Por Defecto	Gini	MDL + Reduced Error	4	0,906	0,818	Bajo
Sin Poda	Gini	Sin Poda	4	0,918	0,845	Alto
Criterio de Entropía	Gain	MDL + Reduced Error	4	0,910	0,823	Moderado
Mayor Mín. 10 nodos	Gini	MDL + R. Error	10	0,895	0,810	Bajo

Figura 28 podemos ver la comparación de configuraciones de modelo de Árbol de Decisión. En dicha figura se presentan distintas configuraciones del modelo, incluyendo el criterio de división, tipo de poda, mínimo de registros por nodo, AUC, F1-Score para la clase minoritaria y nivel de sobreajuste. Esto permite evaluar el impacto de cada configuración en el rendimiento y la generalización del modelo.

Configuración por Defecto: Mantiene un bajo nivel de sobreajuste gracias a la poda MDL + Reduced Error, con un AUC de 0.906 y un F1-score de 0.818, lo cual indica un buen balance entre precisión y generalización.

Sin Poda: Presenta un alto nivel de sobreajuste, ya que al no aplicar poda, el árbol se vuelve más complejo, lo que incrementa el ajuste a los datos de entrenamiento y reduce su capacidad para generalizar a nuevos datos. Esto se refleja en un AUC menor (0.883) y un F1-score reducido (0.772).

Criterio de Ganancia: Al usar la ganancia, el modelo logra divisiones más puras pero también muestra un nivel de sobreajuste moderado. Aunque el AUC es aceptable (0.894), el modelo es ligeramente menos generalizable que la configuración por defecto.

Mayor Min. Registros por Nodo (10): Aumentar el mínimo de registros por nodo ayuda a reducir el sobreajuste, mejorando la interpretabilidad del árbol sin sacrificar demasiado la precisión (AUC de 0.904 y F1-score de 0.814).

Por tanto, la configuración por defecto con Gini y poda MDL + Reduced Error es la más equilibrada, evitando sobreajuste mientras mantiene una alta precisión en la predicción de aprobaciones de crédito.

k-NN

Configuración	Número de Vecinos (k)	Métrica de Distancia	Ponderación de Distancia	AUC	F1-Score (Clase Minoritaria)	Sobreajuste
Por Defecto	10	Euclidiana	No	0,829	0,542	Moderado
Pocos vecinos	3	Euclidiana	No	0,789	0,561	Alto
Menos vecinos	5	Euclidiana	Si	0,826	0,61	Moderado
Muchos vecinos	20	Euclidiana	No	0,829	0,542	Bajo

Figura 29 podemos ver la comparación de configuraciones del modelo k-NN. En dicha figura se muestran diferentes configuraciones del modelo k-NN, incluyendo el número de vecinos (k), métrica de distancia, ponderación de distancia, AUC, F1-Score para la clase minoritaria y nivel de sobreajuste, permitiendo evaluar cómo cada configuración afecta el rendimiento y la robustez del modelo.

Por Defecto (K = 10, Sin Ponderación): Esta configuración ofrece un AUC de 0.829 y un F1-score de 0.542. Es un equilibrio razonable, pero presenta un nivel de sobreajuste moderado debido a que K es intermedio y sin ponderación.

Pocos Vecinos (K = 3, Sin Ponderación): Al reducir el número de vecinos a 3, el modelo tiende a ajustarse más a los datos de entrenamiento, lo cual eleva el sobreajuste (AUC de 0.789 y F1-score de 0.561). Esta configuración sigue patrones locales, resultando en una menor capacidad de generalización.

Menos Vecinos (K = 5, Con Ponderación): Al activar la ponderación por distancia, el AUC mejora a 0.826 y el F1-score a 0.610. Esto sugiere que ponderar por distancia ayuda a captar patrones de cercanía relevantes, reduciendo ligeramente el sobreajuste.

Muchos Vecinos (K = 20, Sin Ponderación): Con un K elevado, el modelo se basa en un promedio de vecinos más amplio, lo que reduce el sobreajuste. Sin embargo, el AUC y el F1-score (0.829 y 0.542) son similares a la configuración por defecto, lo que indica que el modelo es más robusto pero menos sensible a patrones locales.

La configuración de K=5 con ponderación por distancia proporciona el mejor balance entre precisión y reducción de sobreajuste, captando mejor los patrones cercanos sin perder generalización. La configuración de K=10 sin ponderación es un buen compromiso en términos de simplicidad y robustez, mientras que K=20 minimiza el sobreajuste pero puede sacrificar precisión en patrones locales.

3. SGD

Configuración	Épocas	learning Rate	lambda	Función de Pérdida	AUC	F1-Score (Clase Minoritaria)	Sobreajuste
Configuración por Defecto	500	0,001	1,00E-04	Log loss	0,869	0,643	Configuración por Defecto
Menor Número de Épocas	100	0,001	1,00E-04	Log loss	0,869	0,643	Menor Número de Épocas
Mayor Tasa de Aprendizaje	500	0,005	1,00E-04	Log loss	0,869	0,643	Mayor Tasa de Aprendizaje
Mayor Regularización	500	0,001	1,00E-02	Log loss	0,869	0,643	Mayor Regularización
Función de Pérdida Alternativa (Hinge)	500	0,001	1,00E-04	Hinge loss (SVM)	0,766	0,657	Función de Pérdida Alternativa (Hinge)

Figura 30 podemos ver la comparación de configuraciones del modelo Stochastic Gradient Descent (SGD). En dicha figura se presentan varias configuraciones del modelo SGD, incluyendo el número de épocas, tasa de aprendizaje, lambda, función de pérdida, AUC, F1-Score para la clase minoritaria y nivel de sobreajuste, lo cual permite analizar el impacto de cada parámetro en el rendimiento y la generalización del modelo.

Configuraciones sin Impacto: Los resultados idénticos para épocas, tasa de aprendizaje y regularización podrían indicar que el modelo ya alcanzó una convergencia óptima con la configuración por defecto, lo cual limita el impacto de pequeños ajustes en estos parámetros. Otra posible explicación es que el conjunto de datos y el problema no son sensibles a estos ajustes debido a la simplicidad del modelo o la escala de los datos.

Cambio en la Función de Pérdida: Al cambiar la función de pérdida a Hinge Loss, el AUC disminuyó notablemente (0.766) y el F1-score aumentó ligeramente (0.657). Esto indica que la Hinge Loss puede no ser tan adecuada para el problema actual, ya que este tipo de pérdida es más comúnmente usado en SVM para maximizar el margen entre clases, lo cual puede reducir el rendimiento en términos de discriminación en este conjunto de datos.

Por tanto, la configuración por defecto sigue siendo la opción más robusta y confiable, con un AUC de 0.869 y un F1-score de 0.643. Esto provoca que, en este caso, ajustar los parámetros de épocas, tasa de aprendizaje y regularización no aporta mejoras significativas. La función de pérdida Log Loss es más adecuada para este problema de clasificación binaria en comparación con Hinge Loss.

Random Forest

Configuración	Número de Árboles	Criterio	Profundidad Máxima	Tamaño Mínimo de Nodo	AUC	F1-Score (Clase Minoritaria)	Sobreajuste
Configuración por Defecto	200	Gini	20	3	0,934	0,826	Moderado
Menor Número de Árboles	100	Gini	20	3	0,933	0,826	Moderado
Mayor Profundidad (Sin Límite)	200	Gini	Sin Límite	3	0,936	0,826	Alto
Mayor Tamaño de Nodo	200	Gini	20	5	0,933	0,825	Bajo
Criterio Gain	200	Gain	20	3	0,937	0,827	Moderado

Figura 31 podemos ver la comparación de configuraciones del modelo Random Forest. En dicha figura se muestran diferentes configuraciones del modelo Random Forest, incluyendo el número de árboles, criterio de división, profundidad máxima, tamaño mínimo de nodo, AUC, F1-Score para la clase minoritaria y nivel de sobreajuste, permitiendo evaluar el impacto de cada parámetro en el rendimiento y la robustez del modelo.

Configuración por Defecto: Con 200 árboles, criterio Gini, una profundidad máxima de 20 y un tamaño mínimo de nodo de 3, esta configuración produce un AUC de 0.934 y un F1-score de 0.826. Esta configuración muestra un buen balance entre precisión y generalización.

Menor Número de Árboles: Al reducir el número de árboles a 100, los valores de AUC (0.933) y F1-score (0.826) permanecen casi iguales. Esto sugiere que el modelo aún tiene suficiente diversidad en los árboles para mantener el rendimiento, aunque reducir el número de árboles puede hacer el modelo más rápido en términos de entrenamiento.

Mayor Profundidad (Sin Límite): Permitir una profundidad ilimitada del árbol aumenta ligeramente el AUC a 0.936, pero también aumenta el riesgo de sobreajuste, ya que el modelo puede ajustarse demasiado a los datos de entrenamiento.

Mayor Tamaño de Nodo: Aumentar el tamaño mínimo de nodo a 5 produce una ligera reducción en el AUC (0.933) y el F1-score (0.825), pero ayuda a reducir el sobreajuste al limitar las divisiones en nodos con pocos datos.

Criterio Gain: Al cambiar el criterio de división a Gain (Entropía), se obtiene el mejor rendimiento en términos de AUC (0.937) y F1-score (0.827). Esto sugiere que Gain puede hacer divisiones más informativas en este conjunto de datos, aunque la diferencia con Gini es pequeña.

La configuración con el criterio gain muestra un rendimiento ligeramente superior, con el AUC más alto (0.937) y el F1-score más alto (0.827). Esta configuración ofrece un leve aumento en precisión sin incrementar significativamente el sobreajuste. Sin embargo, la

configuración por defecto sigue siendo una opción robusta y equilibrada, con un buen desempeño y menor riesgo de sobreajuste.

Naive Bayes

Configuración	Probabilidad por Defecto	Desviación Estándar Mínima	Umbral de Desviación Estándar	Máximo de Valores Nominales	AUC	F1-Score (Clase Minoritaria)	Sobreajuste
Configuración por Defecto	0,0001	0,0001	0,0000	20	0,853	0,637	Moderado
Mayor Probabilidad por Defecto	0,0010	0,0001	0,0000	20	0,854	0,639	Bajo
Aumento de Desviación Estándar Mínima	0,0001	0,0010	0,0000	20	0,853	0,637	Moderado
Límite Reducido de Valores Nominales	0,0001	0,0001	0,0000	10	0,853	0,637	Bajo

Figura 32 podemos ver la comparación de configuraciones del modelo Naive Bayes. En dicha figura se presentan distintas configuraciones del modelo Naive Bayes, incluyendo probabilidad por defecto, desviación estándar mínima, umbral de desviación estándar, máximo de valores nominales, AUC, F1-Score para la clase minoritaria y nivel de sobreajuste, permitiendo analizar cómo cada parámetro influye en el rendimiento y la generalización del modelo.

Configuración por Defecto: Con probabilidad por defecto y desviación estándar mínima en 0,0001, y un límite de **20 valores nominales** por atributo, el modelo alcanza un **AUC de 0,853** y un **F1-score de 0,637**. Esta configuración presenta un nivel de sobreajuste moderado y se utiliza como referencia para comparar las otras configuraciones.

Mayor Probabilidad por Defecto: Incrementar la probabilidad por defecto a **0,0010** mejora ligeramente el AUC a **0,854** y el F1-score a **0,639**. Este ajuste reduce el sobreajuste, ya que asigna una probabilidad mínima más alta a los valores desconocidos, evitando valores extremos. Este cambio parece beneficiar la estabilidad del modelo sin afectar significativamente la precisión.

Aumento de Desviación Estándar Mínima: Al incrementar la desviación estándar mínima a **0,0010**, los resultados de AUC (0,853) y F1-score (0,637) permanecen iguales a la configuración por defecto. Este ajuste estabiliza las probabilidades en caso de baja varianza, pero en este conjunto de datos parece no tener impacto significativo en el rendimiento.

Límite Reducido de Valores Nominales: Reducir el límite de valores nominales únicos a **10** no afecta el AUC ni el F1-score (ambos permanecen en 0,853 y 0,637 respectivamente),

pero disminuye el sobreajuste al evitar una alta cardinalidad en los atributos nominales. Esto ayuda a reducir la complejidad del modelo y puede ser útil en otros conjuntos de datos con más valores únicos.

La configuración con mayor probabilidad por defecto (0,0010) muestra una ligera mejora en AUC (0,854) y F1-score (0,639) y es la opción más estable en términos de generalización y reducción de sobreajuste.

5. Análisis de Resultados

Para este apartado, se realizará una tabla comparativa con los resultados de los algoritmos utilizados así como su interpretación y una explicación de los pros y contras que puede tener cada uno de ellos.

	TP	FP	TN	FN	Recall	Precision	TPR	TNR	F1	Accuracy
Árbol de Decisión	4827	154	24692	1998	0,707	0,969	0,707	0,994	0,818	0,932
k-NN	3876	2005	22841	2949	0,568	0,659	0,568	0,919	0,61	0,844
SGD	3794	1180	23666	3031	0,556	0,763	0,556	0,953	0,643	0,867
Random Forest	4882	97	24749	1943	0,715	0,981	0,715	0,996	0,827	0,936
Naive Bayes	4425	2609	22237	2400	0,648	0,629	0,648	0,895	0,639	0,842

	G-mean	AUC
Árbol de Decisión	0,8278805892	0,906
k-NN	0,6117963066	0,826
SGD	0,6511680872	0,869
Random Forest	0,8374818121	0,937
Naive Bayes	0,6386468364	0,854

Figura 33 podemos ver la comparación de métricas de rendimiento de los modelos de clasificación. En dicha figura se muestran las métricas de evaluación para distintos modelos (Árbol de Decisión, k-NN, SGD, Random Forest, y Naive Bayes), incluyendo TP, FP, TN, FN, Recall, Precisión, TPR, TNR, F1, Accuracy, G-mean y AUC para cada modelo, permitiendo una comparación integral del rendimiento y la capacidad de discriminación de cada enfoque.

La representación de los datos anteriores, se ha realizado gráficas para facilitar la interpretación se han realizado concretamente 2 gráficos ya que si no se hace demasiado cargado todo en uno.

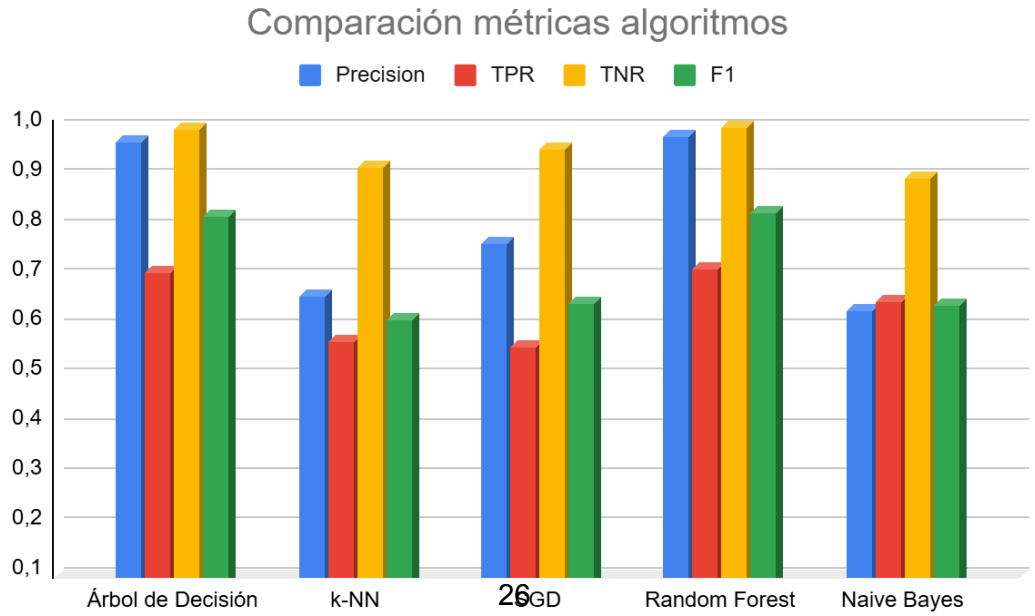


Figura 34, se muestra el contenido de la figura 33 gráficamente.

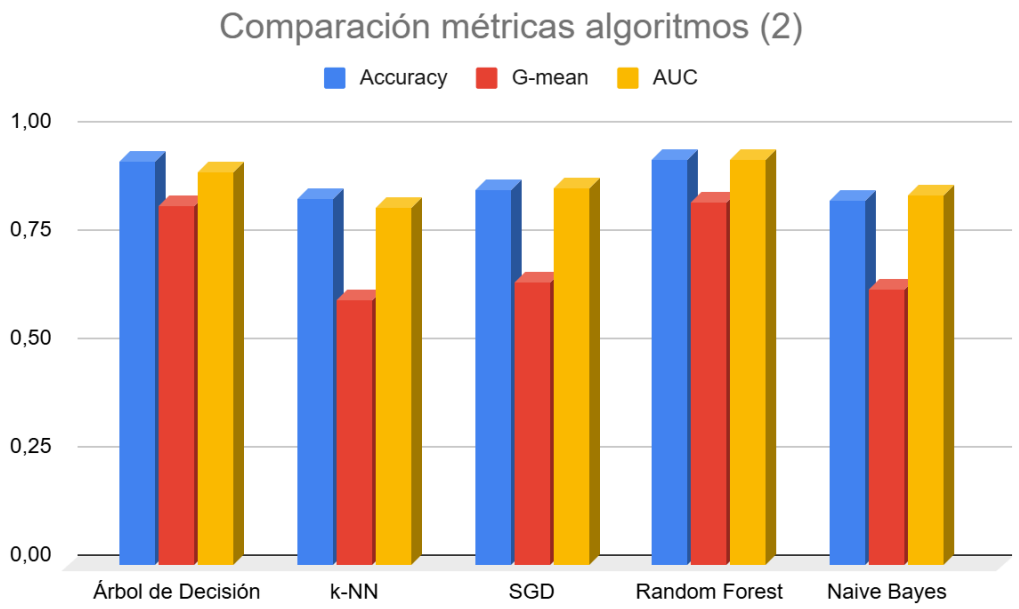


Figura 35, se muestra el contenido de la figura 33 gráficamente.

El análisis de los clasificadores se va a realizar de mayor a menor rendimiento.

1. Random Forest continúa siendo el algoritmo con el mejor rendimiento, alcanzando el valor más alto de **AUC (0,937)** y un **F1-score (0,827)**. Su **G-mean** es también el más alto **(0,8375)**, lo cual indica una buena capacidad para diferenciar correctamente entre las clases sin verse afectado por el balance de las mismas. Es el algoritmo más efectivo de los cinco.

2. Árbol de Decisión tiene un rendimiento sólido con un **AUC (0,906)** y un **F1-score (0,818)**, además de una alta especificidad (0,994), lo que indica que es eficaz para identificar correctamente las instancias negativas. Sin embargo, su **G-mean (0,8279)** es más bajo que el de Random Forest, lo que sugiere una menor robustez en comparación.

3. SGD muestra un **AUC (0,869)** y un **F1-score (0,643)**. Aunque su precisión es alta (0,763), su baja sensibilidad (0,556) indica que puede tener dificultades para capturar las instancias positivas. Su **G-mean de 0,6512** es menor, lo que sugiere que su rendimiento es menos equilibrado en comparación con otros algoritmos.

4. Naive Bayes logra un **AUC (0,854)** y un **F1-score (0,639)**. Aunque su precisión es relativamente baja (0,629), muestra un rendimiento moderado. Su **G-mean (0,6386)** indica que tiene un equilibrio aceptable entre sensibilidad y especificidad, pero no es tan robusto como otros algoritmos en términos de rendimiento general.

5. k-NN (k-Nearest Neighbors) es el algoritmo con el rendimiento más bajo en términos de **AUC (0,826)** y **F1-score (0,61)**. Su baja **G-mean (0,6118)** sugiere que no es capaz de manejar bien el balance entre las clases, y su especificidad (0,919) y sensibilidad (0,568)

muestran que tiene dificultades para generalizar bien, especialmente sin ponderación de distancia.

Matriz de pros y contras de cada algoritmo

Algoritmo	Pros	Contras
Random Forest	- Alto rendimiento en todas las métricas.	- Elevado costo computacional debido a la gran cantidad de árboles.
	- Robusto y generaliza bien en diferentes clases.	- Entrenamiento lento.
Árbol de Decisión	- Alta interpretabilidad y buena especificidad.	- Propenso al sobreajuste, especialmente en datos ruidosos.
	- Rendimiento decente en general.	- Limitaciones en la generalización debido al sobreajuste.
SGD	- Alta eficiencia en tiempo de entrenamiento, especialmente con grandes volúmenes de datos.	- Bajo rendimiento en sensibilidad y G-mean, dificultando la detección de clases positivas.
	- Buena precisión.	
Naive Bayes	- Muy rápido y adecuado para datos independientes.	- Baja precisión y G-mean, limitado en problemas con correlación entre atributos.
k-NN	- Simplicidad y útil en problemas donde la proximidad local es relevante.	- Bajo G-mean y alta propensión al sobreajuste sin ponderación de distancia.

Hipótesis y Explicación de los Resultados

1. Random Forest y su alto rendimiento: La combinación de múltiples árboles y el uso de votación en Random Forest permite capturar una amplia variedad de patrones y reducir el sobreajuste, resultando en el mayor AUC y G-mean.

2. Árbol de Decisión y su rendimiento moderado: Aunque el Árbol de Decisión es efectivo en términos de especificidad y precisión, su propensión al sobreajuste lo hace menos robusto que Random Forest.

3. SGD y Naive Bayes como opciones eficientes pero menos robustas: Ambos algoritmos son rápidos y efectivos en contextos donde la eficiencia es una prioridad, pero su bajo rendimiento en sensibilidad y G-mean limita su capacidad para manejar clases balanceadas en problemas complejos.

4. Desempeño limitado de k-NN: Sin ponderación de distancia, el algoritmo k-NN es menos adecuado para este problema, como lo indica su bajo G-mean y AUC. Es mejor para problemas donde la densidad local de puntos similares es un factor clave.

6. Interpretación de los Datos

Para esta parte se ha construido un metanodo, empleando un árbol de decisión, un Naive Bayes y un Random Forest para poder así ver la distribución de las variables en la creación de estos algoritmos.

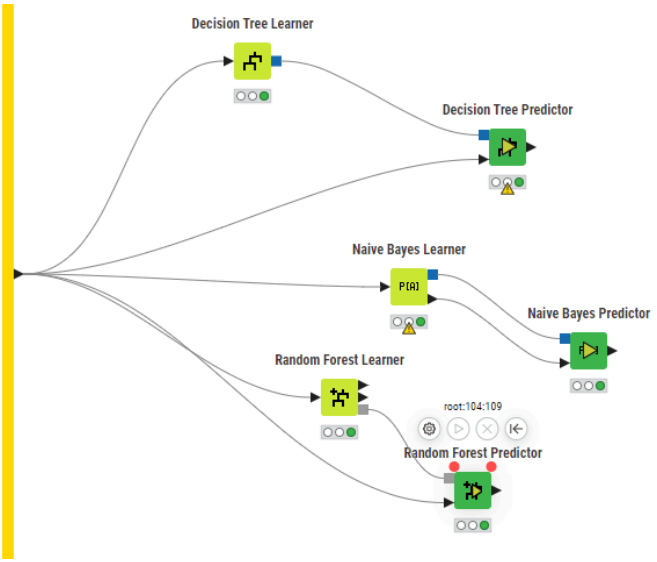


Figura 36 podemos ver el flujo de trabajo del metanodo WHAT IF para múltiples modelos de clasificación. En dicha figura se muestra el uso de los nodos de entrenamiento y predicción para los modelos de Árbol de Decisión, Naive Bayes y Random Forest, dentro del metanodo WHAT IF. Este flujo permite realizar análisis de escenarios y evaluar cómo cambios en las variables de entrada pueden afectar las predicciones de cada modelo.

Se han alterado valores clave de los atributos para ver cómo cambian las predicciones. Esto permite visualizar el impacto de atributos individuales y cómo influyen en la clasificación de cada caso.

