



UNIVERSIDAD
DE GRANADA

PRÁCTICA 1

Análisis Predictivo mediante
Clasificación

Inteligencia de Negocio

DGIIM - 2024/2025

Carmen Azorín Martí

G1

carmenazorin@correo.ugr.es

ÍNDICE

<i>Introducción</i>	4
1. Predicción de aprobación de créditos	4
2. Predicción de segunda cita	5
3. Predicción de enfermedades eritemato-escamosas	6
<i>Procesado de datos</i>	8
1. Predicción de aprobación de créditos	8
2. Predicción de segunda cita	9
3. Predicción de enfermedades eritemato-escamosas	11
<i>Resultados obtenidos</i>	12
1. Predicción de aprobación de créditos	12
Regresión Logística	12
Árbol de decisión	14
Red Neuronal	15
K Nearest Neighbour	17
Naive Bayes	19
2. Predicción de segunda cita	20
Árbol de decisión	20
Random Forest	22
K-Nearest NeighBor	23
Gradient Boosted Trees	25
Regresión Logística	26
3. Predicción de enfermedades eritemato-escamosas	28
Árbol de decisión	28
Random Forest	29
Gradient Boosted Trees	30
K-Nearest Neighbor	31
Red Neuronal MLP	32
<i>Configuración de algoritmos</i>	34
1. Predicción de aprobación de créditos	34
Regresión Logística	34
Árbol de decisión	34
Red Neuronal MLP	35
K-Nearest Neighbors	36
<i>Análisis de resultados</i>	37
1. Predicción de aprobación de créditos	37
2. Predicción de segunda cita	39
3. Predicción de enfermedades eritemato-escamosas	40
<i>Interpretación de los datos</i>	44
1. Predicción de aprobación de créditos	44
Regresión Logística	44
Árbol de decisión	44
Red Neuronal MLP	45

K-Nearest Neighbors	46
2. Predicción de segunda cita	46
Árbol de decisión	46
Random Forest	47
Regresión Logística	48
3. Predicción de enfermedades eritemato-escamosas	48
Árbol de decisión	48
Random Forest	50
Gradient Boosted Trees	50
<i>Bibliografía</i>	53

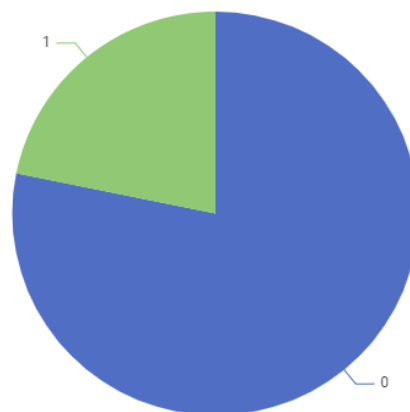
Introducción

1. Predicción de aprobación de créditos

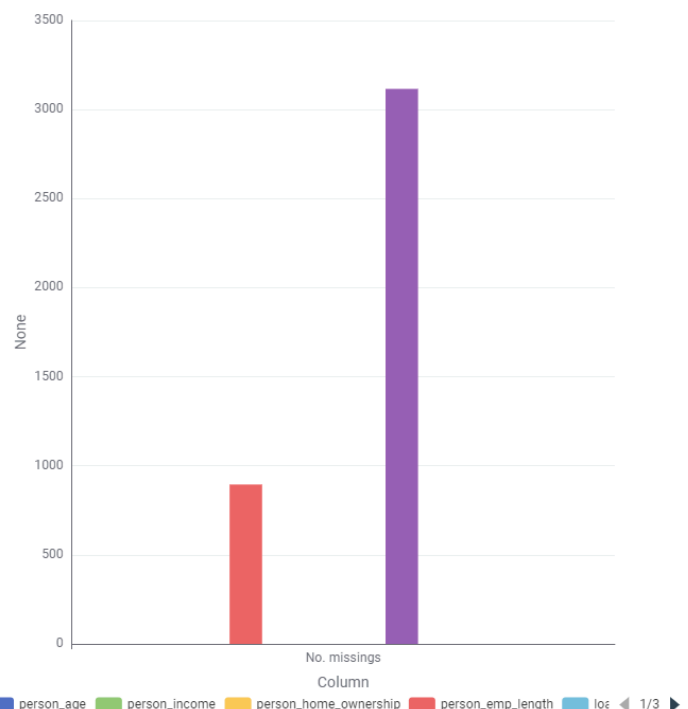
Este problema de clasificación se enfoca en predecir si a un solicitante se le aprobará un préstamo. Contiene información variada sobre características personales y financieras de los solicitantes.

Algunas particularidades del problema:

- *Dimensiones y variables*: 32,581 instancias, con variables numéricas como `loan_int_rate` y `person_age`, y nominales como `person_home_ownership`.
- *Desbalanceo de clase*: identificamos un desbalance en la variable `loan_status`, donde la clase "0" (no aprobado) representa el 78% de las instancias, mientras que la clase "1" (aprobado) solo el 22%. Este desbalance se detectó visualizando la distribución de clases en un Pie Chart, observando la diferencia significativa entre ambas categorías.



- *Valores perdidos*: en el análisis de datos, se encontraron valores faltantes en el conjunto: **895** en `person_emp_length` (aprox. **2.75%**) y **3116** en `loan_int_rate` (cerca del **9.56%**). Aunque el porcentaje de missing values en `person_emp_length` es bajo, el **9.56%** en `loan_int_rate` es más preocupante y debe ser tratado mediante imputación o eliminación de filas.



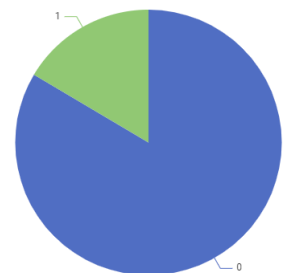
Para este problema de aprobación de créditos, he optado por Regresión Logística, Árbol de Decisión, MLP (Perceptrón Multicapa), K-Nearest Neighbors (KNN) y Naive Bayes. La Regresión Logística es útil aquí debido a su simplicidad y facilidad para interpretar probabilidades en un problema de decisión binaria (aprobación o rechazo). Sin embargo, tiene limitaciones en cuanto a modelar relaciones no lineales, por lo que agregué otros algoritmos. El Árbol de Decisión proporciona interpretabilidad y es fácil de entender para analizar decisiones de préstamo, aunque puede sobreajustarse a los datos si no se poda adecuadamente. El MLP, una red neuronal, permite captar patrones complejos y no lineales en los datos, aunque requiere un mayor tiempo de entrenamiento y puede ser menos interpretable. KNN es sencillo y puede capturar relaciones locales, aunque su rendimiento disminuye si los datos tienen demasiadas características. Finalmente, Naive Bayes es rápido y manejable para grandes conjuntos de datos como este, aunque su supuesto de independencia entre variables puede ser poco realista.

2. Predicción de segunda cita

Este problema de análisis predictivo aborda la probabilidad de que una pareja decida tener una segunda cita tras una primera interacción en un evento de citas rápidas. Cada participante tiene una cita de cuatro minutos con personas del sexo opuesto, y luego evalúan si les gustaría volver a ver a su cita. La variable objetivo a predecir es **match**, la cual indica si ambas partes desean tener una segunda cita.

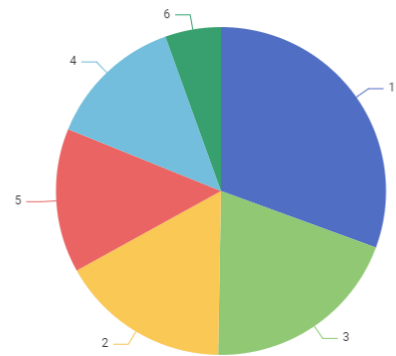
Algunas particularidades del problema:

- **Dimensiones:** 8378 instancias, con una gran cantidad de variables relacionadas con características demográficas, percepciones personales, preferencias en pareja y puntuaciones asignadas a cada cita. Las variables incluyen tanto datos personales y autopercepciones de cada participante, como evaluaciones directas sobre su cita en atributos clave (atractivo, sinceridad, inteligencia, diversión, ambición e intereses compartidos).
- **Tipos de Variables:** existen diferentes tipos de variables, incluidas variables numéricas (edad, puntuaciones de atributos), categóricas (género, raza, campo de estudio) y datos ordinales en algunas preferencias.
- **Desbalanceo de Clases:** la clase objetivo, **match**, está desequilibrada, con un 84% de los registros etiquetados como "no hay match" (0) y solo el 16% como "hay match" (1).
- **Valores Perdidos:** muchas columnas contienen valores perdidos, especialmente aquellas que registran atributos evaluados por la pareja, como **shared_interest_o**, **ambitious_o**, y **funny_o**. Los valores ausentes son numerosos, especialmente en ciertas características, lo que podría deberse a la naturaleza subjetiva de algunas preguntas.



histopatológicas. Las variables son discretas.

- *Desbalanceo de clases*: presenta un desbalanceo significativo en las clases. La clase 1, que representa la enfermedad más común, tiene más del 30% de los casos, mientras que la clase 6 tiene solo un 5.46%, lo que puede sesgar el modelo hacia las clases mayoritarias.
- *Valores perdidos*: contiene sólo 8 valores faltantes en la variable *age*, lo que representa menos del 2% del total. Este pequeño número de valores perdidos no debería afectar significativamente el análisis.



Para este problema de diagnóstico de enfermedades eritemato-escamosas, elegí Árbol de Decisión, Random Forest, KNN, Gradient Boosted Trees y MLP. Este problema es ideal para el uso de modelos de árbol, ya que los síntomas y características histopatológicas permiten una clasificación jerárquica. Random Forest y Gradient Boosted Trees pueden captar mejor las interacciones complejas y compensar el sobreajuste. El MLP es útil en este problema, ya que la naturaleza de los datos histopatológicos y clínicos puede beneficiarse de la capacidad de una red neuronal para captar relaciones no lineales, aunque es menos interpretable. KNN permite hacer una clasificación en función de pacientes similares, aunque podría tener dificultades si el conjunto de datos es muy grande, pero no es nuestro caso.

Procesado de datos

1. Predicción de aprobación de créditos

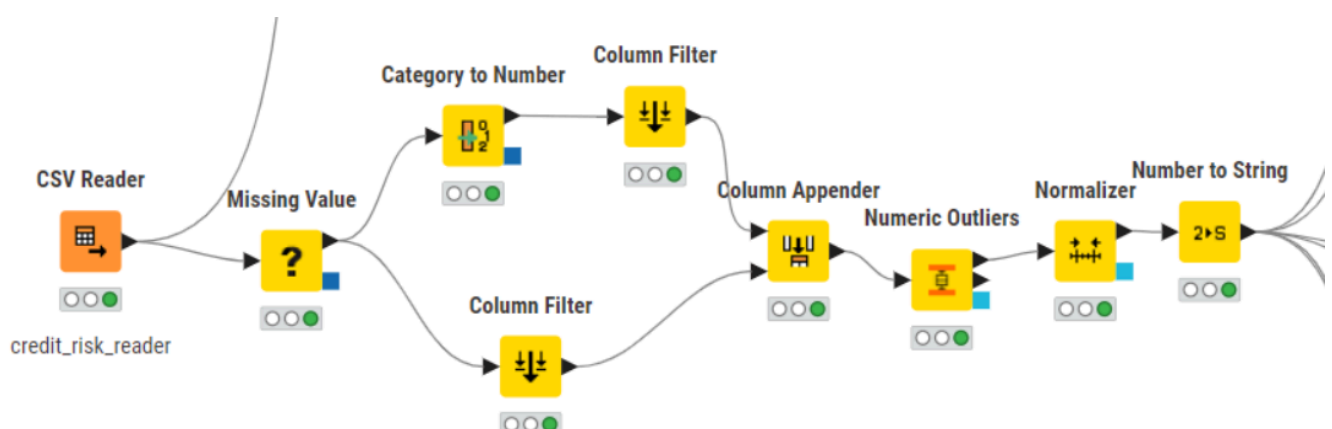
Para abordar la predicción de aprobación de créditos, he realizado un proceso de preprocesamiento que prepara los datos de forma que sean útiles para varios algoritmos (regresión logística, árbol de decisión, red neuronal, KNN y Naive Bayes). Este preprocesamiento ayuda a limpiar y transformar los datos para que los modelos puedan entrenarse de manera más efectiva. Los pasos son los siguientes:

En primer lugar, había un 28% de valores faltantes en los datos, por lo que utilicé la media para completarlos en las variables nominales y la eliminación de filas en las variables categóricas. Al asegurar que cada columna esté completa, permitimos que los algoritmos se ejecuten sin problemas, ya que muchos de ellos requieren una base de datos sin valores vacíos.

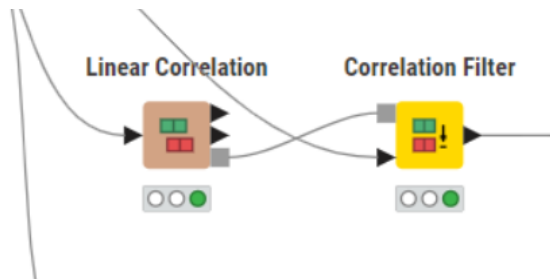
Posteriormente, he convertido las variables categóricas en números usando el nodo **Category to Number**. Este paso es crucial, ya que la mayoría de los algoritmos solo pueden trabajar con datos numéricos. Además, eliminé las columnas categóricas originales y mantuve solo las variables codificadas para evitar duplicados y confusiones en el modelo.

A continuación, he usado el nodo **Column Append** para crear una tabla que incluye tanto las variables numéricas como las codificadas. Para luego eliminar los valores extremos en las variables numéricas utilizando el nodo **Numeric Outliers**. Esto ayuda a que los datos sean más representativos y a que valores atípicos no distorsionen los resultados, algo especialmente importante para algoritmos como KNN y redes neuronales, que son muy sensibles a la escala de los datos. Dejé fuera de este proceso a las variables booleanas **loan_status** (la variable objetivo) y **cb_person_default_on_file** para que los algoritmos de clasificación puedan trabajar con ellas sin problemas.

Después, he normalizado todas las variables numéricas al rango de 0 a 1. Esto es especialmente útil en modelos que se basan en la distancia, como KNN y redes neuronales, porque asegura que todas las variables estén en la misma escala y que ninguna domine a las demás. Por último, convertí **loan_status** en una cadena de texto para que los algoritmos lo traten como una categoría. Esto permite que KNIME y los modelos lo identifiquen correctamente como la etiqueta de clasificación que queremos predecir.



He aplicado una selección de características utilizando los nodos **Linear Correlation** y **Correlation Filter** antes de entrenar el modelo KNN y Naive Bayes, ya que este algoritmo es sensible a la redundancia de las variables. Al eliminar las características altamente correlacionadas, se reduce el ruido en los datos, lo que puede mejorar el rendimiento y la precisión del modelo.



El resultado del nodo **Linear Correlation** nos indica que hay correlación entre algunas de las variables numéricas. Por ejemplo, el correlation value entre **person_age** y **cb_person_cred_hist_length** es de 0.815, lo que demuestra que no son independientes entre ellas. Este dato es crucial para la interpretación del algoritmo de Naive

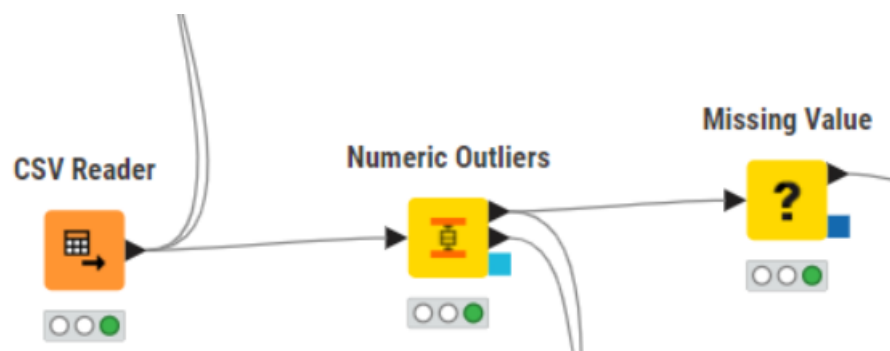
Bayes, ya que asume la independencia de las variables y, en este caso, no lo son.

El nodo de **Correlation Filter** elimina las columnas que están correlacionadas. Hemos considerado que un correlation threshold útil es 0.7. Esto conduce a la eliminación de 2 de las 13 columnas originales que teníamos. Estas once columnas no correlacionadas serán el input de los algoritmos de K-Nearest Neighbour y Naive Bayes.

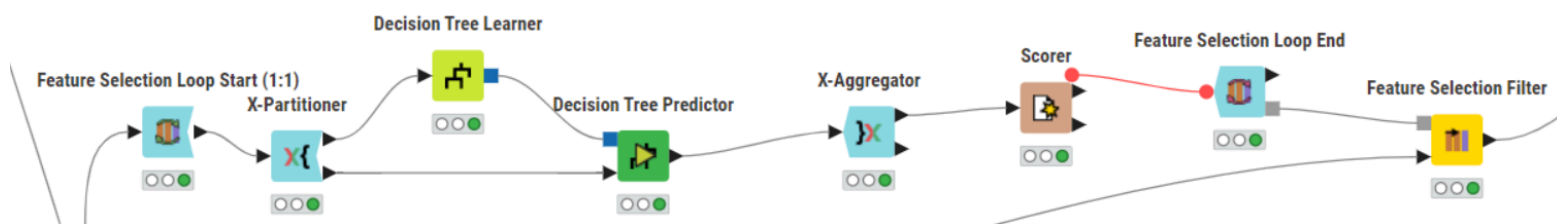
2. Predicción de segunda cita

Tras el análisis de desbalanceo de clases y comprobar un alto número de valores faltantes en algunas columnas. El primer paso fue tratar los outliers numéricos. Usamos el nodo **Numeric Outliers**, que identifica y elimina las filas que contienen valores atípicos, asegurando que los modelos no se vean influenciados por estos datos extremos. Después de este paso, el número de filas se redujo de 8378 a 3108.

