

INTELIGENCIA DE NEGOCIO (2017-2018)
GRADO EN INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE GRANADA

Análisis predictivo empresarial mediante
clasificación



**UNIVERSIDAD
DE GRANADA**

Iván Rodríguez Millán

ivanrodmil@gmail.com

Índice

1	Introducción	7
2	Resultados obtenidos	8
2.1	Decision Tree	8
2.2	Gradient Boosted	13
2.3	Random Forest	17
2.4	Naive Bayes	21
2.5	K-NN	25
2.6	Red Neuronal	30
3	Análisis de los resultados	35
4	Configuración de algoritmos	42
4.1	Decision Tree	42
4.2	Gradient Boosted	47
4.3	Naive Bayes	52
4.4	Random Forest	54
4.5	KNN	56
4.6	Red Neuronal	62
5	Procesado de datos	68
5.1	Decision Tree	68
5.2	Random Forest	78
5.3	Naive Bayes	84
6	Interpretación de resultados	90
7	Contenido adicional	97

Índice de figuras

2.1.	Workflow del algoritmo Decision Tree.	8
2.2.	Validación cruzada aplicando el algoritmo Decision Tree.	8
2.3.	Filtrado de los datos de salida del algoritmo Decision Tree.	9
2.4.	Unión de los datos de dos algoritmos, Decision Tree y Gradient Boosted. .	10
2.5.	Unión final de todos los resultados, creación de la gráfica ROC conjunta e inserción de las columnas restantes.	11
2.6.	Gráfica ROC del algoritmo Decision Tree.	12
2.7.	Workflow del algoritmo Gradient Boosted.	13
2.8.	Validación cruzada aplicando el algoritmo Gradient Boosted.	13
2.9.	Filtrado de los datos de salida del algoritmo Gradient Boosted.	14
2.10.	Unión de los datos de dos algoritmos, Decision Tree y Gradient Boosted. .	15
2.11.	Gráfica ROC del algoritmo Gradient Boosted.	16

2.12. Workflow del algoritmo Random Forest.	17
2.13. Validación cruzada aplicando el algoritmo Random Forest.	17
2.14. Filtrado de los datos de salida del algoritmo Random Forest.	18
2.15. Unión de los datos de dos algoritmos, Random Forest y Naive Bayes.	19
2.16. Gráfica ROC del algoritmo Random Forest.	20
2.17. Workflow del algoritmo Naive Bayes.	21
2.18. Validación cruzada aplicando el algoritmo Naive Bayes.	21
2.19. Filtrado de los datos de salida del algoritmo Naive Bayes.	22
2.20. Unión de los datos de dos algoritmos, Random Forest y Naive Bayes.	23
2.21. Gráfica ROC del algoritmo Naive Bayes.	24
2.22. Workflow del algoritmo K-NN.	25
2.23. Preprocesado de los datos.	25
2.24. Validación cruzada aplicando el algoritmo K-NN.	26
2.25. Filtrado de los datos de salida del algoritmo K-NN.	27
2.26. Unión de los datos de dos algoritmos, K-NN y Red Neuronal.	28
2.27. Gráfica ROC del algoritmo K-NN.	29
2.28. Workflow del algoritmo Red Neuronal.	30
2.29. Preprocesado de los datos.	31
2.30. Validación cruzada aplicando el algoritmo Red Neuronal.	32
2.31. Filtrado de los datos de salida del algoritmo Red Neuronal.	33
2.32. Unión de los datos de dos algoritmos, K-NN y Red Neuronal.	33
2.33. Gráfica ROC del algoritmo Red Neuronal.	34
3.1. Gráfica ROC de todos los algoritmos.	35
3.2. Gráfica del Recall de todos los algoritmos, integrando tanto verdaderos positivos como falsos negativos.	37
3.3. Gráfica del Precision de todos los algoritmos, integrando tanto falsos positivos como verdaderos negativos.	39
3.4. Gráfica Accuracy de todos los algoritmos.	40
3.5. Gráfica Accuracy de todos los algoritmos.	41
4.1. Configuración por defecto del algoritmo Decision Tree.	42
4.2. Configuración modificada del algoritmo Decision Tree, algoritmo 1).	43
4.3. Configuración modificada del algoritmo Decision Tree, algoritmo 2.	44
4.4. Gráfica ROC comparativa del algoritmo Decision Tree con las dos configuraciones.	45
4.5. Configuración por defecto del algoritmo Gradient Boosted.	47
4.6. Configuración modificada del algoritmo Gradient Boosted, algoritmo 1.	48
4.7. Configuración modificada del algoritmo Gradient Boosted, algoritmo 2.	49
4.8. Gráfica ROC comparativa del algoritmo Gradient Boosted con las dos configuraciones..	50
4.9. Configuración por defecto del algoritmo Naive Bayes.	52
4.10. Gráfica ROC del algoritmo Naive Bayes con la configuración por defecto.	53
4.11. Configuración por defecto del algoritmo Random Forest.	54
4.12. Gráfica ROC del algoritmo Random Forest con la configuración por defecto.	55
4.13. Configuración por defecto del algoritmo KNN.	56

4.14. Configuración modificada del algoritmo KNN, algoritmo 1.	57
4.15. Configuración modificada del algoritmo KNN, algoritmo 2.	58
4.16. Configuración modificada del algoritmo KNN, algoritmo 3.	59
4.17. Gráfica ROC comparativa del algoritmo KNN con las dos configuraciones..	60
4.18. Configuración por defecto del algoritmo Red Neuronal.	62
4.19. Configuración modificada del algoritmo Red Neuronal, algoritmo 1.	63
4.20. Configuración modificada del algoritmo Red Neuronal, algoritmo 2.	64
4.21. Configuración modificada del algoritmo Red Neuronal, algoritmo 3.	65
4.22. Gráfica ROC comparativa del algoritmo Red Neuronal con las dos config- uraciones.	66
5.1. Configuración del nodo One to Many.	69
5.2. Configuración del nodo Column Filter.	69
5.3. Configuración del nodo Normalizer.	70
5.4. Workflow de preprocesamiento 1, del algoritmo Decision Tree para las tres versiones del apartado anterior 4.	70
5.5. Gráfica ROC comparativa de las anteriores versiones del Decision Tree con el preprocesamiento 1.	71
5.6. Gráfica ROC comparativa de las anteriores versiones del Decision Tree con el preprocesamiento 2.	73
5.7. Gráfica ROC comparativa teniendo en cuenta la misma versión (1) y dis- tintos preprocesados.	74
5.8. Gráfica ROC comparativa teniendo en cuenta la misma versión (2) y dis- tintos preprocesados.	75
5.9. Gráfica ROC comparativa teniendo en cuenta la misma versión (3) y dis- tintos preprocesados.	76
5.10. Workflow de preprocesamiento 1, del algoritmo Random Forest para la versión del apartado anterior 4.	78
5.11. Configuración del nodo equal size sampling.	80
5.12.	82
5.13.	88
6.1. Gráfica de barras con total de ocurrencias de cada clase.	90
6.2. Gráfica que compara las clases con las clases de trabajo.	91
6.3. Gráfica con número de instancias de cada clase de trabajo.	92
6.4. Gráfica comparativa de la clase de trabajo y la clase del conjunto de datos.	93
6.5. Gráfica Box comparando las dos clases con la edad.	94
6.6. Gráfica comparando las dos clases con la edad.	95

Índice de tablas

2.1. Tabla de datos del algoritmo Decision Tree.	12
2.2. Confusion Matrix del algoritmo Decision Tree.	12
2.3. Tabla de datos del algoritmo Gradient Boosted.	15
2.4. Confusion Matrix del algoritmo Gradient Boosted.	15