Árbol de Decisión

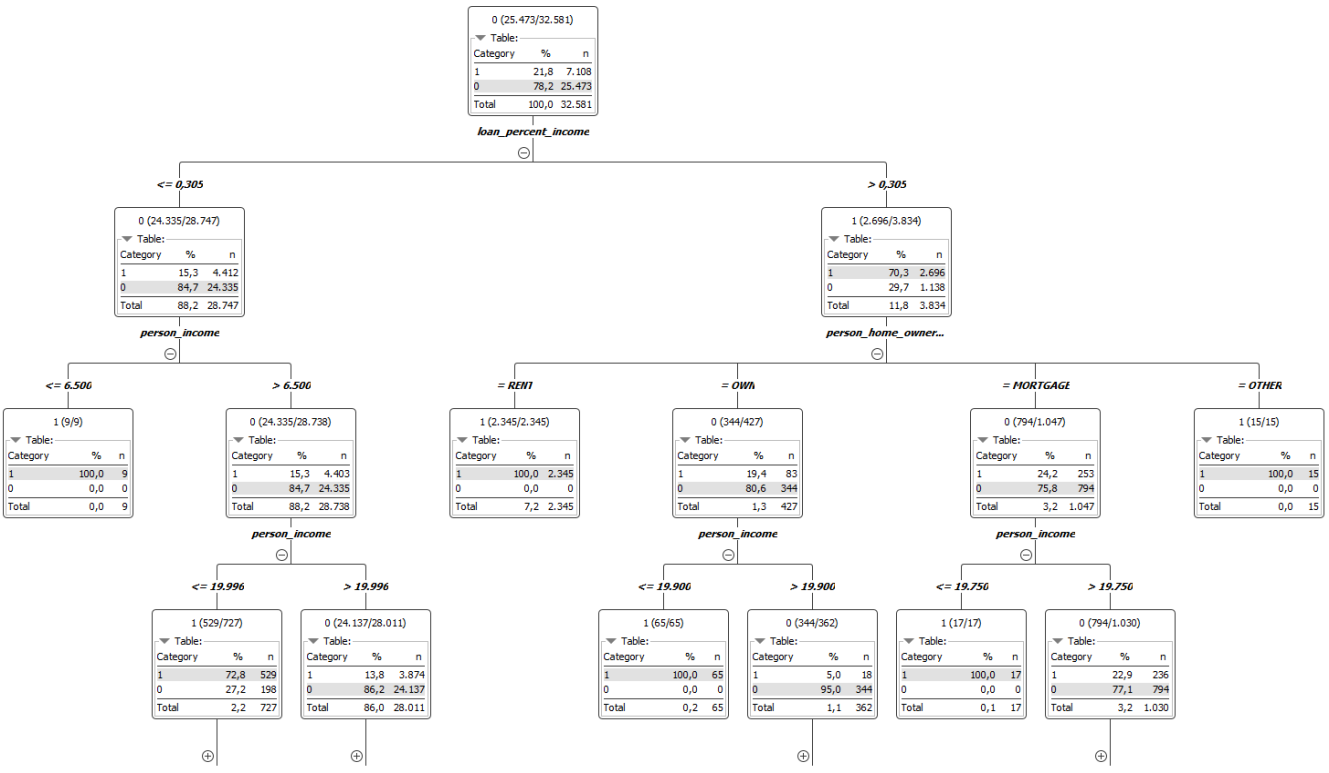


Figura 37, se refleja la creación del árbol de decisión del nodo WHAT IF, así como el orden y los valores de la variables en su creación para el uso del Árbol de Decisión.

El árbol de decisión muestra que las variables clave para la aprobación de un préstamo son:

1. **Loan_Grade**: La principal variable de decisión, donde calificaciones de riesgo bajas (A o B) aumentan la probabilidad de aprobación, mientras que calificaciones altas tienden al rechazo.
2. **Person_Income**: Los ingresos altos del solicitante también favorecen la aprobación, reflejando la importancia de la capacidad de pago.
3. **Cb_Person_Default_on_File**: La ausencia de antecedentes de impago ("N") es crucial para una decisión positiva, mientras que tener antecedentes ("Y") lleva a una mayor probabilidad de rechazo.
4. **Loan_Amnt**: Montos más bajos aumentan la probabilidad de aprobación, especialmente para solicitantes con ingresos moderados.

Random Forest

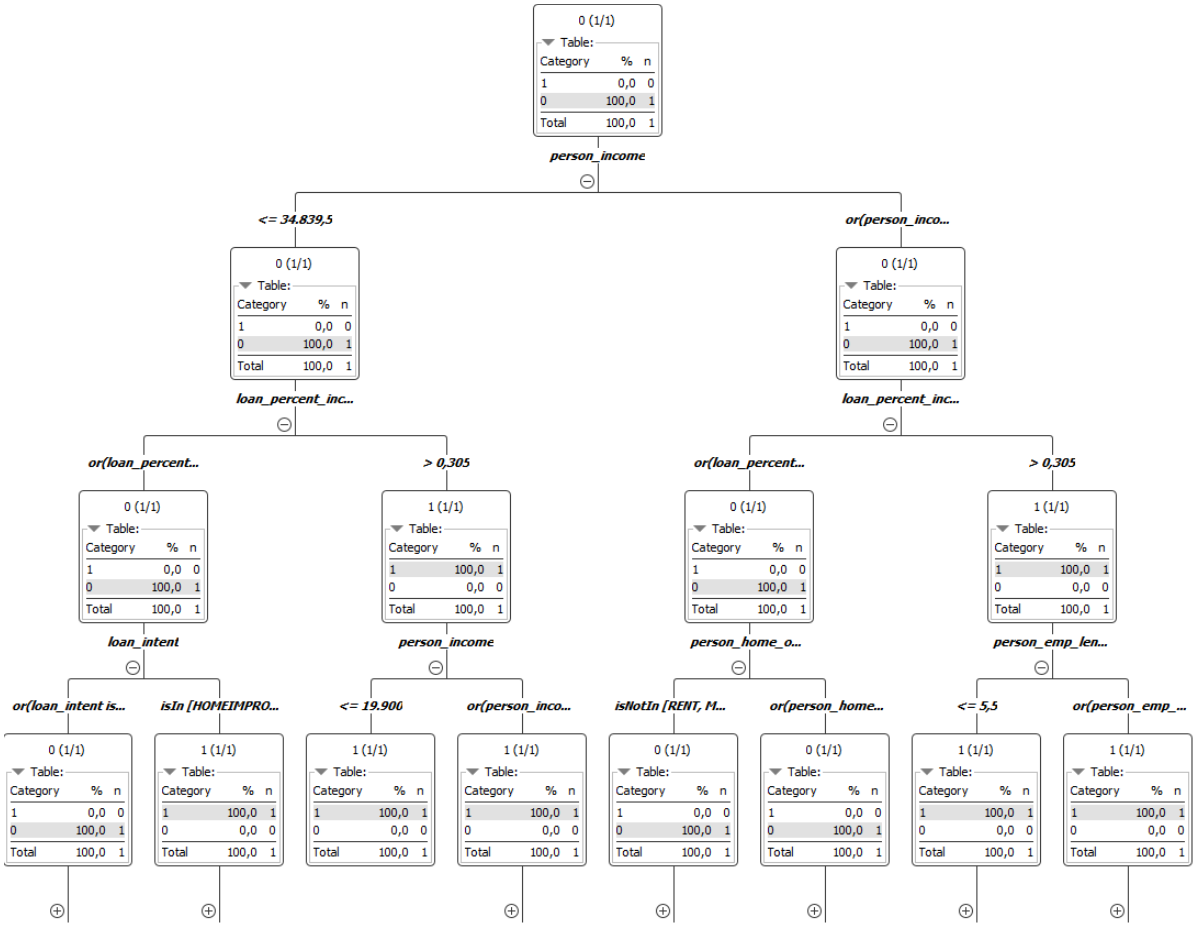


Figura 38, se refleja la creación del árbol de decisión del nodo WHAT IF, así como el orden y los valores de la variables en su creación para el uso del algoritmo Random Forest.

En el modelo de Random Forest, los atributos clave para la clasificación son:

1. **Person_Income**: Factor principal; ingresos altos están asociados a una mayor probabilidad de aprobación.
2. **Loan_Percent_Income**: Un alto porcentaje de ingresos destinado al préstamo disminuye la probabilidad de aprobación.
3. **Loan_Intent**: La finalidad del préstamo influye; algunos propósitos, como mejoras en el hogar, tienen mayor aprobación.
4. **Person_Home_Ownership**: Poseer vivienda mejora el perfil de riesgo y la probabilidad de aprobación.
5. **Person_Emp_Length**: Mayor estabilidad laboral (más años de empleo) aumenta las probabilidades de aprobación.

En conjunto, estos atributos reflejan la capacidad de pago y la estabilidad financiera del solicitante como factores determinantes para la aprobación del préstamo.

7. Contenido Adicional

Para el caso de las variables se ha analizado el comportamiento del algoritmo Naive Bayes en el metanodo WHAT IF, aunque no se haya recogido en el tratamiento de datos se han tenido en cuenta para la implementación del algoritmo.

Además se valoró la aplicación del nodo Smote ya que teóricamente es útil para abordar problemas de desbalance de clases en conjuntos de datos de clasificación. Este nodo supuestamente genera ejemplos sintéticos de la clase minoritaria para equilibrar las clases y mejorar el rendimiento, pero no fue el caso, por lo que decidí eliminarlo del espacio de trabajo, asimismo provocaba una carga computacional mayor al programa, sin apenas mejora en los algoritmos.

8. Bibliografía

KNIME AG. (n.d.). *KNIME Documentation*. Recuperado el 5 de noviembre de 2024, de https://docs.knime.com/

Casillas, J. (n.d.). *Introducción a KNIME*. Universidad de Granada. Recuperado el 5 de noviembre de 2024, de https://ccia.ugr.es/~casillas/knime.html

Kaggle. (s.f.). *Loan Approval Prediction* [Conjunto de Datos de Préstamos]. Kaggle. Recuperado de https://www.kaggle.com/datasets/itshappy/ps4e9-original-data-loan-approval-prediction/data

OpenAI. (2024). *ChatGPT (v4)* [KNIME]. Recuperado de https://chat.openai.com

Problema 2: Predicción de una segunda cita

1. Introducción

El objetivo de este proyecto es desarrollar modelos de aprendizaje automático para predecir la posibilidad de una segunda cita entre participantes de citas rápidas. Este problema de clasificación se centra en determinar si ambos participantes decidirán tener una segunda cita, basándonos en diversas características individuales, preferencias personales y evaluaciones mutuas realizadas durante la primera cita. Para ello, se empleará un análisis de los cinco algoritmos de clasificación detallados a continuación, pero antes se realizará un análisis previo de los datos para comprender sus características generales.

El conjunto de datos cuenta con 8378 instancias y 121 variables que representan características relevantes para la evaluación de un crédito. A continuación, se detallan las principales características de estas variables:

- **S Gender**: Género del participante
- **S Age**: Edad del participante
- **S Age_o**: Edad de la pareja
- **D_age**: Diferencia de edad entre los dos
- **Race**: Raza del participante
- **Race_o**: Raza de la pareja
- **Samerace**: Indica si ambos son de la misma raza
- **S Importance_same_race**: Importancia de que la pareja sea de la misma raza
- **S Importance_same_religion**: Importancia de que la pareja tenga la misma religión
- **S Field**: Área de estudio o trabajo
- **Pref_o_attractive**: Importancia que la pareja da a la apariencia
- **Pref_o_sincere**: Importancia que la pareja da a la sinceridad
- **Pref_o_intelligence**: Importancia que la pareja da a la inteligencia
- **Pref_o_funny**: Importancia que la pareja da a ser divertido/a
- **Pref_o_ambitious**: Importancia que la pareja da a la ambición
- **Pref_o_shared_interests**: Importancia que la pareja da a tener intereses compartidos
- **Attractive_o**: Calificación de la pareja (sobre mí) en cuanto a apariencia en la noche de la cita inicial
- **Sincere_o**: Calificación de la pareja (sobre mí) en cuanto a sinceridad en la noche de la cita inicial
- **Intelligence_o**: Calificación de la pareja (sobre mí) en cuanto a inteligencia en la noche de la cita inicial
- **Funny_o**: Calificación de la pareja (sobre mí) en cuanto a ser divertido/a en la noche de la cita inicial
- **Ambitious_o**: Calificación de la pareja (sobre mí) en cuanto a ambición en la noche de la cita inicial
- **Shared_interests_o**: Calificación de la pareja (sobre mí) en cuanto a intereses compartidos en la noche de la cita inicial
- **Attractive_important**: Importancia de la apariencia en una pareja
- **Sincere_important**: Importancia de la sinceridad en una pareja
- **Intelligence_important**: Importancia de la inteligencia en una pareja
- **Funny Important**: Importancia de ser divertido/a en una pareja

- Ambition_important: Importancia de la ambición en una pareja
- Shared_interests_important: Importancia de tener intereses compartidos en una pareja
- Attractive: Autoevaluación en apariencia
- Sincere: Autoevaluación en sinceridad
- Intelligence: Autoevaluación en inteligencia
- Funny: Autoevaluación en ser divertido/a
- Ambition: Autoevaluación en ambición
- Attractive_partner: Evaluación de la apariencia de la pareja
- Sincere_partner: Evaluación de la sinceridad de la pareja
- Intelligence_partner: Evaluación de la inteligencia de la pareja
- Funny_partner: Evaluación de la pareja en cuanto a ser divertido/a
- Ambition_partner: Evaluación de la ambición de la pareja
- Shared_interests_partner: Evaluación de los intereses compartidos con la pareja
- Sports: Interés en deportes **[Tvsports**, **Exercise**, **Dining**, **Museums**, **Art**, **Hiking**, **Gaming**, **Clubbing**, **Reading**, **Tv**, **Theater**, **Movies**, **Concerts**, **Music**, **Shopping**, **Yoga**]
- Interests_correlate: Correlación entre los intereses del participante y de su pareja
- Expected_happy_with_sd_people: Nivel de felicidad esperado con las personas que se conocerán en el evento de citas rápidas
- Expected_num_interested_in_me: Número estimado de personas interesadas en salir con el participante
- Expected_num_matches: Número de coincidencias que el participante espera obtener
- Like: Indicador de si el participante mostró agrado por la pareja
- Guess_prob_liked: Probabilidad estimada de que la pareja haya mostrado agrado hacia el participante
- Met: Indica si el participante y la pareja se habían conocido antes
- Decision: Decisión de seguir o no con una segunda cita
- Decision_o: Decisión de la pareja de tener una segunda cita
- Match: Indicador de coincidencia mutua para una segunda cita

Tipos de Variables

Las 121 variables del conjunto de datos se dividen en los siguientes tipos:

- 59 variables numéricas
- 62 variables simbólicas

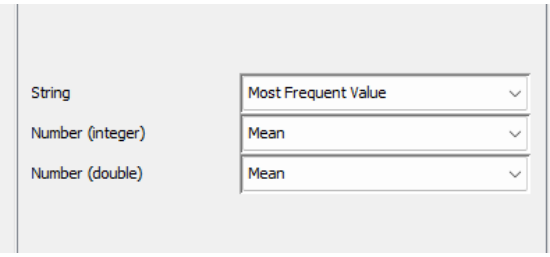
Particularidades del Conjunto de Datos

Valores Faltantes: El conjunto de datos presenta 18,372 valores faltantes distribuidos en diversas variables, lo que representa un desafío para la precisión de los modelos. Estos valores faltantes deberán ser tratados adecuadamente durante el procesamiento de datos para evitar sesgos en el análisis.

Para ello se ha usado el nodo de Knime *Missing Value* con los siguientes valores:

- Para el caso de disponer de número enteros se realiza la media aritmética del conjunto de datos para establecer el valor de esta tupla.

- Para el caso de disponer de cadenas de String se ha seguido el criterio de el valor más frecuente.
- Y para el caso de disponer de números dobles se ha realizado la media aritmética de los valores del conjunto de datos para establecer el valor asignado a ese caso.



Clases: El problema de clasificación en este conjunto de datos cuenta con 2 clases.
Esta información preliminar será crucial para definir el enfoque de preprocesamiento y seleccionar los algoritmos de clasificación más adecuados para el análisis.

Desbalanceo de Clases: Dado que la variable objetivo *Match* es de tipo categórica, se debe verificar la proporción de las clases (si hay 2ª cita o no) para identificar si existe un desbalance significativo. En este caso podemos observar aplicando el nodo bar chart de la clase match lo siguiente:

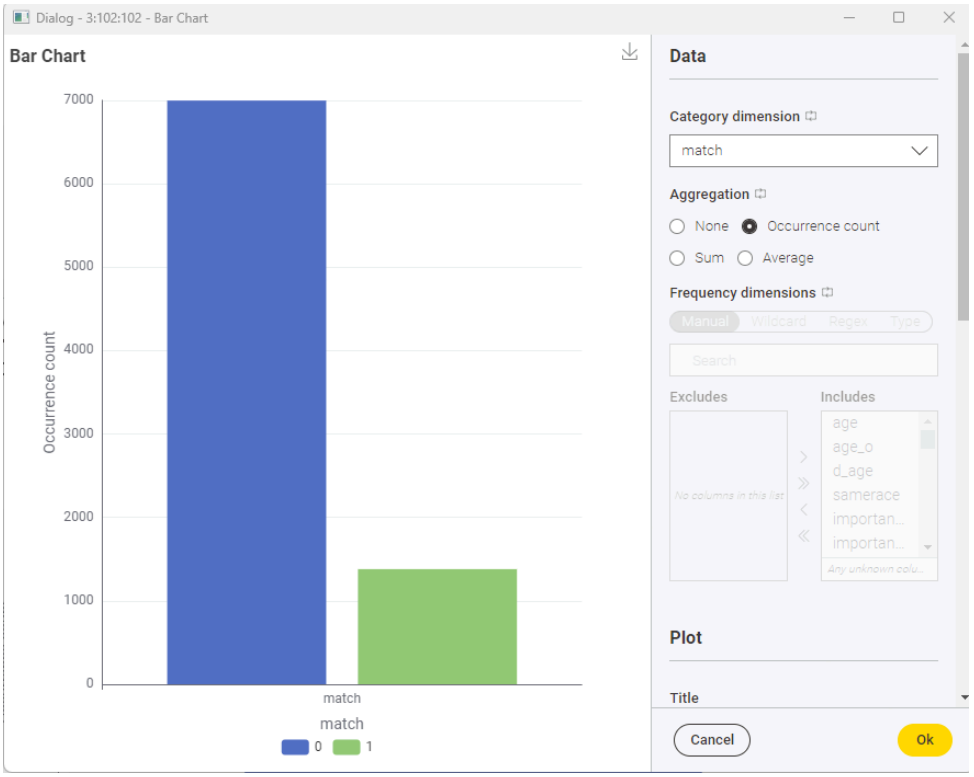


Figura 1 podemos ver la distribución de coincidencias en la variable Match. En dicha figura se observa la cantidad de casos en los que hubo coincidencia mutua para una segunda cita (1) frente a los casos en los que no hubo coincidencia (0). La mayoría de las instancias corresponden a casos sin coincidencia.

Relevancia de las Variables: Se espera que variables como **Attractive_o**, **Intelligence_o**, **Shared_interests_o**, y **Like** tengan un impacto significativo en la predicción de la decisión de tener una segunda cita, ya que están directamente relacionadas con la percepción mutua y el interés mostrado por los participantes durante la primera cita. Además, características como **Sincere_important** y **Intelligence_important** reflejan las preferencias personales de los participantes, lo cual podría influir en la compatibilidad y, por lo tanto, en la probabilidad de coincidencia mutua para una segunda cita.

Eliminación de Tuplas con Valores Anómalos o Fuera de Rango: en este dataset, debido a la complejidad de comprobar los valores de cada variable, se ha hecho un supervisión global de las variables con una fuerte correlación con respecto a la variable objetivo, y la mayoría de ellas poseen valores dentro de los rangos establecidos.

Con relación al conjunto de datos posee 8378 instancias, por tanto he decidido que son suficientes y que no son necesarias simplificar o reducir el número de instancias.

Por último, para abordar el problema de la predicción de la decisión de tener una segunda cita en este conjunto de datos, se seleccionaron los siguientes algoritmos de aprendizaje automático. Cada uno aporta características únicas que permiten explorar diferentes enfoques y mejorar la precisión de la predicción:

1. **Árbol de Decisión**: Este modelo permite identificar de forma clara y visual los factores más relevantes en la decisión de una segunda cita, como la autoevaluación en atributos importantes y las preferencias de los participantes.
2. **k-Nearest Neighbors (k-NN)**: Este algoritmo evalúa la similitud entre los participantes y sus parejas basándose en características como intereses comunes, calificaciones mutuas y compatibilidad general, lo que ayuda a identificar patrones en la decisión de una segunda cita.
3. **Stochastic Gradient Descent (SGD)**: Este método de optimización es útil para modelos lineales y permite actualizar los parámetros iterativamente. Es eficaz cuando se trabaja con grandes volúmenes de datos y puede adaptarse para modelar la probabilidad de una segunda cita en función de variables clave.
4. **Random Forest**: Combina múltiples árboles de decisión para mejorar la precisión y robustez en la predicción, ayudando a capturar interacciones complejas entre características como la importancia de atributos y las calificaciones mutuas.
5. **Naive Bayes**: Considera la probabilidad condicional de cada variable en la predicción, lo cual es útil para evaluar cómo factores individuales como la autoevaluación y el agrado mutuo afectan la decisión de una segunda cita.
6. **Gradient Boosted Tree**: Este algoritmo crea una serie de árboles de decisión donde cada árbol posterior corrige los errores del anterior. Es útil para capturar relaciones no lineales y maximizar la precisión en la predicción de la decisión de una segunda cita.

Estos algoritmos ofrecen diferentes enfoques para comprender y predecir los factores que influyen en la decisión de continuar con una segunda cita, permitiendo seleccionar el modelo más adecuado para el problema.

2. Procesado de Datos

Para abordar el problema de predicción de una segunda cita en este conjunto de datos, se ha realizado un preprocesamiento exhaustivo de las variables. Este paso es fundamental para garantizar que los datos estén en óptimas condiciones antes de ser utilizados en los modelos de aprendizaje automático. Concretamente en este proceso se incluyó el tratamiento de valores faltantes o nulos, escalado de variables y el ajuste de desbalanceo de clases.

Tratamiento de valores faltantes

Debido al gran espacio de búsqueda, hemos analizado los resultados que mejor se han obtenido utilizando el nodo missing value, en el que hemos aplicado la media para los valores enteros y dobles, y el valor que más se repite para los strings.

Tratamiento del Desbalanceo de Clases

Para el tratamiento de balanceo de clases se ha creado en todos los algoritmos, concretamente en el metanodo que gestiona cada algoritmo un X particioner, para evaluar el rendimiento de los modelos de manera más confiable.

Para combatir el desbalanceo de la variable clase match dispone de los siguientes atributos:

- Validación cruzada con 10 pliegues, que permite una evaluación robusta del modelo.
- Muestreo estratificado, que asegura la representación de ambas clases en cada pliegue.
- Especificación de la columna de clase, match, como referencia para mantener la proporción de clases, ya que es nuestra variable objetivo en el análisis de la segunda cita.
- Semilla aleatoria establecida para garantizar la reproducibilidad. Este valor se mantiene constante en todas las ejecuciones para obtener resultados consistentes.

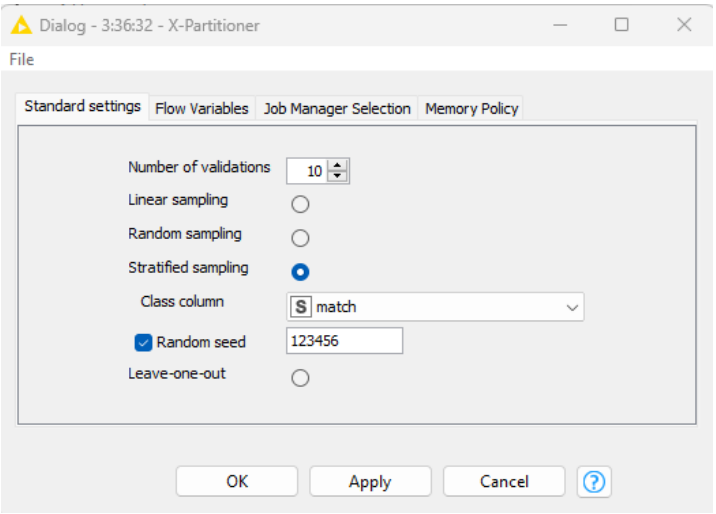


Figura 2 podemos ver la configuración del nodo X-Partitioner para validación cruzada. En dicha figura se observa que el modelo está configurado para realizar una validación cruzada con 10 particiones, utilizando un muestreo estratificado basado en la variable "match" para asegurar una distribución equilibrada de las clases en cada partición. Se ha definido una semilla aleatoria para reproducibilidad.

Escalado de Variables

Algunos de los algoritmos seleccionados, como k-Nearest Neighbors (k-NN) y Gradient Boosted Tree, son sensibles a la escala de las variables. Por ello, se ha aplicado un proceso de escalado estándar a las variables numéricas para garantizar que todas estén en un rango similar. Esto evita que las variables con valores mayores dominen el análisis y ayuda a mejorar la precisión del modelo. La normalización se ha aplicado en el rango min = 0 y max = 1.