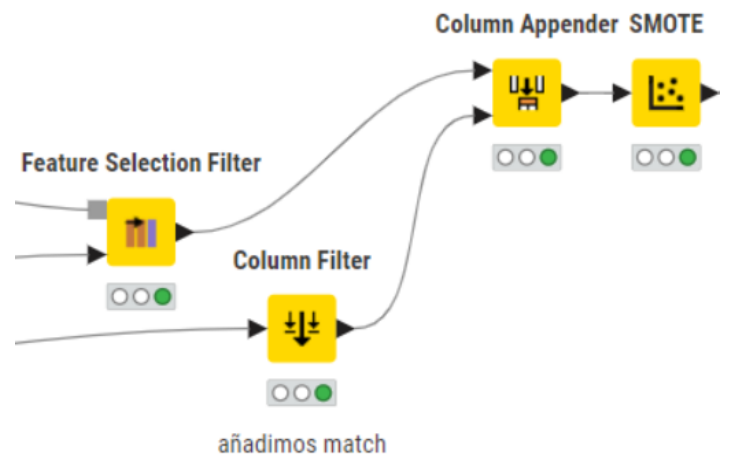
A continuación, nos enfrentamos a los valores faltantes. Utilizamos el nodo **Missing Values**, que nos permitió tratar los valores perdidos en las columnas numéricas y nominales. Para las columnas numéricas, sustituimos los valores faltantes por la media de cada columna, lo que mantiene la distribución de los datos. Para las columnas nominales, eliminamos las filas con valores perdidos, ya que no es adecuado reemplazar estos valores por una moda o por una clase arbitraria. Esto dejó 3107 filas en el conjunto de datos.



Una vez tratados los valores faltantes, realizamos una selección de características con un **Feature Selection Loop**. Usamos un Árbol de Decisión para evaluar qué características eran más relevantes para predecir la variable objetivo. Este proceso se realizó calculando el coeficiente Cohen's Kappa, que mide la consistencia entre las características seleccionadas y la variable a predecir. Seleccionamos el subconjunto de características que generaba el mejor valor de Cohen's Kappa (0.338) y elegimos 65 características relevantes, reduciendo las 119 originales a 65. La selección de características ayuda a reducir la complejidad del modelo y mejora su rendimiento, al centrarse sólo en las variables más importantes.



Para lidiar con el desbalanceo de clases, aplicamos el nodo **SMOTE** (Synthetic Minority Over-sampling Technique). SMOTE genera nuevas instancias sintéticas de la clase minoritaria, lo que ayuda a equilibrar la distribución de clases en los datos y evita que el modelo esté sesgado hacia la clase mayoritaria. Este paso es fundamental para mejorar la precisión y la capacidad de generalización del modelo en situaciones de desbalanceo.

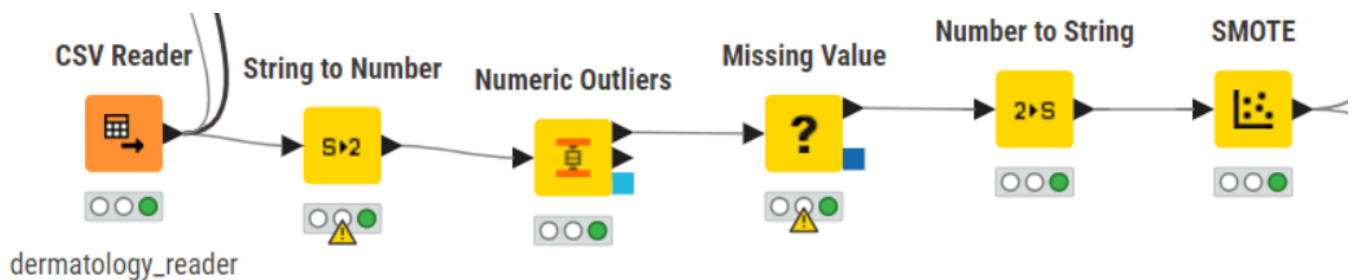


Este preprocesamiento fue adecuado para los algoritmos Decision Tree y Random Forest, que no requirieron ajustes adicionales. Sin embargo, para los modelos Logistic Regression, KNN y Gradient Boosted Trees, añadimos un par de pasos adicionales. Primero, utilizamos el nodo **Category to Number** para convertir las variables categóricas en valores numéricos, ya que estos modelos requieren que todas las características sean numéricas. También aplicamos una normalización de los datos para que todas las características tuvieran un rango similar, lo que es especialmente importante para algoritmos como KNN, que es sensible a la escala de las variables.



3. Predicción de enfermedades eritemato-escamosas

En el proceso de preprocesamiento de datos he convertido la variable `age` de tipo string a tipo entero usando el nodo `String to Number`. Esto es necesario para que todas las variables predictoras se encuentren en formato numérico, facilitando así el análisis y el uso de algoritmos que requieren datos numéricos. La única variable que permanece en formato categórico es `class`, que es la variable objetivo que queremos predecir.

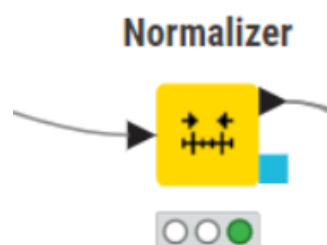


Luego, he usado el nodo `Numeric Outliers` para detectar y reemplazar valores atípicos en las variables numéricas con valores faltantes. Este paso ayuda a reducir el impacto de valores extremos que podrían sesgar el modelo y afectar su precisión.

Posteriormente, he usado el nodo `Missing Value` para reemplazar los valores enteros faltantes con el valor más común en cada variable. Esto garantiza que los datos estén completos y no haya valores faltantes que podrían interrumpir el análisis o causar errores en etapas posteriores.

Finalmente, he aplicado el nodo `SMOTE` (Synthetic Minority Over-sampling Technique) para realizar un sobremuestreo de las clases minoritarias en la variable objetivo. Esta técnica es crucial para abordar el problema del desbalanceo de clases, mejorando así la capacidad del modelo para identificar correctamente las clases minoritarias y evitando que el modelo esté sesgado hacia la clase mayoritaria.

Para los algoritmos de Gradient Boosted Trees, KNN y Red neuronal MLP añadí un nuevo nodo para normalizar los datos.



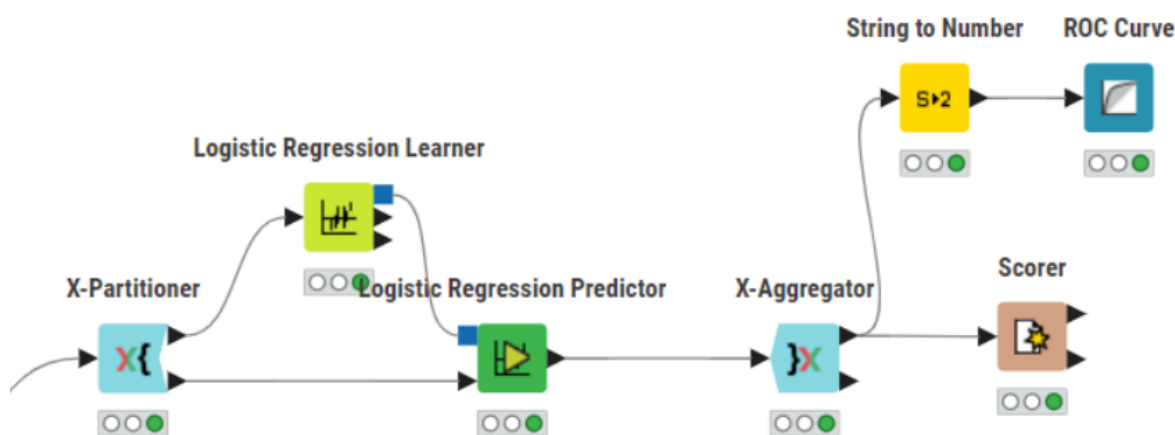
Resultados obtenidos

1. Predicción de aprobación de créditos

Regresión Logística

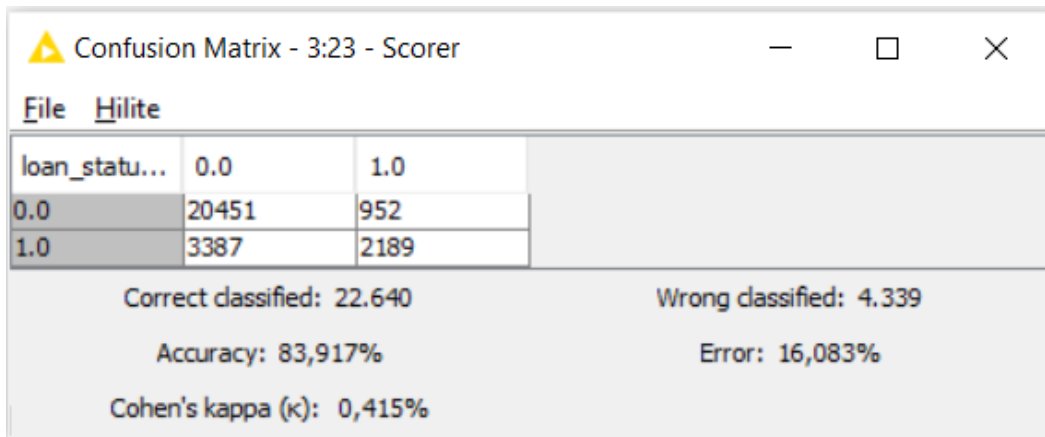
Después de realizar el preprocesamiento de los datos, el siguiente paso ha sido entrenar y evaluar el modelo de regresión logística. Para esto, hemos utilizado una validación cruzada con 5 particiones, mediante los nodos **X-Partitioner** y **X-Aggregator**. Este enfoque nos permite evaluar el rendimiento del modelo de forma más robusta, ya que cada partición de los datos es utilizada tanto para entrenamiento como para prueba en distintos momentos, promediando así los resultados y reduciendo el riesgo de sobreajuste.

Para cada partición, hemos entrenado el modelo usando el nodo **Logistic Regression Learner**, conectándolo con los datos de entrenamiento correspondientes. Posteriormente, cada modelo entrenado ha sido evaluado con los datos de prueba de la partición mediante el nodo **Logistic Regression Predictor**. Esto nos ha permitido generar predicciones para cada partición y comparar los resultados en términos de métricas de rendimiento.



Después de las predicciones, hemos utilizado el nodo **Scorer** para obtener los resultados de rendimiento del modelo. Por un lado, el Scorer nos muestra la matriz de confusión, que nos muestra el número de verdaderos positivos, falsos positivos, verdaderos negativos y falsos negativos.

Y a partir de la matriz calcula otras métricas de evaluación como la exactitud (accuracy) y el coeficiente kappa de Cohen (Cohen's kappa).



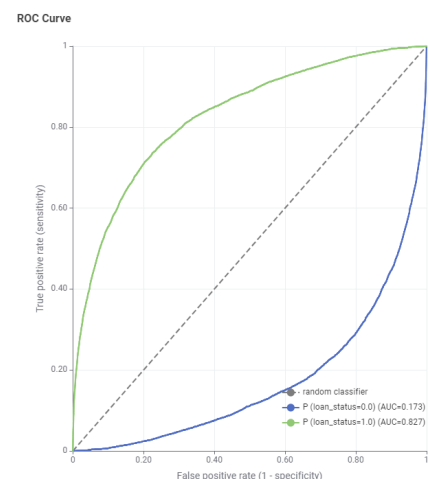
TruePositives	FalsePositives	TrueNegatives	FalseNegatives	Recall	Precision	Sensitivity	Specificity	F-measure	Accuracy	Cohen's kappa
20451	3387	2189	952	0,96	0,86	0,96	0,39	0,90		
2189	952	20451	3387	0,39	0,70	0,39	0,96	0,50		
										0,84 0,42

Por otro lado, el Scorer nos muestra las estadísticas de exactitud. La segunda fila del Scorer representa las métricas de rendimiento al predecir la clase positiva (en este caso, los casos de aprobación de crédito). La precisión (0.70) indica que el modelo es capaz de identificar correctamente el 70% de los casos predichos como positivos, mientras que el recall (0.39) muestra que el modelo captura el 39% de todos los casos positivos reales. La F-measure, que equilibra precisión y recall, tiene un valor moderado de 0.39, lo que sugiere que el modelo aún tiene margen de mejora en términos de captura de verdaderos positivos.

La primera fila del Scorer representa el caso inverso, enfocándose en la predicción de la clase negativa. Aquí, el recall es mucho más alto (0.96), lo que significa que el modelo identifica casi todos los casos negativos correctamente, pero la especificidad es más baja. La F-measure para esta clase es más alta (0.96), lo cual es consistente con el hecho de que el modelo está mucho mejor en la predicción de la clase negativa.

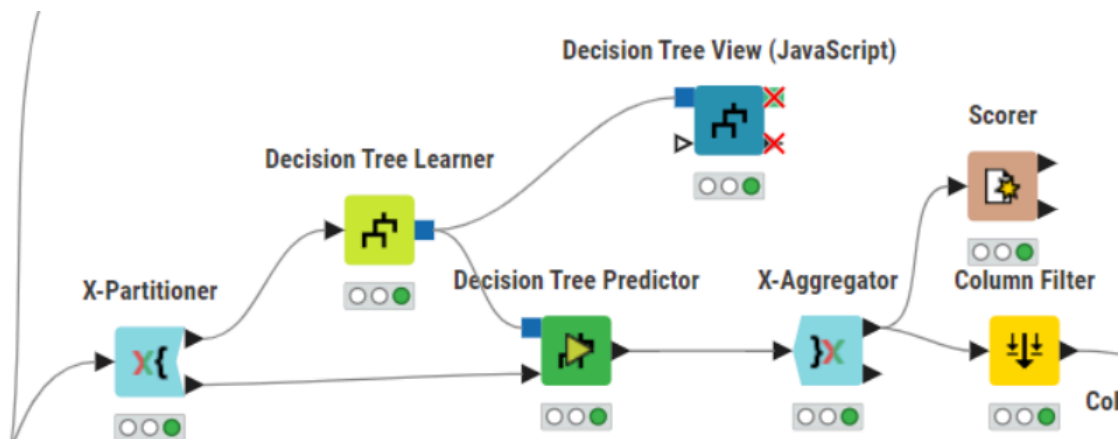
Dado que la clase de "aprobado" está desbalanceada respecto a la de "rechazado", el modelo muestra un rendimiento mucho mejor al predecir la clase mayoritaria (negativa).

Finalmente, hemos calculado la curva ROC conectándola con el nodo de **Logistic Regression Predictor**. En general, un valor de AUC (Área bajo la curva) de 0.5 representa un clasificador aleatorio, mientras que un AUC de 1.0 ó 0.0 representa una predicción perfecta. Vemos que el AUC para la aprobación de créditos es 0.827 y para el rechazo de créditos es 0.173, lo que significa que la predicción está lejos de ser aleatoria en ambos casos



Árbol de decisión

Para abordar el problema de la aprobación de créditos mediante un Árbol de Decisión, he diseñado un algoritmo que usa validación cruzada de 5 particiones, como se nos pide. Este nodo se conecta con el nodo **Decision Tree Learner** usando el training data y se conecta con el nodo **Decision Tree Predictor** usando el testing data. Finalmente, para poder evaluar los resultados, se conecta con **X-aggregator**. El flujo de trabajo quedaría de la siguiente forma,



El nodo de evaluación (**Scorer**) proporcionó los siguientes resultados clave en términos de métricas:

Confusion Matrix - 3:32 - Scorer		
File	Hilite	
loan_statu...	0.0	1.0
0.0	21278	125
1.0	1818	3758
Correct classified: 25.036		Wrong classified: 1.943
Accuracy: 92,798%		Error: 7,202%
Cohen's kappa (κ): 0,753%		

TruePositives	FalsePositives	TrueNegatives	FalseNegatives	Recall	Precision	Sensitivity	Specificity	F-measure	Accuracy	Cohen's kappa
21278,00	1818,00	3758,00	125,00	0,99	0,92	0,99	0,67	0,96		
3758,00	125,00	21278,00	1818,00	0,67	0,97	0,67	0,99	0,79		
										0,93
										0,75

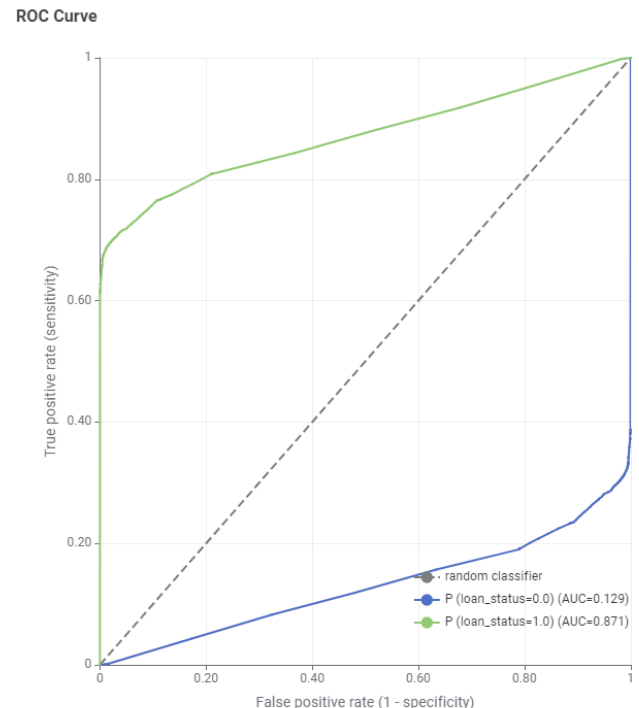
Los resultados obtenidos indican que el modelo de Árbol de Decisión tiene una precisión bastante alta en términos de predicción correcta de ambas clases. La matriz de confusión muestra un número significativo de verdaderos positivos (3758) y verdaderos negativos (21278), con relativamente pocos falsos positivos (125) y falsos negativos (1818). Esto sugiere que el modelo es particularmente eficaz en predecir correctamente las instancias

negativas (es decir, aquellas en las que el cliente probablemente no será aprobado para el crédito).