2.5.	Tabla de datos del algoritmo Random Forest.	19
2.6.	Confusion Matrix del algoritmo Random Forest.	19
2.7.	Tabla de datos del algoritmo Naive Bayes.	23
2.8.	Confusion Matrix del algoritmo Naive Bayes.	23
2.9.	Tabla de datos del algoritmo K-NN.	28
2.10.	Confusion Matrix del algoritmo K-NN.	28
2.11.	Tabla de datos del algoritmo Neural Network.	34
2.12.	Confusion Matrix del algoritmo Neural Network.	34
3.1.	Tabla de datos de todos los algoritmos.	35
4.1.	Tabla comparativa del algoritmo Decision Tree con las configuraciones por defecto y modificadas.	45
4.2.	Tabla comparativa del algoritmo Gradient Boosted con las configuraciones por defecto y modificadas.	49
4.3.	Tabla de resultados del algoritmo Naive Bayes con la configuración por defecto.	52
4.4.	Tabla de resultados del algoritmo Random Forest con la configuración por defecto.	54
4.5.	Tabla comparativa del algoritmo KNN con las configuraciones por defecto y modificadas.	60
4.6.	Tabla comparativa del algoritmo Red Neuronal con las configuraciones por defecto y modificadas.	66
5.1.	Tabla comparativa del algoritmo Decision Tree con las configuraciones del apartado anterior 4 y el primer tipo de preprocesamiento.	70
5.2.	Tabla comparativa del algoritmo Decision Tree con las configuraciones del apartado anterior 4 y el primer tipo de preprocesamiento.	72
5.3.	Tabla comparativa del algoritmo Decision Tree con las configuraciones del apartado anterior 4 y el primer tipo de preprocesamiento.	74
5.4.	Tabla comparativa del algoritmo Decision Tree con las configuraciones del apartado anterior 4 y el primer tipo de preprocesamiento.	75
5.5.	Tabla comparativa del algoritmo Decision Tree con las configuraciones del apartado anterior 4 y el primer tipo de preprocesamiento.	76
5.6.	Tabla comparativa del algoritmo Decision Tree con las configuraciones del apartado anterior 4 y todos los tipos de preprocesamiento.	77
5.7.	Tabla del algoritmo Random Forest con la configuración del apartado anterior 4 y el segundo tipo de preprocesamiento.	79
5.8.	Tabla del algoritmo Random Forest con la configuración del apartado anterior 4 y el segundo tipo de preprocesamiento.	80
5.9.	Tabla del algoritmo Random Forest con la configuración del apartado anterior 4 y el tercer tipo de preprocesamiento.	81
5.10.	Tabla comparativa del algoritmo Random Forest con la configuración del apartado anterior 4 y todos los tipos de preprocesamiento.	82
5.11.	Tabla del algoritmo Naive Bayes con las configuraciones del apartado anterior 4 y el primer tipo de preprocesamiento.	84

5.12. Tabla del algoritmo Naive Bayes con las configuraciones del apartado anterior 4 y el segundo tipo de preprocesamiento.	85
5.13. Tabla del algoritmo Naive Bayes con las configuraciones del apartado anterior 4 y el tercer tipo de preprocesamiento.	86
5.14. Tabla del algoritmo Naive Bayes con las configuraciones del apartado anterior 4 y el cuarto tipo de preprocesamiento.	87
5.15. Tabla del algoritmo Naive Bayes con las configuraciones del apartado anterior 4 y el primer tipo de preprocesamiento.	88

1. Introducción

Esta práctica ha sido llevada a cabo para la asignatura de Inteligencia de Negocio de la universidad de Granada, asignatura de cuarto curso del Grado en Ingeniería Informática. Veremos el uso de algoritmos de aprendizaje supervisado de clasificación para realizar el análisis predictivo de los datos de la empresa que se describirán a continuación.

Así mismo, vamos a trabajar con el conjunto de datos Adult [1] para estudiar la influencia de determinados factores sobre la capacidad de ingresos anuales de una persona, en particular, si sus ingresos anuales excederán de los 50 mil dólares. Los datos están extraídos del censo de una base de datos de 1994.

Información de los atributos más relevantes:

- Edad. Numérica
- Clase de trabajo. Categórica.
- Educación. Categórica.
- Estado conyugal. Categórica.
- Relación. Categórica.
- Ocupación. Categórica.
- Raza. Categórica.
- Sexo. Categórica.
- País nativo. Categórica.
- Capital ganado. Numérica.
- Capital perdido. Numérica.
- Horas por semana. Numérica.
- educación (Número). Numérica.
- fnlwgt. Numérica.

Son en total 14 atributos, y hay 48842 instancias, con valores perdidos.

Principalmente en esta práctica debemos abordar el estudio de los distintos algoritmos de clasificación mediante el diseño experimental apropiado extrayendo conclusiones finales. Para ello debemos de aplicar 6 algoritmos de clasificación distintos, y a su vez sobre algunos de ellos, realizar pequeñas modificaciones para estudiar los comportamientos de los mismos sobre el problema que se estudia.

Toda la experimentación se ha realizado con validación cruzada de 5 particiones. También se han extraído tablas comparativas para mostrar las diferencias entre los distintos algoritmos, gráficas ROC, matrices de confusión, G-mean, etc.

2. Resultados obtenidos

2.1. Decision Tree

En primer lugar usaremos el algoritmo de clasificación Decision Tree (Árbol de Decisión)[3], el cual a partir de un conjunto de datos con los distintos atributos permite determinar a que clase pertenece el caso de estudio.

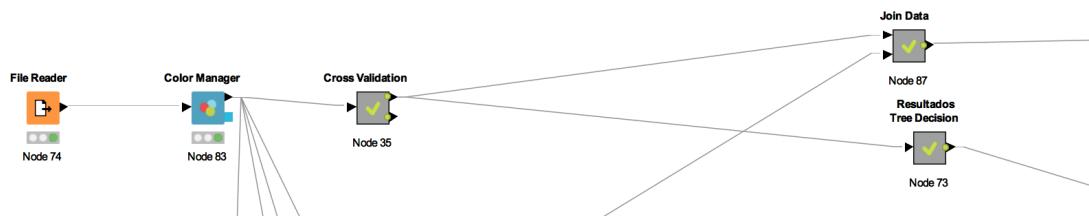


Figura 2.1: Workflow del algoritmo Decision Tree.

El flujo de trabajo de todos los algoritmos comienza en el nodo File Reader (Lector de fichero) donde se cargan los datos, seguidamente tendremos el nodo Color Manager (Gestor de color) para diferenciar por colores las instancias con número de ingresos mayor o igual a 50K de las instancias con número de ingresos menor que 50K.

Una vez nos encontramos en este punto podemos ver las diferencias significativas entre los distintos algoritmos:

El nodo Cross Validation:

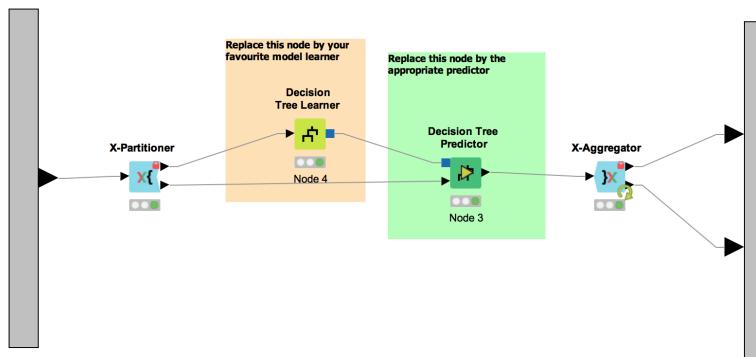


Figura 2.2: Validación cruzada aplicando el algoritmo Decision Tree.

Dentro de este nodo Cross Validation como podemos ver está el nodo "X-Partitioner", que básicamente nos permite configurar el número de validaciones (En nuestro caso 5.) y el muestreo estratificado. Después tenemos los nodos del algoritmo, en primer lugar el Decision Tree Learner (nodo encargado del entrenamiento), el cuál se encarga de recoger los datos de entrenamiento e inducir a partir de ellos un árbol de decisión y en segundo lugar el nodo Decision Tree Predictor el cual una vez obtenido un árbol de decisión se lo pasamos por uno de los puertos (el puerto de color azul), y por el otro puerto el conjunto de datos de test, para así predecir el valor de la clase de nuevas muestras. Y por último nos encontramos con el nodo "Y-Aggregator" que se encarga de volver a reproducir los tres nodos anteriores como si de un bucle se tratara, y también se encarga de sacar una tabla con las predicciones.

Posterior al nodo Cross Validation tenemos el nodo Resultados Tree Decision, que contiene la siguiente información:

En primer lugar nos encontramos con el nodo "Scorer" que nos permite mostrar la matriz de confusión y las estadísticas de precisión del algoritmo. Los nodos posteriores tienen la única función de filtrar estos datos y cambiar el identificador de la clase a predecir por el nombre del algoritmo que le hayamos aplicado, de tal forma que podamos identificar los resultados obtenidos por cualquier algoritmo.