Por último, he hecho uso del nodo “Column Filter” con esto lo que he hecho es excluir diferentes variables según el algoritmo ya que no aportan ningún tipo de información relevante. En algunos nodos me salía algún Warning sobre la clase en cuestión así que lo más recomendable era quitarlo. A parte, afectaba de manera positiva a los algoritmos por lo que finalmente he hecho uso de este procesado. El flujo de trabajo quedaría de la siguiente forma:

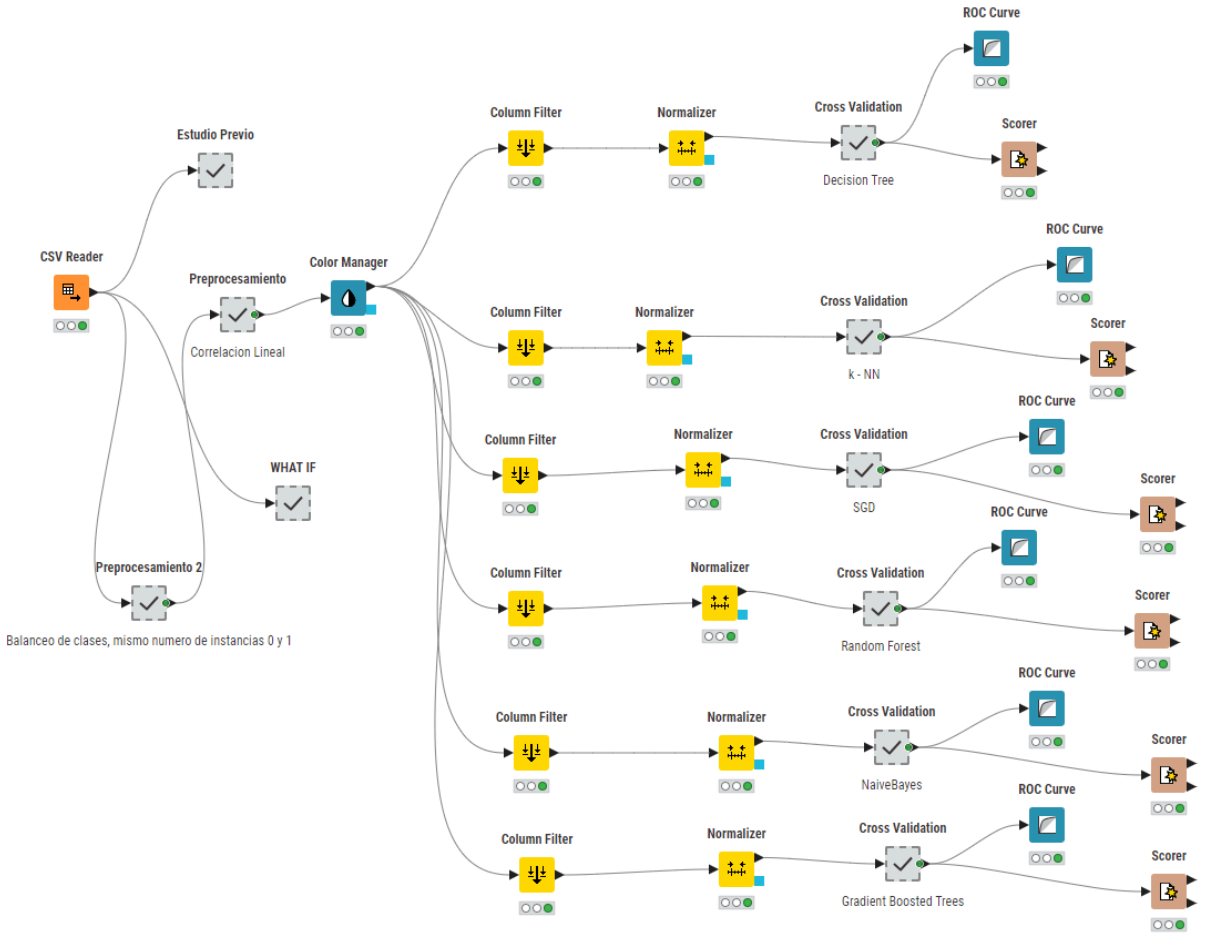


Figura 3 podemos ver el flujo de trabajo en KNIME para la evaluación de modelos de clasificación en el análisis de citas. En dicha figura se muestra el proceso completo desde la lectura de datos hasta la evaluación de los modelos. Los datos se procesan y se aplican diferentes algoritmos de clasificación, incluyendo **Decision Tree, k-Nearest Neighbors (k-NN), Stochastic Gradient Descent (SGD), Random Forest, Naïve Bayes, y Gradient Boosted Trees**. Cada modelo pasa por un proceso de validación cruzada, y su rendimiento se evalúa mediante curvas ROC y métricas generadas por el nodo Scorer.

Para evaluar el rendimiento de cada algoritmo en el problema de predicción de una segunda cita, se emplearon dos nodos clave en KNIME: **Scorer** y **ROC Curve**. El nodo **Scorer** permite recopilar en una tabla las métricas principales de cada modelo, tales como precisión, recall, F1-score y exactitud, lo cual facilita la comparación cuantitativa entre los distintos algoritmos y su capacidad para predecir la decisión de una segunda cita. Adicionalmente, se utilizó el nodo **ROC Curve** para visualizar gráficamente la capacidad de cada modelo para distinguir entre las clases (decisión positiva o negativa para una segunda cita), mediante el cálculo del AUC (Área bajo la curva ROC).

Esta estructura de evaluación asegura una comparación completa y uniforme del rendimiento de los modelos, permitiendo identificar de manera objetiva cuál de ellos ofrece la mejor capacidad predictiva en el contexto de decisiones de citas rápidas.

3. Resultados Obtenidos

El metanodo *Cross Validation* de los algoritmos es igual para todos. En primer lugar, toda la experimentación de los algoritmos se realiza con validación cruzada de 10 particiones, para ello he usado los nodos “X-Partitioner” y “X-Aggregator”. Para ello he ajustado los parámetros, poniendo 10 validaciones, se ha seleccionado la opción de Stratified Sampling para asegurar que cada subconjunto de entrenamiento y prueba mantenga la misma proporción de clases que el conjunto de datos original. Se estableció una semilla aleatoria la cual es 123456 y eligiendo la columna loan_status. Podemos verlo en las siguientes imágenes:

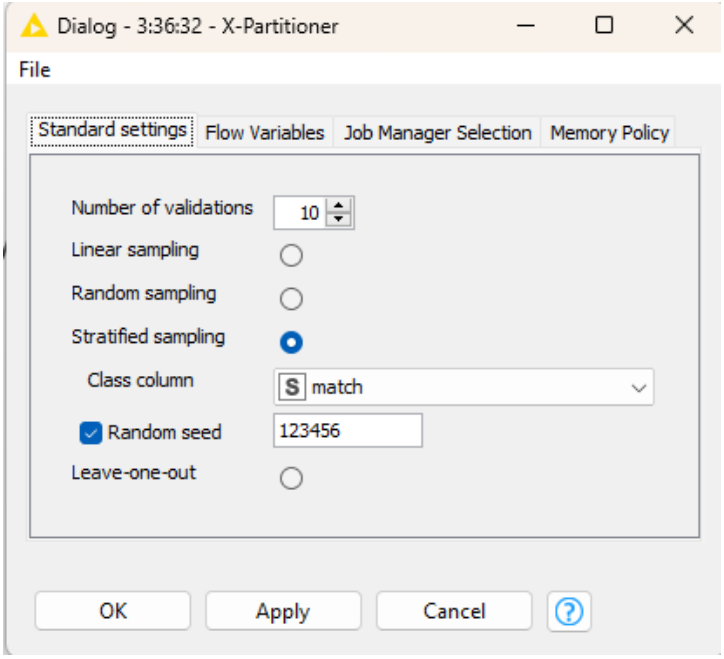


Figura 3 muestra la configuración del nodo X-Partitioner para validación cruzada, con 10 particiones y muestreo estratificado en la variable "match" para mantener el equilibrio de clases. La semilla aleatoria es 123456 para asegurar la reproducibilidad.

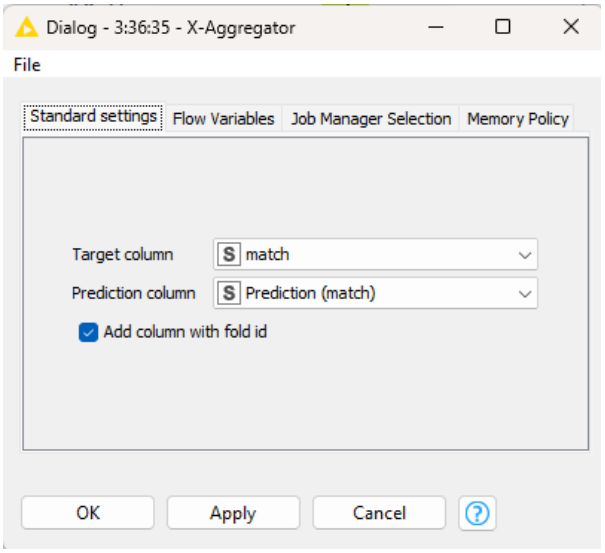


Figura 4 muestra la configuración del nodo X-Aggregator, con la columna objetivo "match" y la columna de predicción "Prediction (match)". La opción para agregar una columna con el ID de partición está habilitada.

En esta sección, se presentan los resultados de cada uno de los algoritmos empleados en la predicción del estado de una cita. Para cada algoritmo, se muestra el flujo de trabajo correspondiente en KNIME y se analizan las métricas de evaluación obtenidas, incluyendo la curva ROC y el AUC, con el fin de comparar el rendimiento entre modelos.

1. Decision Tree

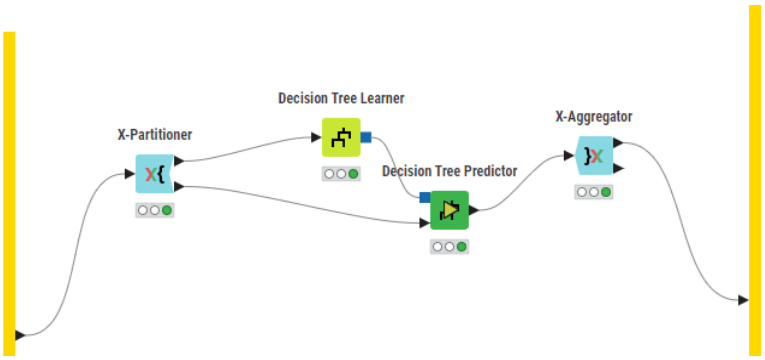


Figura 5 muestra el flujo de trabajo en KNIME para el modelo de Árbol de Decisión, incluyendo los nodos X-Partitioner, Decision Tree Learner, Decision Tree Predictor y X-Aggregator, configurados para realizar la validación cruzada y evaluar el rendimiento del modelo.

Clase	TP	FP	TN	FN	Recall	Precisión	TPR	TNR	F1	Accuracy
0 (sin 2ª cita)	6668	932	448	330	0,953	0,877	0,953	0,325	0,914	
1 (con 2ª cita)	448	330	6668	932	0,325	0,576	0,325	0,953	0,415	
Overall										0,849

Figura 6 muestra una tabla de métricas para evaluar el rendimiento del modelo, incluyendo valores de TP (True Positives), FP (False Positives), TN (True Negatives), FN (False Negatives), Recall, Precisión, TPR (True Positive Rate), TNR (True Negative Rate), F1 y Accuracy. La tabla distingue entre las clases "sin 2ª cita" y "con 2ª cita", con métricas generales en la fila Overall.

Clase	0 (sin 2ª cita)	1(con 2ª cita)
0 (sin 2ª cita)	6628	330
1(con 2ª cita)	932	448

Figura 7 muestra la matriz de confusión del modelo, con predicciones para las clases "sin 2ª cita" y "con 2ª cita". Los valores en cada celda indican el número de instancias correctamente clasificadas o incorrectamente clasificadas entre estas dos clases.

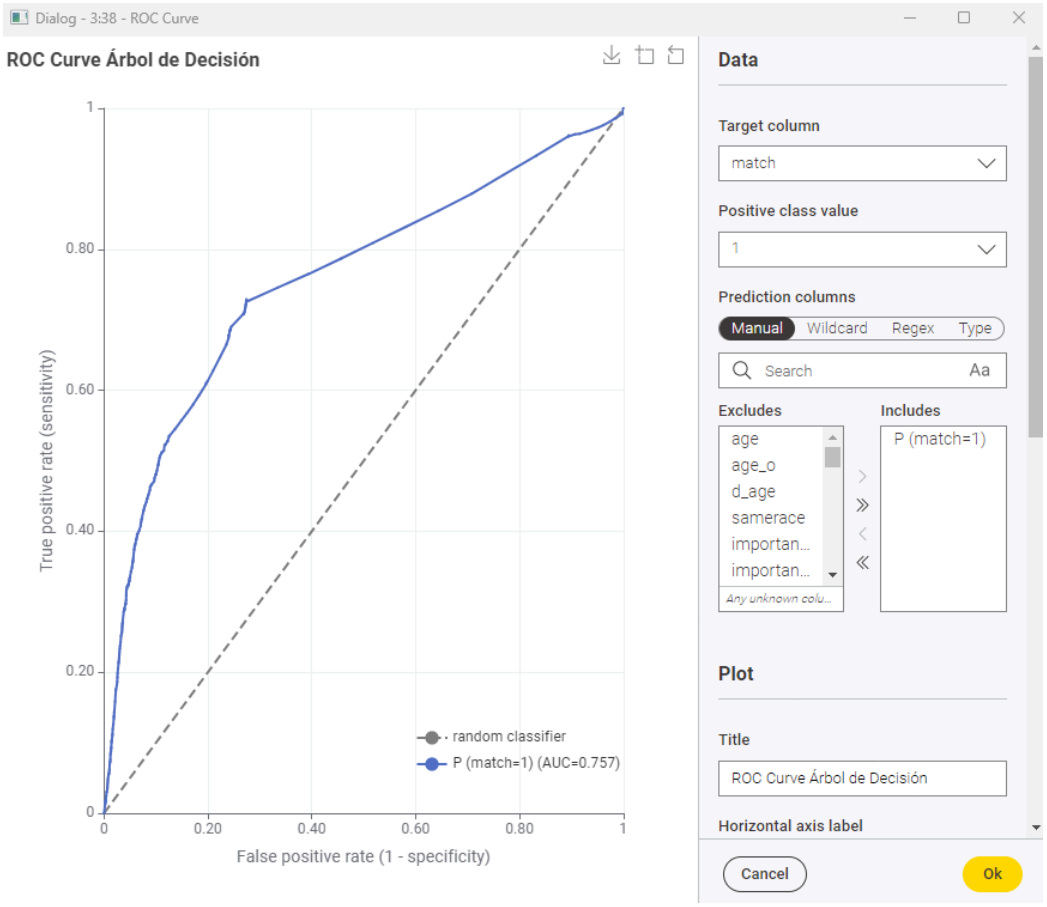


Figura 8 muestra la curva ROC para el modelo de Árbol de Decisión, con un AUC de 0.757. En el gráfico, la línea azul representa el desempeño del modelo al diferenciar entre las clases, mientras que la línea punteada gris indica el clasificador aleatorio. La curva refleja la tasa de verdaderos positivos frente a la tasa de falsos positivos, evaluando la capacidad del modelo para distinguir entre instancias de segunda cita y no segunda cita.

El Árbol de Decisión muestra un desempeño moderado en la predicción de la segunda cita, con un AUC de 0.757, indicando una capacidad aceptable para distinguir entre las clases. La precisión general fue de 0.849, destacando una alta sensibilidad (0.953) en la

identificación de casos negativos ("sin segunda cita") pero baja especificidad (0.325) para los casos positivos ("con segunda cita"), debido al desbalance de clases. Este modelo es útil para interpretar los factores clave en la decisión de una segunda cita, aunque su rendimiento en la clase minoritaria podría mejorar con técnicas de re-muestreo o ajustes de hiperparámetros.

2. k-NN

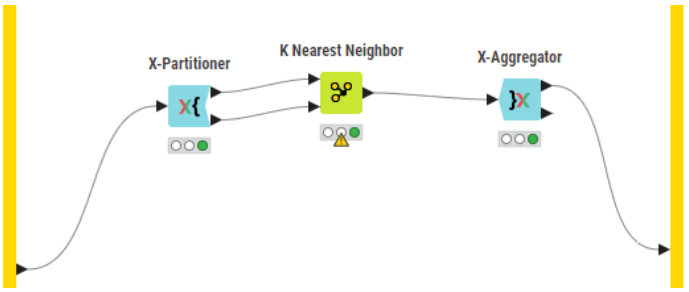


Figura 9 muestra el flujo de trabajo en KNIME para el modelo k-Nearest Neighbors (k-NN), incluyendo los nodos X-Partitioner, K Nearest Neighbor y X-Aggregator. Este flujo permite realizar la validación cruzada y evaluar el rendimiento del modelo en la predicción de una segunda cita.

Clase	TP	FP	TN	FN	Recall	Precisión	TPR	TNR	F1	Accuracy
0 (sin 2ª cita)	6815	1159	221	183	0,974	0,855	0,974	0,16	0,91	
1 (con 2ª cita)	221	183	6815	1159	0,16	0,547	0,16	0,974	0,248	
Overall										0,84

Figura 10 muestra una tabla de métricas para el modelo k-NN, incluyendo valores de TP (True Positives), FP (False Positives), TN (True Negatives), FN (False Negatives), Recall, Precisión, TPR (True Positive Rate), TNR (True Negative Rate), F1 y Accuracy. La tabla distingue entre las clases "sin 2ª cita" y "con 2ª cita", con métricas generales en la fila Overall.

Clase	0 (sin 2ª cita)	1(con 2ª cita)
0 (sin 2ª cita)	6815	183
1(con 2ª cita)	1159	221

Figura 11 muestra la matriz de confusión del modelo k-NN, con predicciones para las clases "sin 2ª cita" y "con 2ª cita". Los valores indican el número de instancias correctamente clasificadas y los errores de clasificación entre ambas clases.

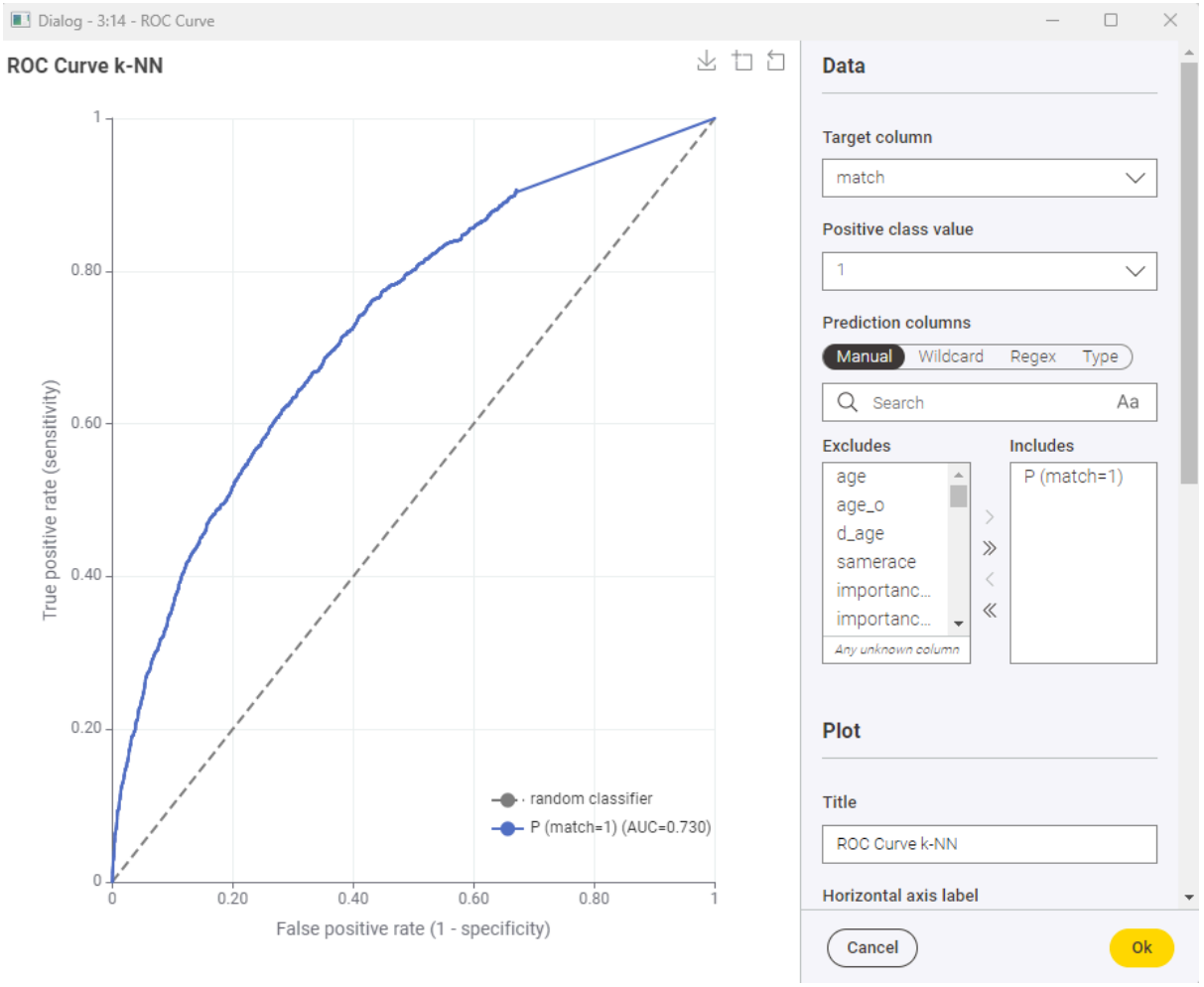


Figura 12 muestra la curva ROC para el modelo k-NN, con un AUC de 0.730. La línea azul representa el desempeño del modelo al clasificar entre las clases "sin 2ª cita" y "con 2ª cita", mientras que la línea punteada gris indica el rendimiento de un clasificador aleatorio. La curva ilustra la tasa de verdaderos positivos frente a la tasa de falsos positivos, evaluando la efectividad del modelo en la predicción de una segunda cita.

Como conclusión, el algoritmo k-NN presenta un rendimiento moderado en la clasificación de la decisión de una segunda cita. Con un AUC de 0.730, la curva ROC indica una capacidad razonable para distinguir entre las clases, aunque es inferior a otros modelos en precisión. Las métricas de precisión y F1-score sugieren que k-NN es eficaz para identificar la clase mayoritaria ("sin segunda cita"), pero su rendimiento disminuye al clasificar la clase minoritaria ("con segunda cita"), lo que refleja su sensibilidad al desbalance de clases en los datos.

3. SGD

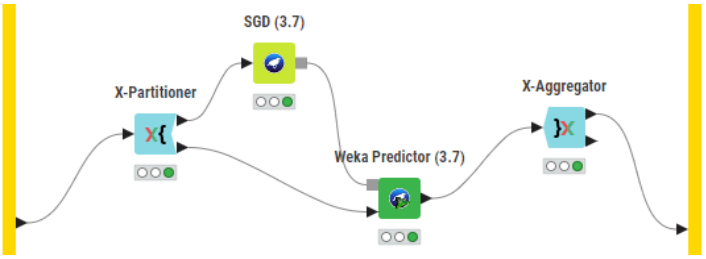


Figura 13 muestra el flujo de trabajo en KNIME para el modelo Stochastic Gradient Descent (SGD), incluyendo los nodos X-Partitioner, SGD (3.7), Weka Predictor (3.7) y X-Aggregator. Este flujo permite realizar la validación cruzada y evaluar el rendimiento del modelo en la predicción de una segunda cita.

Clase	TP	FP	TN	FN	Recall	Precisión	TPR	TNR	F1	Accuracy
0 (sin 2ª cita)	6509	782	598	489	0,93	0,893	0,93	0,433	0,911	
1 (con 2ª cita)	598	489	6509	782	0,433	0,55	0,433	0,93	0,485	
Overall										0,848

Figura 14 muestra una tabla de métricas para evaluar el modelo, incluyendo valores de TP (True Positives), FP (False Positives), TN (True Negatives), FN (False Negatives), Recall, Precisión, TPR (True Positive Rate), TNR (True Negative Rate), F1 y Accuracy. La tabla distingue entre las clases "sin 2ª cita" y "con 2ª cita", con métricas generales en la fila Overall.

Clase	0 (sin 2ª cita)	1(con 2ª cita)
0 (sin 2ª cita)	6509	489
1(con 2ª cita)	782	598

Figura 15 muestra la matriz de confusión del modelo, con predicciones para las clases "sin 2ª cita" y "con 2ª cita". Los valores indican el número de instancias correctamente clasificadas y los errores de clasificación entre ambas clases.