La precisión es alta en ambas filas, lo cual indica que cuando el modelo predice una aprobación de crédito, suele estar en lo correcto. El valor de sensibilidad (recall) de 0.67 en la fila de abajo implica que, aunque el modelo identifica correctamente una buena cantidad de casos positivos, no todos los positivos son detectados, dejando algunos aprobados sin identificar (falsos negativos). Esto podría ser problemático en un escenario donde perder aprobaciones potenciales tiene un alto costo.

La especificidad (0.99) es alta, lo que indica que el modelo es muy eficiente en la identificación de casos negativos, reduciendo la cantidad de aprobaciones erróneas. Finalmente, la medida F de 0.79 en la segunda fila refleja un equilibrio entre precisión y recall, aunque la sensibilidad baja indica que el modelo podría mejorarse en la detección de casos positivos.

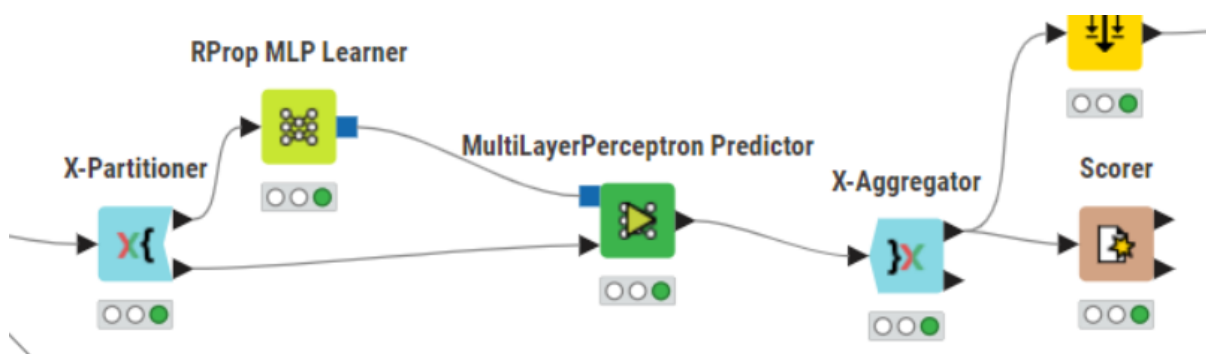
La ROC curve nos indica un AUC=0.129 para la predicción de rechazo de créditos y un AUC=0.871 para la predicción de aprobación de créditos.



Red Neuronal

El proceso de implementación del algoritmo de red neuronal ha seguido una estructura similar a la de otros modelos, comenzando con el preprocesamiento de los datos. Después, para realizar la validación cruzada con 5 particiones, he utilizado el nodo **X-Partitioner** para dividir los datos y el nodo **X-Aggregator** para combinar los resultados, asegurando una validación adecuada en cada partición.

Para entrenar el modelo, he empleado el nodo **RProp MLP Learner**, que es adecuado para redes neuronales multilayer perceptron utilizando el algoritmo de retropropagación (RProp). Finalmente, para evaluar el rendimiento del modelo, he usado el nodo **MultiLayerPerceptron Predictor**, que predice las clases basadas en el modelo entrenado.



Los resultados del nodo **Scorer** muestran un rendimiento bastante bueno en general, pero con áreas para mejorar, especialmente en la predicción de la clase 1.

Confusion Matrix - 3:63 - Scorer		
File	Hilite	
loan_statu...	0.0	1.0
0.0	20769	634
1.0	2781	2795
Correct classified: 23.564		
Wrong classified: 3.415		
Accuracy: 87,342%		
Error: 12,658%		
Cohen's kappa (κ): 0,55%		

TruePositives	FalsePositives	TrueNegatives	FalseNegatives	Recall	Precision	Sensitivity	Specificity	F-measure	Accuracy	Cohen's kappa
20769,00	2781,00	2795,00	634,00	0,97	0,88	0,97	0,50	0,92		
2795,00	634,00	20769,00	2781,00	0,50	0,82	0,50	0,97	0,62		
									0,87	0,55

En la matriz de confusión, vemos que el modelo ha acertado en 20,769 casos de clase 0 y en 2,795 casos de clase 1, pero también ha cometido errores: ha clasificado incorrectamente 634 instancias de clase 0 como clase 1 (falsos positivos) y ha fallado en 2,781 ocasiones al no identificar correctamente las instancias de clase 1 (falsos negativos).

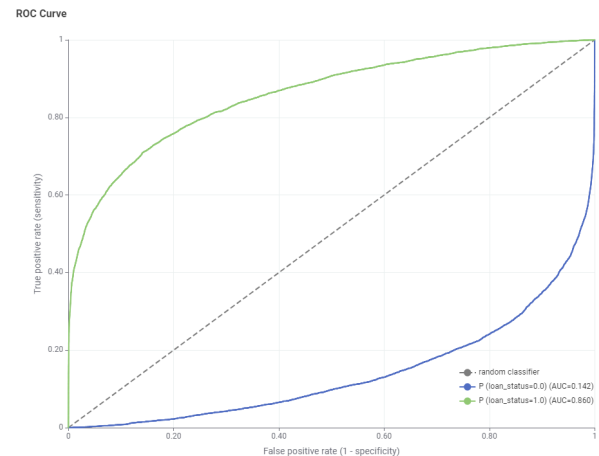
En cuanto a las estadísticas, el modelo muestra un buen desempeño para la clase 0, con un recall de 0.97, lo que significa que detecta el 97% de las instancias de clase 0 correctamente. La precision es 0.88, lo que indica que cuando predice clase 0, la mayoría de las predicciones son correctas. Sin embargo, la specificity de la clase 0 es solo 0.50, lo que sugiere que tiene dificultades para diferenciar entre las clases cuando se trata de predecir la clase 1. La F-measure de la clase 0 es alta (0.92), lo que refleja un buen equilibrio entre recall y precision.

Por otro lado, para la clase 1, el rendimiento no es tan bueno. El recall es solo 0.50, lo que significa que el modelo detecta correctamente solo la mitad de las instancias de clase 1, dejando muchas sin identificar (falsos negativos). La precision de clase 1 es de 0.81, lo que indica que cuando predice clase 1, la predicción es correcta en la mayoría de los casos, pero el bajo recall muestra que pierde muchas instancias de clase 1. La specificity es alta (0.97), lo que indica que el modelo tiene una buena capacidad para predecir correctamente la clase 0. La F-measure para la clase 1 es más baja (0.62), lo que refleja el desbalance entre recall y precision en esta clase.

El accuracy global del modelo es de 87.3%, lo que es bastante bueno, pero el Cohen's Kappa de 0.55 sugiere que, aunque hay una buena concordancia entre las predicciones y

las clases reales, todavía hay margen de mejora, especialmente para la clase 1. En resumen, el modelo es muy efectivo para predecir la clase 0, pero podría mejorarse en la predicción de la clase 1, tal vez explorando técnicas de balanceo de clases o ajustando los umbrales de decisión.

La ROC curve nos indica un AUC=0.142 para la predicción de rechazo de créditos y un AUC=0.860 para la predicción de aprobación de créditos.

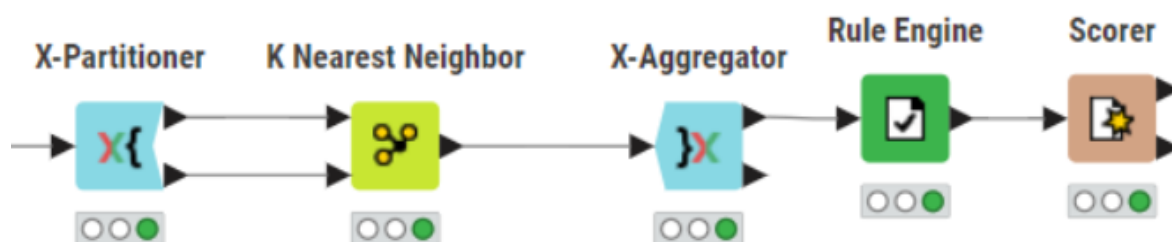


K Nearest Neighbour

Para crear el algoritmo KNN, he empezado con el preprocesamiento común a todos los algoritmos, que incluye la normalización de los datos, la conversión de las variables categóricas a numéricas y el tratamiento de los valores faltantes. Sin embargo, en este caso he añadido un paso extra de preprocesamiento, que fue la selección de características. He usado el nodo **Correlation Filter** y **Linear Correlation** para eliminar las características menos relevantes y reducir el ruido, lo que ayuda a mejorar la precisión del modelo.

Luego, como en los otros algoritmos, he usado la validación cruzada con 5 particiones para evaluar el rendimiento del modelo. En lugar de separar explícitamente los datos de entrenamiento y prueba, pasé ambos, training data y testing data, al nodo **K Nearest Neighbour**. Este nodo está diseñado para realizar tanto el entrenamiento como la validación, por lo que no es necesario dividir los datos de antemano.

Una vez entrenado el modelo, he usado el nodo **X-Aggregator** para combinar las particiones generadas por la validación cruzada. Sin embargo, no se había creado una columna de partición para identificar las predicciones correctas o incorrectas. Para resolver esto, he utilizado el nodo **Rule Engine**, donde he definido una regla que asigna un valor de 1 cuando la predicción del modelo (en la columna **knn**) es igual al valor real de **loan_status** (es decir, cuando la predicción es correcta), y un valor de 0 cuando son diferentes (predicción incorrecta).



Finalmente, pasé los resultados al nodo **Scorer**, que generó las métricas de evaluación, incluyendo la matriz de confusión y las estadísticas de precisión:

loan_statu...	0.0	1.0
0.0	20460	943
1.0	2470	3106

Correct classified: 23.566 Wrong classified: 3.413

Accuracy: 87,349% Error: 12,651%

Cohen's kappa (κ): 0,571%

TruePositives	FalsePositives	TrueNegatives	FalseNegatives	Recall	Precision	Sensitivity	Specificity	F-measure	Accuracy	Cohen's kappa
20460	2470	3106	943	0,96	0,89	0,96	0,56	0,92		
3106	943	20460	2470	0,56	0,77	0,56	0,96	0,65	0,87	0,57

Los resultados del modelo KNN muestran un buen rendimiento general, con algunas áreas de mejora en la predicción de la clase 1. En la matriz de confusión, se observa que el modelo ha clasificado correctamente 20460 casos de la clase 0 y 3106 de la clase 1. Sin embargo, también ha cometido errores, con 943 falsos positivos (clasificando casos de clase 0 como clase 1) y 2470 falsos negativos (no identificando correctamente casos de clase 1).

En cuanto a las estadísticas, para la clase 0, el modelo tiene un recall de 0.956, lo que significa que identifica correctamente el 95.6% de los casos de esta clase, con una precisión de 0.892, lo que indica que cuando predice clase 0, lo hace con bastante precisión. La specificity es 0.557, lo que refleja que el modelo tiene más dificultades para identificar correctamente los casos de clase 1, aunque el sensitivity es alto, indicando que detecta bien la clase 0. La F-measure de 0.923 muestra un buen balance entre recall y precision.

Para la clase 1, el modelo tiene un recall de 0.557, lo que significa que solo identifica correctamente la mitad de los casos de clase 1, lo que es una área de mejora significativa. La precision es más baja, 0.767, lo que indica que aunque el modelo predice clase 1 con razonable acierto, también comete más errores. La specificity es alta (0.956), lo que significa que es muy bueno para predecir clase 0 correctamente. La F-measure de 0.645 refleja un balance inferior entre precisión y recall para la clase 1.

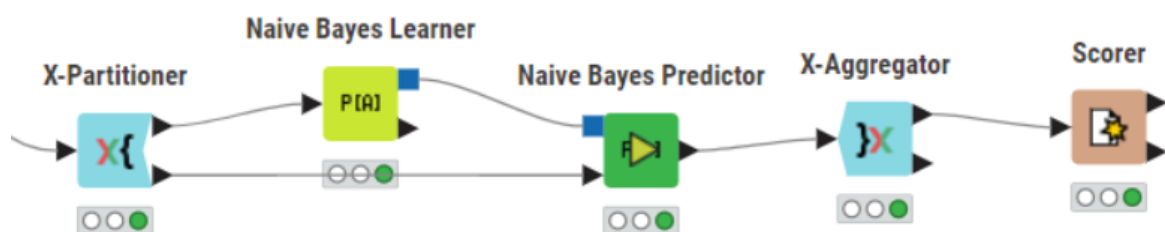
En términos generales, el accuracy del modelo es del 87.3%, lo que es bastante bueno, pero el Cohen's Kappa de 0.571 indica que aunque el modelo tiene una concordancia razonable con las etiquetas reales, todavía hay margen para mejorar, especialmente en la predicción de la clase 1.

Este algoritmo es fácil de entender e interpretar. No requiere entrenamiento explícito, ya que simplemente memoriza los datos de entrenamiento. Sin embargo, al igual que el anterior, requiere mucho tiempo de cómputo, ya que la clasificación se realiza en tiempo real durante la predicción.

En este caso era realmente importante normalizar los valores de las características, ya que es muy sensible a la escala. Además, como ya hemos comentado, es sensible al ruido en los datos.

Naive Bayes

Para implementar el algoritmo de Naive Bayes, utilicé el nodo **Naive Bayes Learner** para entrenar el modelo y el nodo **Naive Bayes Predictor** para evaluarlo, tras realizar el preprocesamiento acompañado de la eliminación de variables correlacionadas, para reducir la dependencia de datos. Naive Bayes es rápido, eficiente y adecuado para grandes volúmenes de datos, especialmente cuando las características son independientes. Sin embargo, su principal limitación es la suposición de independencia entre características, lo que puede afectar su rendimiento si estas están correlacionadas. Además, puede requerir suavizado para manejar valores de probabilidad cero. En resumen, es un buen algoritmo cuando las condiciones son adecuadas, pero tiene algunas restricciones.



Los resultados del nodo **Scorer** son los siguientes:

Confusion Matrix - 3:76 - Scorer				
File Hilit				
loan_statu...	0.0	1.0		
0.0	18968	2435		
1.0	2236	3340		
Correct classified: 22.308			Wrong classified: 4.671	
Accuracy: 82,687%			Error: 17,313%	
Cohen's kappa (κ): 0,479%				

El modelo Naive Bayes muestra un buen rendimiento para predecir la clase positiva ("1"), con un recall y precisión de 0.89, lo que indica que identifica correctamente la mayoría de los casos positivos. Sin embargo, tiene dificultades con la clase negativa ("0"), con un recall

Row ID	TP	FP	TN	FN	Recall	Precision	Sensitivity	Specificity	F-measure	Accuracy	Cohen's kappa
0,00	18968,00	2236,00	3340,00	2435,00	0,89	0,89	0,89	0,60	0,89		
1,00	3340,00	2435,00	18968,00	2236,00	0,60	0,58	0,60	0,89	0,59		
										0,83	0,48

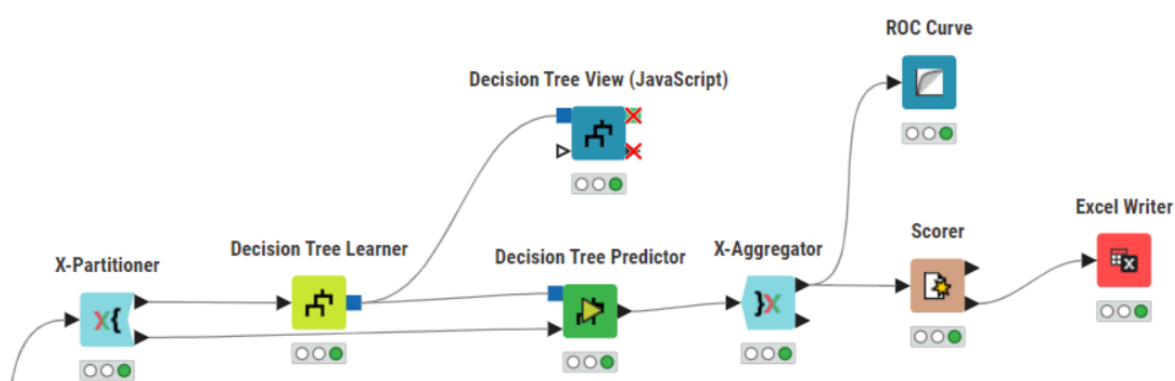
de 0.60 y precisión de 0.58, lo que sugiere que comete más errores al predecir negativos. La accuracy es del 83%, pero el Cohen's Kappa de 0.48 indica que el modelo tiene un rendimiento moderado en comparación con una predicción aleatoria. En resumen, el modelo es efectivo para los casos positivos, pero necesita mejorar en la identificación de los negativos.

El método Naive Bayes es muy rápido de entrenar y ejecutar y, como el primer algoritmo, ofrece probabilidades como resultado, que es útil cuando se necesita una estimación de incertidumbre. Aunque se asume independencia, en la práctica puede funcionar sorprendentemente bien incluso con algunas dependencias entre características, como podría ser nuestro problema.

2. Predicción de segunda cita

Árbol de decisión

En el caso del modelo de Decision Tree, se ha realizado una validación cruzada de 5 particiones, utilizando los nodos **X-Partitioner** y **X-Aggregator**. El proceso de validación cruzada nos permite entrenar y evaluar el modelo de manera más robusta, ya que se entrena en diferentes subconjuntos de datos y se evalúa en otros, lo que da una mejor estimación del rendimiento del modelo en datos no vistos. Los nodos **Decision Tree Learner** y **Decision Tree Predictor** se encargan de entrenar el modelo y predecir las clases para las instancias del conjunto de prueba, respectivamente.



Los resultados del nodo **Scorer** para la clasificación del modelo **Decision Tree** muestran una serie de métricas que nos permiten evaluar la calidad del modelo.