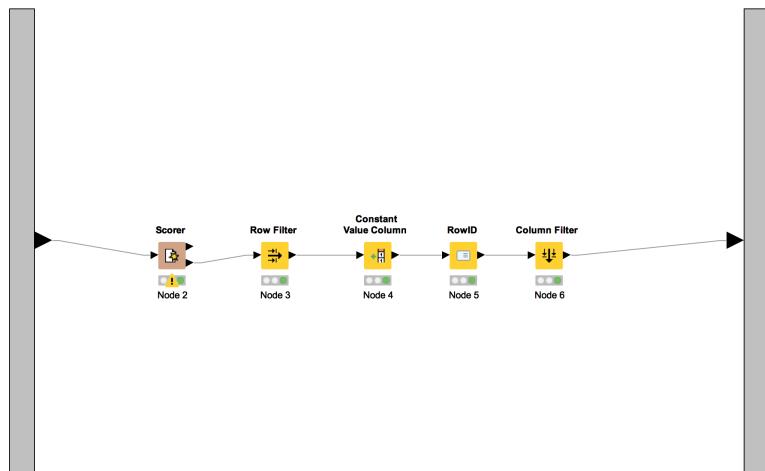


Figura 2.3: Filtrado de los datos de salida del algoritmo Decision Tree.

Paralelamente a este nodo llamado Resultados Tree Decision, tenemos otro llamado Join Data, que basicamente une los resultados de 2 algoritmos para poder realizar la Gráfica ROC más adelante de forma comparativa, en donde se refleje en una única gráfica el AUC (Área bajo la curva) de todos los algoritmos. Los nodos ColumnRename nos sirven para renombrar la variable "P(>=50K)" para saber cada valor de que algoritmo proviene.

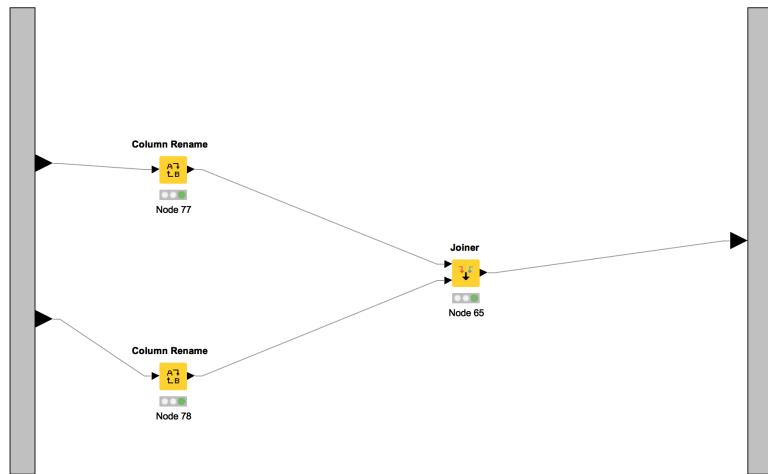


Figura 2.4: Unión de los datos de dos algoritmos, Decision Tree y Gradient Boosted.

Tras el paso de unir cada 2 algoritmos los datos, pasamos a unir a su vez estos resultados para acabar desembocando en lo siguiente: En primer lugar tenemos el nodo

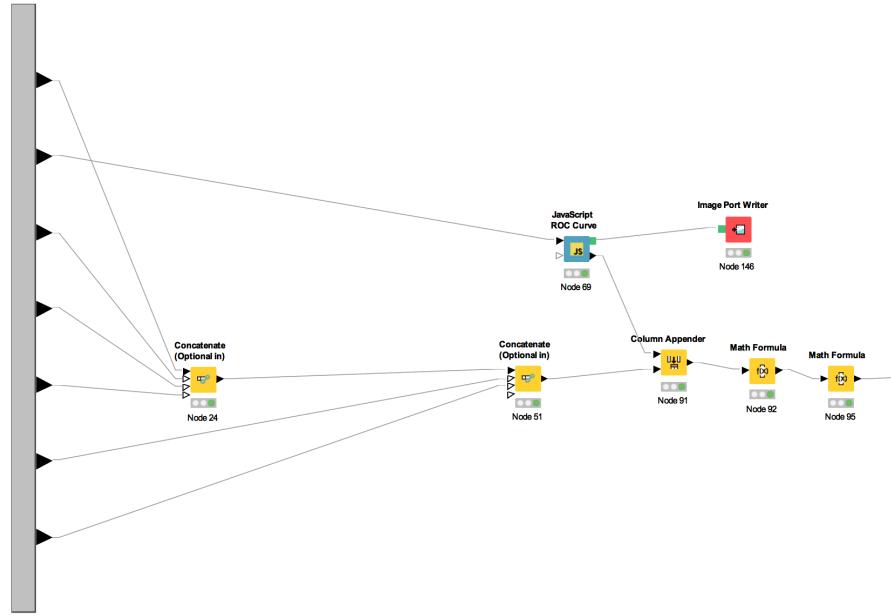


Figura 2.5: Unión final de todos los resultas, creación de la gráfica ROC conjunta e inserción de las columnas restantes.

Concatenate[14] que nos sirve para unir en una tabla todos los datos extraidos por los diferentes algoritmos, tenemos 2 porque uno solo nos sirve como mucho para unir 4 algoritmos. Paralelamente a estos nodos "Concatenate" tenemos el nodo JavaScript ROC Curve para obtener la gráfica ROC y el valor de AUC (Area Under Curve, área bajo la curva) [15]. Con el nodo ColumnAppender unimos a la tabla de los resultados de todos los algoritmos la columna nueva con el AUC. Los nodos MathFormula posteriores sirven para añadir las columnas ".Accuracy"[16] y "G-Mean."^a la tabla final. ¹

¹Esta parte es única para todos los algoritmos, por ello solo se mostrará en este apartado.

Algorithm	TruePositives	FalsePositives	TrueNegatives	FalseNegatives	Recall	Precision	Specificity	F-measure	G-mean	Accuracy	AUC
Decision Tree	6969	3744	32991	4541	0.6054	0.6505	0.8980	0.6271	0.7374	0.8282	0.8055

Tabla 2.1: Tabla de datos del algoritmo Decision Tree.

	<=50K	>50K
<=50K	32991	3744
>50K	4541	6969

Tabla 2.2: Confusion Matrix del algoritmo Decision Tree.

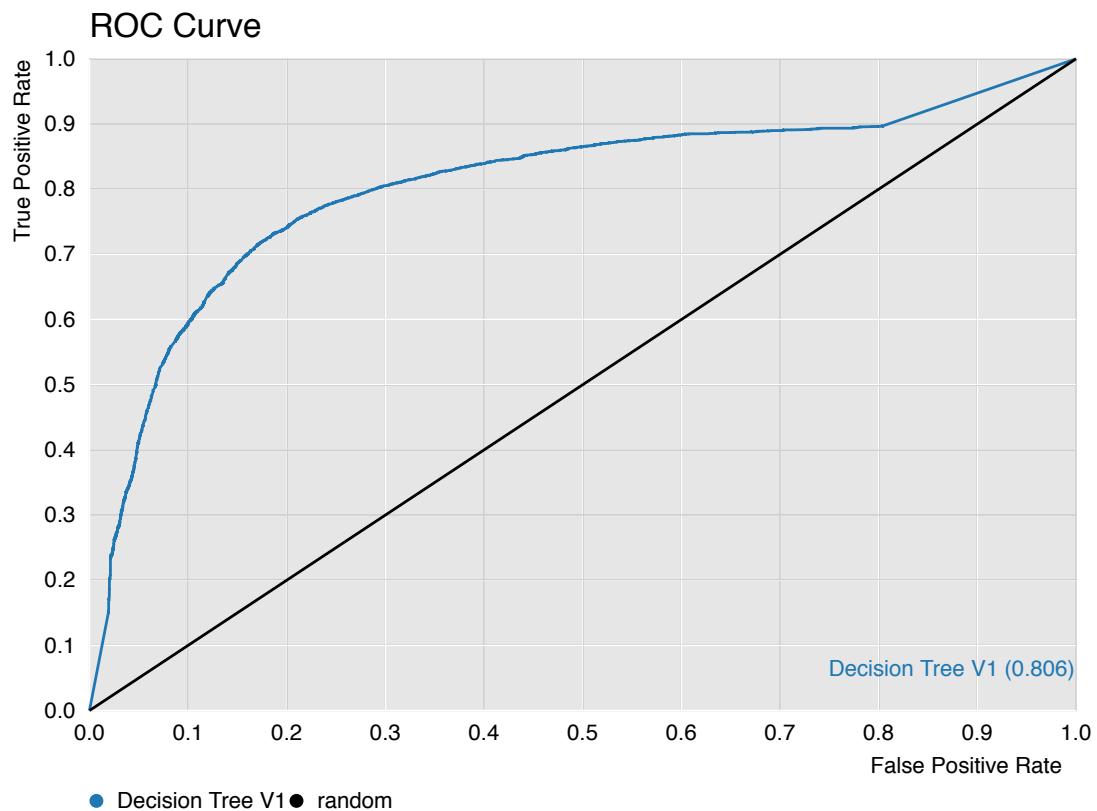


Figura 2.6: Gráfica ROC del algoritmo Decision Tree.