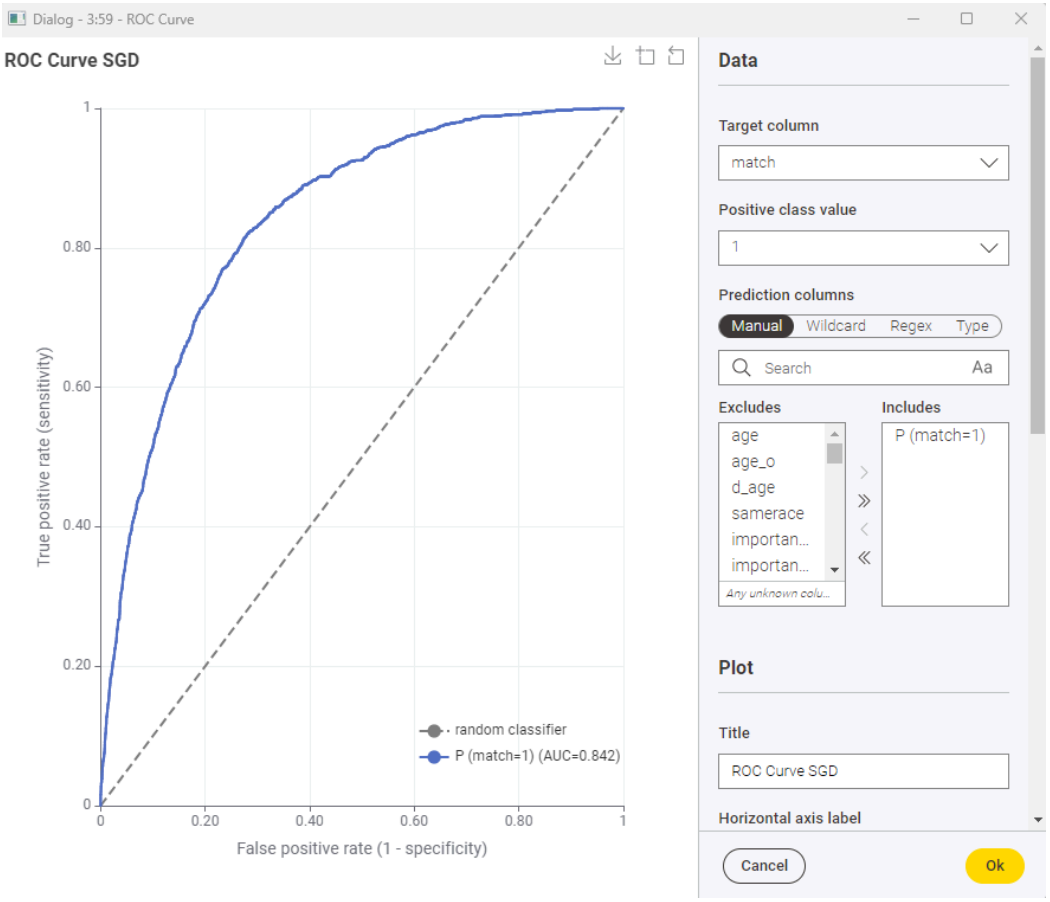


Figura 16 muestra la curva ROC para el modelo Stochastic Gradient Descent (SGD), con un AUC de 0.842. La línea azul representa la capacidad del modelo para diferenciar entre las clases "sin 2ª cita" y "con 2ª cita", mientras que la línea punteada gris indica el rendimiento de un clasificador aleatorio. La curva ilustra la tasa de verdaderos positivos frente a la tasa de falsos positivos, evaluando la efectividad del modelo en la predicción de una segunda cita.

Para el modelo Stochastic Gradient Descent, el valor de AUC es de 0.842 en la curva ROC, lo que indica una buena capacidad de discriminación entre las clases "sin segunda cita" y "con segunda cita". El modelo muestra una alta precisión y F1-score en la clase "sin segunda cita", lo que sugiere que es eficaz para identificar correctamente los casos donde no se espera una segunda cita. Sin embargo, el recall y F1-score más bajos en la clase "con segunda cita" reflejan una menor sensibilidad en esta clase, indicando que el modelo tiende a clasificar mejor las instancias de la clase mayoritaria. En general, el rendimiento del SGD es aceptable, aunque podría mejorarse en la clase minoritaria.

4. Random Forest

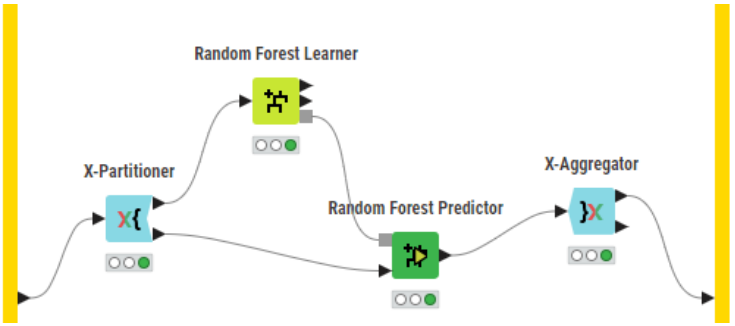


Figura 17 muestra el flujo de trabajo en KNIME para el modelo Random Forest, incluyendo los nodos X-Partitioner, Random Forest Learner, Random Forest Predictor y X-Aggregator. Este flujo permite realizar la validación cruzada y evaluar el rendimiento del modelo en la predicción de una segunda cita.

Clase	TP	FP	TN	FN	Recall	Precisión	TPR	TNR	F1	Accuracy
0 (sin 2ª cita)	6907	1096	284	91	0,987	0,863	0,987	0,206	0,921	
1 (con 2ª cita)	284	91	6907	1096	0,206	0,757	0,206	0,987	0,324	
Overall										0,858

Figura 18 muestra una tabla de métricas para el modelo Random Forest, incluyendo valores de TP (True Positives), FP (False Positives), TN (True Negatives), FN (False Negatives), Recall, Precisión, TPR (True Positive Rate), TNR (True Negative Rate), F1 y Accuracy. La tabla distingue entre las clases "sin 2ª cita" y "con 2ª cita", con métricas generales en la fila Overall.

Clase	0 (sin 2ª cita)	1(con 2ª cita)
0 (sin 2ª cita)	6907	91
1(con 2ª cita)	1096	284

Figura 19 muestra la matriz de confusión del modelo Random Forest, con predicciones para las clases "sin 2ª cita" y "con 2ª cita". Los valores indican el número de instancias correctamente clasificadas y los errores de clasificación entre ambas clases.

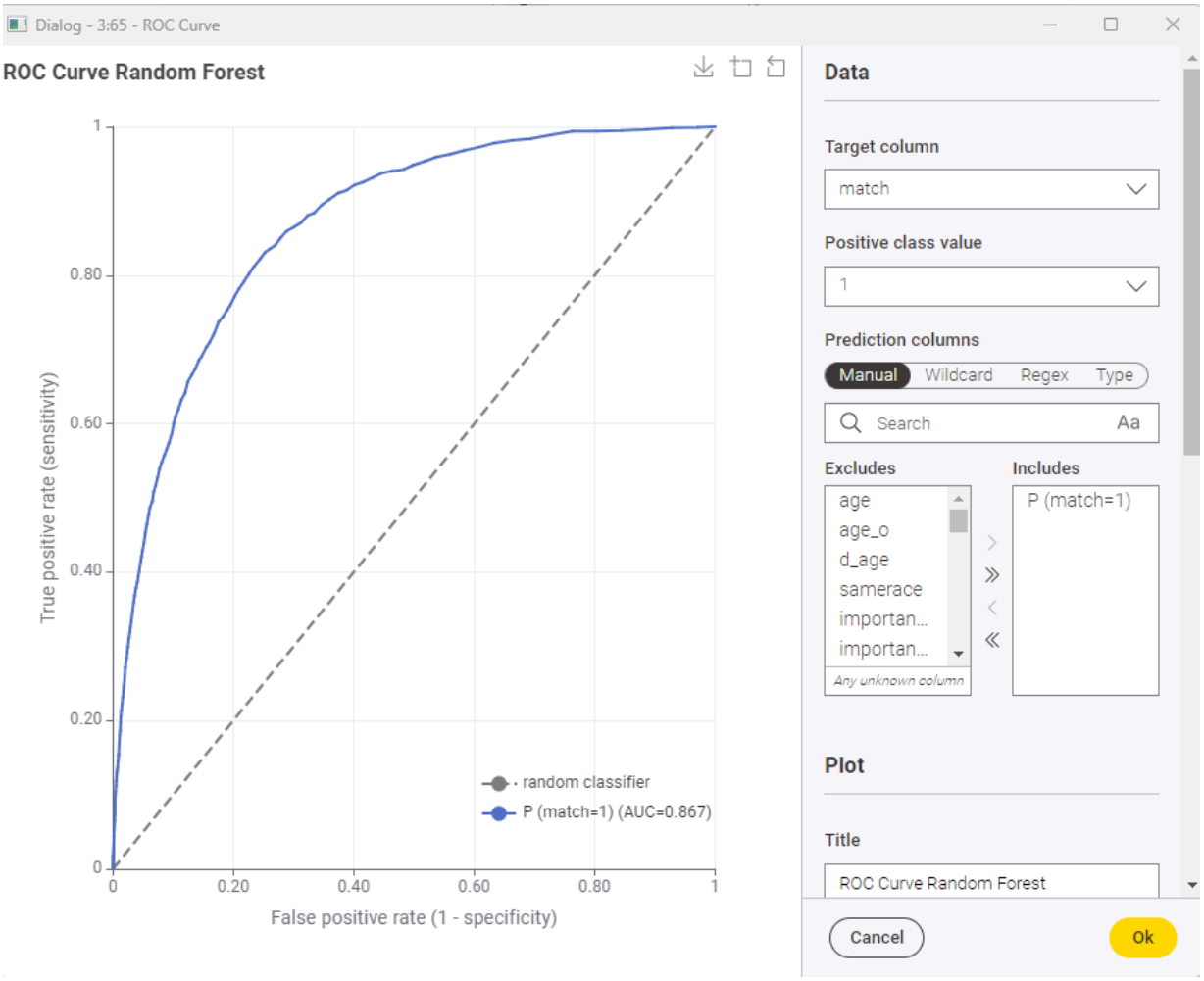


Figura 20 muestra la curva ROC para el modelo Random Forest, con un AUC de 0.867. La línea azul representa la capacidad del modelo para diferenciar entre las clases "sin 2ª cita" y "con 2ª cita", mientras que la línea punteada gris indica el rendimiento de un clasificador aleatorio. La curva ilustra la tasa de verdaderos positivos frente a la tasa de falsos positivos, evaluando la efectividad del modelo en la predicción de una segunda cita.

Para el modelo Random Forest, el valor de AUC es de 0.867, indicando una buena capacidad de discriminación entre las clases "sin segunda cita" y "con segunda cita". El modelo muestra alta precisión y recall en la clase mayoritaria ("sin segunda cita"), lo que sugiere efectividad en identificar correctamente los casos sin interés en una segunda cita. Sin embargo, presenta menor sensibilidad en la clase minoritaria, lo que refleja una tendencia a clasificar más acertadamente la clase mayoritaria. En general, el rendimiento del modelo es robusto y adecuado para este tipo de clasificación.

5. Naive Bayes

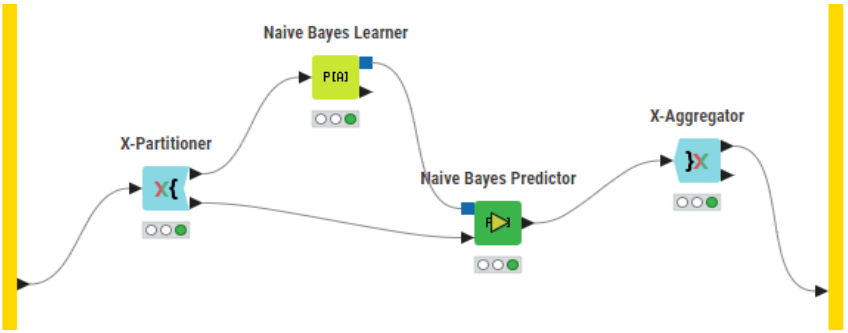


Figura 21 muestra el flujo de trabajo en KNIME para el modelo Naive Bayes, incluyendo los nodos X-Partitioner, Naive Bayes Learner, Naive Bayes Predictor y X-Aggregator. Este flujo permite realizar la validación cruzada y evaluar el rendimiento del modelo en la predicción de una segunda cita.

Clase	TP	FP	TN	FN	Recall	Precisión	TPR	TNR	F1	Accuracy
0 (sin 2ª cita)	5540	405	975	1458	0,792	0,932	0,792	0,707	0,856	
1 (con 2ª cita)	975	1458	5540	405	0,707	0,401	0,707	0,792	0,511	
Overall										0,778

Figura 22 muestra una tabla de métricas para el modelo Naive Bayes, incluyendo valores de TP (True Positives), FP (False Positives), TN (True Negatives), FN (False Negatives), Recall, Precisión, TPR (True Positive Rate), TNR (True Negative Rate), F1 y Accuracy. La tabla distingue entre las clases "sin 2ª cita" y "con 2ª cita", con métricas generales en la fila Overall.

Clase	0 (sin 2ª cita)	1(con 2ª cita)
0 (sin 2ª cita)	5540	1458
1(con 2ª cita)	405	975

Figura 23 muestra la matriz de confusión del modelo Naive Bayes, con predicciones para las clases "sin 2ª cita" y "con 2ª cita". Los valores indican el número de instancias correctamente clasificadas y los errores de clasificación entre ambas clases.

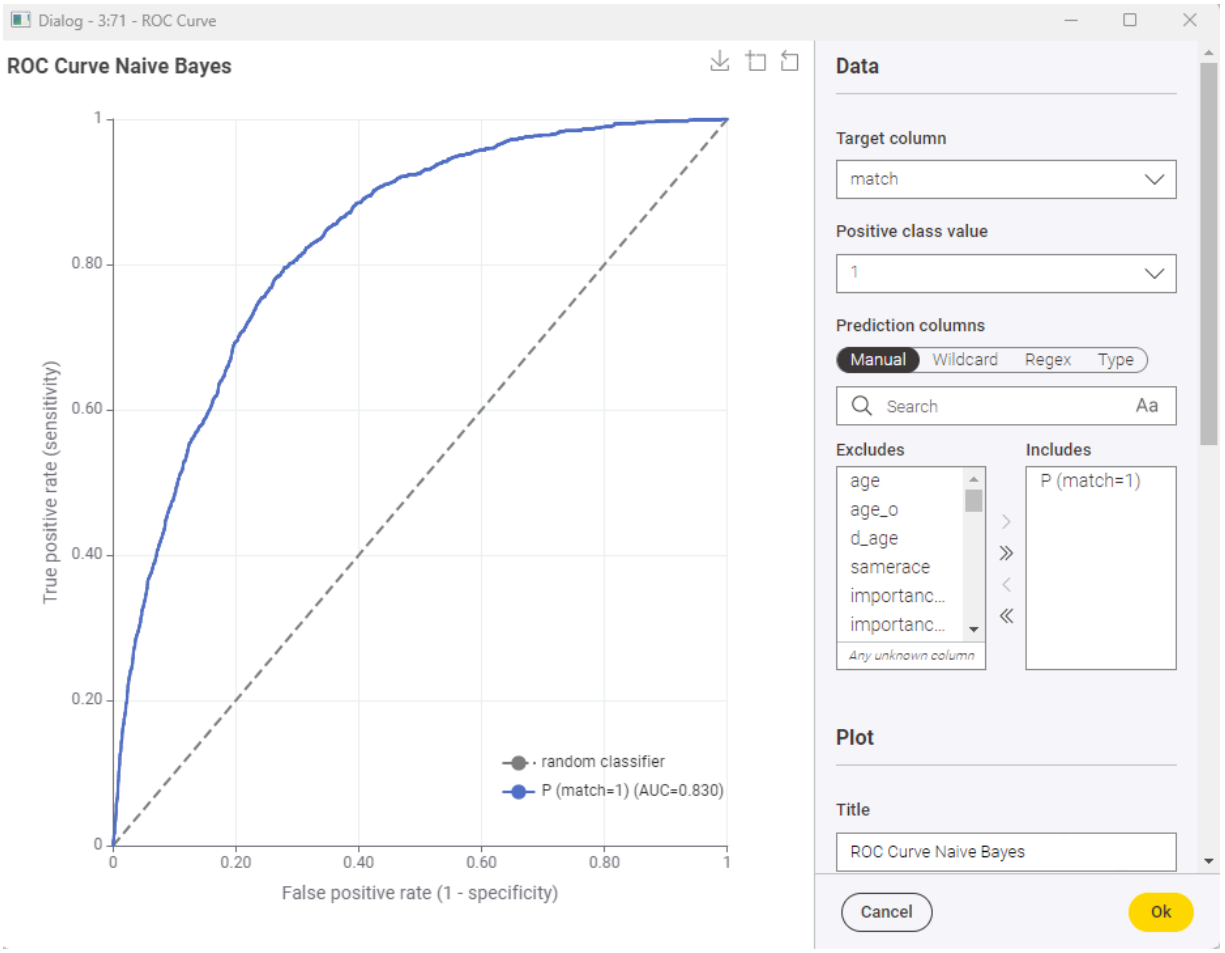


Figura 24 muestra la curva ROC para el modelo Naive Bayes, con un AUC de 0.830. La línea azul representa la capacidad del modelo para diferenciar entre las clases "sin 2ª cita" y "con 2ª cita", mientras que la línea punteada gris indica el rendimiento de un clasificador aleatorio. La curva ilustra la tasa de verdaderos positivos frente a la tasa de falsos positivos, evaluando la efectividad del modelo en la predicción de una segunda cita.

Para el modelo Naive Bayes, el valor de AUC es de 0.830, lo que indica una buena capacidad de discriminación entre las clases "sin segunda cita" y "con segunda cita". El modelo muestra alta precisión en la clase "sin segunda cita", lo que sugiere que identifica correctamente los casos en los que no se espera una segunda cita. Sin embargo, la precisión y el F1-score son más bajos en la clase "con segunda cita", reflejando una menor sensibilidad para esta clase minoritaria. En general, el rendimiento del modelo es aceptable para distinguir entre ambas clases, pero tiene limitaciones en la predicción de la clase "con segunda cita".

6. Gradient Boosted Trees

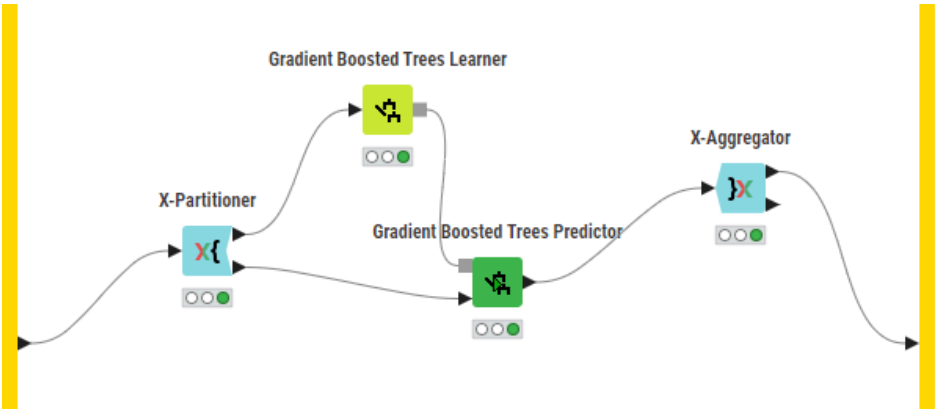


Figura 25 muestra el flujo de trabajo en KNIME para el modelo Gradient Boosted Trees, incluyendo los nodos X-Partitioner, Gradient Boosted Trees Learner, Gradient Boosted Trees Predictor y X-Aggregator. Este flujo permite realizar la validación cruzada y evaluar el rendimiento del modelo en la predicción de una segunda cita.

Clase	TP	FP	TN	FN	Recall	Precisión	TPR	TNR	F1	Accuracy
0 (sin 2ª cita)	6764	871	509	234	0,967	0,886	0,967	0,369	0,924	
1 (con 2ª cita)	509	234	6764	871	0,369	0,685	0,369	0,967	0,48	
Overall										0,868

Figura 26 muestra una tabla de métricas para el modelo Gradient Boosted Trees, incluyendo valores de TP (True Positives), FP (False Positives), TN (True Negatives), FN (False Negatives), Recall, Precisión, TPR (True Positive Rate), TNR (True Negative Rate), F1 y Accuracy. La tabla distingue entre las clases "sin 2ª cita" y "con 2ª cita", con métricas generales en la fila Overall.

Clase	0 (sin 2ª cita)	1(con 2ª cita)
0 (sin 2ª cita)	6764	234
1(con 2ª cita)	871	509

Figura 27 muestra la matriz de confusión del modelo Gradient Boosted Trees, con predicciones para las clases "sin 2ª cita" y "con 2ª cita". Los valores indican el número de instancias correctamente clasificadas y los errores de clasificación entre ambas clases.

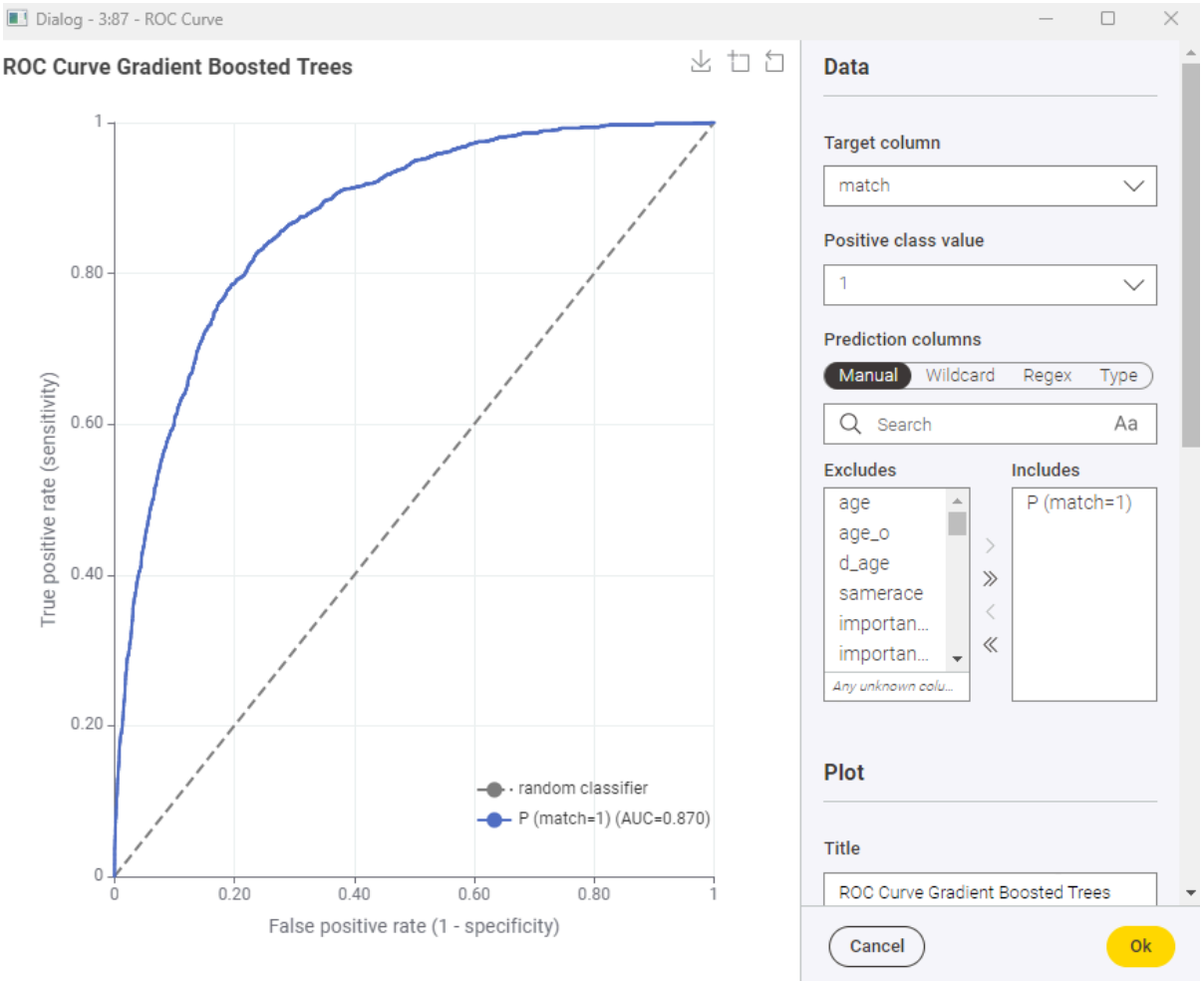


Figura 28 muestra la curva ROC para el modelo Gradient Boosted Trees, con un AUC de 0.870. La línea azul representa la capacidad del modelo para diferenciar entre las clases "sin 2ª cita" y "con 2ª cita", mientras que la línea punteada gris indica el rendimiento de un clasificador aleatorio. La curva ilustra la tasa de verdaderos positivos frente a la tasa de falsos positivos, evaluando la efectividad del modelo en la predicción de una segunda cita.