Confusion Matrix - 5:115 - Scorer

match \ Pr...	0	1
0	2240	366
1	425	2181

Correct classified: 4.421 Wrong classified: 791
 Accuracy: 84,823% Error: 15,177%
 Cohen's kappa (κ): 0,696%

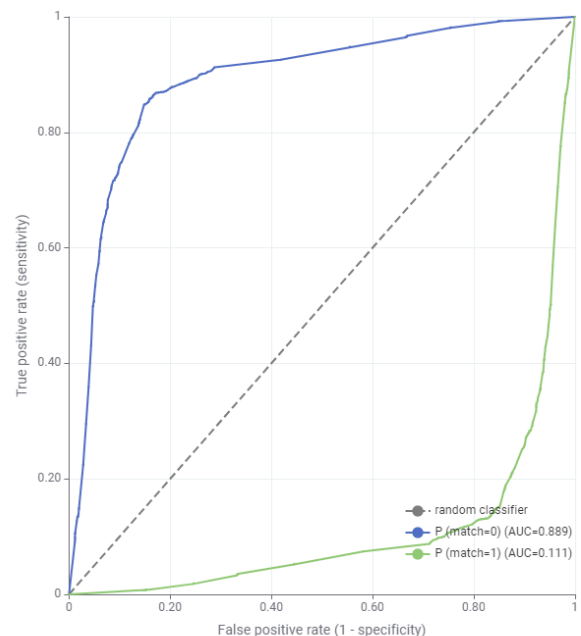
TP	FP	TN	FN	Recall	Precision	Sensitivity	Specificity	F-measure	Accuracy	Cohen's kappa
2240	425	2181	366	0,86	0,84	0,86	0,84	0,85	?	?
2181	366	2240	425	0,84	0,86	0,84	0,86	0,85	?	?
?	?	?	?	?	?	?	?	?	0,85	0,70

En términos de recall (0.86), el modelo tiene una alta capacidad para identificar correctamente las instancias positivas (match=1). Esto significa que el modelo no está dejando escapar muchos casos en los que la clase debería haber sido positiva, lo cual es una buena señal. La precision (0.84) indica que, cuando el modelo predice que un caso es positivo, tiene una probabilidad bastante alta de que sea realmente positivo. Además, la sensitivity y specificity también son buenas, con valores similares (0.86 y 0.84, respectivamente), lo que sugiere que el modelo está equilibrado en cuanto a su capacidad para detectar tanto los casos positivos como negativos.

La F-measure (0.85) es una media armónica entre precision y recall, y un valor cercano a 1 indica que el modelo tiene un buen equilibrio entre estos dos factores. Finalmente, el Cohen's kappa es de 0.70, lo que indica una buena concordancia entre las predicciones del modelo y las etiquetas reales, tomando en cuenta la posibilidad de que las predicciones hayan sido hechas al azar.

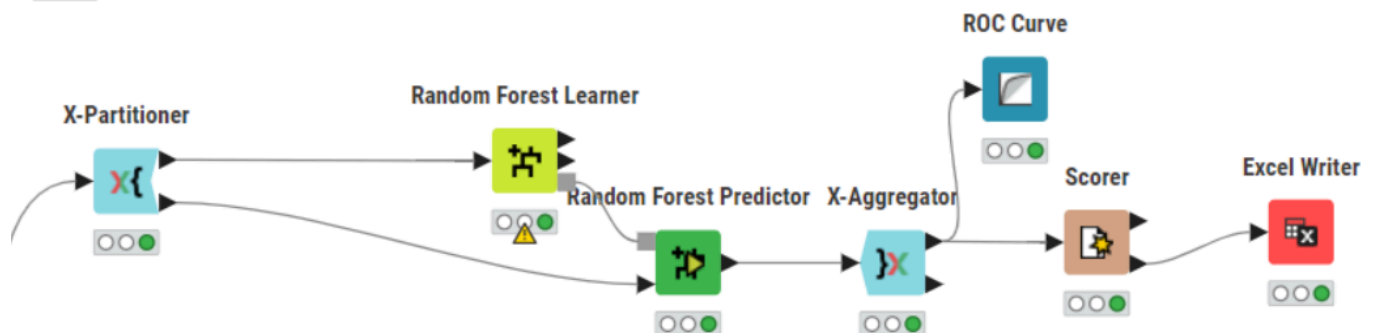
Finalmente, al mirar la ROC curve y los AUC (área bajo la curva), vemos que el AUC para $P(\text{match}=0)$ es 0.889 y el AUC para $P(\text{match}=1)$ es 0.111, lo que indica que el modelo tiene una excelente capacidad para distinguir entre los casos negativos (no match) y los casos positivos (matches).

ROC Curve



Random Forest

En el modelo de Random Forest, también he realizado una validación cruzada de 5 particiones utilizando los nodos **Random Forest Learner** y **Random Forest Predictor** para entrenar y evaluar el modelo.



Los resultados obtenidos del nodo **Scorer** muestran un rendimiento impresionante en general.

Confusion Matrix - 5:121 - Scorer				
File Hilite				
match \Pr...	0	1		
0	2445	161		
1	117	2489		
Correct classified: 4.934			Wrong classified: 278	
Accuracy: 94,666%			Error: 5,334%	
Cohen's kappa (κ): 0,893%				

TP	FP	TN	FN	Recall	Precision	Sensitivity	Specificity	F-measure	Accuracy	Cohen's kappa
2445,00	117,00	2489,00	161,00	0,94	0,95	0,94	0,96	0,95	?	?
2489,00	161,00	2445,00	117,00	0,96	0,94	0,96	0,94	0,95	?	?
?	?	?	?	?	?	?	?	?	0,95	0,89

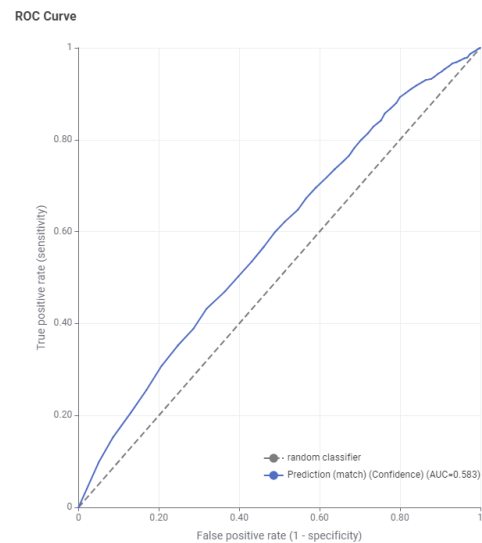
Al observar las métricas de rendimiento, el modelo tiene una alta cantidad de True Positives (2445 y 2489) y una cantidad relativamente baja de False Positives (117 y 161). Esto indica que el modelo es bastante eficiente en identificar correctamente los matches, con un buen equilibrio en las predicciones positivas. La Recall o sensibilidad es sobresaliente, alcanzando valores de 0.94 y 0.96, lo que significa que el modelo identifica la mayoría de los verdaderos matches, lo cual es crucial en este tipo de problema.

La Precision también es alta, con valores de 0.95 y 0.94, lo que indica que, cuando el modelo predice un match, tiene una alta probabilidad de que efectivamente sea un match. La Specificity es igualmente buena, con valores de 0.96 y 0.94, lo que sugiere que el

modelo también es bastante preciso al identificar los casos negativos (no match). La F-measure de 0.95 es un buen indicador de que el modelo está equilibrando correctamente la sensibilidad y la precisión, lo que es clave para obtener un modelo robusto.

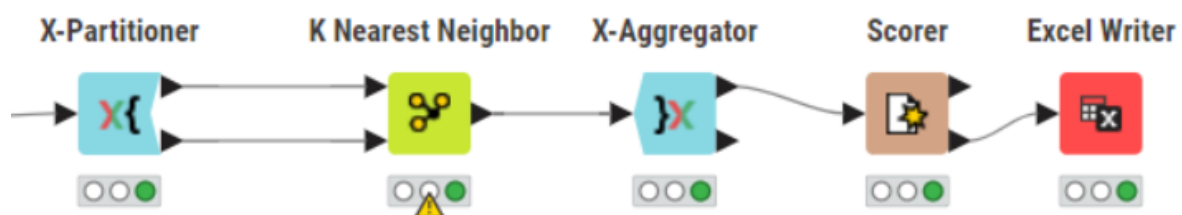
La accuracy es alta, con un 0.95 y el Cohen's kappa es de 0.89 en la última partición, lo que indica una excelente concordancia entre las predicciones del modelo y los valores reales, un valor muy sólido que refleja un buen rendimiento general.

En cuanto a la ROC curve y el AUC (área bajo la curva), el AUC para la predicción de match es 0.583. Este AUC es relativamente bajo y sugiere que, aunque el modelo tiene un buen rendimiento general, no es tan efectivo al distinguir entre las clases cuando se trata de predecir los casos positivos (matches). Un AUC cercano a 0.5 indicaría un rendimiento aleatorio, y aunque 0.583 es superior a eso, podría mejorarse.



K-Nearest NeighBor

En el caso del modelo KNN (K-Nearest Neighbors), también realicé una validación cruzada de 5 particiones utilizando los nodos **KNN Learner** y **KNN Predictor** para entrenar y evaluar el modelo.



Los resultados obtenidos del nodo **Scorer** revelan un rendimiento interesante y, en ciertos aspectos, muy positivo.

Confusion Matrix - 5:134 - Scorer		
File Hilite		
match \ Cl...	0	1
0	1833	773
1	30	2576
Correct classified: 4.409		Wrong classified: 803
Accuracy: 84,593%		Error: 15,407%
Cohen's kappa (κ): 0,692%		

Algorithm	RowID	TP	FP	TN	FN	Recall	Precision	Sensitivity	Specificity	F-measure	Accuracy	Cohen's kappa
KNN	0	1833	30	2576	773	0,70	0,98	0,70	0,99	0,82	?	?
KNN	1	2576	773	1833	30	0,99	0,77	0,99	0,70	0,87	?	?
KNN	?	?	?	?	?	?	?	?	?	?	0,85	0,69

La accuracy general del modelo es de 0.85, lo que indica que el modelo tiene un rendimiento bastante bueno en general, clasificando correctamente el 85% de los casos. Sin embargo, al observar las métricas de True Positives (TP), True Negatives (TN), False Positives (FP) y False Negatives (FN), podemos ver que hay un desbalance entre los resultados de las distintas particiones.

En la primera partición, el modelo tiene un Recall (sensibilidad) de 0.70, lo que significa que identifica correctamente el 70% de los casos positivos (matches). Esta cifra es relativamente baja, lo que sugiere que el modelo podría estar perdiendo muchos de los verdaderos positivos. Sin embargo, en la segunda partición, la Recall se incrementa considerablemente a 0.99, lo que implica que el modelo es muy efectivo en la identificación de matches en esa partición específica.

La Precision es de 0.98 en la primera partición, lo que indica que cuando el modelo predice un match, tiene una probabilidad muy alta de que esa predicción sea correcta. No obstante, en la segunda partición, la Precision disminuye a 0.77, lo que sugiere que, aunque la tasa de Recall es alta, el modelo también está clasificando incorrectamente más casos como positivos, aumentando los falsos positivos.

El F-measure en la primera partición es 0.82 y en la segunda 0.87, lo que muestra un buen equilibrio entre Recall y Precision, lo cual es positivo para el modelo en general.

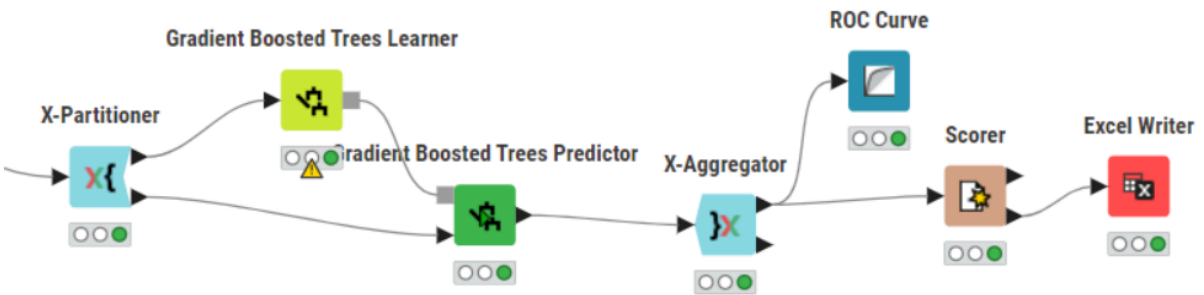
La Specificity, que mide la capacidad del modelo para identificar correctamente los negativos, es muy alta en la primera partición (0.99) y disminuye a 0.70 en la segunda, lo que sugiere que el modelo tiene un rendimiento desigual en diferentes particiones.

Finalmente, la Cohen's Kappa es de 0.69, lo que indica una buena concordancia entre las predicciones del modelo y los valores reales, aunque no perfecta. Esto sugiere que el modelo es razonablemente efectivo, pero podría beneficiarse de ajustes adicionales.

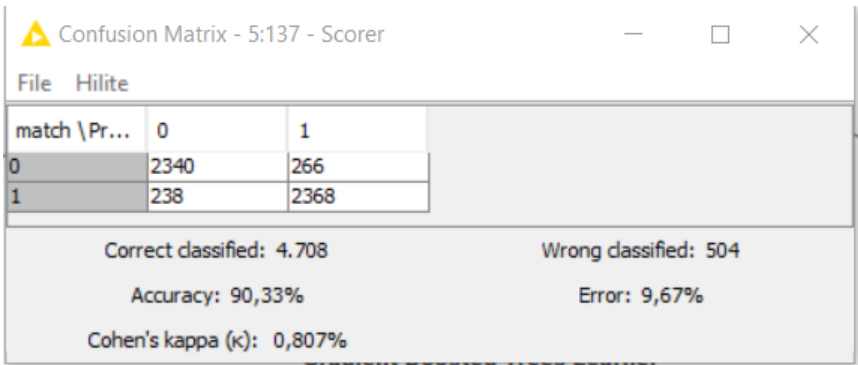
En resumen, el modelo KNN tiene un rendimiento sólido con una accuracy de 0.85, pero la gran variabilidad en las métricas de las distintas particiones indica que puede haber algún sobreajuste o que el modelo está siendo influenciado por características específicas de los datos.

Gradient Boosted Trees

En el caso del modelo Gradient Boosted Trees (GBT), también se ha realizado una validación cruzada de 5 particiones utilizando los nodos **Gradient Boosted Trees Learner** y **Gradient Boosted Trees Predictor** para entrenar y evaluar el modelo.



Los resultados obtenidos del nodo **Scorer** muestran un rendimiento global muy positivo.



TP	FP	TN	FN	Recall	Precision	Sensitivity	Specificity	F-measure	Accuracy	Cohen's kappa
2340	238	2368	266	0,90	0,91	0,90	0,91	0,90	?	?
2368	266	2340	238	0,91	0,90	0,91	0,90	0,90	?	?
?	?	?	?	?	?	?	?	?	0,90	0,81

La accuracy global del modelo es de 0.90, lo que muestra que el modelo tiene un alto nivel de precisión general, clasificando correctamente el 90% de los casos. Este es un buen resultado y sugiere que el modelo es bastante efectivo en la predicción de la variable de interés.

En cuanto a las métricas de True Positives (TP), True Negatives (TN), False Positives (FP) y False Negatives (FN), podemos observar que el modelo ha logrado un Recall de 0.90 y 0.91 en las dos particiones, lo que significa que el modelo ha sido capaz de identificar correctamente entre el 90% y el 91% de los casos positivos (matches). Este es un buen

indicador de que el modelo está capturando una gran cantidad de los casos positivos y no está dejando muchos casos importantes fuera.

La Precision es también bastante alta, con valores de 0.91 y 0.90 en las dos particiones. Esto indica que cuando el modelo predice un match, tiene una probabilidad muy alta de que esta predicción sea correcta, lo que es positivo en términos de reducir los falsos positivos.

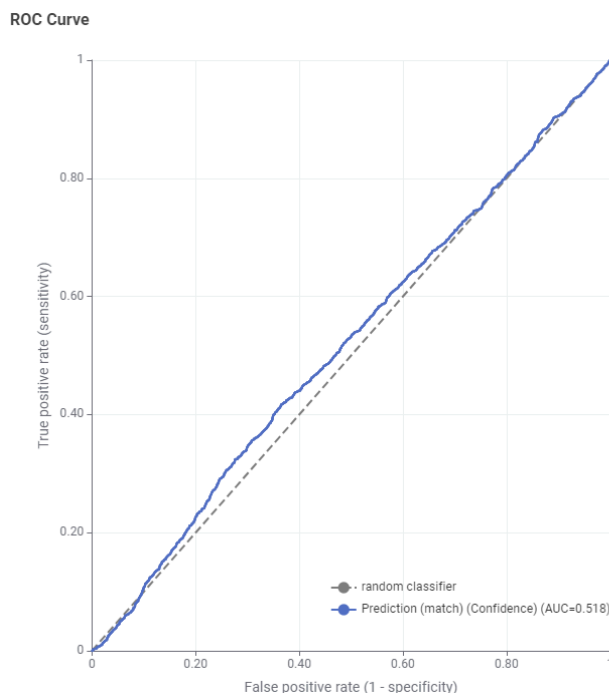
El F-measure es de 0.90 en ambas particiones, lo que significa que hay un buen balance entre la Precision y el Recall, una característica deseable en los modelos de clasificación.

La Specificity, que mide la capacidad del modelo para identificar correctamente los negativos, es de 0.91 en ambas particiones, lo que indica que el modelo es efectivo tanto para los casos positivos como para los negativos, manteniendo un buen equilibrio en su rendimiento.

El Cohen's Kappa es de 0.81, lo que refleja una excelente concordancia entre las predicciones del modelo y los valores reales, indicando un rendimiento muy sólido.

La curva ROC muestra un valor de $AUC = 0.518$ para la predicción de la clase positiva (match). Este valor es bastante bajo y podría indicar que la curva está calculándose de forma incorrecta, ya que un AUC tan bajo sugiere que el modelo no está diferenciando bien entre las clases positivas y negativas. Es recomendable revisar la implementación de la

curva ROC y verificar si los valores de predicción están correctamente calculados.