2.2. Gradient Boosted

El segundo algoritmo que aplicamos es Gradient Boosted[2], el cual utiliza árboles de regresión para construir un conjunto de árboles.

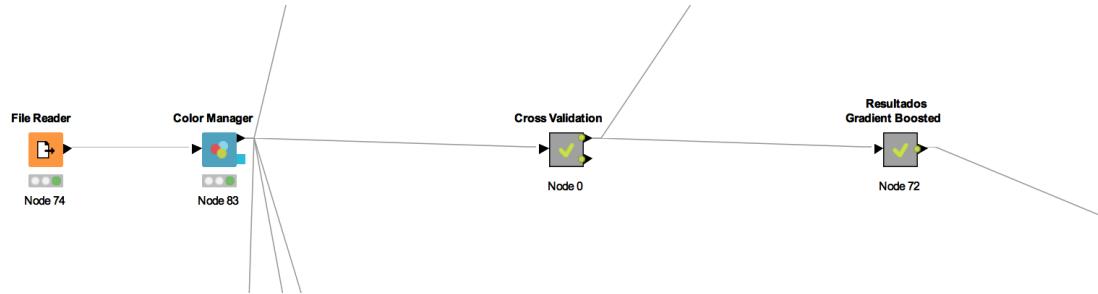


Figura 2.7: Workflow del algoritmo Gradient Boosted.

El flujo de trabajo de todos los algoritmos comienza en el nodo File Reader (Lector de fichero) donde se cargan los datos, seguidamente tendremos el nodo Color Manager (Gestor de color) para diferenciar por colores las instancias con número de ingresos mayor o igual a 50K de las instancias con número de ingresos menor que 50K.

Una vez nos encontramos en este punto podemos ver las diferencias significativas entre los distintos algoritmos:

El nodo Cross Validation:

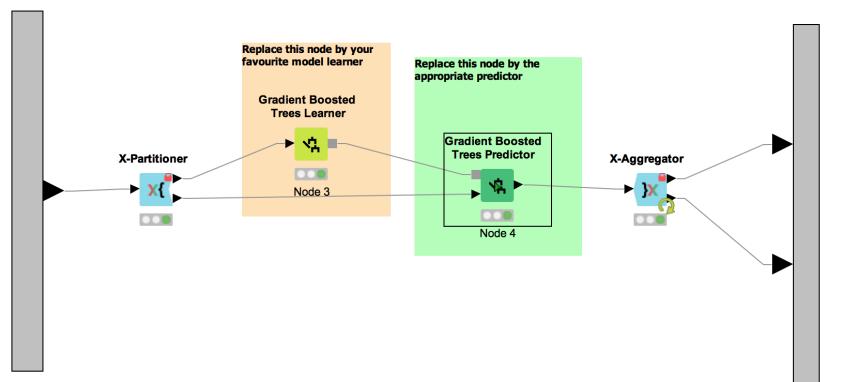


Figura 2.8: Validación cruzada aplicando el algoritmo Gradient Boosted.

Dentro de este nodo Cross Validation como podemos ver está el nodo "X-Partitioner", que básicamente nos permite configurar el número de validaciones (En nuestro caso 5.) y el muestreo estratificado. Después tenemos los nodos del algoritmo, en primer lugar el Gradient Boosted Trees Learner (nodo encargado del entrenamiento) en segundo lugar el nodo Gradient Boosted Trees Predictor el cual clasifica los datos usando un modelo de Gradient Boosted Trees. Y por último nos encontramos con el nodo "Y-Aggregator" que se encarga de volver a reproducir los tres nodos anteriores como si de un bucle se tratara, y también se encarga de sacar una tabla con las predicciones.

Posterior al nodo Cross Validation tenemos el nodo Resultados Gradient Boosted, que contiene la siguiente información:

En primer lugar nos encontramos con el nodo "Scorer" que nos permite mostrar la matriz de confusión y las estadísticas de precisión del algoritmo. Los nodos posteriores tienen la única función de filtrar estos datos y cambiar el identificador de la clase a predecir por el nombre del algoritmo que le hayamos aplicado, de tal forma que podamos identificar los resultados obtenidos por cualquier algoritmo.

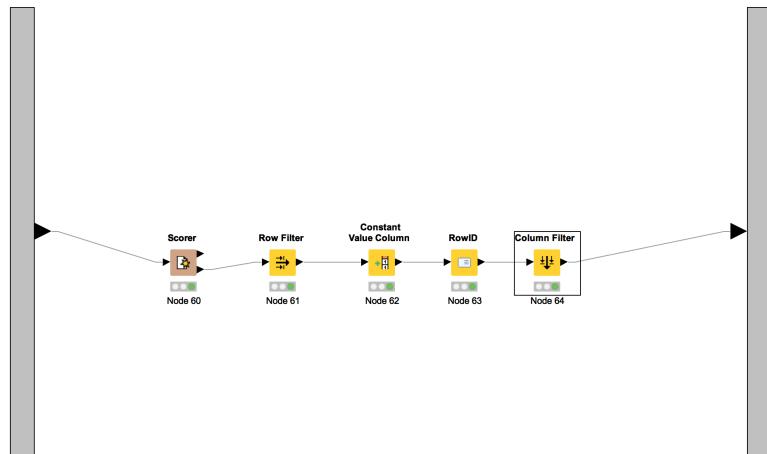


Figura 2.9: Filtrado de los datos de salida del algoritmo Gradient Boosted.

Paralelamente a este nodo llamado Resultados Gradient Boosted, tenemos otro llamado Join Data, que basicamente une los resultados de 2 algoritmos para poder realizar la Gráfica ROC más adelante de forma comparativa, en donde se refleje en una única gráfica el AUC (Área bajo la curva) de todos los algoritmos. Esto se verá mas adelante, por ello en este apartado simplemente comento el porque de la existencia de tal nodo.

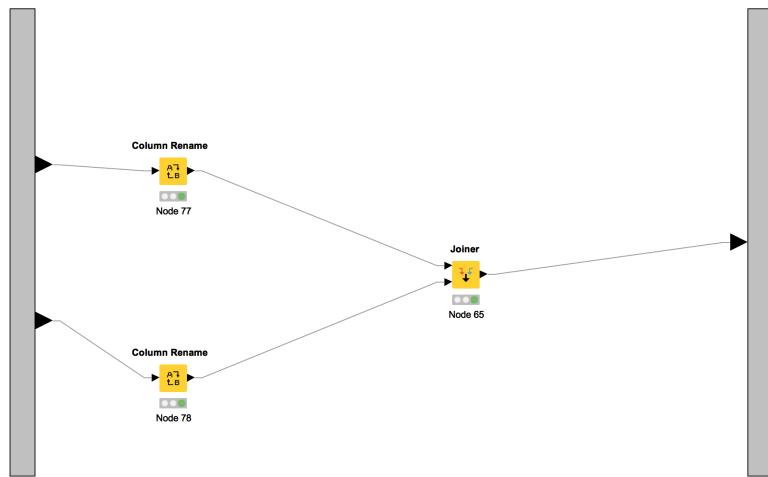


Figura 2.10: Unión de los datos de dos algoritmos, Decision Tree y Gradient Boosted.

Algorithm	TruePositives	FalsePositives	TrueNegatives	FalseNegatives	Recall	Precision	Specificity	F-measure	G-mean	Accuracy	AUC
Gradient Boosted	7376	2157	34998	4311	0.6311	0.7737	0.9419	0.6951	0.7710	0.8675	0.9191

Tabla 2.3: Tabla de datos del algoritmo Gradient Boosted.

		<=50K	>50K
<=50K	>50K		
<=50K	34998	2157	
>50K	4311	7376	

Tabla 2.4: Confusion Matrix del algoritmo Gradient Boosted.