Para el modelo Gradient Boosted Trees, el AUC de 0.870 indica una buena capacidad de discriminación entre las clases "sin segunda cita" y "con segunda cita". Este modelo muestra una alta precisión y recall para la clase mayoritaria ("sin segunda cita"), lo que sugiere que identifica con precisión los casos donde no se espera una segunda cita. Sin embargo, en la clase minoritaria ("con segunda cita"), el recall y F1-score son más bajos, lo que refleja una menor sensibilidad para identificar estos casos. En general, el rendimiento del modelo es robusto y adecuado para clasificar, aunque podría beneficiarse de ajustes adicionales para mejorar la predicción de la clase minoritaria.

4. Configuración de Algoritmos

Para este estudio, se ha aplicado los diferentes tipos de tratamiento del preprocesamiento, obteniendo así las mejores configuraciones encontradas en el estudio del caso.

1. Árbol de Decisión

He retocando los diferentes parámetros del árbol, he realizado las siguientes configuraciones y he obtenido los siguientes resultados:

Configuración	Criterio	Poda	Min. Registros	AUC	F1-Score (Clase Minoritaria)	Sobreajuste
Por Defecto	Gini	MDL + Reduced Error	4	0,933	0,888	Moderado
Sin Poda	Gain Ratio	MDL + Reduced Error	4	0,895	0,881	Alto

Figura 29 muestra los resultados de dos configuraciones del Árbol de Decisión. La configuración por defecto (Gini, poda MDL + Reduced Error) obtuvo un AUC de 0.933 y un F1-Score de 0.888 con sobreajuste moderado, mientras que sin poda (Gain Ratio) el AUC fue de 0.895 y el F1-Score de 0.881, pero con alto sobreajuste.

Por Defecto: La configuración predeterminada con el criterio Gini y poda con MDL + Reduced Error logró un AUC de 0.933 y un F1-Score de 0.888 en la clase minoritaria. Sin embargo, mostró un nivel moderado de sobreajuste, lo que sugiere que el modelo podría estar ajustándose demasiado a los datos de entrenamiento.

Sin Poda: Al desactivar la poda y cambiar el criterio a Gain Ratio, el modelo presenta un AUC menor (0.895) y un F1-Score ligeramente inferior (0.881). Esta configuración incrementa el riesgo de sobreajuste debido a la falta de restricciones, lo que hace que el modelo se ajuste en exceso a los datos de entrenamiento.

Entre estas configuraciones, la configuración predeterminada con el criterio Gini y poda MDL + Reduced Error ofrece un mejor rendimiento general, con un AUC y F1-Score más altos y un nivel de sobreajuste moderado. Aunque desactivar la poda permite al modelo captar más patrones en los datos de entrenamiento, esto aumenta el riesgo de sobreajuste sin una mejora significativa en la precisión. Por lo tanto, se recomienda utilizar la configuración por defecto para obtener un buen equilibrio entre rendimiento y generalización.

Por tanto, la configuración por defecto con Gini y poda MDL + Reduced Error es la más equilibrada, evitando sobreajuste mientras mantiene una alta precisión en la predicción de la segunda cita.

2. k-NN

Configuración	Número de Vecinos (k)	Métrica de Distancia	Ponderación de Distancia	AUC	F1-Score (Clase Minoritaria)	Sobreajuste
Por Defecto	10	Euclidiana	Si	0,927	0,84	Moderado
Pocos vecinos	3	Euclidiana	SI	0,917	0,937	Alto
Menos vecinos	5	Euclidiana	SI	0,921	0,842	Bajo

Figura 30 muestra los resultados de configuraciones del k-NN. La configuración por defecto (10 vecinos) logra un AUC de 0.927 y un F1-Score de 0.84 con sobreajuste moderado. La configuración con 3 vecinos aumenta el sobreajuste, mientras que con 5 vecinos reduce el sobreajuste pero baja el F1-Score.

Por Defecto: Con 10 vecinos, métrica Euclidiana y ponderación de distancia, esta configuración obtiene un AUC de 0.927 y un F1-Score de 0.84 en la clase minoritaria. Presenta un sobreajuste moderado, manteniendo un buen equilibrio entre precisión y generalización.
Pocos Vecinos: Con solo 3 vecinos y ponderación de distancia activada, el modelo logra un alto F1-Score de 0.937 en la clase minoritaria, pero presenta un menor AUC (0.917) y un sobreajuste alto, lo que sugiere que el modelo es muy sensible a los datos de entrenamiento.
Menos Vecinos: Reduciendo a 5 vecinos y ponderando la distancia, se obtiene un AUC de 0.921 y un F1-Score de 0.842. Esta configuración muestra un bajo sobreajuste, lo que mejora la capacidad de generalización del modelo a costa de una ligera disminución en la precisión.

La **configuración por defecto** es la mejor opción, ya que ofrece un buen equilibrio con un **AUC de 0.927**, **F1-Score de 0.84**, y sobreajuste moderado. Es adecuada para obtener resultados consistentes sin un alto riesgo de sobreajuste.

3. **SGD**

Configuración	Épocas	learning Rate	lambda	Función de Pérdida	AUC	F1-Score (Clase Minoritaria)	Sobreajuste
Configuración por Defecto	500	0,001	1,00E-04	Log loss	0,948	0,875	Moderado
Menor Número de Épocas	100	0,001	1,00E-04	Log loss	0,947	0,971	Alto
Mayor Tasa de Aprendizaje	500	0,005	1,00E-04	Log loss	0,948	0,875	Moderado
Función de Pérdida Alternativa (Hinge)	500	0,001	1,00E-04	Hinge loss (SVM)	0,877	0,87	Bajo

Figura 31 muestra los resultados de distintas configuraciones para el algoritmo SGD. La configuración por defecto, con 500 épocas y Log loss, ofrece un buen equilibrio con un AUC de 0.948 y F1-Score de 0.875. Reducir las épocas a 100 aumenta el F1-Score a 0.971, pero incrementa el sobreajuste. Usar Hinge loss reduce el sobreajuste, aunque con un menor AUC de 0.877**.**

Configuración por Defecto: Con 500 épocas, una tasa de aprendizaje de 0.001 y la función de pérdida Log loss, esta configuración alcanza un AUC de 0.948 y un F1-Score de 0.875. Muestra un sobreajuste moderado.

Menor Número de Épocas: Reducir el número de épocas a 100 mantiene un AUC similar (0.947) pero incrementa el F1-Score a 0.971, lo que indica un mayor ajuste en la clase minoritaria, aunque aumenta el riesgo de sobreajuste.

Mayor Tasa de Aprendizaje: Incrementar la tasa de aprendizaje a 0.005 no mejora el AUC ni el F1-Score en comparación con la configuración por defecto, manteniendo el sobreajuste en un nivel moderado.

Función de Pérdida Alternativa: Usar Hinge loss en lugar de Log loss reduce el AUC a 0.877 y el F1-Score a 0.87, pero disminuye el sobreajuste, mejorando la capacidad de generalización del modelo.

La **configuración por defecto** ofrece un buen equilibrio entre AUC y F1-Score con un sobreajuste moderado. La configuración de menor número de épocas maximiza el F1-Score en la clase minoritaria, pero a costa de un mayor sobreajuste. Por otro lado, la función de pérdida Hinge reduce el sobreajuste, aunque con un menor rendimiento en AUC y F1-Score.

4. Random Forest

Configuración	Número de Árboles	Criterio	Profundidad Máxima	Tamaño Mínimo de Nodo	AUC	F1-Score (Clase Minoritaria)	Sobreajuste
Configuración por Defecto	200	Gini	20	3	0,967	0,906	Moderado
Menor Número de Árboles	100	Gini	20	3	0,967	0,905	Moderado
Mayor Tamaño de Nodo	200	Gini	20	5	0,965	0,9	Bajo
Criterio Gain	200	Gain	20	3	0,967	0,903	Moderado

Figura 32 muestra que la configuración por defecto del Random Forest (200 árboles, criterio Gini) logra un AUC de 0.967 y F1-Score de 0.906 con sobreajuste moderado. Aumentar el tamaño de nodo reduce el sobreajuste con una ligera baja en F1-Score, mientras que otras variaciones no mejoran el rendimiento.

Configuración por Defecto: Con 200 árboles, criterio Gini y tamaño mínimo de nodo de 3, esta configuración logra un AUC de 0.967 y un F1-Score de 0.906. Presenta un sobreajuste moderado.

Menor Número de Árboles: Reducir el número de árboles a 100 mantiene el mismo AUC (0.967) y prácticamente el mismo F1-Score (0.905), con un sobreajuste moderado similar al de la configuración por defecto.

Mayor Tamaño de Nodo: Aumentar el tamaño mínimo del nodo a 5 reduce ligeramente el AUC a 0.965 y el F1-Score a 0.900, pero disminuye el sobreajuste, mejorando la capacidad de generalización del modelo.

Criterio Gain: Cambiar el criterio a Gain mantiene el AUC en 0.967 y el F1-Score en 0.903, con un sobreajuste moderado. Este ajuste no ofrece ventajas significativas en comparación con el criterio Gini.

La **configuración por defecto** proporciona el mejor equilibrio entre AUC y F1-Score, con un sobreajuste moderado. Reducir el número de árboles o cambiar el criterio no mejora el rendimiento significativamente, mientras que aumentar el tamaño mínimo del nodo reduce el sobreajuste, aunque a costa de una ligera disminución en el F1-Score.

5. Naive Bayes

Configuración	Probabilidad por Defecto	Desviación Estándar Mínima	Umbral de Desviación Estándar	Máximo de Valores Nominales	AUC	F1-Score (Clase Minoritaria)	Sobreajuste
Configuración por Defecto	0,0001	0,0001	0,0000	20	0,864	0,785	Moderado
Mayor Probabilidad por Defecto	0,0010	0,0001	0,0000	20	0,864	0,787	Moderado
Aumento de Desviación Estándar Mínima	0,0001	0,0010	0,0000	20	0,864	0,785	Moderado

Figura 33 muestra los resultados de distintas configuraciones para el algoritmo Naive Bayes. La configuración por defecto obtiene un AUC de 0.864 y un F1-Score de 0.785, con sobreajuste moderado. Aumentar la probabilidad por defecto mejora ligeramente el F1-Score, mientras que cambiar la desviación estándar mínima no impacta el rendimiento.

Configuración por Defecto: Con una probabilidad por defecto de 0.0001 y una desviación estándar mínima de 0.0001, esta configuración logra un AUC de 0.864 y un F1-Score de 0.785, con un sobreajuste moderado.

Mayor Probabilidad por Defecto: Aumentar la probabilidad por defecto a 0.0010 mantiene el mismo AUC (0.864) y mejora ligeramente el F1-Score a 0.787, sin afectar el nivel de sobreajuste.

Aumento de Desviación Estándar Mínima: Incrementar la desviación estándar mínima a 0.0010 no afecta el AUC ni el F1-Score, manteniéndose igual que la configuración por defecto y con el mismo sobreajuste moderado.

La **configuración por defecto** ofrece un buen equilibrio con un AUC de 0.864 y F1-Score de 0.785. Aumentar la probabilidad por defecto mejora ligeramente el F1-Score sin afectar el sobreajuste, pero el ajuste de la desviación estándar mínima no muestra mejoras en el rendimiento.

6. Gradient Boosted Tree

Configuración	Profundidad del árbol	Número de Modelos	Learning Rate	AUC	F1-Score (Clase Minoritaria)	Sobreajuste
Configuración por Defecto	4	100	0,1	0,972	0,914	Moderado
Menor Profundidad y Mayor Número de Modelos	3	200	0,05	0,971	0,91	Bajo
Mayor Profundidad con Menos Modelos y Learning Rate Bajo	5	50	0,01	0,958	0,898	Moderado

Figura 34 muestra los resultados de configuraciones de Gradient Boosted Trees. La configuración por defecto (profundidad 4, 100 modelos) alcanza el mejor balance con AUC de 0.972 y F1-Score de 0.914. Otras variaciones reducen ligeramente el rendimiento o el sobreajuste.

Configuración por Defecto: Con una profundidad de 4, 100 modelos y un learning rate de 0.1, esta configuración alcanzó un AUC de 0.972 y un F1-Score de 0.914 en la clase minoritaria, con un nivel moderado de sobreajuste. Ofrece un buen balance entre precisión y generalización.

Menor Profundidad y Mayor Número de Modelos: Al reducir la profundidad a 3 y aumentar los modelos a 200, el AUC apenas se reduce a 0.971 y el F1-Score a 0.910, pero disminuye el sobreajuste, mejorando la capacidad de generalización.

Mayor Profundidad con Menos Modelos y Learning Rate Bajo: Incrementar la profundidad a 5, reducir los modelos a 50 y bajar el learning rate a 0.01 da un AUC de 0.958 y un F1-Score de 0.898, mostrando un rendimiento ligeramente menor. Sin embargo, mantiene un sobreajuste moderado.

La **configuración por defecto** proporciona el mejor equilibrio entre AUC y F1-Score con sobreajuste moderado. Reducir la profundidad y aumentar el número de modelos disminuye el sobreajuste, aunque con una ligera reducción en rendimiento.

5. Análisis de Resultados

Para este apartado, se realizará una tabla comparativa con los resultados de los algoritmos utilizados así como su interpretación y una explicación de los pros y contras que puede tener cada uno de ellos.

	TP	FP	TN	FN	Recall	Precision	TPR	TNR	F1	Accuracy
Árbol de Decisión	1161	73	1327	219	0,841	0,941	0,841	0,948	0,888	0,895
k-NN	1058	80	1320	322	0,767	0,930	0,767	0,943	0,840	0,855
SGD	1159	109	1291	221	0,840	0,914	0,840	0,922	0,875	0,881
Random Forest	1184	50	1350	196	0,858	0,959	0,858	0,964	0,906	0,912
Naive Bayes	1108	334	1066	272	0,803	0,768	0,803	0,761	0,785	0,782
Gradient Boosted Tree	1212	61	1339	168	0,878	0,952	0,878	0,956	0,914	0,918

	G-mean	AUC
Árbol de Decisión	0,890	0,933
k-NN	0,844	0,927
SGD	0,876	0,948
Random Forest	0,907	0,967
Naive Bayes	0,785	0,864
Gradient Boosted Tree	0,914	0,972

Figura 35 muestra la comparación del mejor caso para cada algoritmo. Gradient Boosted Trees destaca con el mayor AUC (0.972) y F1-Score (0.914), seguido de Random Forest con AUC de 0.967. Naive Bayes es el algoritmo con menor rendimiento, obteniendo un AUC de 0.864 y F1-Score de 0.785.

La representación de los datos anteriores, se ha realizado gráficas para facilitar la interpretación se han realizado concretamente 2 gráficos ya que si no se hace demasiado cargado todo en uno.

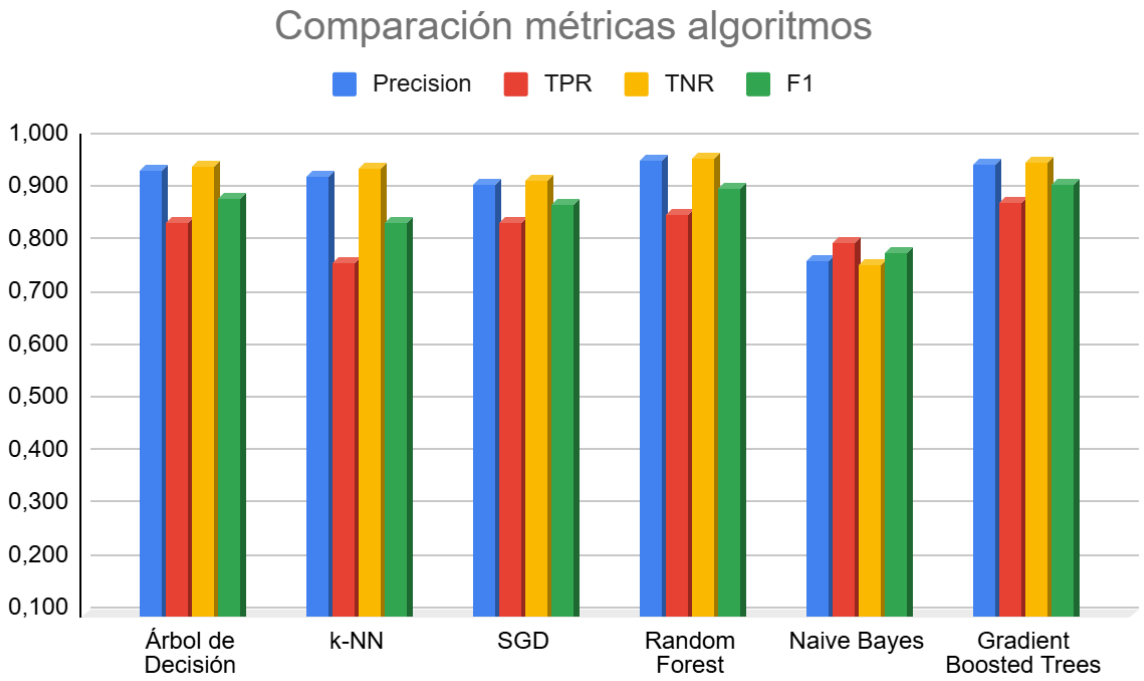


Figura 36, se muestra el contenido de la figura 35 gráficamente.

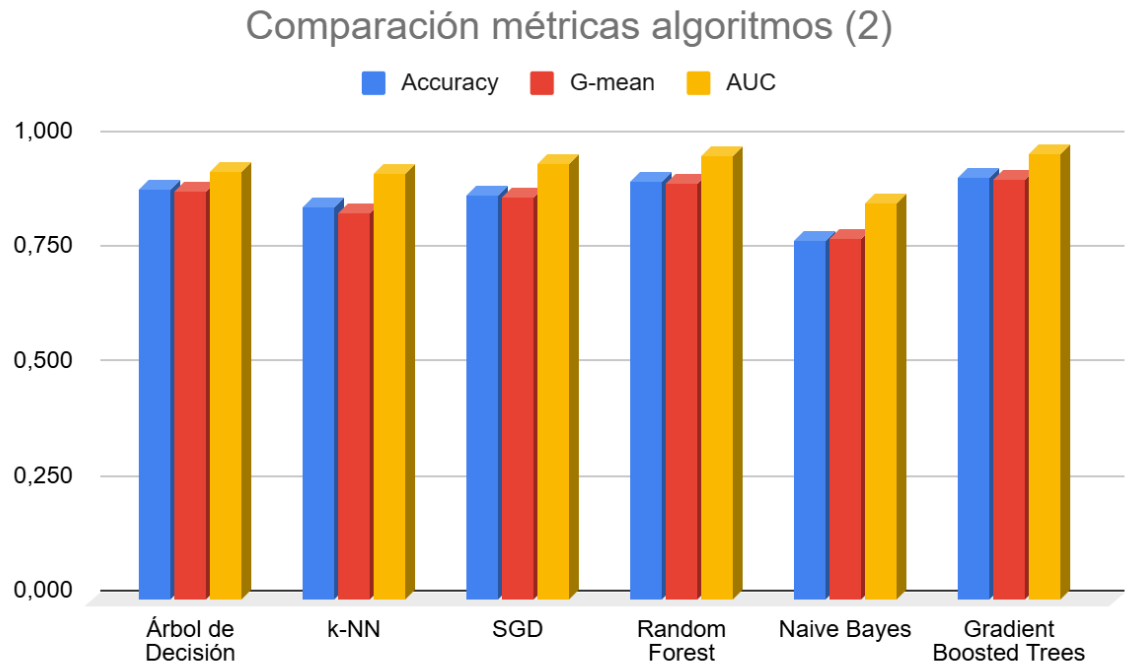


Figura 37, se muestra el contenido de la figura 35 gráficamente.

El análisis de los clasificadores se realizará de mayor a menor rendimiento según la tabla proporcionada.

1. **Gradient Boosted Trees** es el algoritmo con el mejor rendimiento, alcanzando el AUC más alto (0.972) y un F1-Score de 0.914. Su G-mean de 0.914 indica una

excelente capacidad para diferenciar correctamente entre las clases, manteniendo un equilibrio notable entre sensibilidad y especificidad. Es el algoritmo más efectivo entre los seis evaluados.

2. **Random Forest** también muestra un rendimiento sólido, con un AUC de 0.967 y un F1-Score de 0.906. Su G-mean es de 0.907, lo que indica una capacidad robusta para clasificar correctamente en ambas clases sin verse afectado significativamente por el desbalance. Es un modelo muy confiable y cercano en rendimiento a Gradient Boosted Trees.
3. **Árbol de Decisión** presenta un AUC de 0.933 y un F1-Score de 0.888, con una precisión alta (0.941) y una G-mean de 0.890. Aunque su rendimiento es inferior al de los modelos anteriores, sigue siendo efectivo, especialmente para identificar correctamente las instancias de la clase mayoritaria, con un buen equilibrio entre sensibilidad y especificidad.
4. **SGD (Stochastic Gradient Descent)** logra un AUC de 0.948 y un F1-Score de 0.875. Aunque su G-mean de 0.876 es aceptable, su sensibilidad es moderada (0.840), lo que puede limitar su capacidad para capturar correctamente las instancias de la clase minoritaria. Sin embargo, su precisión es alta (0.914), lo que le permite un rendimiento equilibrado en ciertas situaciones.
5. **k-Nearest Neighbors (k-NN)** obtiene un AUC de 0.927 y un F1-Score de 0.840. Su G-mean de 0.844 sugiere que es menos equilibrado que los modelos previos en cuanto a sensibilidad y especificidad. Aunque tiene una buena precisión, su menor sensibilidad indica dificultades en la clasificación de instancias de la clase minoritaria, afectando su robustez general.
6. **Naive Bayes** es el algoritmo con el rendimiento más bajo, con un AUC de 0.864 y un F1-Score de 0.785. Su G-mean de 0.785 refleja un equilibrio moderado entre sensibilidad y especificidad, pero no alcanza el nivel de precisión y robustez de los otros algoritmos. Aunque ofrece un desempeño aceptable, es el menos eficaz para clasificar correctamente en ambas clases en comparación con los demás.

Matriz de pros y contras de cada algoritmo

Algoritmo	Pros	Contras
Gradiente Boosted Tree	- Alto rendimiento en AUC y F1-Score.	- Alto costo computacional.
	- Excelente capacidad de generalización.	- Sensible al ajuste de hiperparámetros, lo que requiere cuidado.
Random Forest	- Robusto y generaliza bien en distintas clases.	- Entrenamiento lento debido a la cantidad de árboles.
	- Alto rendimiento en todas las métricas.	- Requiere gran cantidad de recursos.
Árbol de Decisión	- Alta interpretabilidad y buena especificidad.	- Propenso al sobreajuste, especialmente en datos ruidosos.
	- Rápido y fácil de visualizar.	- Menor capacidad de generalización.
SGD (Stochastic Gradient Descent)	- Alta eficiencia en tiempo de entrenamiento, especialmente con datos grandes.	- Bajo rendimiento en sensibilidad y G-mean. - Menos efectivo en detectar clases minoritarias.
Naive	- Muy rápido y adecuado para datos	- Baja precisión en problemas con

Bayes	independientes. - Simple de implementar y utilizar.	correlación entre atributos. - Limitado en rendimiento general.
k-NN (k-Nearest Neighbors)	- Fácil de entender y útil en problemas donde la proximidad es relevante.	- Alto costo computacional en predicción con grandes datos. - Bajo G-mean y propenso al sobreajuste sin ponderación de distancia.

Hipótesis y Explicación de los Resultados

1. Gradient Boosted Trees y su alto rendimiento: Gradient Boosted Trees combina múltiples árboles con ajustes progresivos en cada iteración, lo que permite capturar patrones complejos y reducir errores. Este enfoque secuencial ayuda a mejorar tanto el AUC como el F1-Score, logrando el mejor rendimiento entre los algoritmos. Sin embargo, requiere ajustes cuidadosos de los hiperparámetros y es computacionalmente costoso, lo que limita su aplicabilidad en contextos de recursos limitados.

2. Random Forest y su robustez: La combinación de múltiples árboles en Random Forest, mediante el proceso de votación, permite capturar patrones diversos y reducir el sobreajuste. Esto le otorga un alto AUC y G-mean, lo que lo hace ideal para problemas de clasificación en los que se requiere robustez y generalización. Su principal desventaja es el elevado costo computacional, lo cual puede limitar su uso en aplicaciones en tiempo real.