Regresión Logística

Para el modelo de Regresión Logística, realicé una validación cruzada de 5 particiones con los nodos **Logistic Regression Learner** y **Logistic Regression Predictor**

Confusion Matrix - 5:151 - Scorer

match \Pr...	0	1
0	2002	604
1	476	2130

Correct classified: 4.132 Wrong classified: 1.080

Accuracy: 79,279% Error: 20,721%

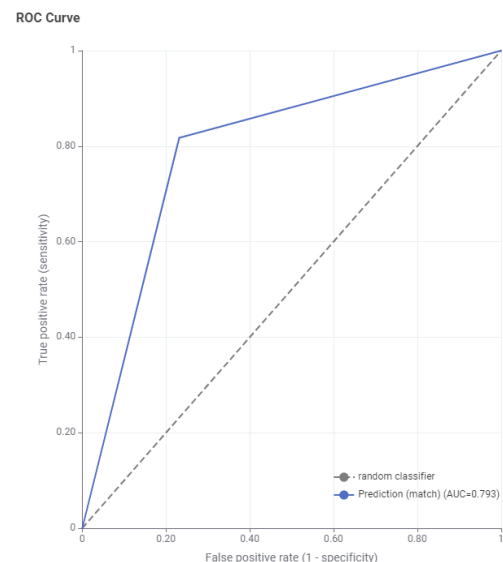
Cohen's kappa (κ): 0,586%

[illegible]

La Recall es de 0.77 en una partición y de 0.82 en otra, indicando que el modelo identifica correctamente entre el 77% y el 82% de los casos positivos. La Precision es de 0.81 y 0.78 en las particiones, lo que significa que cuando el modelo predice un match, entre el 78% y el 81% de estas predicciones son correctas. La F-measure, que combina Precision y Recall, es de 0.79 y 0.80 en las particiones. Estos valores reflejan un buen equilibrio en las predicciones positivas y negativas, pero están por debajo de otros algoritmos, lo que indica que el modelo de regresión logística está algo limitado en este contexto.

El Cohen's Kappa, de 0.59, muestra una moderada concordancia entre las predicciones del modelo y los valores reales. Aunque refleja cierta precisión, también sugiere que el modelo no captura completamente la complejidad de los datos.

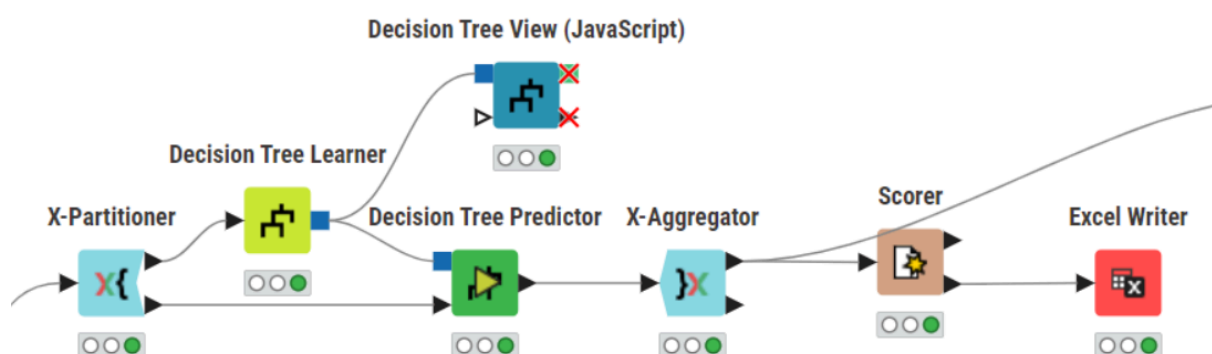
Finalmente, la curva ROC muestra un valor de AUC de 0.793, lo cual es un indicador positivo de la capacidad del modelo para diferenciar entre clases positivas y negativas. Sin embargo, este AUC sigue siendo inferior al obtenido con algunos de los otros modelos, sugiriendo que la regresión logística puede estar limitada por la linealidad asumida en las relaciones entre las características.



3. Predicción de enfermedades eritemato-escamosas

Árbol de decisión

En el modelo de Árbol de Decisión, se utilizó validación cruzada de 5 particiones, aplicando el **Decision Tree Learner** y su predictor.



Los resultados indican una precisión promedio adecuada, aunque algo inferior en comparación con otros modelos como la Red Neuronal MLP. La precisión (precision) varía entre 0,71 y 0,94, reflejando fluctuaciones en la identificación de verdaderos positivos. La sensibilidad (recall) también muestra variación, oscilando entre 0,71 y 0,97, lo cual indica

que el modelo tiene un desempeño algo inconsistente en la detección de positivos en diferentes particiones.

Confusion Matrix - 3:115 - Scorer

File Hilite

class \ Pre...	2	1	3	5	4	6
2	80	0	19	3	5	5
1	0	109	0	3	0	0
3	2	0	104	1	4	1
5	0	3	3	106	0	0
4	4	0	17	0	90	1
6	1	3	2	0	0	106

Correct classified: 595 Wrong classified: 77

Accuracy: 88,542% Error: 11,458%

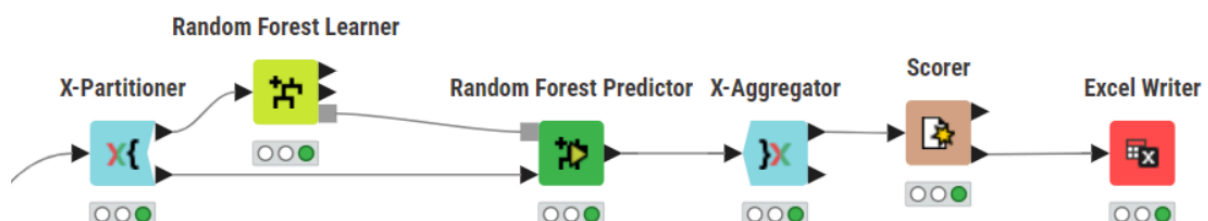
Cohen's kappa (κ): 0,863%

Algorithm	RowID	TP	FP	TN	FN	Recall	Precision	Sensitivity	Specificity	F-measure	Accuracy	Cohen's kappa
Decision Tree	2	80	7	553	32	0,7143	0,9195	0,7143	0,9875	0,8040		
Decision Tree	1	109	6	554	3	0,9732	0,9478	0,9732	0,9893	0,9604		
Decision Tree	3	104	41	519	8	0,9286	0,7172	0,9286	0,9268	0,8093		
Decision Tree	5	106	7	553	6	0,9464	0,9381	0,9464	0,9875	0,9422		
Decision Tree	4	90	9	551	22	0,8036	0,9091	0,8036	0,9839	0,8531		
Decision Tree	6	106	7	553	6	0,9464	0,9381	0,9464	0,9875	0,9422		
Decision Tree											0,8854	0,8625

El accuracy promedio se sitúa en 0,89, mientras que el índice de Cohen's Kappa es de 0,86, lo que sugiere un nivel de acuerdo razonablemente alto entre el modelo y las etiquetas reales, pero menos robusto en comparación con otros algoritmos. En conclusión, aunque el modelo de Árbol de Decisión logra un rendimiento razonable en general, su desempeño presenta más variabilidad y es menos consistente que el modelo de Red Neuronal MLP, en especial en la precisión y sensibilidad en varias particiones.


Random Forest

Para el modelo de Random Forest, se aplicó una validación cruzada con 5 particiones, utilizando el **Random Forest Learner** y su predictor.



Los resultados muestran un rendimiento consistente y alto en precisión, sensibilidad, y métricas generales. La precisión del modelo oscila entre 0,91 y 1,00, lo que indica que Random Forest identifica correctamente la mayoría de los verdaderos positivos sin exceso de falsos positivos. La sensibilidad es igualmente alta, con valores entre 0,93 y 1,00, lo que

refleja una fuerte capacidad del modelo para detectar verdaderos positivos de forma consistente en todas las particiones.


Confusion Matrix - 3:121 - Scorer

—
□
✕

File Hilite

class \ Pre...	2	1	3	5	4	6
2	104	0	5	2	0	1
1	0	111	0	1	0	0
3	2	0	108	0	2	0
5	0	0	1	111	0	0
4	1	0	5	0	106	0
6	0	0	0	0	0	112

Correct classified: 652

Accuracy: 97,024%

Cohen's kappa (κ): 0,964%

Wrong classified: 20

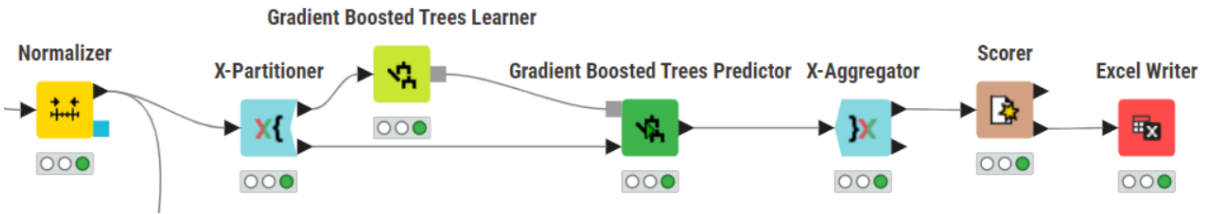
Error: 2,976%

Algorithm	RowID	TP	FP	TN	FN	Recall	Precision	Sensitivity	Specificity	F-measure	Accuracy	Cohen's kappa
Random Forest	2	104	3	557	8	0,9286	0,9720	0,9286	0,9946	0,9498		
Random Forest	1	111	0	560	1	0,9911	1,0000	0,9911	1,0000	0,9955		
Random Forest	3	108	11	549	4	0,9643	0,9076	0,9643	0,9804	0,9351		
Random Forest	5	111	3	557	1	0,9911	0,9737	0,9911	0,9946	0,9823		
Random Forest	4	106	2	558	6	0,9464	0,9815	0,9464	0,9964	0,9636		
Random Forest	6	112	1	559	0	1,0000	0,9912	1,0000	0,9982	0,9956		
Random Forest											0,9702	0,9643

El accuracy promedio de este modelo es de 0,97, mientras que el índice de Cohen’s Kappa alcanza 0,96, lo que muestra un excelente nivel de acuerdo entre las predicciones del modelo y las etiquetas reales. Estos resultados reflejan la robustez de Random Forest para clasificar correctamente, y la alta precisión en distintas métricas confirma que este algoritmo maneja tanto la variabilidad como la complejidad de los datos mejor que otros modelos.

Gradient Boosted Trees

Para el modelo de Gradient Boosted Trees, se utilizó la validación cruzada con 5 particiones, empleando los nodos de aprendizaje y predicción específicos para este algoritmo.



Los resultados obtenidos reflejan un rendimiento sólido en términos de precisión y sensibilidad, mostrando consistencia en todas las particiones. La precisión varía entre 0,93 y 0,99, lo que indica un buen control sobre los falsos positivos, mientras que la sensibilidad

se mantiene alta, entre 0,92 y 1,00, lo que demuestra una capacidad efectiva para identificar verdaderos positivos en la mayoría de los casos.

Confusion Matrix - 3:137 - Scorer

File	Hilite					
class \Pre...	2	1	3	5	4	6
2	103	0	5	1	2	1
1	0	108	0	3	0	1
3	2	0	104	1	5	0
5	1	1	2	108	0	0
4	3	0	1	0	108	0
6	0	0	0	0	0	112

Correct classified: 643

Wrong classified: 29

Accuracy: 95,685%

Error: 4,315%

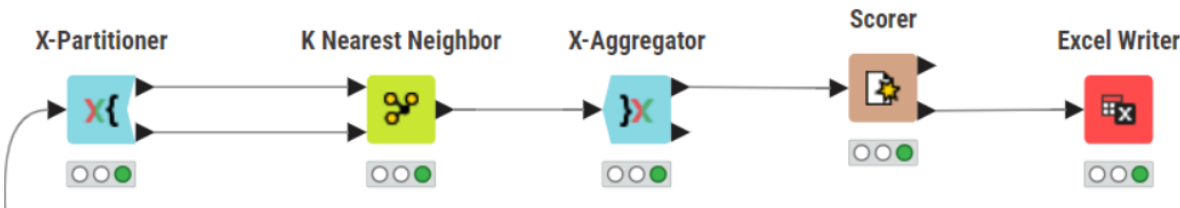
Cohen's kappa (κ): 0,948%

Algorithm	RowID	TP	FP	TN	FN	Recall	Precision	Sensitivity	Specificity	F-measure	Accuracy	Cohen's kappa
Gradient Boosted Trees	2	103	6	554	9	0,9196	0,9450	0,9196	0,9893	0,9321		
Gradient Boosted Trees	1	108	1	559	4	0,9643	0,9908	0,9643	0,9982	0,9774		
Gradient Boosted Trees	3	104	8	552	8	0,9286	0,9286	0,9286	0,9857	0,9286		
Gradient Boosted Trees	5	108	5	555	4	0,9643	0,9558	0,9643	0,9911	0,9600		
Gradient Boosted Trees	4	108	7	553	4	0,9643	0,9391	0,9643	0,9875	0,9515		
Gradient Boosted Trees	6	112	2	558	0	1,0000	0,9825	1,0000	0,9964	0,9912		
Gradient Boosted Trees											0,9568	0,9482

El modelo logra un accuracy promedio de 0,96, y un índice de Cohen's Kappa de 0,95, lo que indica un nivel de acuerdo considerable entre las predicciones del modelo y las etiquetas reales. Estos valores muestran que Gradient Boosted Trees es un modelo adecuado para este conjunto de datos, ya que balancea bien la precisión y la sensibilidad. En resumen, Gradient Boosted Trees proporciona un rendimiento estable y confiable, logrando clasificaciones precisas y coherentes en diferentes particiones del conjunto de datos.

K-Nearest Neighbor

En el modelo de K-Nearest Neighbors (KNN), se implementó la validación cruzada de 5 particiones utilizando los nodos de aprendizaje y predicción de KNN, y los resultados obtenidos muestran un rendimiento adecuado en términos de precisión y sensibilidad, aunque con cierta variabilidad entre particiones.



La precisión oscila entre 0,91 y 1,00, lo que indica una buena capacidad para minimizar falsos positivos, mientras que la sensibilidad (recall) varía entre 0,87 y 1,00, destacando un nivel sólido en la identificación de verdaderos positivos.

Confusion Matrix - 3:161 - Scorer

File Hilite

class \ Clas...	2	1	3	5	4	6
2	96	0	6	1	7	0
1	0	104	0	5	0	2
3	0	0	105	2	3	0
5	0	0	2	89	0	0
4	4	0	2	0	100	1
6	0	0	0	0	0	112

Correct classified: 606 Wrong classified: 35

Accuracy: 94,54% Error: 5,46%

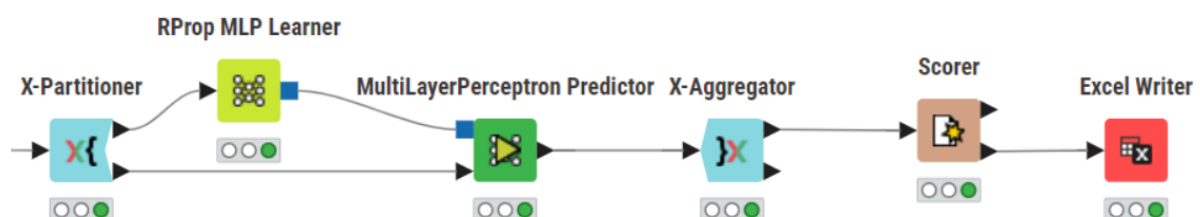
Cohen's kappa (κ): 0,934%

Algorithm	RowID	TP	FP	TN	FN	Recall	Precision	Sensitivity	Specificity	F-measure	Accuracy	Cohen's kappa
KNN	2	96	4	527	14	0,8727	0,9600	0,8727	0,9925	0,9143		
KNN	1	104	0	530	7	0,9369	1,0000	0,9369	1,0000	0,9674		
KNN	3	105	10	521	5	0,9545	0,9130	0,9545	0,9812	0,9333		
KNN	5	89	8	542	2	0,9780	0,9175	0,9780	0,9855	0,9468		
KNN	4	100	10	524	7	0,9346	0,9091	0,9346	0,9813	0,9217		
KNN	6	112	3	526	0	1,0000	0,9739	1,0000	0,9943	0,9868		
KNN											0,9454	0,9344

El modelo alcanzó un accuracy promedio de 0,95 y un índice de Cohen's Kappa de 0,93, lo que refleja un alto grado de concordancia entre las predicciones y las etiquetas reales. Estos valores sugieren que KNN es adecuado para este conjunto de datos y logra un buen equilibrio entre precisión y sensibilidad. En general, KNN demuestra ser eficaz en la clasificación de las instancias, con un rendimiento estable y fiable a lo largo de las particiones del conjunto de datos.


Red Neuronal MLP

En el modelo de Red Neuronal Multicapa (MLP), se ha usado una validación cruzada de 5 particiones, obteniendo buenos resultados en todas las métricas de evaluación.



La precisión se encuentra en un rango entre 0,92 y 0,99, lo que indica una gran capacidad del modelo para predecir correctamente los verdaderos positivos y evitar falsos positivos. La

sensibilidad varía entre 0,91 y 1,00, reflejando su capacidad de identificar los casos positivos con muy pocos errores.