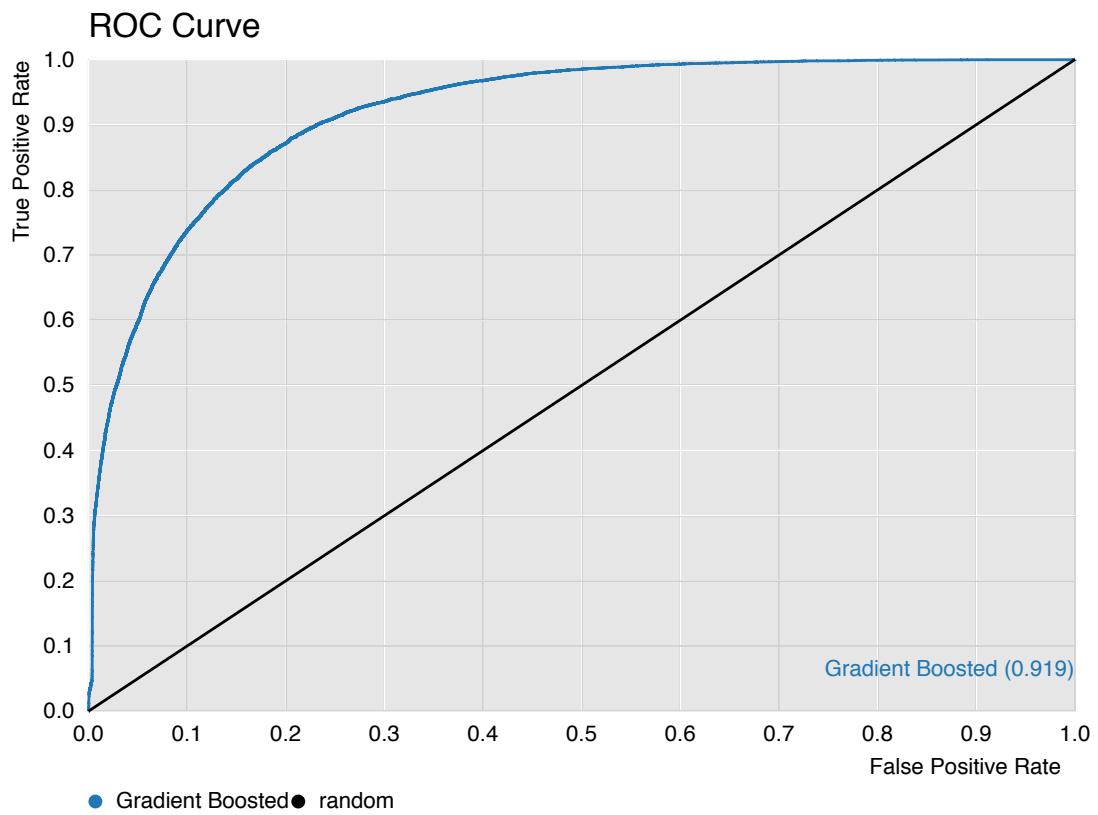


Figura 2.11: Gráfica ROC del algoritmo Gradient Boosted.

2.3. Random Forest

El tercer algoritmo que usaremos será Random Forest[4], el cuál aprende un bosque aleatorio, que consiste en un número definido de árboles de decisión.

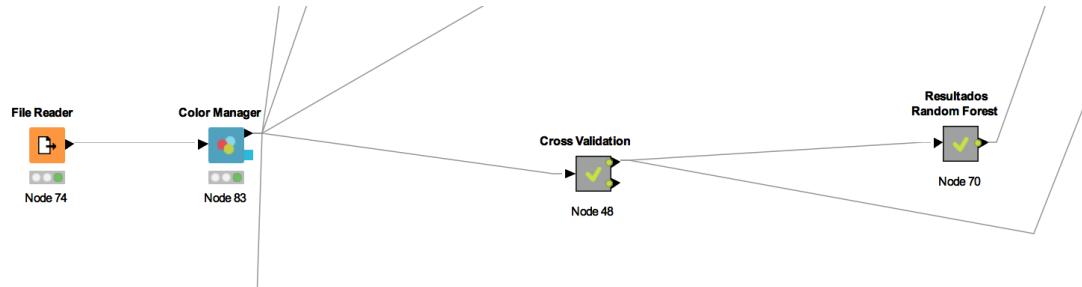


Figura 2.12: Workflow del algoritmo Random Forest.

El flujo de trabajo de todos los algoritmos comienza en el nodo File Reader (Lector de fichero) donde se cargan los datos, seguidamente tendremos el nodo Color Manager (Gestor de color) para diferenciar por colores las instancias con número de ingresos mayor o igual a 50K de las instancias con número de ingresos menor que 50K.

Una vez nos encontramos en este punto podemos ver las diferencias significativas entre los distintos algoritmos:

El nodo Cross Validation:

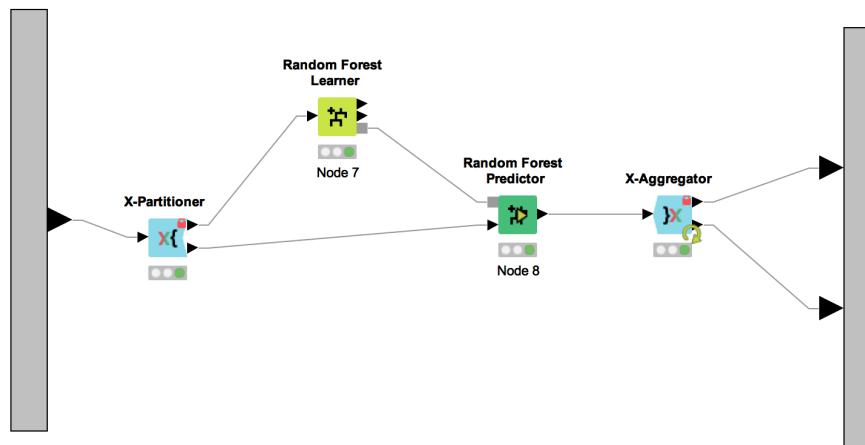


Figura 2.13: Validación cruzada aplicando el algoritmo Random Forest.

Dentro de este nodo Cross Validation como podemos ver está el nodo "X-Partitioner", que básicamente nos permite configurar el número de validaciones (En nuestro caso 5.) y el muestreo estratificado. Después tenemos los nodos del algoritmo, en primer lugar el Random Forest Learner (nodo encargado del entrenamiento) y en segundo lugar el nodo Random Forest Predictor el cual predice patrones de acuerdo a los árboles individuales en un modelo de bosque aleatorio. Y por último nos encontramos con el nodo "Y-Aggregator" que se encarga de volver a reproducir los tres nodos anteriores como si de un bucle se tratara, y también se encarga de sacar una tabla con las predicciones.

Posterior al nodo Cross Validation tenemos el nodo Resultados Random Forest, que contiene la siguiente información:

En primer lugar nos encontramos con el nodo "Scorer" que nos permite mostrar la matriz de confusión y las estadísticas de precisión del algoritmo. Los nodos posteriores tienen la única función de filtrar estos datos y cambiar el identificador de la clase a predecir por el nombre del algoritmo que le hayamos aplicado, de tal forma que podamos identificar los resultados obtenidos por cualquier algoritmo.

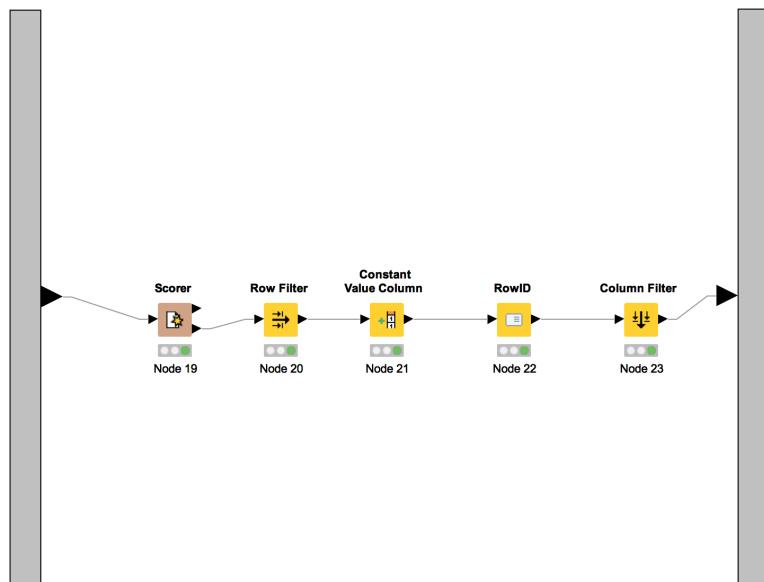


Figura 2.14: Filtrado de los datos de salida del algoritmo Random Forest.

Paralelamente a este nodo llamado Resultados Random Forest, tenemos otro llamado Join Data, que basicamente une los resultados de 2 algoritmos para poder realizar la Gráfica ROC más adelante de forma comparativa, en donde se refleje en una única gráfica el AUC (Área bajo la curva) de todos los algoritmos. Esto se verá mas adelante, por ello en este apartado simplemente comento el porque de la existencia de tal nodo.