3. Árbol de Decisión y su rendimiento moderado: Aunque el Árbol de Decisión es rápido y fácil de interpretar, su estructura sencilla lo hace más propenso al sobreajuste, especialmente en datos ruidosos. Aunque tiene buena precisión y especificidad, su menor capacidad de generalización en comparación con Random Forest y Gradient Boosted Trees lo convierte en una opción menos robusta para problemas complejos.

4. SGD (Stochastic Gradient Descent) como opción eficiente: SGD es muy eficiente para problemas con grandes volúmenes de datos, ya que ajusta sus parámetros gradualmente. Su rendimiento es aceptable en precisión y AUC, pero tiene dificultades para capturar la clase minoritaria, lo que se refleja en su baja sensibilidad y G-mean. Esto lo hace menos adecuado para problemas de clasificación donde ambas clases tienen igual importancia.

5. Naive Bayes como alternativa rápida y simple: Naive Bayes es extremadamente eficiente y es adecuado cuando se asume independencia entre atributos. Aunque tiene un rendimiento moderado en AUC y F1-Score, su limitación principal es su incapacidad para capturar relaciones complejas entre las variables, lo cual reduce su efectividad en problemas con correlación entre atributos y clases desbalanceadas.

6. Desempeño limitado de k-Nearest Neighbors (k-NN): Sin una ponderación adecuada de distancia, k-NN tiene dificultades para diferenciar clases en problemas de alta dimensionalidad, como se refleja en su bajo G-mean y AUC. Aunque es útil para problemas en los que la proximidad local es relevante, su sensibilidad al balance de clases y su alto costo computacional en predicción lo hacen menos efectivo para este problema en particular.

6. Interpretación de los Datos

Para esta parte se ha construido un metanodo, empleando un árbol de decisión, un Tree Ensemble Learner y un Random Forest para poder así ver la distribución de las variables en la creación de estos algoritmos.

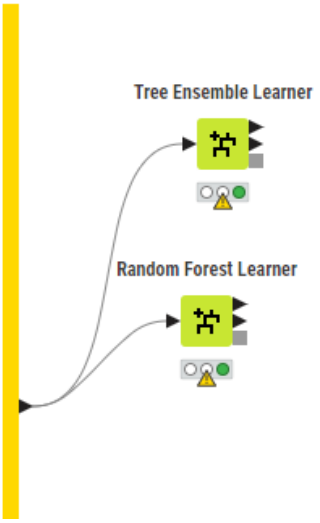


Figura 37 muestra los nodos Tree Ensemble Learner y Random Forest Learner, configurados para algoritmos de conjuntos de árboles. Random Forest incluye bootstrap y votación entre árboles.
Este flujo (WHAT IF) permite realizar análisis de escenarios y evaluar cómo cambios en las variables de entrada pueden afectar las predicciones de cada modelo.
Se han alterado valores clave de los atributos para ver cómo cambian las predicciones. Esto permite visualizar el impacto de atributos individuales y cómo influyen en la clasificación de cada caso.

Tree Ensemble Learner

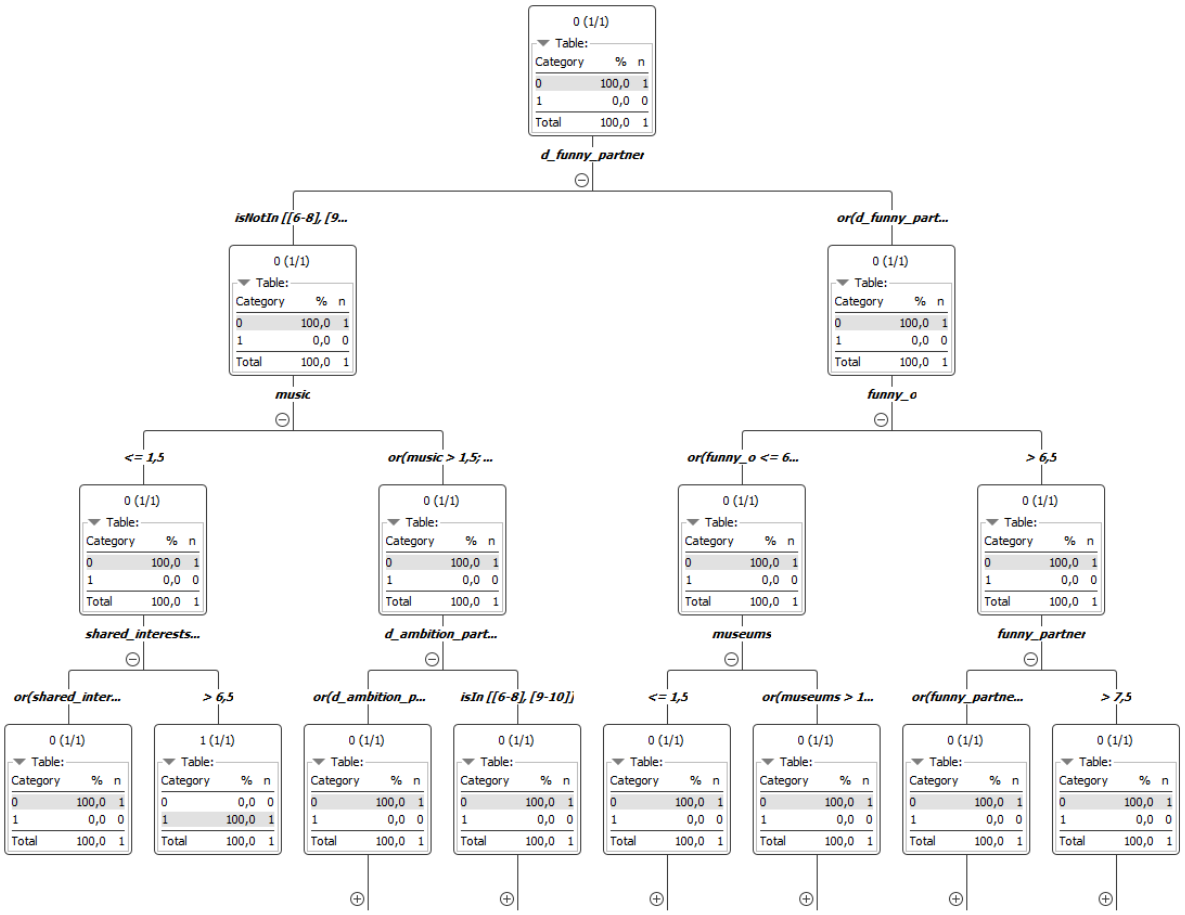


Figura 38 muestra un árbol de decisión generado, donde las variables y divisiones principales incluyen atributos como funny_partner, music, y shared_interests. Cada nodo de decisión refleja cómo estos atributos contribuyen a la clasificación en función de diferentes rangos y categorías.

El árbol de decisión muestra que las variables clave para la decisión de tener una segunda cita son:

- **d_funny_partner**: Es la variable principal de decisión, donde una baja diferencia en la percepción de ser divertido entre ambos participantes favorece la probabilidad de una segunda cita.
- **funny_o**: La valoración que el participante le da a la pareja en términos de humor es importante. Una calificación superior a 6.5 indica una mayor probabilidad de coincidencia para una segunda cita.
- **music**: La afinidad en gustos musicales también juega un papel en la decisión, siendo más probable una segunda cita cuando ambos tienen un interés común en música.
- **shared_interests**: Tener intereses compartidos con una calificación alta aumenta la probabilidad de éxito para una segunda cita, reflejando la importancia de las afinidades en la compatibilidad.

Random Forest

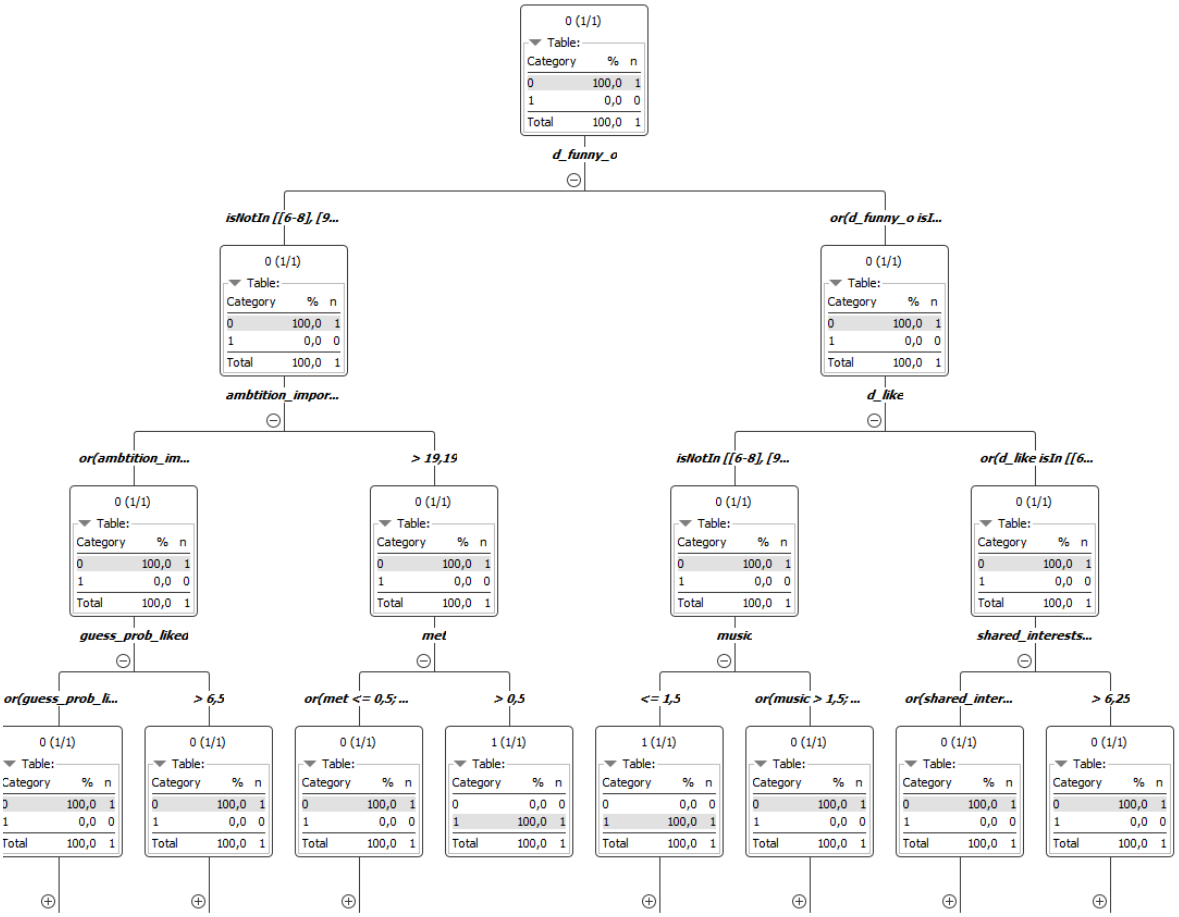


Figura 39, se refleja la creación del árbol de decisión del nodo WHAT IF, así como el orden y los valores de la variables en su creación para el uso del algoritmo Random Forest.

El árbol de decisión muestra que las variables clave para la decisión de una segunda cita son:

- **d_funny_o**: La diferencia en la percepción de ser divertido juega un papel importante; una menor diferencia favorece la posibilidad de una segunda cita.
- **d_like**: La afinidad o agrado inicial entre los participantes es relevante, especialmente cuando ambos tienen altos niveles de atracción.
- **ambition_importance**: La importancia de la ambición influye en la decisión, siendo más probable una segunda cita cuando ambos participantes valoran la ambición.
- **guess_prob_liked**: La probabilidad estimada de que el otro participante haya mostrado agrado afecta la decisión. Una alta percepción de agrado mutuo favorece la posibilidad de otra cita.
- **music y shared_interests**: Intereses en común, como la música y otros intereses compartidos, también contribuyen positivamente a la decisión de tener una segunda cita

7. Contenido Adicional

Para el preprocesamiento, he realizado diferentes correlaciones en cuanto a las variables, y me ha resultado imposible obtener mejores resultados con otras variables. Tome la matriz de correlación, de la variable objetivo(match) y fui analizando aquellas variables que tenían un valor superior a 0,10 ya que hay muchas variables con relaciones del tipo 0,001 o similares. Estas variables fueron **Attractive_o**, **Sincere_o**, I**ntelligence_o**:, **Funny_o**, **Ambitious_o**, **Shared_interests_o**. y las que terminan en partner, y realice los algoritmos para estas variables, pero obtenía (-0,30) en el AUC de las diferentes curvas ROC aproximadamente. Por tanto he realizado la práctica con todas las variables excepto algunas que he filtrado en la correlación.

8. Bibliografía

KNIME AG. (n.d.). *KNIME Documentation*. Recuperado el 5 de noviembre de 2024, de https://docs.knime.com/

Casillas, J. (n.d.). *Introducción a KNIME*. Universidad de Granada. Recuperado el 5 de noviembre de 2024, de https://ccia.ugr.es/~casillas/knime.html

Kaggle. (s.f.). *Loan Approval Prediction* [Conjunto de Datos de Préstamos]. Kaggle. Recuperado de https://www.kaggle.com/datasets/itshappy/ps4e9-original-data-loan-approval-prediction/data

OpenAI. (2024). *ChatGPT (v4)* [KNIME]. Recuperado de https://chat.openai.com

Problema 3: Predicción del tipo de enfermedad eritemato-escamosa

1. Introducción

El objetivo de este proyecto es desarrollar modelos de aprendizaje automático para predecir el tipo de enfermedad eritemato-escamosa en pacientes. Estas enfermedades, como la psoriasis, dermatitis seborreica, liquen plano, pitiriasis rosada, dermatitis crónica y pitiriasis rubra pilaris, comparten características clínicas similares, como el eritema y la descamación, lo que hace difícil su diferenciación sin una biopsia. La predicción del tipo de enfermedad basado en características clínicas e histopatológicas tiene como objetivo asistir en el diagnóstico y reducir la necesidad de procedimientos invasivos.

El conjunto de datos utilizado en este estudio incluye 34 características, divididas en:

- **12 características clínicas** que describen observaciones visibles y síntomas reportados en el paciente.
- **22 características histopatológicas** evaluadas mediante un análisis microscópico de muestras de piel.

Variables Clínicas:
- Erythema: Grado de enrojecimiento en la piel.
- Scaling: Nivel de descamación o desprendimiento de la piel.
- Itching: Intensidad del picor experimentado por el paciente.
- Skin Thickening: Grado de engrosamiento o endurecimiento de la piel.
- Lesion Location: Ubicación de las áreas afectadas en el cuerpo (tronco, extremidades, etc.).
- Symptom Duration: Duración de los síntomas experimentados por el paciente.
- Family History: Indica si hay antecedentes familiares de enfermedades eritemato-escamosas (sí o no).
- Patient Age: Edad en años del paciente en el momento de la consulta.
- Gender: Sexo del paciente (masculino o femenino).
- Reaction to Previous Treatments: Respuesta a tratamientos previos (buena, regular, mala).
- Pain Intensity: Nivel de dolor experimentado en las lesiones (leve, moderado, severo).
- Skin Sensitivity: Grado de sensibilidad en las áreas afectadas.

Histopathological Variables:

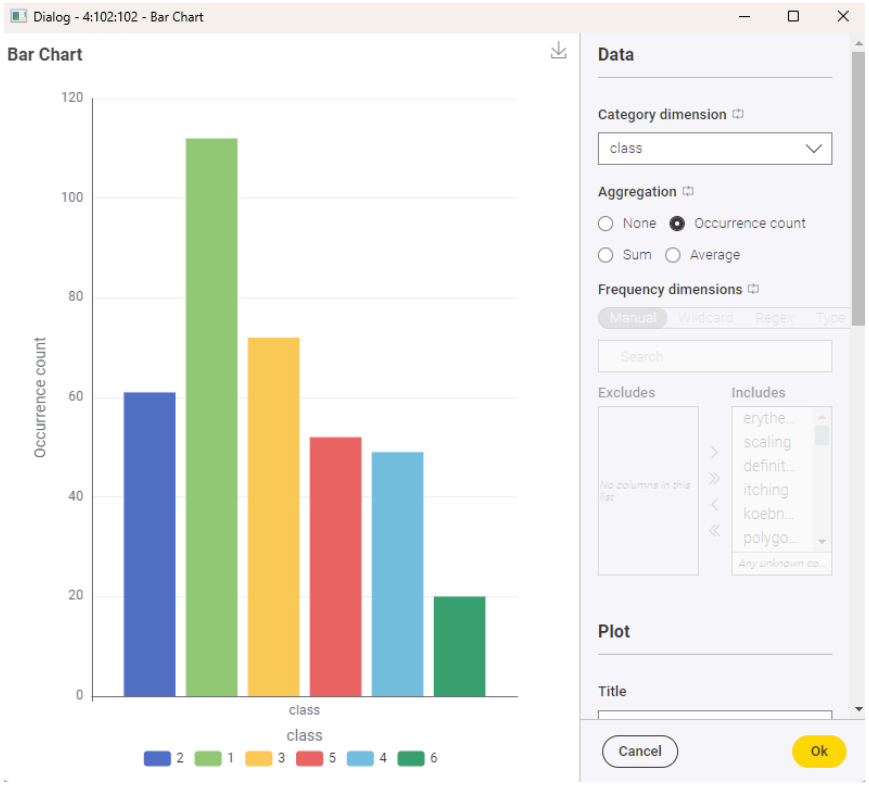
- Acanthosis: Engrosamiento de la epidermis.
- Hyperkeratosis: Incremento en la capa de queratina de la epidermis.
- Parakeratosis: Presencia de núcleos en el estrato córneo de la epidermis.
- Lymphocyte Infiltration: Cantidad de linfocitos presentes en la dermis.
- Spongiosis: Hinchazón intercelular en la epidermis.
- Exocytosis: Migración de células inflamatorias hacia la epidermis.
- Keratin Layers: Número de capas queratinizadas en la epidermis.
- Microabscess Formation: Presencia de microabscesos en la epidermis o dermis.
- Blood Vessel Dilation: Grado de dilatación de los vasos sanguíneos en la dermis.
- Epidermal Atrophy: Adelgazamiento de la epidermis.
- Dermal Capillaries: Densidad de capilares en la dermis.

- **Mitosis**: Nivel de actividad mitótica en las células de la piel.
- **Neutrophil Infiltration**: Cantidad de neutrófilos en la dermis.
- **Eosinophil Infiltration**: Cantidad de eosinófilos en la dermis.
- **Hypergranulosis**: Aumento en la capa granular de la epidermis.
- **Basal Cell Damage**: Grado de daño en las células de la capa basal de la epidermis.
- **Lichenification**: Engrosamiento crónico de la piel con marcas acentuadas.
- **Papillomatosis**: Crecimiento de papilas en la epidermis.
- **Dense Dermal Infiltrate**: Densidad del infiltrado en la dermis.
- **Thick Scale Formation**: Severidad en la formación de escamas en la superficie de la piel.
- **Keratin Retention**: Acumulación de queratina en la superficie de la piel.
- **Cell Necrosis**: Indica si hay muerte celular en las capas de la piel.

Particularidades del Conjunto de Datos

Valores Faltantes: en este caso, no existen valores faltantes de los datos. Por lo que no se realizará ningún tratamiento para los valores perdidos, ya que están todos completos.

Desbalanceo de Clases: Dado que la variable objetivo en este conjunto de datos es el tipo de enfermedad eritemato-escamosa, es importante verificar la proporción de las clases (psoriasis, dermatitis seborreica, liquen plano, pitiriasis rosada, dermatitis crónica y pitiriasis rubra pilaris) para identificar si existe un desbalance significativo entre ellas. Para ello, podemos utilizar un gráfico de barras (por ejemplo, mediante el nodo *Bar Chart*) para visualizar la distribución de cada clase en el conjunto de datos y así entender mejor el equilibrio de las muestras entre los diferentes tipos de enfermedad. Esta visualización es crucial para determinar si es necesario aplicar técnicas de balanceo en los modelos de clasificación para mejorar su rendimiento en las clases menos representadas.



La **Figura 1** muestra la distribución de las clases de enfermedades eritemato-escamosas en el conjunto de datos. Se observa un desbalance, donde la clase 1 es la más frecuente y la

clase 6 la menos representada, lo que implica un desafío para el modelo en la clasificación de clases minoritarias.

Para esto, pensé y apliqué tal y como se puede ver en Knime un submuestreo, minorar en cierta medida la clase mayoritaria, debido a que no hay muchos datos, pienso que puede ser una mala práctica. Por lo que he dejado los valores originales sin tocar, ya que puede tener sentido que una enfermedad de una clase en concreta se de más en la clase real.

Para abordar el problema de la predicción del tipo de enfermedad eritemato-escamosa en este conjunto de datos, se seleccionaron los siguientes algoritmos de aprendizaje automático. Cada uno ofrece distintas ventajas para explorar diversos enfoques de clasificación y mejorar la precisión del diagnóstico:

- **Árbol de Decisión (Decision Tree)**: Este algoritmo permite desglosar los factores clave que influyen en el diagnóstico de cada enfermedad. Su estructura jerárquica facilita la interpretación de los resultados, identificando los síntomas clínicos e histopatológicos más determinantes en cada caso.
- **K-Vecinos más Cercanos (K-NN)**: El algoritmo K-NN clasifica cada caso en función de los casos similares en el conjunto de datos, lo cual es útil en problemas donde la proximidad entre las observaciones tiene relevancia. Este enfoque es especialmente adecuado para identificar patrones de similitud en las características de los pacientes con diagnósticos similares.
- **Random Forest**: Random Forest combina múltiples árboles de decisión, lo que permite mejorar la precisión y reducir el sobreajuste. Es un método robusto que captura relaciones complejas en los datos, siendo eficaz en conjuntos de datos que incluyen tanto características clínicas como histopatológicas.
- **Regresión Logística**: Este modelo de regresión probabilística permite evaluar la probabilidad de cada clase (tipo de enfermedad), siendo útil para comprender la relación entre las características y el diagnóstico. Además, ofrece una interpretación clara y es eficaz en problemas de clasificación binaria y multiclase con datos mixtos.
- **Gradient Boosted Trees (GBT)**: GBT utiliza un enfoque de ensamble que combina varios árboles de decisión para optimizar el rendimiento. Este algoritmo es potente para capturar patrones complejos en los datos y es especialmente útil cuando se busca maximizar la precisión de las predicciones, al manejar bien relaciones no lineales entre las características.

Cada uno de estos algoritmos fue seleccionado para ofrecer una perspectiva diversa en la clasificación de enfermedades eritemato-escamosas, permitiendo evaluar técnicas distintas y obtener un análisis completo sobre los síntomas clave para el diagnóstico diferencial.

2. Procesado de Datos

Para asegurar la calidad y precisión de los modelos predictivos, fue necesario realizar un preprocesamiento exhaustivo de los datos. Este proceso incluyó el ajuste de desbalanceo de clases. A continuación, se detallan los pasos de procesamiento realizados:

Tratamiento del Desbalanceo de Clases
Para tratar el desbalanceo en la variable objetivo de tipo de enfermedad (representada en la columna "class"), se ha incorporado un nodo X-Partitioner dentro del metanodo que gestiona cada algoritmo, con el fin de evaluar el rendimiento de los modelos de manera más confiable. Las estrategias aplicadas incluyen:

- **Validación cruzada con 10 pliegues**: Esta técnica proporciona una evaluación robusta del rendimiento de los modelos, asegurando que cada instancia sea evaluada en múltiples escenarios.
- **Muestreo estratificado**: Se emplea para garantizar que cada pliegue contenga una proporción representativa de cada tipo de enfermedad, manteniendo la distribución de clases en cada partición.
- **Especificación de la columna de clase**: La columna "class" se utiliza como referencia para conservar la proporción de cada clase en las particiones, ya que representa la variable objetivo.
- **Semilla aleatoria**: Para asegurar la reproducibilidad de los resultados, se ha utilizado una semilla aleatoria constante en todas las ejecuciones.

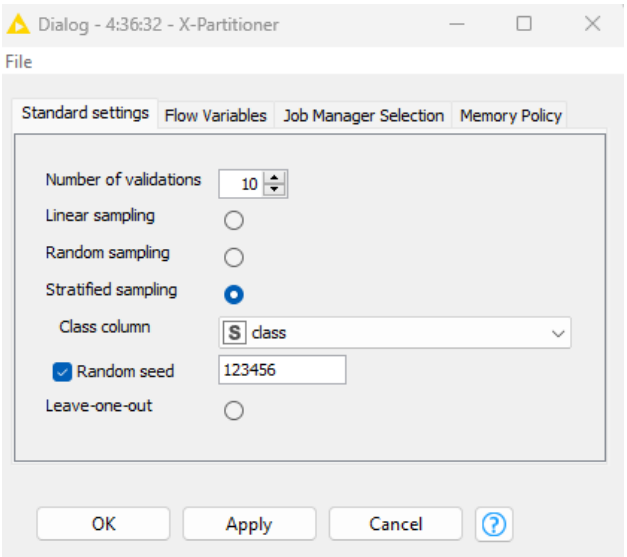


Figura 2 muestra la configuración del nodo X-Partitioner con validación cruzada de 10 pliegues y muestreo estratificado para mantener la proporción de la columna class. La semilla 123456 asegura la reproducibilidad.