Confusion Matrix - 3:166 - Scorer

File

Hilite

class \ Pre...	2	1	3	5	4	6
2	104	1	2	0	2	1
1	1	107	2	1	0	0
3	5	0	100	1	4	0
5	0	1	0	90	0	0
4	3	0	3	0	101	0
6	0	0	0	0	0	112

Correct classified: 614

Wrong classified: 27

Accuracy: 95,788%

Error: 4,212%

Cohen's kappa (κ): 0,949%

Algorithm	RowID	TP	FP	TN	FN	Recall	Precision	Sensitivity	Specificity	F-measure	Accuracy	Cohen's kappa
MLP	2	104	9	522	6	0,9455	0,9204	0,9455	0,9831	0,9327		
MLP	1	107	2	528	4	0,9640	0,9817	0,9640	0,9962	0,9727		
MLP	3	100	7	524	10	0,9091	0,9346	0,9091	0,9868	0,9217		
MLP	5	90	2	548	1	0,9890	0,9783	0,9890	0,9964	0,9836		
MLP	4	101	6	528	6	0,9439	0,9439	0,9439	0,9888	0,9439		
MLP	6	112	1	528	0	1,0000	0,9912	1,0000	0,9981	0,9956		
MLP											0,9579	0,9494

El accuracy promedio de la MLP es de 0,96, y el índice de Cohen's Kappa es de 0,95, lo que indica una fuerte concordancia entre las predicciones del modelo y las etiquetas reales, superando a otros algoritmos en consistencia. La F-measure, que se encuentra entre 0,92 y 0,99, muestra que el modelo logra un equilibrio óptimo entre precisión y sensibilidad. En conjunto, estos resultados reflejan que la Red Neuronal MLP es un modelo muy efectivo para esta tarea, con un rendimiento sobresaliente en la clasificación precisa y fiable de las instancias del conjunto de datos.

Configuración de algoritmos

1. Predicción de aprobación de créditos

Regresión Logística

En el caso del algoritmo de regresión logística podemos comparar dos configuraciones distintas tuyas. La primera de ellas (apartado de resultados obtenidos) usa Stochastic Average Gradient que optimiza el gradiente de forma eficiente, adecuado para conjuntos de datos grandes y modelos que necesitan rapidez en la clasificación. La segunda configuración usa Iteratively Reweighted Least Squares (IRLS), permite un ajuste más preciso, optimizando los pesos iterativamente.

Algorithm	Row ID	TP	FP	TN	FN	Recall	Precisin	Senstivty	Spcfty	F-mre	Accurcy	Cohen's kappa
Configuración I		20451	3387	2189	952	0,96	0,86	0,96	0,39	0,90		
Configuración I	0	2189	952	20451	3387	0,39	0,70	0,39	0,96	0,50		
Configuración I	1										0,84	0,42
Configuración II	0	20444	3369	2207	959	0,96	0,86	0,96	0,40	0,90		
Configuración II	1	2207	959	20444	3369	0,40	0,70	0,40	0,96	0,51		
Configuración II											0,84	0,42

IRLS mejora levemente el ajuste del modelo sin comprometer significativamente la eficiencia, siendo una buena elección para conjuntos de datos con clases desbalanceadas o ruido. Sin embargo, Stochastic Average Gradient es una opción sólida para cuando se requiere mayor velocidad de clasificación.

Árbol de decisión

Para el algoritmo de árbol de decisión hemos hecho hasta cuatro configuraciones distintas. La original (apartado de Obtención de resultados) usaba 5 como número mínimo de records por nodo y usaba Gain Ratio como medida de calidad.

La segunda configuración usa 1 como mínimo número de registros por nodo, esto ha permitido al árbol profundizar más en los datos, logrando así una mayor sensibilidad (recall) y precisión (f-measure) para la clase positiva, con una precisión general del 92.6%. Sin embargo, este ajuste también incrementa el riesgo de sobreajuste, ya que el árbol ahora es capaz de adaptarse a detalles específicos de los datos de entrenamiento que pueden no generalizar bien en nuevos datos.

La tercera configuración usa 5 como mínimo número de registros por nodo y Gini Index como medida de calidad. Usar el gini index maximiza la pureza en cada nodo, capturando de forma precisa las variaciones en los datos.

La cuarta configuración usa 15 como número mínimo de records por nodo y también Gini Index. Al aumentar el número mínimo de registros a 15, el árbol realiza menos divisiones y generaliza mejor. Esta configuración es más robusta y menos susceptible a captar ruido, a costa de una ligera reducción en la capacidad de captar patrones muy específicos.

Algorithm	Row ID	TP	FP	TN	FN	Recall	Precisin	Senstivty	Spcfty	F-mre	Accuracy	Cohen's kappa
Configuración I	0	21278	1818	3758	125	0,99	0,92	0,99	0,67	0,96		
Configuración I	1	3758	125	21278	1818	0,67	0,97	0,67	0,99	0,79		
Configuración I											0,93	0,75
Configuración II	0	21271	1857	3719	132	0,99	0,92	0,99	0,67	0,96		
Configuración II	1	3719	132	21271	1857	0,67	0,97	0,67	0,99	0,79		
Configuración II											0,93	0,75
Configuración III	0	21277	1797	3779	126	0,99	0,92	0,99	0,68	0,96		
Configuración III	1	3779	126	21277	1797	0,68	0,97	0,68	0,99	0,80		
Configuración III											0,93	0,76
Configuración IV	0	21268	1835	3741	135	0,99	0,92	0,99	0,67	0,96		
Configuración IV	1	3741	135	21268	1835	0,67	0,97	0,67	0,99	0,79		
Configuración IV											0,93	0,75

El gini index con 5 registros mínimos proporciona el ajuste más detallado, aunque se enfrenta a un riesgo de sobreajuste.

Red Neuronal MLP

La primera configuración (la original) de este algoritmo usaba 2 hidden layers y 10 hidden neurons per layer. Esta configuración permite captar patrones en los datos, aunque la cantidad de neuronas es limitada. Esto puede restringir la capacidad de la red para captar variabilidad compleja en los datos, especialmente en problemas con más detalles.

La segunda configuración varía a 5 hidden layers. Incrementar el número de capas aumenta la profundidad del modelo, permitiéndole captar patrones complejos de forma más efectiva. Sin embargo, esto también incrementa el riesgo de sobreajuste, ya que la red tiene más parámetros para ajustar en función de los datos de entrenamiento.

La tercera configuración usa 2 hidden layers, como en la original, pero aumenta a 20 el número de hidden neurons per layer. Permite a la red captar patrones detallados sin necesidad de más profundidad. Aunque mejora la capacidad de clasificación, el riesgo de sobreajuste también aumenta debido a la mayor capacidad de ajuste de la red, especialmente si los datos contienen ruido.

Algorithm	Row ID	TP	FP	TN	FN	Recall	Precisin	Sensitivty	Spcfty	F-mre	Accurcy	Cohen's kappa
Configuración I	0	20769	2781	2795	634	0,97	0,88	0,97	0,50	0,92		
Configuración I	1	2795	634	20769	2781	0,50	0,82	0,50	0,97	0,62		
Configuración I											0,87	0,55
Configuración II	0	20759	2846	2730	644	0,97	0,88	0,97	0,49	0,92		
Configuración II	1	2730	644	20759	2846	0,49	0,81	0,49	0,97	0,61		
Configuración II											0,87	
Configuración III	0	20700	2502	3074	703	0,97	0,89	0,97	0,55	0,93		
Configuración III	1	3074	703	20700	2502	0,55	0,81	0,55	0,97	0,66		
Configuración III											0,88	0,59

Aumentar las capas y las neuronas mejora la capacidad de la red para captar patrones complejos, aunque puede llevar a un sobreajuste.

K-Nearest Neighbors

La configuración original usa 3 vecinos, mientras que la nueva configuración usa 5 vecinos. Al aumentar el número de vecinos de 3 a 5, el modelo se vuelve menos sensible a outliers y reduce un poco su varianza, mejorando su precisión general hasta 0.89 y aumentando tanto la sensibilidad como la especificidad.

Algorithm	Row ID	TP	FP	TN	FN	Recall	Precisin	Sensitivty	Spcfty	F-mre	Accurcy	Cohen's kappa
Configuración I	0	20489	2308	3268	914	0,96	0,90	0,96	0,59	0,93		
Configuración I	1	3268	914	20489	2308	0,59	0,78	0,59	0,96	0,67		
Configuración I											0,88	0,60
Configuración II	0	20771	2424	3152	632	0,97	0,90	0,97	0,57	0,93		
Configuración II	1	3152	632	20771	2424	0,57	0,83	0,57	0,97	0,67		
Configuración II											0,89	0,61

Con más vecinos, el modelo basa sus decisiones en un contexto más amplio, lo que hace que sea menos probable que responda a valores atípicos. Este cambio aumenta la estabilidad del modelo y reduce el riesgo de sobreajuste, aunque a costa de un posible menor ajuste a la particularidad de ciertos casos complejos.

Análisis de resultados

1. Predicción de aprobación de créditos

He reunido los resultados de cada algoritmo en KNIME mediante el nodo **Concatenate** para combinar las métricas generadas por el **Scorer**, como precisión, recall, sensibilidad, especificidad, F-measure, accuracy y Cohen's kappa. Luego, he exportado la tabla combinada a Excel para organizar mejor los datos, calcular promedios y medianas y observar patrones generales. También he analizado las curvas ROC de cada modelo, evaluando el área bajo la curva (AUC) para medir su capacidad de discriminación entre clases. Este proceso ha facilitado una comparación clara y fundamentada del rendimiento de cada modelo y permite justificar las diferencias observadas en su eficacia al predecir la aprobación de créditos.

Algorithm	Row ID	TP	FP	TN	FN	Recall	Precision	Sensitivity	Specificity	F-measure
Logistic Regression	0	20451	3387	2189	952	0,96	0,86	0,96	0,39	0,90
Decision Tree	0	21278	1818	3758	125	0,99	0,92	0,99	0,67	0,96
Red Neuronal MLP	0	20769	2781	2795	634	0,97	0,88	0,97	0,50	0,92
KNN	0	20489	2308	3268	914	0,96	0,90	0,96	0,59	0,93
Naive Bayes	0	18976	2241	3335	2427	0,89	0,89	0,89	0,60	0,89
Media		20392,6	2507	3069	1010,4	0,95	0,89	0,95	0,55	0,92
Mediana		20489	2308	3268	914	0,96	0,89	0,96	0,59	0,92
Algorithm	Row ID	TP	FP	TN	FN	Recall	Precision	Sensitivity	Specificity	F-measure
Logistic Regression	1	2189	952	20451	3387	0,39	0,70	0,39	0,96	0,50
Decision Tree	1	3758	125	21278	1818	0,67	0,97	0,67	0,99	0,79
Red Neuronal MLP	1	2795	634	20769	2781	0,50	0,82	0,50	0,97	0,62
KNN	1	3268	914	20489	2308	0,59	0,78	0,59	0,96	0,67
Naive Bayes	1	3335	2427	18976	2241	0,60	0,58	0,60	0,89	0,59
Media		3069	1010,4	20392,6	2507	0,55	0,77	0,55	0,95	0,64
Mediana		3268	914	20489	2308	0,59	0,78	0,59	0,96	0,62

En cuanto a las métricas de precisión y recall, el modelo de decision tree muestra los resultados más altos, con un recall de 0,99 y una precisión de 0,92. Esto sugiere que el decision tree es especialmente eficaz para identificar correctamente los casos positivos (es decir, cuando el crédito es aprobado), manteniendo al mismo tiempo una baja tasa de falsos positivos. La red neuronal MLP y el modelo KNN también lograron buenos valores de precisión y recall, con la red neuronal alcanzando un recall de 0,97 y una precisión de 0,88,

mientras que el KNN obtuvo un recall de 0,96 y precisión de 0,90. En contraste, el modelo naive bayes, aunque presenta una precisión similar (0,89), tiene el recall más bajo (0,89), lo cual puede deberse a su suposición de independencia entre las variables, que en la práctica no siempre se cumple.

La sensibilidad y especificidad también arrojan información valiosa. Los modelos que presentan una mayor sensibilidad (capacidad de detectar los positivos correctamente) son decision tree y red neuronal MLP, con valores de 0,99 y 0,97, respectivamente. Sin embargo, es importante señalar que el Naive Bayes mantiene una especificidad más alta que la red neuronal, lo cual indica que es más preciso al identificar casos negativos (cuando no se aprueba el crédito), si bien su sensibilidad es inferior. La especificidad del Naive Bayes es de 0,89, algo por debajo de los otros modelos, que rondan entre 0,95 y 0,99.

Algorithm	Accuracy	Cohen's kappa
Logistic Regression	0,84	0,42
Decision Tree	0,93	0,75
Red Neuronal MLP	0,87	0,55
KNN	0,88	0,60
Naive Bayes	0,83	0,48
Media	0,87	0,56
Mediana	0,87	0,55

En cuanto a las métricas de precisión global (accuracy y F-measure), la decision tree también lidera con un accuracy de 0,93 y una F-measure de 0,96. Esto sugiere que, en promedio, el decision tree clasifica los datos de forma correcta en la mayoría de los casos. El modelo KNN también presenta una accuracy destacable de 0,88, lo cual lo convierte en una opción competitiva. Naive Bayes, sin embargo, tiene el accuracy más bajo (0,83), lo cual podría estar relacionado con su rendimiento en presencia de variables dependientes.

En cuanto a la ROC AUC, observamos que el decision tree nuevamente se destaca con un AUC de 0,871, seguido de cerca por la red neuronal con un AUC de 0,860. Estos valores indican que ambos algoritmos tienen un excelente rendimiento en términos de separación entre las clases positivas y negativas. El modelo KNN obtiene un AUC algo menor de 0,772, pero sigue siendo sólido, mientras que naive bayes tiene el AUC más bajo (0,743), lo que indica una capacidad de discriminación más limitada.

Finalmente, si analizamos el Cohen's kappa, un indicador que mide la concordancia entre el modelo y las clases verdaderas ajustado por el azar, vemos que el decision tree alcanza el valor más alto (0,75), lo cual sugiere que este modelo tiene la mayor precisión ajustada. Le sigue el KNN (0,60) y la red neuronal (0,55). Estos valores respaldan los resultados de accuracy, ya que muestran que estos modelos superan al azar en la clasificación.

En conclusión, el decision tree parece ser el modelo más robusto para este problema de predicción de aprobación de créditos, ya que obtiene altos valores en precisión, recall, y AUC, además de un excelente Cohen's kappa. La red neuronal MLP y el KNN también

ofrecen un rendimiento fuerte, con buenos balances en las métricas evaluadas. El Naive Bayes, aunque presenta resultados aceptables, es el modelo con menor capacidad predictiva, debido a las dependencias entre variables que afectan su supuesto de independencia.

2. Predicción de segunda cita

Veamos, de forma análoga al caso anterior, una comparación del rendimiento de los cinco algoritmos usados para este problema:

Algorithm	Row ID	TP	FP	TN	FN	Recall	Precision	Sensitivity	Specificity	F-measure
Decision Tree	0	2240	425	2181	366	0,86	0,84	0,86	0,84	0,85
Random Forest	0	2445,00	117,00	2489,00	161,00	0,94	0,95	0,94	0,96	0,95
KNN	0	1833	30	2576	773	0,70	0,98	0,70	0,99	0,82
GradientBoosted	0	2340	238	2368	266	0,90	0,91	0,90	0,91	0,90
Log reg	0	2002	476	2130	604	0,77	0,81	0,77	0,82	0,79
Media		2172	257,2	2348,8	434	0,83	0,90	0,83	0,90	0,86
Mediana		2240	238	2368	366	0,86	0,91	0,86	0,91	0,85
Algorithm	Row ID	TP	FP	TN	FN	Recall	Precision	Sensitivity	Specificity	F-measure
Decision Tree	1	2181	366	2240	425	0,84	0,86	0,84	0,86	0,85
Random Forest	1	2489,00	161,00	2445,00	117,00	0,96	0,94	0,96	0,94	0,95
KNN	1	2576	773	1833	30	0,99	0,77	0,99	0,70	0,87
GradientBoosted	1	2368	266	2340	238	0,91	0,90	0,91	0,90	0,90
Log reg	1	2130	604	2002	476	0,82	0,78	0,82	0,77	0,80
Media		2348,8	434	2172	257,2	0,90	0,85	0,90	0,83	0,87
Mediana		2368	366	2240	238	0,91	0,86	0,91	0,86	0,87

Algorithm	Accuracy	Cohen's kappa
Decision Tree	0,85	0,70
Random Forest	0,95	0,89
KNN	0,85	0,69
GradientBoosted	0,90	0,81
Log reg	0,79	0,59
Media	0,87	0,73
Mediana	0,85	0,70

En este análisis comparativo de algoritmos, los resultados muestran que Random Forest es el mejor clasificador, con una precisión promedio de 0,95 y un Cohen's Kappa de 0,89, lo cual indica una alta concordancia y efectividad en identificar correctamente ambas clases. Su capacidad para manejar datos complejos y ruidosos, combinando múltiples árboles para

reducir el sobreajuste, le permite captar las relaciones presentes en los datos con mayor precisión que otros modelos. Le sigue Gradient Boosted Trees con una precisión de 0,90 y Kappa de 0,81, destacando en especificidad y sensibilidad, aunque algo más susceptible al sobreajuste en caso de datos no preprocesados óptimamente.

Decision Tree logra una precisión moderada de 0,85 y un Kappa de 0,70. Aunque es más fácil de interpretar, su estructura simple lo limita para capturar patrones no lineales complejos en el dataset. K-Nearest Neighbors (KNN) muestra resultados dispares, con una alta especificidad de 99%, pero una menor capacidad para identificar la clase positiva, reflejada en un menor recall. La Regresión Logística, con una precisión de 0,79 y Kappa de 0,59, es el modelo con menor rendimiento, posiblemente debido a su limitación para captar relaciones no lineales en los datos.

La robustez de los modelos de ensamble (Random Forest y Gradient Boosted Trees) es evidente, ya que manejan mejor las interacciones complejas y los outliers sin verse tan afectados por variaciones en los datos. En cambio, algoritmos como la Regresión Logística y KNN son más sensibles a la representación y la linealidad de las características. El preprocesamiento ha sido fundamental, especialmente para algoritmos como KNN y Regresión Logística, que dependen más de la escala y distribución de los datos.