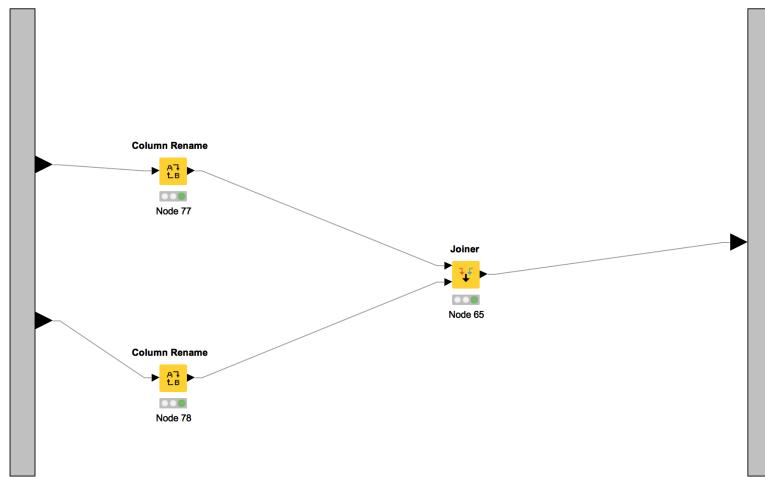


Figura 2.15: Unión de los datos de dos algoritmos, Random Forest y Naive Bayes.

Algorithm	TruePositives	FalsePositives	TrueNegatives	FalseNegatives	Recall	Precision	Specificity	F-measure	G-mean	Accuracy	AUC
Random Forest	7127	1837	35318	4560	0.6098	0.7950	0.9505	0.6902	0.7613	0.8690	0.9108

Tabla 2.5: Tabla de datos del algoritmo Random Forest.

		<=50K	>50K
<=50K	35318		1837
>50K	4560		7127

Tabla 2.6: Confusion Matrix del algoritmo Random Forest.

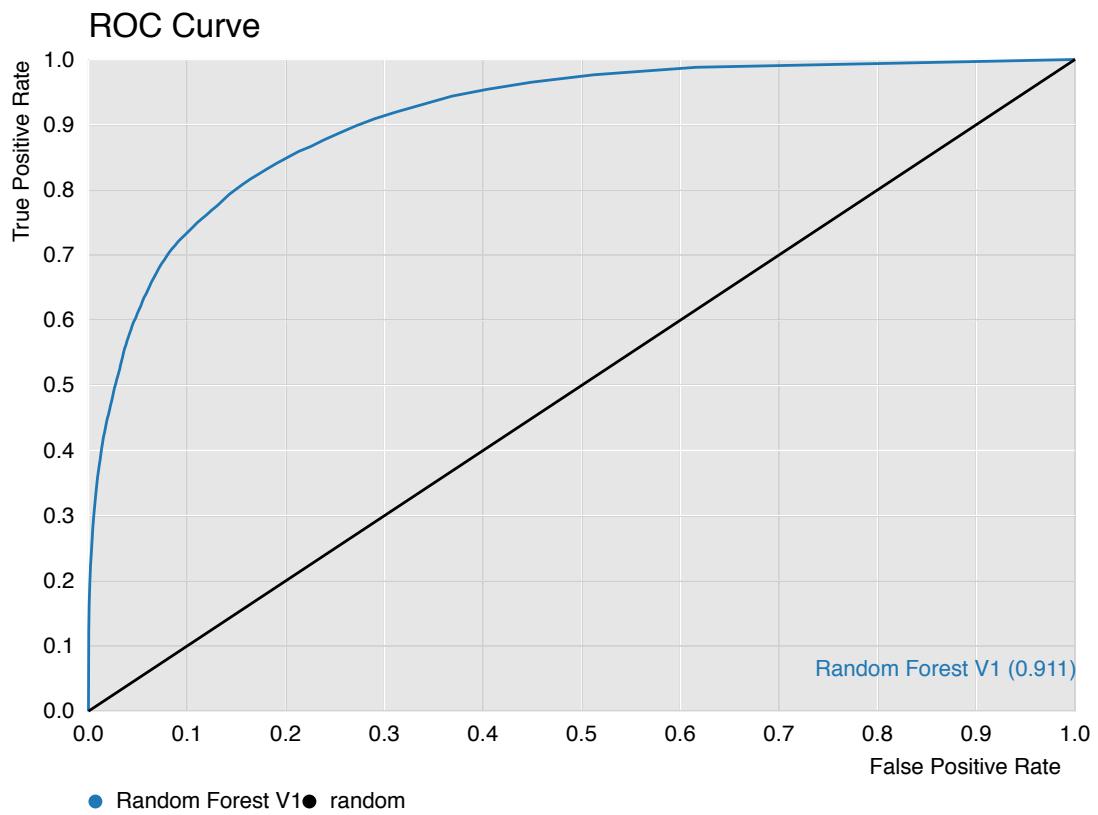


Figura 2.16: Gráfica ROC del algoritmo Random Forest.

2.4. Naive Bayes

El tercer algoritmo que usaremos será Naive Bayes[5] que es un clasificador probabilístico, está basada en el teorema de Bayes.

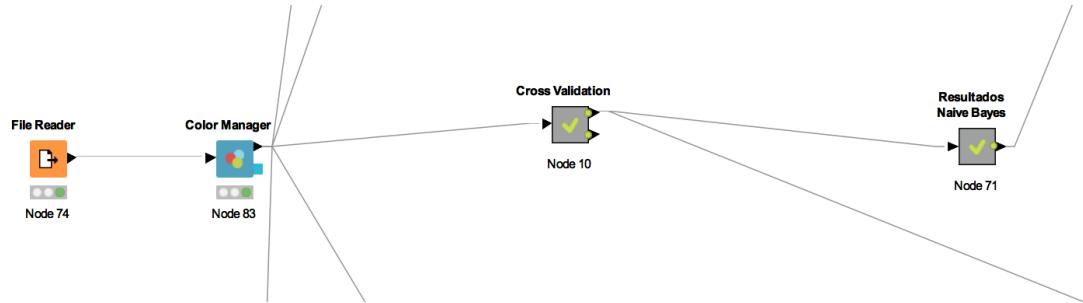


Figura 2.17: Workflow del algoritmo Naive Bayes.

El flujo de trabajo de todos los algoritmos comienza en el nodo File Reader (Lector de fichero) donde se cargan los datos, seguidamente tendremos el nodo Color Manager (Gestor de color) para diferenciar por colores las instancias con número de ingresos mayor o igual a 50K de las instancias con número de ingresos menor que 50K.

Una vez nos encontramos en este punto podemos ver las diferencias significativas entre los distintos algoritmos:

El nodo Cross Validation:

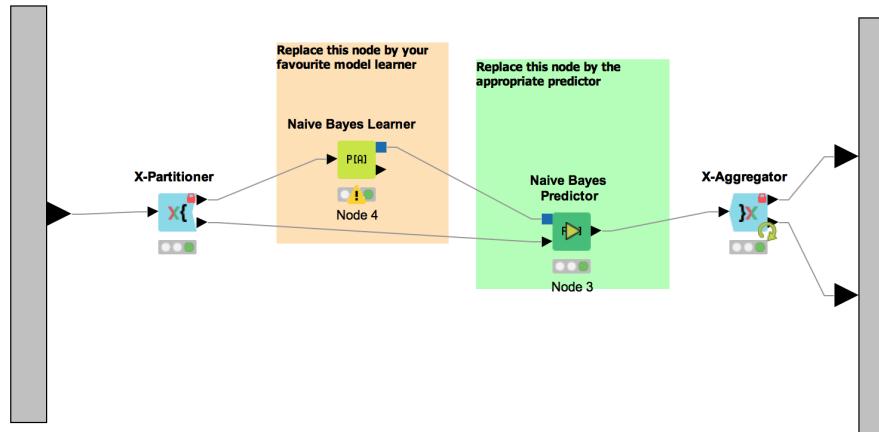


Figura 2.18: Validación cruzada aplicando el algoritmo Naive Bayes.

Dentro de este nodo Cross Validation como podemos ver está el nodo "X-Partitioner", que básicamente nos permite configurar el número de validaciones (En nuestro caso 5.) y el muestreo estratificado. Después tenemos los nodos del algoritmo, en primer lugar el Naive Bayes Learner (nodo encargado del entrenamiento) el cual crea un modelo bayesiano y en segundo lugar el nodo Naive Bayes Predictor el cual predice la clase basada en el modelo aprendido .Y por último nos encontramos con el nodo "Y-Aggregator" que se encarga de volver a reproducir los tres nodos anteriores como si de un bucle se tratara, y también se encarga de sacar una tabla con las predicciones.