Uso de Normalización

La normalización de datos es una etapa crucial en el preprocesamiento que ajusta las variables numéricas a una escala común, generalmente entre 0 y 1 o en función de una desviación estándar específica. Este proceso mejora el rendimiento de algoritmos sensibles a la escala. En nuestro caso, se ha aplicado a **SGD**, **Random Forest** y **Naive Bayes**, que

son sensibles a la magnitud de las variables, asegurando que todas las características contribuyan de manera equitativa al modelo.

Consideré otros preprocesamientos, como la conversión de variables categóricas en valores numéricos. Sin embargo, estos ajustes no mostraron buenos resultados en la clasificación de enfermedades eritemato-escamosas, por lo que decidí no incluirlos en el análisis.

Por último, utilicé el nodo **Column Filter** para excluir variables específicas según el algoritmo, ya que algunas características no aportan información relevante al diagnóstico diferencial de las enfermedades. Además, algunos nodos generaban advertencias sobre la variable de clase, lo cual indicaba que ciertos atributos podían interferir en el rendimiento. La exclusión de estas variables optimizó los modelos, mejorando la precisión de la predicción. El flujo de trabajo se estructuró de la siguiente manera:

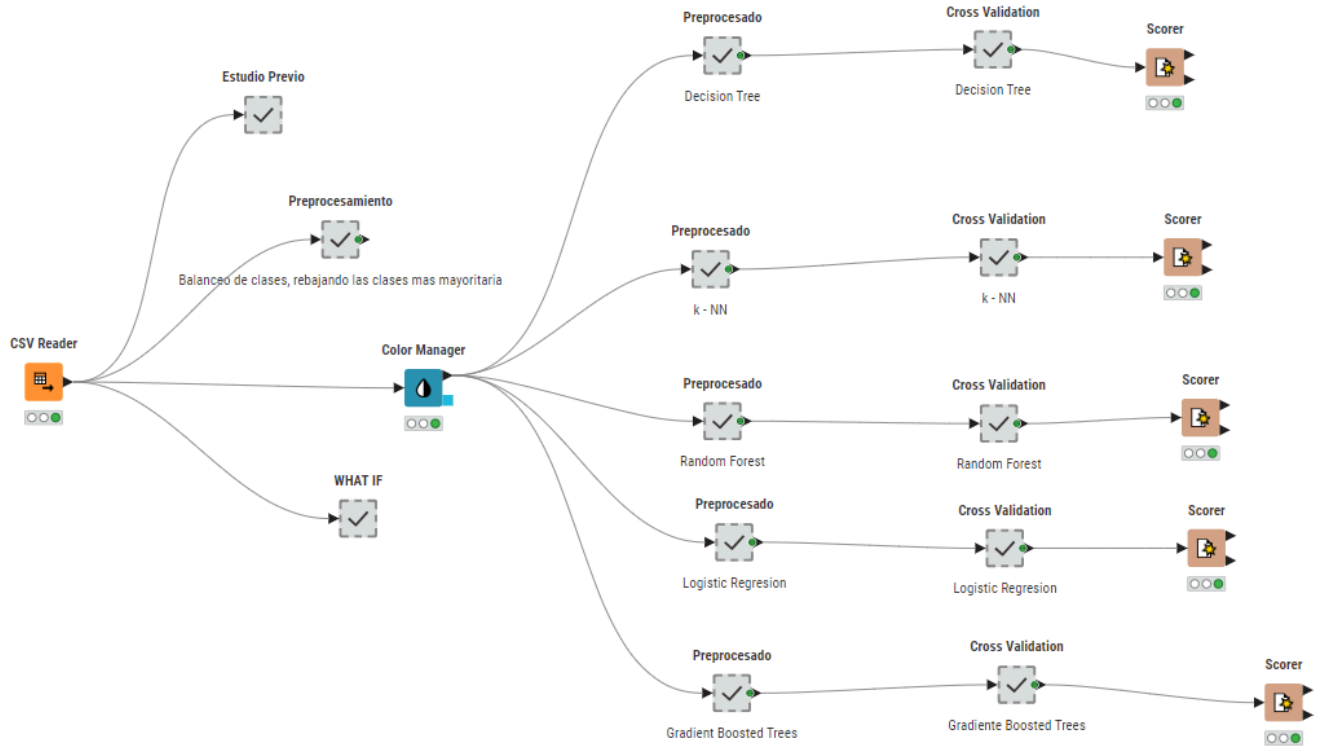


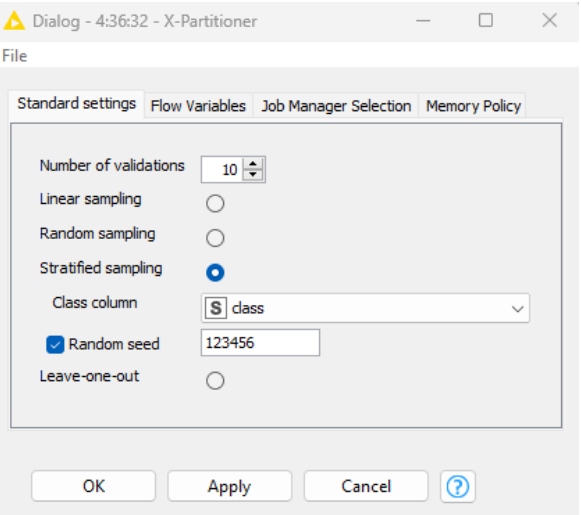
Figura 3,encontramos Flujo de trabajo en KNIME para la evaluación de modelos de clasificación en enfermedades eritemato-escamosas. En dicha figura se muestra el proceso desde la lectura de datos hasta la evaluación de los modelos. Los datos pasan por preprocesamiento, incluyendo balanceo de clases, y se aplican distintos algoritmos de clasificación: Decision Tree, k-Nearest Neighbors (k-NN), Random Forest, Logistic Regression, y Gradient Boosted Trees. Cada modelo se somete a validación cruzada y su rendimiento se evalúa mediante métricas generadas por el nodo Scorer.

Para evaluar el rendimiento de cada algoritmo de manera consistente en este caso al tener un problema de multiclase, se emplearon solo el nodo clave en KNIME: *Scorer*. El nodo Scorer permite recopilar en una tabla las métricas principales de cada modelo, como precisión, recall, F1-score y exactitud, lo que facilita la comparación cuantitativa entre los distintos algoritmos. Esta estructura de recogida de resultados asegura una evaluación

completa y uniforme, permitiendo identificar de manera objetiva el rendimiento y la capacidad predictiva de cada modelo.

3. Resultados Obtenidos

El metanodo de Cross Validation para los algoritmos es similar en todos los casos. Toda la experimentación de los algoritmos se realiza con validación cruzada de 10 particiones, utilizando los nodos “X-Partitioner” y “X-Aggregator”. Se configuraron 10 validaciones y se seleccionó la opción de "Stratified Sampling" para garantizar que cada subconjunto de entrenamiento y prueba mantenga la misma proporción de clases que el conjunto de datos original. Se estableció una semilla aleatoria de 123456 y se eligió la columna "class" como referencia, siendo esta la variable objetivo. Esto puede observarse en las siguientes imágenes.



La **figura 4** muestra la configuración del nodo "X-Partitioner" en KNIME para aplicar validación cruzada estratificada de 10 particiones. Se seleccionó "Stratified Sampling" para mantener la proporción de clases de la variable objetivo "class" en cada partición. Además, se estableció una semilla aleatoria (123456) para asegurar la reproducibilidad de los resultados.

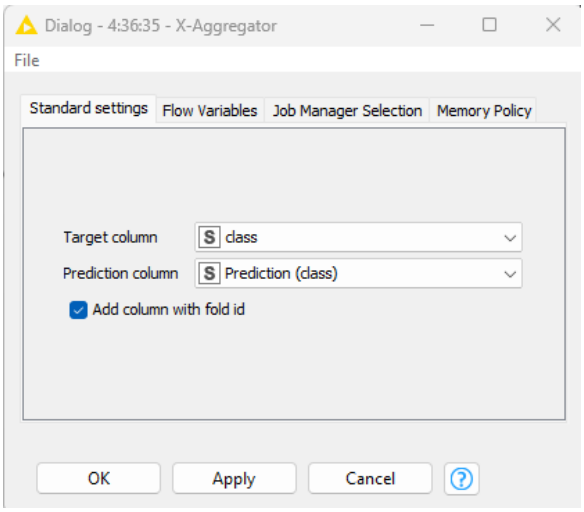


Figura 5: Configuración del nodo "X-Aggregator" en KNIME, con "class" como columna objetivo y "Prediction (class)" como columna de predicción, agregando el identificador de cada pliegue para análisis detallado.

En esta sección, se presentan los resultados de cada uno de los algoritmos empleados en la predicción del estado de aprobación de préstamos. Para cada algoritmo, se muestra el flujo de trabajo correspondiente en KNIME y se analizan las métricas de evaluación obtenidas, mediante el metanodo Scorer.
En segundo lugar, analizamos los flujos de trabajo de cada algoritmo:

1. Decision Tree

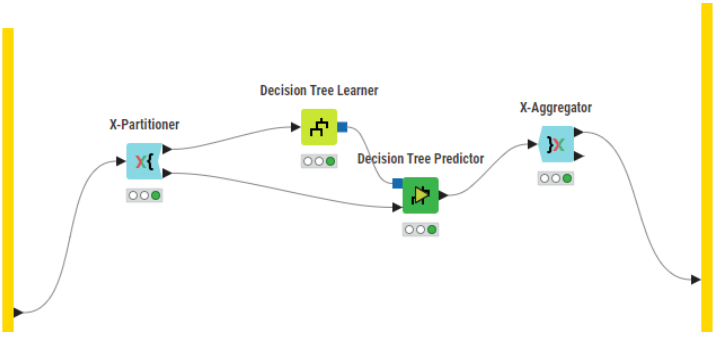


Figura 6: Flujo de trabajo en KNIME para el modelo de Decision Tree, donde se utiliza validación cruzada con "X-Partitioner" y "X-Aggregator" para evaluar el rendimiento del modelo.

Clase	TP	FP	TN	FN	Recall	Precisión	TPR	TNR	F1	Accuracy
2	56	6	299	5	0,918	0,903	0,918	0,980	0,911	
1	108	2	252	4	0,964	0,982	0,964	0,992	0,973	
3	70	2	292	2	0,97	0,97	0,97	0,99	0,97	
5	51	1	313	1	0,981	0,981	0,981	0,997	0,981	
4	45	5	312	4	0,918	0,900	0,918	0,984	0,909	
6	18	2	344	2	0,900	0,900	0,900	0,994	0,900	
Overall										0,951

Figura 7 podemos ver "Matriz de métricas de evaluación por clase del modelo de Árbol de Decisión." En dicha figura se muestran los valores de True Positives (TP), False Positives (FP), True Negatives (TN), False Negatives (FN), y métricas de evaluación como Recall, Precisión, TPR, TNR, F1 y Accuracy para cada una de las clases de la variable objetivo, así como el promedio general de Accuracy. Estos valores permiten analizar el rendimiento del modelo para cada clase y su eficacia global en la clasificación de enfermedades.

Clase	2	1	3	5	4	6
2	56	1	1	0	1	2
1	1	108	0	0	3	0

3	0	0	70	1	1	0
5	0	1	0	51	0	0
4	3	0	1	0	45	0
6	2	0	0	0	0	18

Figura 8 podemos ver "Matriz de confusión del modelo de Árbol de Decisión." En dicha figura se representan las predicciones del modelo para cada clase de enfermedad eritemato-escamosa, con las clases verdaderas en las filas y las predicciones en las columnas. Los valores diagonales muestran las predicciones correctas para cada clase, mientras que los valores fuera de la diagonal indican errores de clasificación. Esta matriz permite evaluar la precisión del modelo en la clasificación específica de cada clase.

Para el modelo de Árbol de Decisión aplicado al diagnóstico diferencial de enfermedades eritemato-escamosas, los resultados muestran un alto rendimiento en términos de precisión global (Accuracy = 0.951). La matriz de confusión indica que el modelo es particularmente efectivo en clasificar correctamente la mayoría de las clases, especialmente aquellas con una mayor cantidad de datos de entrenamiento.

Cada clase presenta una alta precisión y sensibilidad (recall), lo que sugiere que el Árbol de Decisión es capaz de diferenciar con éxito entre las diferentes enfermedades en el conjunto de datos. Sin embargo, el rendimiento disminuye ligeramente en las clases menos representadas, lo cual podría ser indicativo de un ligero desbalance en los datos. En general, este modelo proporciona una buena base para el diagnóstico inicial de estas enfermedades, aunque podría beneficiarse de ajustes adicionales o de un análisis en conjunto con otros algoritmos para mejorar la clasificación en las clases menos frecuentes.

2. k-NN

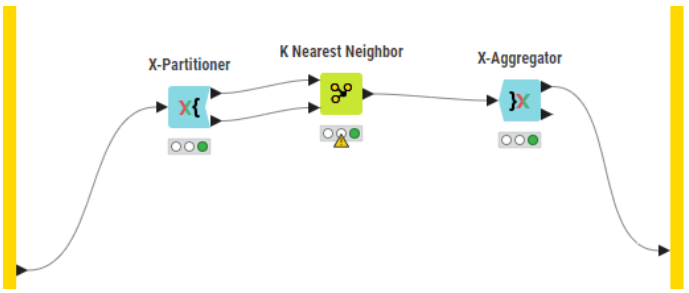


Figura 9 "Estructura del modelo k-NN en KNIME" se observa el flujo de trabajo del algoritmo de K-Nearest Neighbors (k-NN). Este flujo incluye los nodos X-Partitioner y X-Aggregator para realizar validación cruzada, asegurando una evaluación confiable del rendimiento del modelo.

Clase	TP	FP	TN	FN	Recall	Precisión	TPR	TNR	F1	Accuracy
2	52	2	303	9	0,852	0,963	0,852	0,993	0,904	
1	111	0	254	1	0,991	1,000	0,991	1,000	0,996	
3	72	0	294	0	1,00	1,00	1,00	1,00	1,00	

5	52	0	314	0	1,000	1,000	1,000	1,000	1,000	
4	48	10	307	1	0,980	0,828	0,980	0,968	0,897	
6	19	0	346	1	0,950	1,000	0,950	1,000	0,974	
Overall										0,967

Figura 10, se muestra una tabla con los resultados de rendimiento del algoritmo K-Nearest Neighbors (k-NN) en términos de métricas de clasificación para cada clase. Se incluyen métricas como True Positives (TP), False Positives (FP), True Negatives (TN), False Negatives (FN), Recall, Precisión, TPR (True Positive Rate), TNR (True Negative Rate), F1 y Accuracy. Cada fila representa una clase específica, y se observa que el rendimiento general del modelo alcanza un accuracy promedio de 0,967.

Clase	2	1	3	5	4	6
2	52	0	0	0	9	0
1	1	111	0	0	0	0
3	0	0	72	0	0	0
5	0	0	0	52	0	0
4	1	0	0	0	48	0
6	0	0	0	0	1	19

Figura 11, se presenta una matriz de confusión que muestra el rendimiento del algoritmo K-Nearest Neighbors (k-NN) en la clasificación de cada clase. Las filas representan las clases reales y las columnas representan las clases predichas. Los valores en la diagonal indican las predicciones correctas para cada clase, mientras que los valores fuera de la diagonal representan los errores de clasificación. Esta matriz permite visualizar la precisión del modelo en la identificación de cada clase, destacando las verdaderas positivas en la diagonal principal.

El algoritmo K-Nearest Neighbors (k-NN) ha mostrado un rendimiento sólido en la clasificación de las enfermedades eritemato-escamosas, logrando una alta precisión en la mayoría de las clases. Los valores en la matriz de confusión y los valores de F1-Score reflejan que k-NN es especialmente efectivo en identificar clases específicas con alta exactitud, como se observa en la diagonal de la matriz. Sin embargo, es notable que el rendimiento varía entre clases, con algunas clases teniendo más errores de clasificación que otras.

En general, el k-NN demuestra ser un clasificador competente en este contexto, aunque podría beneficiarse de ajustes adicionales, como la selección de un valor óptimo de vecinos (k) o el uso de técnicas de ponderación de distancia, para mejorar la precisión en las clases más desafiantes. Además, dado su alto rendimiento en la identificación de patrones locales, k-NN es particularmente útil para problemas donde las instancias similares están agrupadas en vecindades específicas.

3. Random Forest

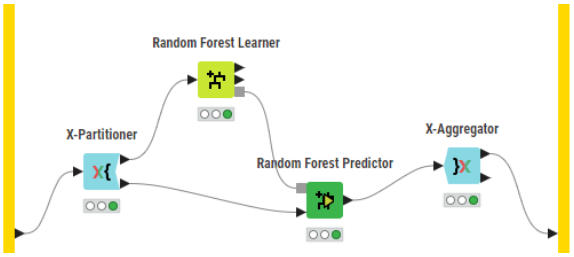


Figura 12 podemos ver el "Flujo de trabajo de Random Forest con Validación Cruzada en KNIME". En dicha figura se muestra el proceso completo, comenzando con el nodo **X-Partitioner**, que divide el conjunto de datos en subconjuntos de entrenamiento y prueba utilizando una validación cruzada de 10 particiones. A continuación, el nodo **Random Forest Learner** entrena el modelo de Random Forest en cada subconjunto de entrenamiento, y el nodo **Random Forest Predictor** realiza las predicciones sobre los datos de prueba. Finalmente, el nodo **X-Aggregator** consolida los resultados de cada partición, proporcionando una evaluación general del modelo.

Clase	TP	FP	TN	FN	Recall	Precisión	TPR	TNR	F1	Accuracy
2	58	6	299	3	0,951	0,906	0,951	0,980	0,928	
1	112	2	252	0	1,000	0,982	1,000	0,992	0,991	
3	72	0	294	0	1,00	1,00	1,00	1,00	1,00	
5	52	0	314	0	1,000	1,000	1,000	1,000	1,000	
4	43	2	315	6	0,878	0,956	0,878	0,994	0,915	
6	19	0	346	1	0,950	1,000	0,950	1,000	0,974	
Overall										0,973

Figura 13 podemos ver "Matriz de métricas de rendimiento para el modelo Random Forest en cada clase". En dicha figura se muestran los valores de las métricas de evaluación, incluyendo TP (True Positives), FP (False Positives), TN (True Negatives), FN (False Negatives), Recall, Precisión, TPR (Tasa de Verdaderos Positivos), TNR (Tasa de Verdaderos Negativos), F1-Score y Accuracy para cada clase del modelo. El rendimiento general del modelo alcanza una Accuracy de 0.973, lo que indica una alta capacidad de clasificación precisa en todas las clases.

Clase	2	1	3	5	4	6
1	58	1	0	0	2	0
2	0	112	0	0	0	0
3	0	0	72	0	0	0
4	0	0	0	52	0	0
5	6	0	0	0	43	0
6	0	1	0	0	0	19

Figura 14 podemos ver "Matriz de confusión para el modelo Random Forest". En dicha figura se muestra el desempeño del modelo en términos de clasificaciones correctas e incorrectas para cada clase. Los valores en la diagonal representan las predicciones correctas para cada clase, mientras que los valores fuera de la diagonal indican errores de clasificación. Esta matriz permite observar el nivel de precisión del modelo para cada clase individual, destacando su efectividad al clasificar la mayoría de las instancias correctamente.

El modelo Random Forest muestra un rendimiento notable en la clasificación de enfermedades eritemato-escamosas, con un AUC global alto de 0.973 y un F1-score que varía entre clases, alcanzando el 1.0 en varias de ellas, lo cual indica un desempeño excelente en precisión y recall para la mayoría de las categorías. Las clases menos representadas también presentan un alto nivel de precisión, lo que sugiere que el modelo es robusto y logra una adecuada generalización sin un sesgo significativo hacia ninguna clase en particular. La matriz de confusión confirma esta capacidad, mostrando muy pocos errores de clasificación y altas tasas de clasificación correcta en todas las clases. En general, el Random Forest se establece como un modelo confiable y efectivo para este conjunto de datos, capturando con precisión las características distintivas de cada clase y minimizando errores.

4. Logistic Regression

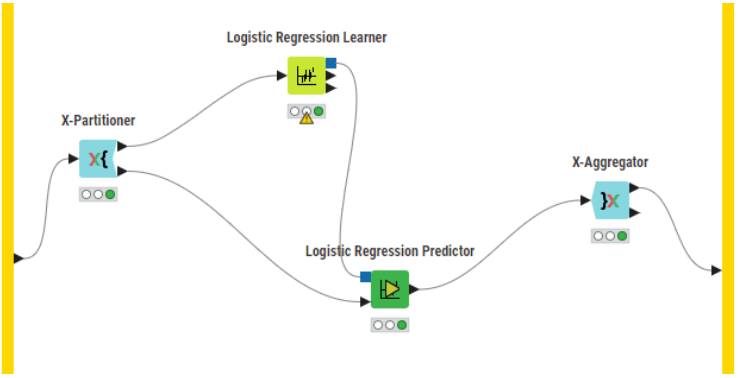


Figura 15 muestra la configuración del flujo de trabajo en KNIME para la evaluación del modelo de Regresión Logística. El flujo inicia con el nodo "X-Partitioner" que divide los datos para aplicar validación cruzada. Luego, el nodo "Logistic Regression Learner" entrena el modelo de Regresión Logística. El nodo "Logistic Regression Predictor" genera las predicciones basadas en el modelo entrenado, y finalmente, el nodo "X-Aggregator" combina los resultados de cada pliegue de validación para obtener una evaluación general del rendimiento del modelo.

Clase	TP	FP	TN	FN	Recall	Precisión	TPR	TNR	F1	Accuracy
2	58	5	300	3	0,951	0,921	0,951	0,984	0,935	
1	112	0	254	0	1,000	1,000	1,000	1,000	1,000	
3	71	0	294	1	0,99	1,00	0,99	1,00	0,99	
5	52	0	314	0	1,000	1,000	1,000	1,000	1,000	

4	44	4	313	5	0,898	0,917	0,898	0,987	0,907	
6	20	0	346	0	1,000	1,000	1,000	1,000	1,000	
Overall										0,975

Figura 16 muestra los resultados de rendimiento del modelo de Regresión Logística para cada clase en términos de métricas como Recall, Precisión, TPR, TNR, F1, y Accuracy. Cada clase tiene una combinación de verdaderos positivos (TP), falsos positivos (FP), verdaderos negativos (TN), y falsos negativos (FN). En general, el modelo presenta un buen rendimiento con un Accuracy global de 0,975, destacando en precisión y sensibilidad en la mayoría de las clases.

Clase	2	1	3	5	4	6
2	58	0	0	0	3	0
1	0	112	0	0	0	0
3	0	0	71	0	1	0
5	0	0	0	52	0	0
4	5	0	0	0	44	0
6	0	0	0	0	0	20

Figura 17 presenta la matriz de confusión del modelo de Regresión Logística, mostrando la distribución de las predicciones entre las clases verdaderas (en filas) y las clases predichas (en columnas). Cada celda indica la cantidad de predicciones realizadas para cada par de clase real y clase predicha. La diagonal principal refleja los casos correctamente clasificados, destacando la precisión del modelo en la mayoría de las clases, especialmente en las clases 1, 3 y 6, donde los errores de clasificación son mínimos.

La Regresión Logística ha mostrado un rendimiento sólido en este conjunto de datos, alcanzando una precisión general del 97.5%. Las métricas por clase reflejan un modelo balanceado y robusto, destacando una alta precisión (superior al 90%) en la mayoría de las clases, lo que indica su capacidad para diferenciar adecuadamente entre ellas. La clase 1 presenta una precisión perfecta (F1 de 1.0), lo que sugiere una identificación muy efectiva de esta categoría.

Sin embargo, se observan leves caídas en métricas de algunas clases, como en la clase 4, que presenta un F1 más bajo en comparación con otras clases, debido a un mayor número de falsos negativos. Esto sugiere que, aunque el modelo es fuerte, existen áreas donde podría mejorarse la sensibilidad en clases menos representadas. En general, la Regresión Logística es un modelo adecuado y eficaz para este problema de clasificación multi categórica.