3. Predicción de enfermedades eritemato-escamosas

Veamos una comparación del rendimiento de los cinco algoritmos usados para este problema:

Algorithm	Row ID	TP	FP	TN	FN	Recall	Precision	Sensitivity	Specificity	F-measure
Decision Tree	1	109	6	554	3	0,9732	0,9478	0,9732	0,9893	0,9604
Random Forest	1	111	0	560	1	0,9911	1,0000	0,9911	1,0000	0,9955
Gradient Boosted Trees	1	108	1	559	4	0,9643	0,9908	0,9643	0,9982	0,9774
KNN	1	104	0	530	7	0,9369	1,0000	0,9369	1,0000	0,9674
MLP	1	107	2	528	4	0,9640	0,9817	0,9640	0,9962	0,9727
Media		107,8	1,8	546,2	3,8	0,9659	0,9841	0,9659	0,9967	0,9747
Mediana		108	1	554	4	0,9643	0,9908	0,9643	0,9982	0,9727
Algorithm	Row ID	TP	FP	TN	FN	Recall	Precision	Sensitivity	Specificity	F-measure
Decision Tree	2	80	7	553	32	0,7143	0,9195	0,7143	0,9875	0,8040
Random Forest	2	104	3	557	8	0,9286	0,9720	0,9286	0,9946	0,9498
Gradient Boosted Trees	2	103	6	554	9	0,9196	0,9450	0,9196	0,9893	0,9321
KNN	2	96	4	527	14	0,8727	0,9600	0,8727	0,9925	0,9143

[illegible]

Algorithm	Rowl D	TP	FP	TN	FN	Recall	Precision	Sensitivity	Specificity	F-measure
Decision Tree	6	106	7	553	6	0,9464	0,9381	0,9464	0,9875	0,9422
Random Forest	6	112	1	559	0	1,0000	0,9912	1,0000	0,9982	0,9956
Gradient Boosted Trees	6	112	2	558	0	1,0000	0,9825	1,0000	0,9964	0,9912
KNN	6	112	3	526	0	1,0000	0,9739	1,0000	0,9943	0,9868
MLP	6	112	1	528	0	1,0000	0,9912	1,0000	0,9981	0,9956
Media		110,8	2,8	544,8	1,2	0,9893	0,9753	0,9893	0,9949	0,9823
Mediana		112	2	553	0	1,0000	0,9825	1,0000	0,9964	0,9912

Algorithm	Accuracy	Cohen's kappa
Decision Tree	0,8854	0,8625
Random Forest	0,9702	0,9643
Gradient Boosted Trees	0,9568	0,9482
KNN	0,9454	0,9344
MLP	0,9579	0,9494
Media	0,9432	0,9318
Mediana	0,9568	0,9482

Al comparar los resultados obtenidos para los diferentes modelos, observamos que Random Forest se destaca consistentemente como el mejor algoritmo en este problema, logrando la mayor precisión y especificidad en casi todas las pruebas, con valores cercanos a 0,9911 y 0,9982 respectivamente. Su capacidad para combinar múltiples árboles de decisión le permite capturar diferentes patrones en los datos, resultando en predicciones altamente estables y minimizando el riesgo de falsos positivos y negativos. Esta robustez se debe a que cada árbol de decisión contribuye con una perspectiva diferente, haciendo que el modelo general sea más resistente a pequeñas variaciones en los datos.

Gradient Boosted Trees, aunque también produce resultados sólidos, muestra una ligera desventaja en especificidad en comparación con Random Forest. Su enfoque de entrenamiento iterativo permite ajustar las predicciones para los errores en cada ronda, lo cual refina el modelo y lo hace muy eficaz, pero también introduce un poco más de riesgo de sobreajuste. Esto se refleja en sus valores de precisión y F-measure, que no alcanzan los mismos niveles que Random Forest en todos los casos. Sin embargo, este modelo es muy adecuado cuando es necesario optimizar en casos difíciles de clasificar, ya que es excelente para mejorar iterativamente las predicciones.

El MLP (Multilayer Perceptron), que también da buenos resultados, se encuentra en un punto intermedio en cuanto a rendimiento. Con una precisión de 0,9640 y una F-measure de 0,9727, el MLP logra capturar patrones complejos gracias a su estructura de red neuronal. Sin embargo, en algunos casos presenta un mayor número de falsos positivos en

comparación con Random Forest, lo cual podría deberse a que la red neuronal necesita ajustes específicos para captar correctamente los patrones presentes en este conjunto de datos.

Por otro lado, el KNN tiene altos valores de sensibilidad y especificidad, aunque su precisión es algo menor en comparación con los modelos más complejos. Este modelo se comporta bien al clasificar instancias con patrones claros, pero puede verse afectado por la cantidad de vecinos seleccionados y la distancia entre puntos. Aunque logra buenos resultados en algunas pruebas, puede volverse inestable en presencia de puntos de datos cercanos de clases distintas, lo cual explica su desempeño inferior en comparación con los modelos basados en árboles o redes neuronales.

Finalmente, Decision Tree es el modelo con un desempeño más modesto, especialmente en comparación con Random Forest, del cual es la base. Al utilizar un solo árbol, no tiene el beneficio del promediado de errores que encontramos en Random Forest, lo que resulta en una menor precisión y sensibilidad. Aunque el árbol de decisión es rápido y fácil de interpretar, es más susceptible a los errores debido a la variabilidad en los datos y, por tanto, tiene una menor capacidad para generalizar en comparaciones directas.

Interpretación de los datos

1. Predicción de aprobación de créditos

Regresión Logística

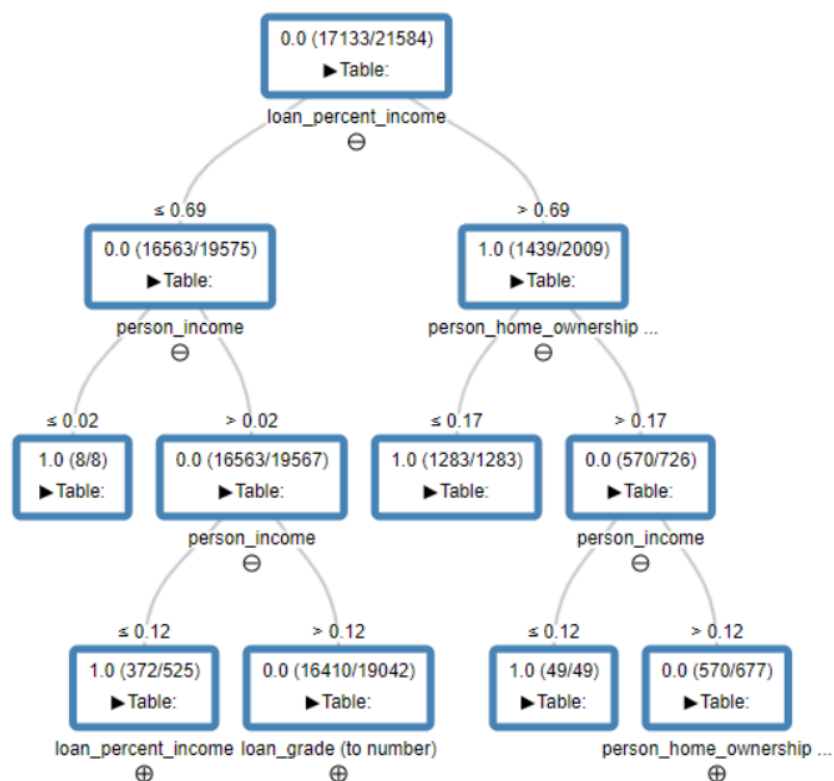
En la regresión logística, los coeficientes representan el impacto de cada variable independiente en la probabilidad de pertenecer a una determinada clase (aprobación o rechazo del crédito). He insertado un nodo de **Table View** conectado al **Logistic Regression Learner** mediante los coeficientes y la tabla encontrada es la siguiente:

Variable	Coeff
cb_person_cred_hist_length	-0,04
person_home_ownership	-1,04
loan_intent	0,73
loan_grade	-0,39
cb_person_default_on_file	-0,22
person_age	0,02
person_income	-0,32
person_emp_length	-0,18
loan_amnt	-2,66
loan_int_rate	4,06
loan_percent_income	5,39
cb_person_cred_hist_length (#1)	-0,04
Constant	-3,95

Para interpretar la tabla debemos tener en cuenta que los coeficientes representan el cambio logarítmico en las probabilidades de la variable dependiente por el cambio de unidad en el predictor. Por lo que vemos, los factores más determinantes en el riesgo de crédito parecen ser la tasa de interés (**loan_int_rate**), el porcentaje del préstamo en relación con el ingreso (**loan_percent_income**), que ambos aumentan el riesgo de impago. La antigüedad laboral (**person_emp_length**) y el nivel de ingresos (**person_income**), que reducen el riesgo de impago y aumentan la probabilidad de ser aprobado el crédito.

Árbol de decisión

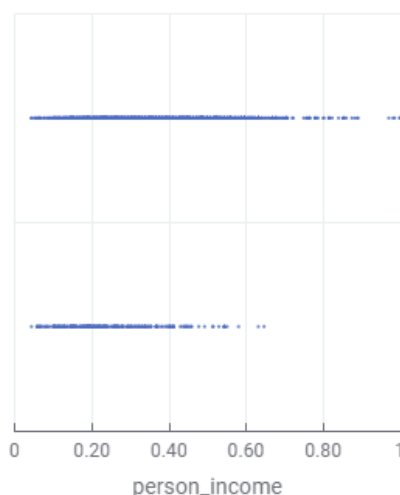
Para interpretar los árboles de decisión hemos usado el nodo **Decision Tree View**, que nos permite ir mostrando las ramas que necesitamos.



Cada división en el árbol representa una regla que el modelo ha aprendido para clasificar los datos. Los factores en el nivel superior del árbol son los factores más determinantes, ya que segmentan el conjunto de datos en los grupos principales. En nuestro caso, la variable de `loan_percent_income`, `person_income` y `person_home_ownership`. Además, en los nodos de decisión, los valores utilizados para dividir datos indican puntos críticos que marcan la diferencia entre las clases.

Si nos fijamos, los datos aparecen normalizados debido al preprocesamiento de los datos al entrenar el modelo. Para interpretar el árbol sin la normalización podríamos entrenarlo aparte con datos sin normalizar, que no afectará al desempeño, ya que este algoritmo no se ve perjudicado por la falta de normalización. Otra opción sería desnormalizar manualmente los valores.

Red Neuronal MLP



Este algoritmo se considera de caja negra, ya que es muy poco interpretable. Gracias a la interpretación de los algoritmos anteriores, podemos crear una matriz de Scatter Plot que determine las predicciones en función de otras variables.

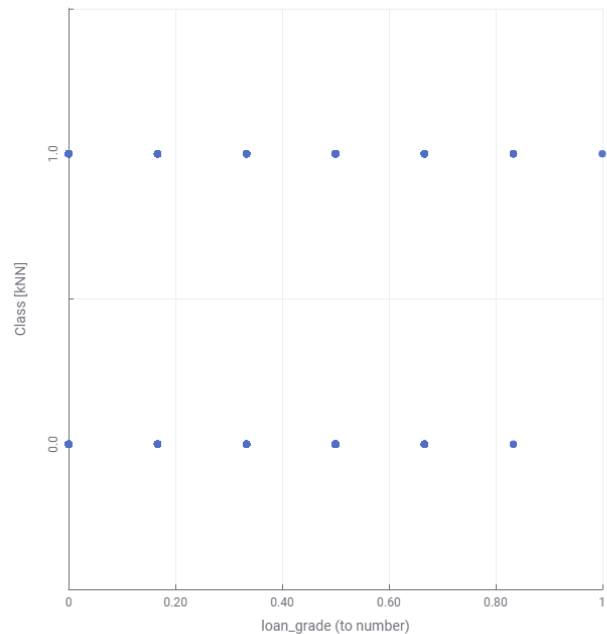
Podemos observar que un alto valor de `person_income` sugiere una predicción de aprobación del crédito. En caso contrario, deberíamos considerar otras variables.

K-Nearest Neighbors

Para este algoritmo también podemos usar un Scatter Plot como el siguiente. Podemos ver que un valor de `loan_grade` muy alto sugiere una predicción de la clase 1, es decir, riesgo de impago y, por tanto, rechazo del crédito.

Como habíamos comentado, estos gráficos están creados a partir de un algoritmo que recibe los datos categóricos codificados y normalizados. Por tanto, debemos transformar el valor de `loan_grade` 1 a categórico, que es la clase G. Es decir, un `loan_grade` de G implica una predicción de rechazo del crédito.

Scatter Plot Matrix

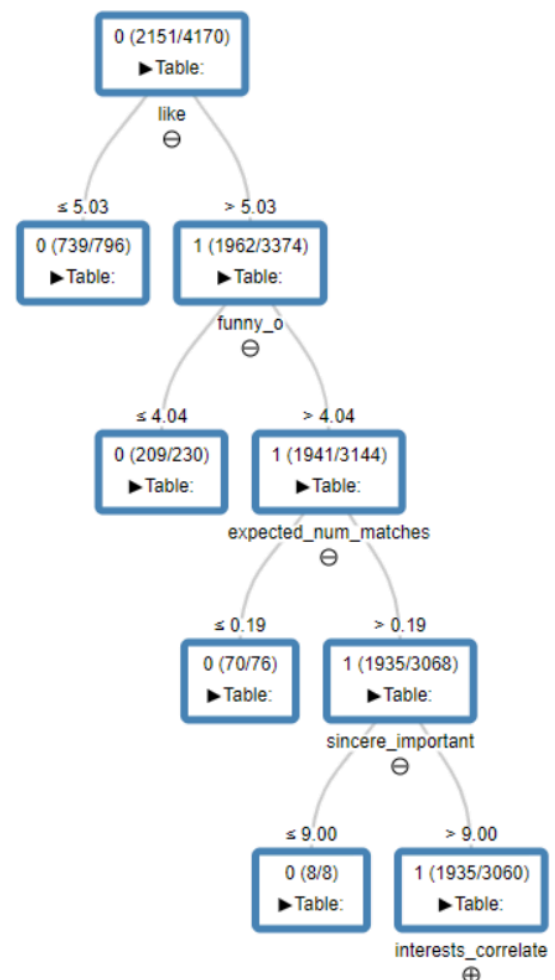


2. Predicción de segunda cita

Árbol de decisión

Este árbol de decisión clasifica una muestra en dos clases (0 o 1) y utiliza una serie de atributos para hacer la distinción. Comienza evaluando el valor de `like`, dividiendo los casos en función de si este valor es menor o igual a 5.03 o mayor. Cuando `like` es bajo (≤ 5.03), los casos se agrupan mayoritariamente en la clase 0 (con 739 casos sobre un total de 796). Sin embargo, cuando `like` supera este umbral, el árbol explora otros atributos.

Si `like` es alto (> 5.03), el árbol analiza el valor de `funny_o`. Si este es menor o igual a 4.04, la mayoría de los casos se asignan nuevamente a la clase 0 (209 de 230). Pero si `funny_o` también es alto, el árbol profundiza aún más, evaluando `expected_num_matches`. Un valor bajo de esta característica (≤ 0.19) lleva principalmente a la clase 0 (70 de 76 casos), mientras que un valor mayor sigue hacia otra división.

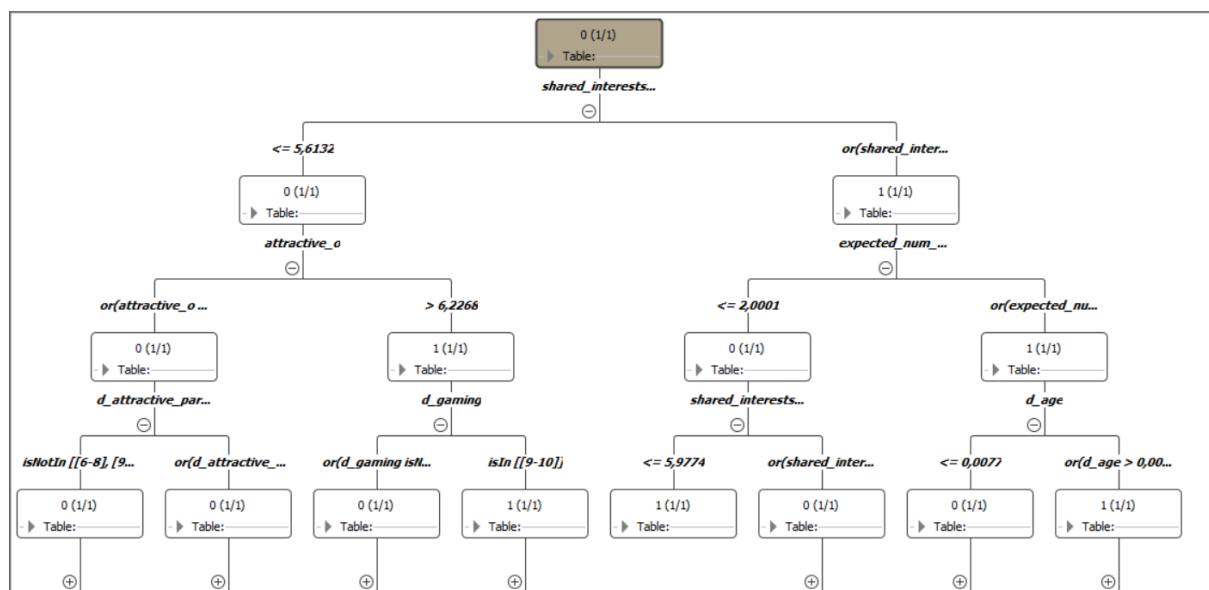


Para los casos con valores altos de `like`, `funny_o`, y `expected_num_matches`, se evalúa `sincere_important`. Si `sincere_important` es bajo (≤ 9), el árbol asigna todos los casos restantes a la clase 0. En cambio, si también es alto, el árbol examina finalmente `interests_correlate`. Si este último atributo también supera cierto umbral, todos los casos restantes se clasifican en la clase 1 (1935 de 3060).

En resumen, el árbol sugiere que la clasificación en la clase 1 se asocia con valores altos de `like`, `funny_o`, `expected_num_matches`, y `sincere_important`. En cambio, los valores bajos de estos atributos tienden a clasificar los casos en la clase 0.