Posterior al nodo Cross Validation tenemos el nodo Resultados Naive Bayes, que contiene la siguiente información:

En primer lugar nos encontramos con el nodo "Scorer" que nos permite mostrar la matriz de confusión y las estadísticas de precisión del algoritmo. Los nodos posteriores tienen la única función de filtrar estos datos y cambiar el identificador de la clase a predecir por el nombre del algoritmo que le hayamos aplicado, de tal forma que podamos identificar los resultados obtenidos por cualquier algoritmo.

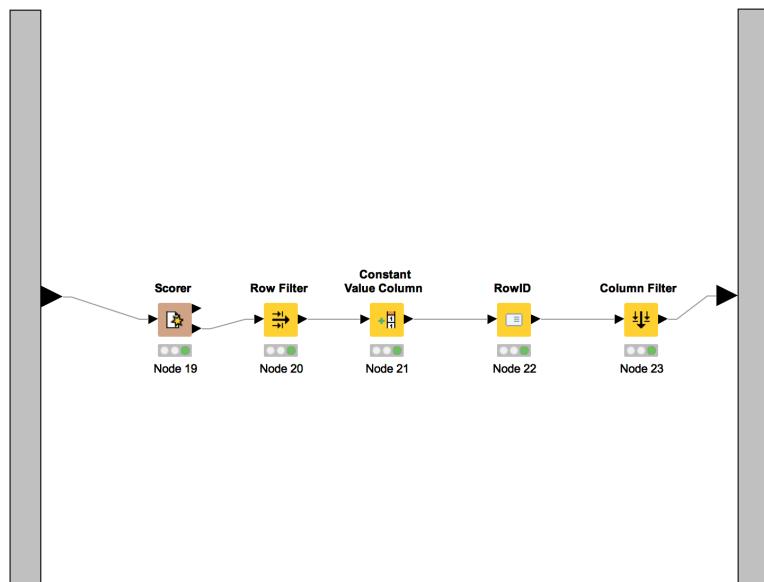


Figura 2.19: Filtrado de los datos de salida del algoritmo Naive Bayes.

Paralelamente a este nodo llamado Resultados Naive Bayes, tenemos otro llamado Join Data, que basicamente une los resultados de 2 algoritmos para poder realizar la Gráfica ROC más adelante de forma comparativa, en donde se refleje en una única gráfica el AUC (Área bajo la curva) de todos los algoritmos. Esto se verá mas adelante, por ello en este apartado simplemente comento el porque de la existencia de tal nodo.

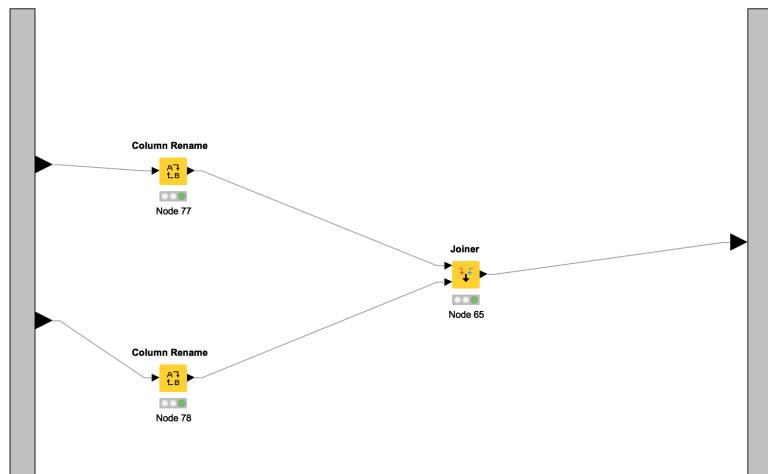


Figura 2.20: Unión de los datos de dos algoritmos, Random Forest y Naive Bayes.

Algorithm	TruePositives	FalsePositives	TrueNegatives	FalseNegatives	Recall	Precision	Specificity	F-measure	G-mean	Accuracy	AUC
Naive Bayes	8959	6279	30876	2728	0.7665	0.5879	0.8310	0.6654	0.7981	0.8155	0.891

Tabla 2.7: Tabla de datos del algoritmo Naive Bayes.

		<=50K	>50K
<=50K	30876		6279
>50K	2728		8959

Tabla 2.8: Confusion Matrix del algoritmo Naive Bayes.

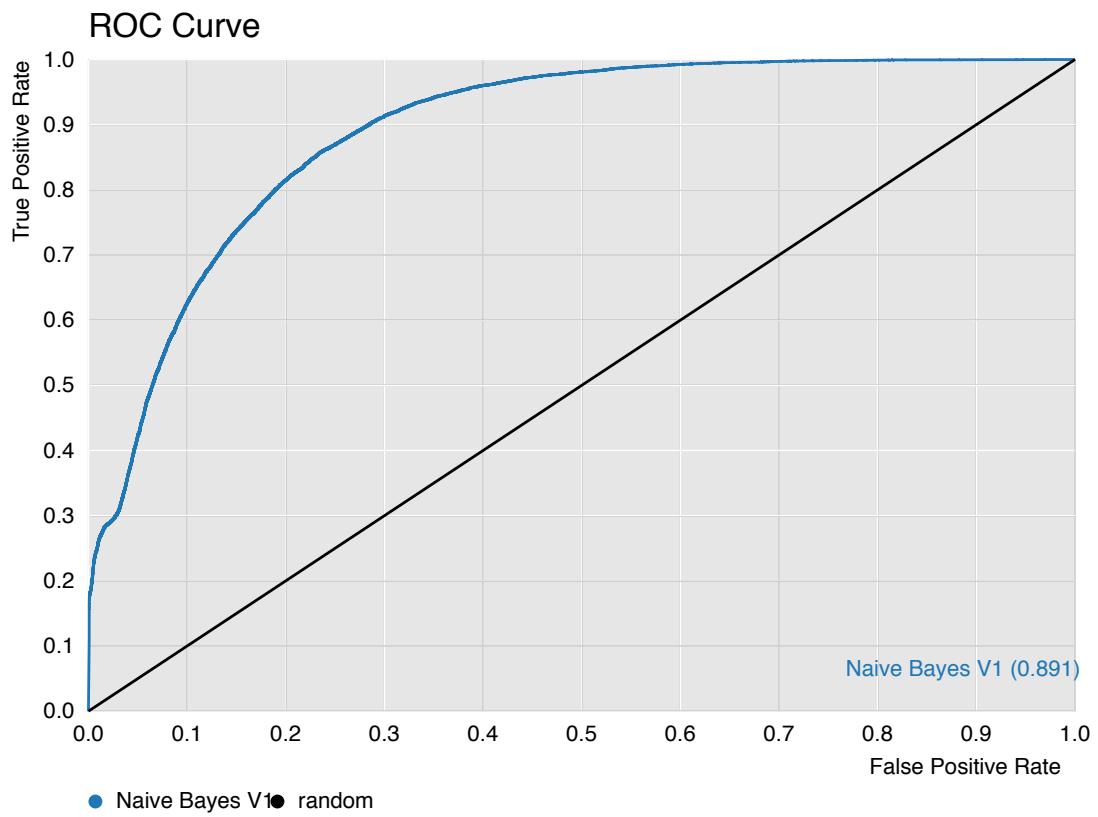


Figura 2.21: Gráfica ROC del algoritmo Naive Bayes.

2.5. K-NN

El tercer algoritmo que usaremos será K-NN (Vecino más cercano)[6], básicamente calcula las distancias con todos los miembros de la base de datos (o colección) y se toma un número K que son los que obtuvieron la menor distancia.

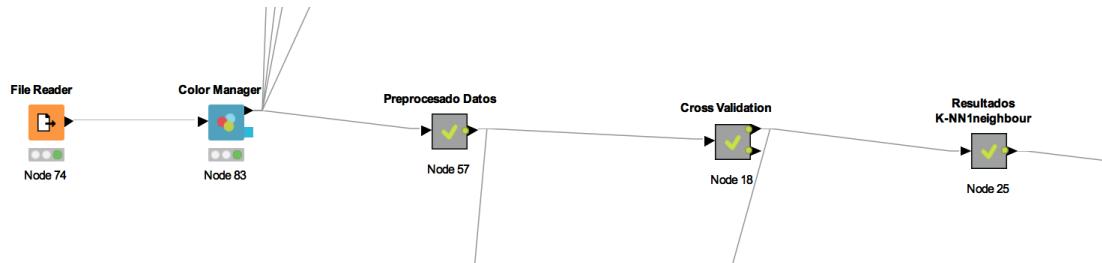


Figura 2.22: Workflow del algoritmo K-NN.