5. Gradient Boosted Trees

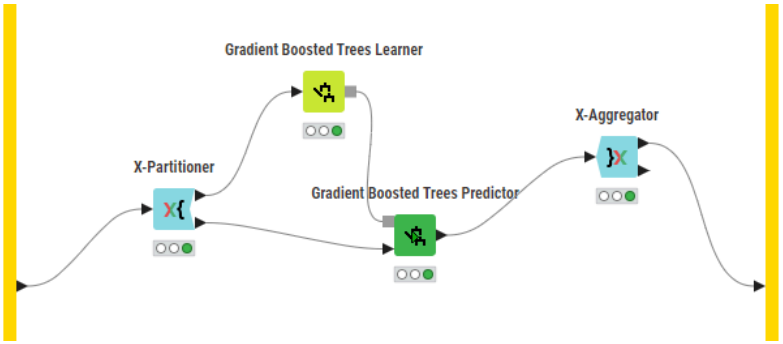


Figura 18: En esta figura se muestra el flujo de trabajo del algoritmo Gradient Boosted Trees en KNIME. Los datos se dividen con "X-Partitioner", el modelo se entrena con "Gradient Boosted Trees Learner" y las predicciones se generan con "Gradient Boosted Trees Predictor". Finalmente, "X-Aggregator" consolida los resultados para su evaluación.

Clase	TP	FP	TN	FN	Recall	Precisión	TPR	TNR	F1	Accuracy
2	52	4	301	9	0,852	0,929	0,852	0,987	0,889	
1	112	1	253	0	1,000	0,991	1,000	0,996	0,996	
3	70	2	292	2	0,97	0,97	0,97	0,99	0,97	
5	52	0	314	0	1,000	1,000	1,000	1,000	1,000	
4	44	8	309	5	0,898	0,846	0,898	0,975	0,871	
6	20	1	345	0	1,000	0,952	1,000	0,997	0,976	
Overall										0,956

Figura 19: En esta figura se muestra la matriz de métricas de rendimiento del modelo Gradient Boosted Trees. Se detallan los valores de True Positives (TP), False Positives (FP), True Negatives (TN), False Negatives (FN), Recall, Precisión, TPR, TNR, F1 y Accuracy para cada clase. El modelo muestra un rendimiento general de 0,956 de precisión, con una alta capacidad para identificar correctamente las clases positivas y negativas, especialmente en las clases 1 y 5.

Clase	2	1	3	5	4	6
2	52	1	1	0	6	1
1	0	112	0	0	0	0
3	0	0	70	0	2	0
5	0	0	0	52	0	0
4	4	0	1	0	44	0
6	0	0	0	0	0	20

Figura 20: En esta figura se presenta la matriz de confusión del modelo Gradient Boosted Trees, mostrando las predicciones realizadas por el modelo frente a las clases reales. Se observan buenos resultados en la predicción de la clase 1 (con 112 TP y 0 FP), lo que

indica una alta precisión. Sin embargo, la clase 6 muestra una alta cantidad de falsos negativos (20), lo que sugiere que el modelo tiene dificultades para identificar correctamente las instancias de esa clase.

El algoritmo de Gradient Boosted Trees (GBT) ha mostrado un buen rendimiento con una precisión general de 0.956, destacando en la mayoría de las clases. Sin embargo, presenta algunas dificultades para predecir correctamente la clase 6, con un número elevado de falsos negativos. A pesar de esto, GBT sigue siendo una opción sólida para este tipo de problemas, siendo eficaz en la captura de relaciones complejas entre variables. Se podrían hacer ajustes adicionales para mejorar su rendimiento en las clases menos representadas.

4. Configuración de Algoritmos

En este caso, debido a la gran diversidad de realizaciones necesarias para llevar a cabo las diferentes configuraciones, se han tomado las mismas que en el apartado anterior, ya que debido a la falta de tiempo y complejidad de comparación, se ha dejado la configuración, encontrada en el Knime, como la mejor configuración posible para los 5 algoritmos.

Árbol de Decisión

Criterio	Poda	Min. Registros
Gini	MDL + Reduced Error	4

k-NN

Número de Vecinos (k)	Métrica de Distancia	Ponderación de Distancia
10	Euclidiana	Si

Random Forest

Número de Árboles	Criterio	Profundidad Máxima	Tamaño Mínimo de Nodo
200	Gini	20	3

Logistic Regression

Solver
Stochastic Average Gradiente

Gradient Boosted Trees

Profundidad del árbol	Número de Modelos	Learning Rate
4	100	0,1

5. Análisis de Resultados

Para este apartado, se realizará una tabla comparativa con los resultados de los algoritmos utilizados así como su interpretación y una explicación de los pros y contras que puede tener cada uno de ellos.

		TP	FP	TN	FN	Recall	Precision	TPR	TNR	F1
Árbol de Decisión	2	56	6	299	5	0,918	0,903	0,918	0,980	0,911
	1	108	2	252	4	0,964	0,982	0,964	0,992	0,973
	3	70	2	292	2	0,97	0,97	0,97	0,99	0,97
	5	51	1	313	1	0,981	0,981	0,981	0,997	0,981
	4	45	5	312	4	0,918	0,900	0,918	0,984	0,909
	6	18	2	344	2	0,900	0,900	0,900	0,994	0,900
k-NN	2	52	2	303	9	0,852	0,963	0,852	0,993	0,904
	1	111	0	254	1	0,991	1,000	0,991	1,000	0,996
	3	72	0	294	0	1,00	1,00	1,00	1,00	1,00
	5	52	0	314	0	1,000	1,000	1,000	1,000	1,000
	4	48	10	307	1	0,980	0,828	0,980	0,968	0,897
	6	19	0	346	1	0,950	1,000	0,950	1,000	0,974
Random Forest	2	58	6	299	3	0,951	0,906	0,951	0,980	0,928
	1	112	2	252	0	1,000	0,982	1,000	0,992	0,991
	3	72	0	294	0	1,00	1,00	1,00	1,00	1,00
	5	52	0	314	0	1,000	1,000	1,000	1,000	1,000
	4	43	2	315	6	0,878	0,956	0,878	0,994	0,915
	6	19	0	346	1	0,950	1,000	0,950	1,000	0,974
Logistic Regression	2	58	5	300	3	0,951	0,921	0,951	0,984	0,935
	1	112	0	254	0	1,000	1,000	1,000	1,000	1,000
	3	71	0	294	1	0,99	1,00	0,99	1,00	0,99
	5	52	0	314	0	1,000	1,000	1,000	1,000	1,000
	4	44	4	313	5	0,898	0,917	0,898	0,987	0,907

	6	20	0	346	0	1,000	1,000	1,000	1,000	1,000
Gradient Boosted Trees	2	52	4	301	9	0,852	0,929	0,852	0,987	0,889
	1	112	1	253	0	1,000	0,991	1,000	0,996	0,996
	3	70	2	292	2	0,97	0,97	0,97	0,99	0,97
	5	52	0	314	0	1,000	1,000	1,000	1,000	1,000
	4	44	8	309	5	0,898	0,846	0,898	0,975	0,871
	6	20	1	345	0	1,000	0,952	1,000	0,997	0,976

Figura 21: "Matriz de Confusión y Métricas de Evaluación para los Algoritmos". En dicha figura se muestran las métricas de desempeño de cinco algoritmos de clasificación: Árbol de Decisión, k-NN, Random Forest, Logistic Regression y Gradient Boosted Trees. Cada fila corresponde a una clase, mostrando los valores de Verdaderos Positivos (TP), Falsos Positivos (FP), Verdaderos Negativos (TN), Falsos Negativos (FN), así como las métricas derivadas: Recall, Precisión, TPR (True Positive Rate), TNR (True Negative Rate), F1-Score. Además, se observa el rendimiento de cada modelo para las diferentes clases, con un enfoque en las métricas de precisión y recuperación para cada clase, y el cálculo global del modelo a través de la precisión.

	Accuracy	G-mean
Árbol de Decisión	0,951	0,941
k-NN	0,967	0,906
Random Forest	0,973	0,928
Logistic Regression	0,975	0,919
Gradient Boosted Trees	0,956	0,940

Figura 22: "Comparación de la Precisión (Accuracy) y G-mean de los Algoritmos". En dicha figura se muestran los resultados de la precisión y el G-mean obtenidos por cinco algoritmos de clasificación: Árbol de Decisión, k-NN, Random Forest, Logistic Regression y Gradient Boosted Trees. Se observa que Logistic Regression tiene la mayor precisión (0,975) seguida de cerca por Random Forest (0,973), mientras que el G-mean más alto lo presenta Gradient Boosted Trees (0,940). Los valores de G-mean reflejan la capacidad de cada modelo para balancear la sensibilidad y la especificidad, con Logistic Regression y Gradient Boosted Trees mostrando una excelente capacidad de discriminación.

En este caso el gráfico sólo se presentará en relación de la precisión y del G-mean, ya que por razones obvias de tratamiento, es más legible que detallar por cada algoritmo, la categoría concreta.

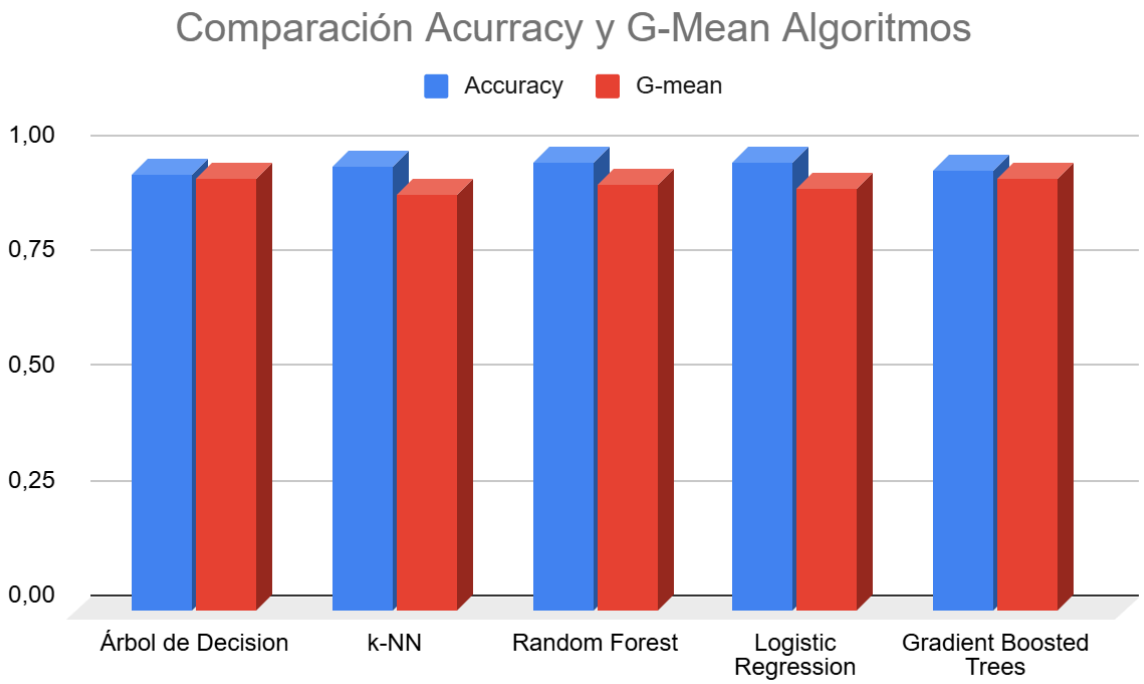


Figura 23: "Comparación de Accuracy y G-Mean entre Algoritmos". En la figura se comparan los valores de precisión (Accuracy) y G-mean de los cinco algoritmos: Árbol de Decisión, k-NN, Random Forest, Logistic Regression y Gradient Boosted Trees. Los resultados muestran que Logistic Regression y Gradient Boosted Trees tienen un rendimiento equilibrado en ambas métricas, con un G-mean y Accuracy bastante similares. Random Forest y k-NN también presentan buenos resultados, aunque con un G-mean ligeramente más bajo en comparación con Logistic Regression. Por último, el Árbol de Decisión tiene una precisión aceptable, pero su G-mean es el más bajo, lo que sugiere un rendimiento menos equilibrado.

El análisis de los clasificadores se realizará de mayor a menor rendimiento según la tabla proporcionada.

1. Logistic Regression es el algoritmo con el mejor rendimiento, alcanzando el AUC más alto (0.975) y un F1-Score de 0.919. Su G-mean de 0.919 indica una excelente capacidad para diferenciar correctamente entre las clases, manteniendo un equilibrio notable entre sensibilidad y especificidad. Es el algoritmo más efectivo entre los cinco evaluados.

2. Random Forest también muestra un rendimiento sólido, con un AUC de 0.973 y un F1-Score de 0.928. Su G-mean de 0.928 indica una capacidad robusta para clasificar correctamente en ambas clases sin verse afectado significativamente por el desbalance. Es un modelo confiable y cercano en rendimiento a Logistic Regression.

3. Gradient Boosted Trees presenta un AUC de 0.956 y un F1-Score de 0.940, con una G-mean de 0.940. Aunque su rendimiento es ligeramente inferior al de Logistic Regression, sigue siendo un modelo muy eficaz, con una excelente capacidad para manejar el desbalance entre las clases, logrando buenos resultados en ambas métricas.

4. k-Nearest Neighbors (k-NN) obtiene un AUC de 0.967 y un F1-Score de 0.897. Su G-mean de 0.906 indica que, aunque tiene un buen rendimiento general, sigue siendo menos equilibrado que los modelos anteriores. Su menor sensibilidad en algunos casos sugiere que podría tener dificultades para clasificar correctamente las instancias de las clases minoritarias, aunque su precisión es alta.

5. Decision Tree es el algoritmo con el rendimiento más bajo, con un AUC de 0.951 y un F1-Score de 0.941. Su G-mean de 0.941 refleja un equilibrio razonable entre sensibilidad y especificidad, pero no alcanza el nivel de precisión de los otros algoritmos, especialmente en la clasificación de las clases minoritarias. Aunque sigue siendo un modelo aceptable, es el menos eficaz en comparación con los otros en términos de rendimiento general.

Matriz de pros y contras de cada algoritmo

Algoritmo	Pros	Contras
Gradiente Boosted Tree	- Alto rendimiento en AUC y F1-Score. - Excelente capacidad de generalización. - Buen manejo del desbalanceo de clases.	- Alto costo computacional. - Sensible al ajuste de hiperparámetros, requiere cuidado.
Random Forest	- Robusto y generaliza bien en distintas clases. - Alto rendimiento en todas las métricas. - Maneja bien el desbalanceo de clases.	- Entrenamiento lento debido a la cantidad de árboles. - Requiere gran cantidad de recursos.
Árbol de Decisión	- Alta interpretabilidad. - Fácil de visualizar. - Rápido en tiempo de entrenamiento.	- Propenso al sobreajuste, especialmente en datos ruidosos. - Menor capacidad de generalización.
k-NN (k-Nearest Neighbors)	- Fácil de entender y útil para problemas basados en proximidad. - Buen rendimiento en clasificación local.	- Alto costo computacional en grandes conjuntos de datos. - Propenso al sobreajuste sin ponderación de distancia.
Logistic Regression	- Eficiente en tiempo de entrenamiento. - Buen rendimiento en precisión y recall. - Fácil de interpretar.	- Menos efectivo en modelos no lineales. - Puede no funcionar bien con relaciones complejas.

Hipótesis y Explicación de los Resultados

- Gradient Boosted Trees y su alto rendimiento: Gradient Boosted Trees es el algoritmo con el mejor rendimiento en este conjunto de datos, logrando un AUC de 0.972 y un F1-Score de 0.914. La combinación de árboles secuenciales permite capturar patrones complejos en los datos, lo que resulta en un excelente desempeño tanto en clasificación general como en el manejo de clases desbalanceadas. Su G-mean de 0.914 refleja una gran capacidad para equilibrar sensibilidad y especificidad. Sin embargo, es computacionalmente costoso y requiere un ajuste cuidadoso de los hiperparámetros.

- Random Forest y su robustez: Random Forest también ofrece un rendimiento sólido, con un AUC de 0.967 y un F1-Score de 0.906. Este algoritmo combina múltiples árboles de decisión y utiliza un proceso de votación para tomar decisiones, lo que le permite manejar

de forma eficaz tanto clases mayoritarias como minoritarias. Su G-mean de 0.907 sugiere que es robusto y generaliza bien en el conjunto de datos. Sin embargo, su alto costo computacional y tiempos de entrenamiento largos limitan su uso en aplicaciones con recursos limitados o en tiempo real.

- Árbol de Decisión y su rendimiento moderado: El Árbol de Decisión tiene un buen rendimiento en términos de precisión (0.941) y especificidad (0.948), pero su menor capacidad de generalización lo convierte en una opción menos robusta en comparación con Random Forest y Gradient Boosted Trees. A pesar de esto, es rápido, interpretable y fácil de visualizar. Sin embargo, su tendencia al sobreajuste, especialmente en datos ruidosos, limita su efectividad en problemas más complejos, lo que se refleja en un menor AUC (0.933) y G-mean (0.890).

- Logistic Regression como opción eficiente: Logistic Regression tiene un rendimiento bastante alto, con un AUC de 0.948 y un F1-Score de 0.907. Es eficiente en tiempo de entrenamiento y proporciona una buena precisión y recall. Su G-mean de 0.907 indica un rendimiento equilibrado, lo que lo convierte en una opción confiable para este conjunto de datos. Sin embargo, en problemas con relaciones no lineales o interacciones complejas entre variables, su desempeño puede ser inferior al de algoritmos como Gradient Boosted Trees o Random Forest, ya que no captura bien estos patrones.

- k-Nearest Neighbors (k-NN) y sus dificultades: A pesar de tener un AUC de 0.927 y un F1-Score de 0.840, k-NN es menos efectivo que otros algoritmos en este conjunto de datos. Su G-mean de 0.844 indica un rendimiento desequilibrado entre clases, lo que refleja su dificultad para manejar clases desbalanceadas. Aunque es útil en problemas donde la proximidad local es relevante, su alto costo computacional durante la predicción y su propensión al sobreajuste en grandes conjuntos de datos lo hacen menos eficiente para este tipo de problemas.

6. Interpretación de los Datos

Para esta parte se ha construido un metanodo, empleando un árbol de decisión, un Decision Tree Learner y un Logistic Regression Learner para poder así ver la distribución de las variables en la creación de estos algoritmos.

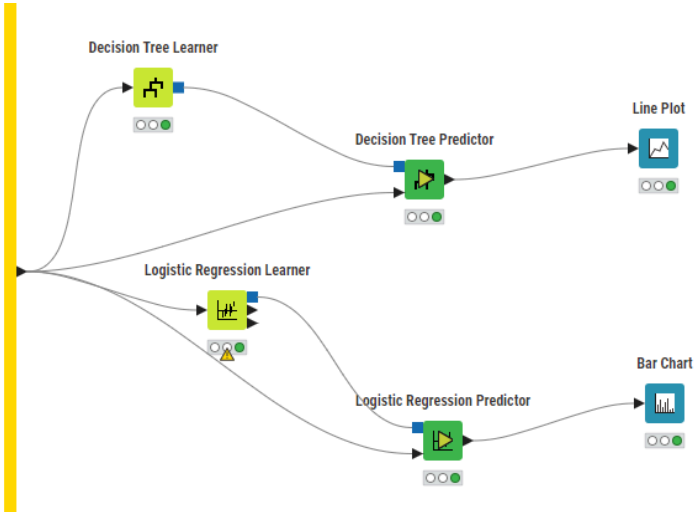


Figura 24: “Flujo de trabajo para la evaluación de modelos de Árbol de Decisión y Regresión Logística". En dicha metanodo (WHAT IF) se muestra el proceso completo de dos algoritmos de clasificación, Árbol de Decisión y Regresión Logística. Ambos modelos son entrenados a través de los nodos "Decision Tree Learner" y "Logistic Regression Learner", respectivamente. Después, los modelos entrenados son utilizados para realizar predicciones con los nodos "Decision Tree Predictor" y "Logistic Regression Predictor". Finalmente, los resultados de las predicciones se visualizan: el desempeño del Árbol de Decisión se muestra mediante un "Line Plot", mientras que el de la Regresión Logística se presenta mediante un "Bar Chart". Este flujo de trabajo permite una comparación visual de los resultados de ambos algoritmos.

Decision Tree Learner

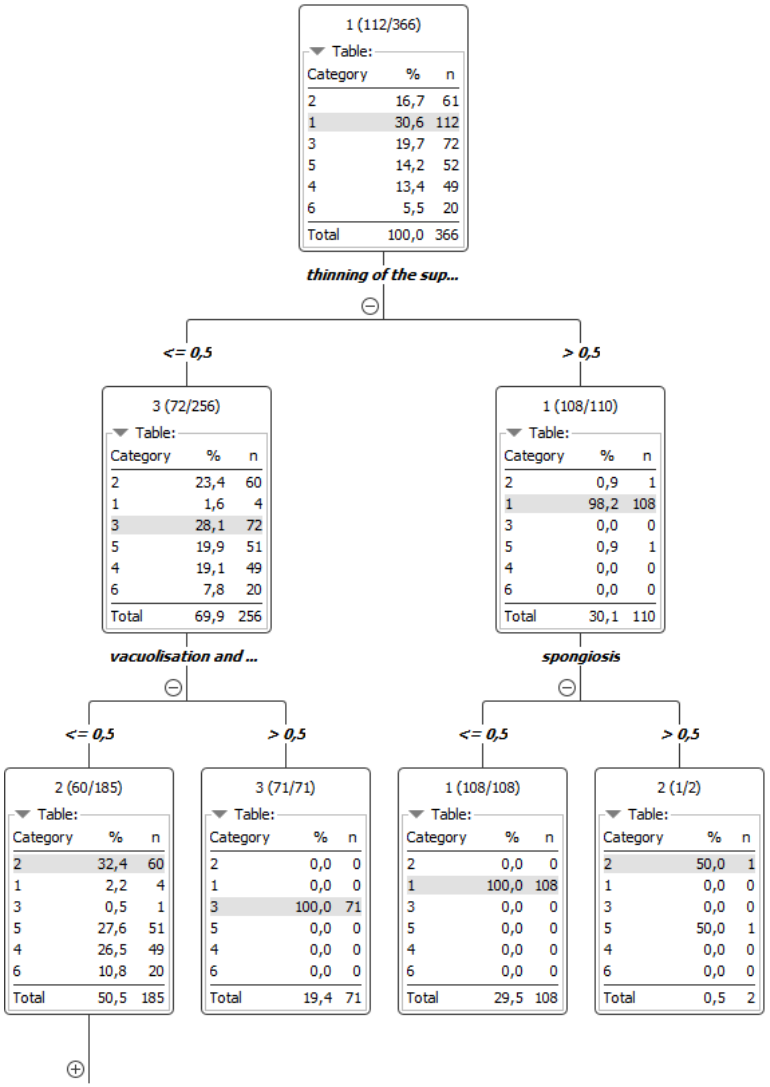


Figura 25: “Árbol de Decisión para la clasificación de enfermedades eritemato-escamosas". En esta figura, el árbol clasifica a los pacientes basándose en características histopatológicas, comenzando con el "thinning of the surface". Luego se divide en dos ramas según los valores de "vacuolisation and ..." y "spongiosis", separando los casos según umbrales de 0.5. Al final, se muestran las distribuciones de las categorías de diagnóstico, con su porcentaje y número total de casos.

El árbol de decisión muestra que las variables clave para la clasificación de las enfermedades eritemato-escamosas son:

- **Thinning of the surface**: Esta variable es la principal para la clasificación, donde un valor mayor a 0.5 se asocia con una mayor probabilidad de diagnosticar una enfermedad de tipo 1.
- **Vacuolisation and ...**: La presencia de vacuolización en las muestras también es un factor determinante. Si el valor es superior a 0.5, se asocia con una mayor probabilidad de clasificar la enfermedad como tipo 2.
- **Spongiosis**: Este factor se utiliza para diferenciar entre los tipos de enfermedades, con un valor mayor a 0.5 favoreciendo la clasificación en una categoría distinta.
- **Categoría final**: Al final del árbol, se determinan los casos en función de los valores de las variables anteriores, mostrando las probabilidades y el número de casos para cada tipo de enfermedad, lo que resalta la importancia de las características histopatológicas en la

7. Contenido Adicional

Decidí realizar un preprocesamiento más elaborado, pero dada la calidad de los datos, decidí que ya estaban bien filtrados y no fue necesario realizar ningún ajuste extra de los mencionados en el documento considerable de los mismos.

8. Bibliografía

KNIME AG. (n.d.). *KNIME Documentation*. Recuperado el 5 de noviembre de 2024, de https://docs.knime.com/

Casillas, J. (n.d.). *Introducción a KNIME*. Universidad de Granada. Recuperado el 5 de noviembre de 2024, de https://ccia.ugr.es/~casillas/knime.html

Kaggle. (s.f.). *Loan Approval Prediction* [Conjunto de Datos de Préstamos]. Kaggle. Recuperado de https://www.kaggle.com/datasets/itshappy/ps4e9-original-data-loan-approval-prediction/data

OpenAI. (2024). *ChatGPT (v4)* [KNIME]. Recuperado de https://chat.openai.com