Random Forest

El algoritmo Random Forest se puede interpretar gracias al árbol visible por el nodo `Random Forest Learner`.



Este árbol de Random Forest utiliza `shared_interests` como nodo principal para dividir los datos, lo que sugiere que los intereses compartidos son una variable relevante para la clasificación en este contexto. Dependiendo del valor de `shared_interests`, el árbol explora otros atributos como `attractive_o`, `d_gaming`, `d_attractive_par`, `expected_num_matches`, y `d_age`, que permiten refinar la clasificación en función de características adicionales como la edad, la atracción percibida y los intereses en juegos.

A diferencia de un árbol de decisión individual, el Random Forest combina múltiples árboles, de los cuales este es solo uno, para generar un resultado final. Aunque cada árbol en un Random Forest puede parecer complejo, la combinación de muchos árboles evita el sobreajuste y mejora la precisión. Además, este modelo es más robusto y generaliza mejor en comparación con un solo árbol de decisión, que puede ser más simple pero también más propenso a sobreajustarse a los datos de entrenamiento.

Regresión Logística

El algoritmo de la regresión logística es interpretable gracias a los coeficientes generados por el nodo **Logistic Regression Learner**. La tabla es:

Variable	Coeff.	Std. Err.	z-score
shopping	0,7443723201	0,1004714713	7,40879287
like	-1,020294197	0,06822718899	-14,95436367
d_age	0,13	0,06	2,01
samerace	0,16	0,05	3,20
pref_o_attractive	0,32	0,06	4,89
attractive_o	-0,5452857062	0,05664044937	-9,627143008
funny_o	-0,53	0,06	-8,37

En el análisis de los coeficientes del modelo de regresión logística, observamos que varias variables tienen una relación significativa con la variable de interés ($P < 0.05$). Por ejemplo, variables como **d_age** (coeficiente: 0.13), **samerace** (0.16) y **pref_o_attractive** (0.32) muestran una relación positiva, indicando que a medida que estos valores aumentan, también lo hace la probabilidad de pertenecer a la clase objetivo.

Por otro lado, variables como **attractive_o** (-0.55), **funny_o** (-0.53) y **like** (-1.02) tienen coeficientes negativos significativos, lo que sugiere que mayores valores en estas variables reducen la probabilidad de estar en la clase objetivo. La variable shopping destaca con el coeficiente positivo más alto (0.74), lo cual indica una fuerte asociación positiva.

Algunas variables, como **sports**, **museums**, y **art**, no son significativas ($P > 0.05$), sugiriendo que podrían ser excluidas del modelo sin afectar mucho su desempeño. Vemos que atributos relacionados con preferencias personales y percepción de la otra persona tienen el mayor impacto en la probabilidad de clasificación, mientras que actividades de ocio parecen tener un rol menos relevante.

3. Predicción de enfermedades eritemato-escamosas

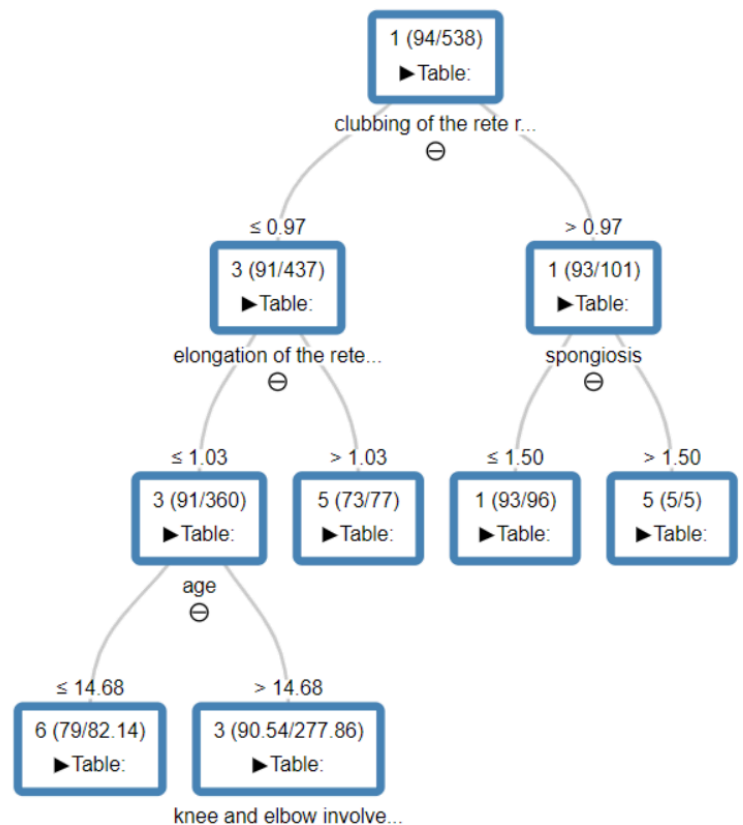
Árbol de decisión

Este árbol de decisión clasifica una muestra en 6 clases y utiliza una serie de atributos para hacer la distinción. Comienza evaluando el valor de **clubbing_of_the_rete_ridges**, dividiendo los casos en función de si este valor es menor o igual a 0.97 o mayor. Cuando es alto (≤ 0.97), los casos se agrupan mayoritariamente en la clase 1 (con 93 casos sobre un total de 101). Sin embargo, cuando **clubbing_of_the_rete_ridges** es menor, el árbol explora otros atributos.

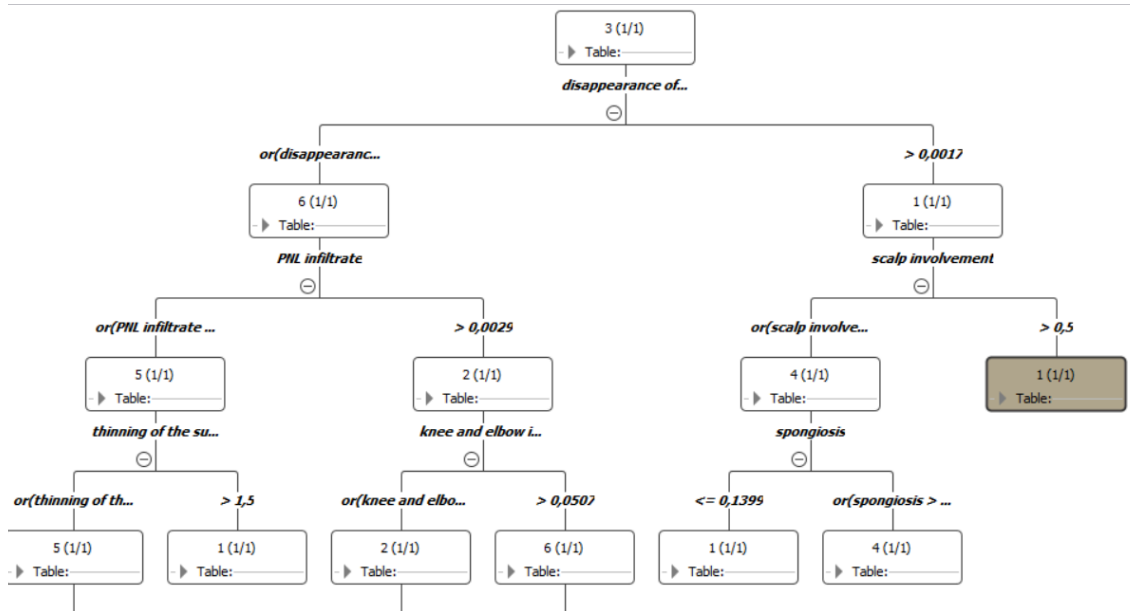
Si es bajo, el árbol analiza el valor de **elongation of the rete ridges**. Si este es mayor a 1.03, la mayoría de los casos se asignan nuevamente a la clase 5 (73 de 77). Pero si es bajo, el árbol profundiza aún más, evaluando **age**. Un valor bajo de esta característica (≤ 14.68) lleva principalmente a la clase 6 (79 de 82.14 casos), mientras que un valor mayor sigue hacia otra división.

El árbol continúa su clasificación de la misma manera para las ramas de niveles más bajos.

En resumen, el árbol sugiere que la clasificación en la clase 1 se asocia con valores altos de **clubbing of the rete ridges**; si el valor de esta variable es bajo y el **elongation of the rete ridges** el alto, sugiere la clasificación en la clase 5. En caso contrario, un valor bajo de **age** y un valor alto de **knee_and_elbow_involvement** sugiere la clasificación en la clase 6. En caso contrario, un valor alto de **PNL_infiltrate** sugiere la clasificación en la clase 2. Finalmente, tras varias ramas, podríamos decir que la clasificación entre las clases 3 y 4 depende del valor de **itching**.



Random Forest

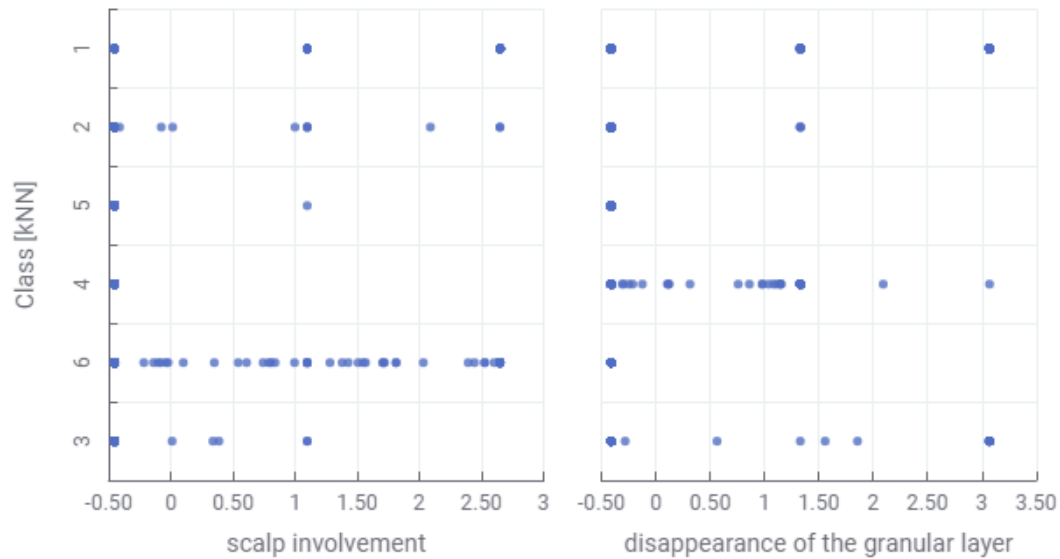


Este árbol de Random Forest utiliza **disappearance of the granular layer** como nodo principal para dividir los datos. Dependiendo de este valor, el árbol explora otros atributos como **PNL infiltrate**, **scalp involvement**, **spongiosis**, y **thinning of the peripapillary epidermis**, que permiten refinar la clasificación.

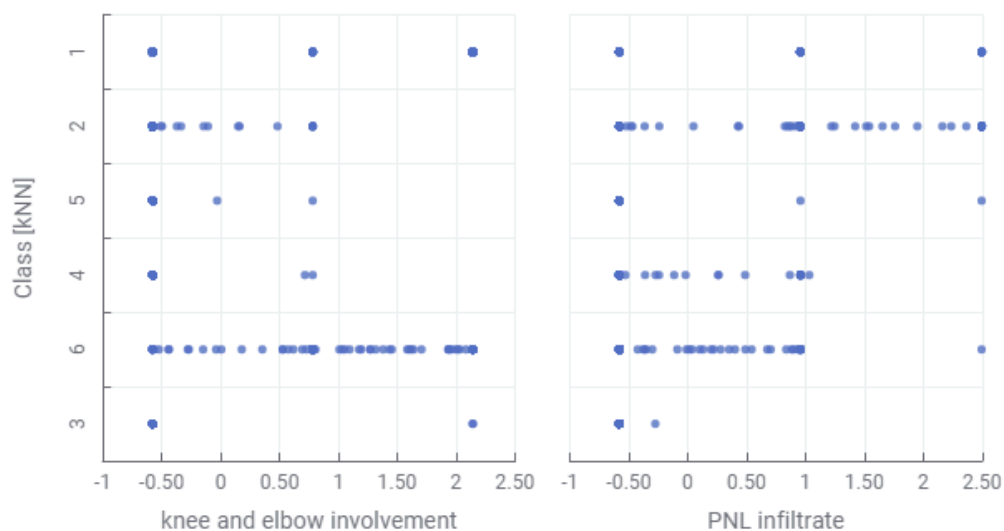
El Random Forest genera su resultado final al combinar múltiples árboles de decisión, en lugar de depender de uno solo. Aunque un árbol de decisión individual puede ser menos complejo, es más susceptible al sobreajuste y a ajustarse demasiado a los datos de entrenamiento. Al usar una colección de árboles, el Random Forest reduce este problema, mejora la precisión y ofrece un modelo más robusto y capaz de generalizar mejor en comparación con un único árbol.

Gradient Boosted Trees

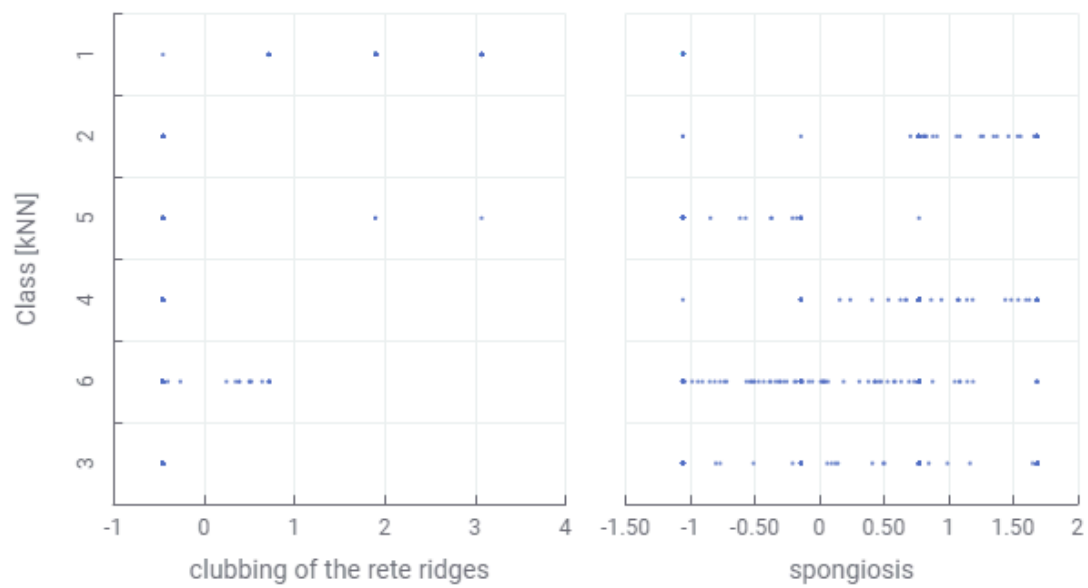
Para interpretar los resultados de este algoritmo he usado un Scatter Plot Matrix, que me permite comprobar las relaciones de la predicción con respecto a alguna variable. He ejecutado aquellos gráficos que se pueden comparar con los resultados dados en los anteriores algoritmos.



Este gráfico concuerda con los resultados propuestos por el algoritmo Random Forest. Vemos que si los valores de **disappearance of the granular layer** son altos, entonces probablemente estemos ante un caso de clase 1, 3 ó 4. En dicho caso, podríamos comprobar si el valor de **scalp involvement** es alto, puesto que esto supondría una predicción de la clase 1, ya que las clases 3 y 4 están relacionadas con valores bajos de esta variable.



Podemos interpretar este gráfico como la rama izquierda del Random Forest. En dicho caso, un valor alto de **PNL_infiltrate** sugeriría una predicción de las clases 1, 2, 5 ó 6. En dicho caso, contemplaremos el valor de **knee_and_elbow_involvement**, que si es alto implicaría una predicción de las clases 1 ó 6, como muestra el árbol.



También podemos ver el sentido del árbol de decisión. Como variable principal usa **clubbing_of_the_rate_ridges**: si es alto, implica una predicción de la clase 1 ó 5. En dicho caso, podemos contemplar el valor de **spongiosis**, ya que un valor medio o alto sugeriría una predicción de la clase 5, al igual que el árbol de decisión.

Bibliografía

Bajaj, Vardaan. "Deep Dive Into Logistic Regression and Data Pre-Processing." *Medium*, 19

June 2020,

<https://medium.com/analytics-vidhya/machine-learning-ii-logistic-regression-explained-data-pre-processing-hands-on-kaggle-728e6a9d4bbf>. Accessed 5 November

2024.

Data Science Training. "Feature Selection Loop End [KNIME Analytics Platform]." *YouTube*,

24 septiembre 2023, https://youtu.be/IXO2uZaT7hw?si=0Hw_-TwqIi254BGJ.

"Interpreting Coefficients in Logistic Regression: A Guide." *LinkedIn*, 1 January 2024,

<https://www.linkedin.com/advice/0/how-can-you-interpret-coefficients-logistic-regression-t53gf>. Accessed 9 November 2024.

"Linear Correlation Coefficient: A Complete Insight." *Voxco*,

<https://www.voxco.com/blog/linear-correlation-coefficient/>. Accessed 10 November

2024.

"Logistic Regression – KNIME Community Hub." *KNIME Community Hub*, 20 October 2023,

https://hub.knime.com/knime/spaces/Examples/04_Analytics/04_Classification_and_Predictive_Modelling/06_Logistic_Regression~SOUGsGodd-2vs8HJ/current-state.

Accessed 5 November 2024.

"Multi Layer Perceptron – KNIME Community Hub." *KNIME Community Hub*, 20 October

2023,

https://hub.knime.com/knime/spaces/Examples/04_Analytics/04_Classification_and_Predictive_Modelling/02_Example_for_Learning_a_Neural_Network~xUrGdCLkLrNe8EBz/current-state.

Accessed 10 November 2024.