El flujo de trabajo de todos los algoritmos comienza en el nodo File Reader (Lector de fichero) donde se cargan los datos, seguidamente tendremos el nodo Color Manager (Gestor de color) para diferenciar por colores las instancias con número de ingresos mayor o igual a 50K de las instancias con número de ingresos menor que 50K.

Una vez nos encontramos en este punto podemos ver las diferencias significativas entre los distintos algoritmos:

El preprocessado de los datos:

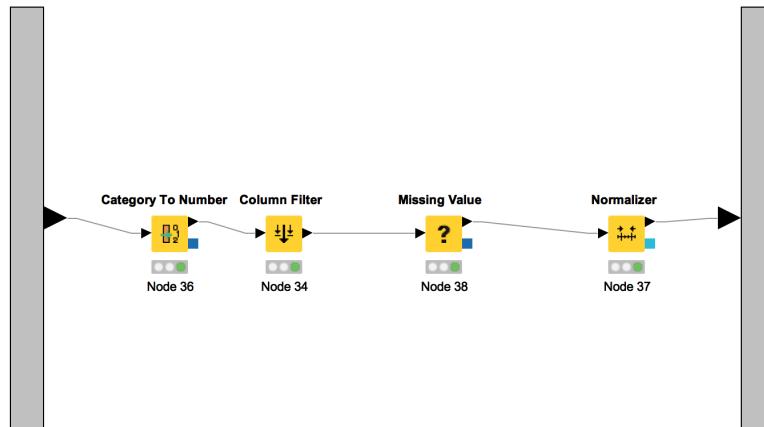


Figura 2.23: Preprocesado de los datos.

La parte de preprocessado es necesaria para el funcionamiento de los algoritmos K-NN y Red Neuronal. Por ejemplo estos algoritmos requieren que las columnas categóricas con datos de tipo cadena (String) sean transformadas a nuevas columnas numéricas. Este papel lo desempeña el nodo "Category to Number". Tras este nodo nos encontramos con el nodo "column filter"[9], que nos permite filtrar las columnas de tal forma que podemos eliminar desde este punto en adelante los datos innecesarios. Para poder ejecutar los algoritmos también es necesario utilizar el nodo "Missing Value" que nos permite tratar los valores perdidos de una forma diferente para que no haya errores, en este caso lo que se hace es escoger la media en donde haya valores perdidos. Este comportamiento será modificado en siguientes secciones, en busca de un mejor modelo. Y por último tenemos el nodo "normalizer"[8], el cual normaliza los valores de todas las columnas numéricas.

El nodo Cross Validation:

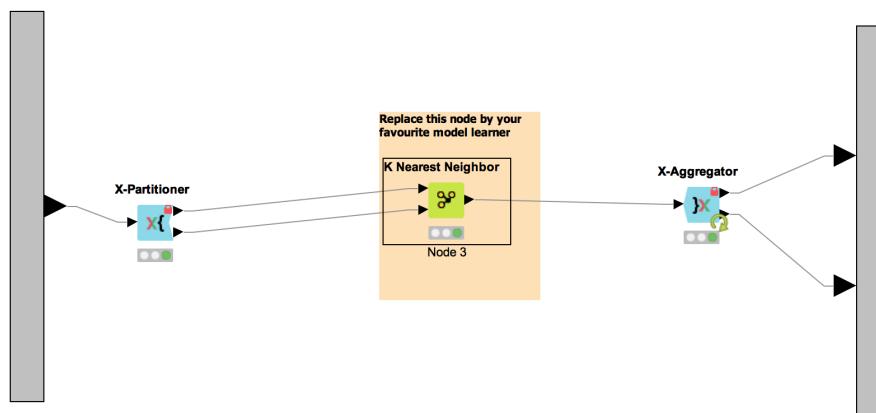


Figura 2.24: Validación cruzada aplicando el algoritmo K-NN.

Dentro de este nodo Cross Validation como podemos ver está el nodo "X-Partitioner", que básicamente nos permite configurar el número de validaciones (En nuestro caso 5.) y el muestreo estratificado. Despues tenemos el nodo K nearest neighbour, este nodo tiene la peculiaridad de que no necesita un nodo previo para los datos de entrenamiento, hace ambas cosas el mismo nodo (Training y tests). Y por último nos encontramos con el nodo "Y-Aggregator" que se encarga de volver a reproducir los tres nodos anteriores como si de un bucle se tratara, y también se encarga de sacar una tabla con las predicciones.

Posterior al nodo Cross Validation tenemos el nodo Resultados K-NN, que contiene la siguiente información:

En primer lugar nos encontramos con el nodo "Scorer" que nos permite mostrar la matriz de confusión y las estadísticas de precisión del algoritmo. Los nodos posteriores tienen la única función de filtrar estos datos y cambiar el identificador de la clase a predecir por el nombre del algoritmo que le hayamos aplicado, de tal forma que podamos identificar los resultados obtenidos por cualquier algoritmo.

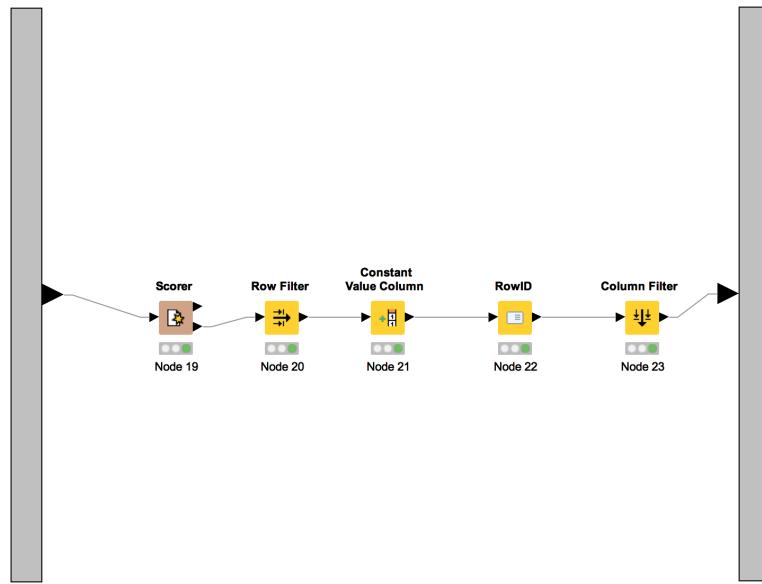


Figura 2.25: Filtrado de los datos de salida del algoritmo K-NN.

Paralelamente a este nodo llamado Resultados K-NN, tenemos otro llamado Join Data, que basicamente une los resultados de 2 algoritmos para poder realizar la Gráfica ROC más adelante de forma comparativa, en donde se refleje en una única gráfica el AUC (Área bajo la curva) de todos los algoritmos. Esto se verá mas adelante, por ello en este apartado simplemente comento el porque de la existencia de tal nodo.

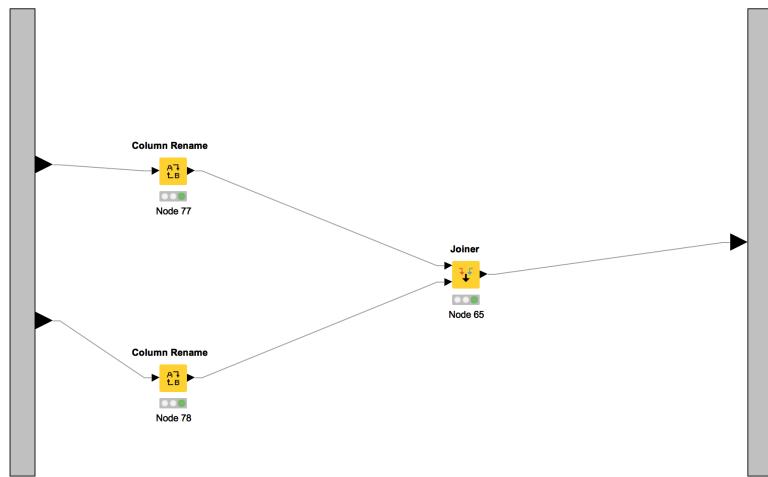


Figura 2.26: Unión de los datos de dos algoritmos, K-NN y Red Neuronal.

Algorithm	TruePositives	FalsePositives	TrueNegatives	FalseNegatives	Recall	Precision	Specificity	F-measure	G-mean	Accuracy	AUC
K-NN	6588	4962	32193	5099	0.5637	0.5703	0.8664	0.8664	0.6988	0.7940	0.7157

Tabla 2.9: Tabla de datos del algoritmo K-NN.

		<=50K	>50K
<=50K	32193		4962
>50K	5099		6588

Tabla 2.10: Confusion Matrix del algoritmo K-NN.

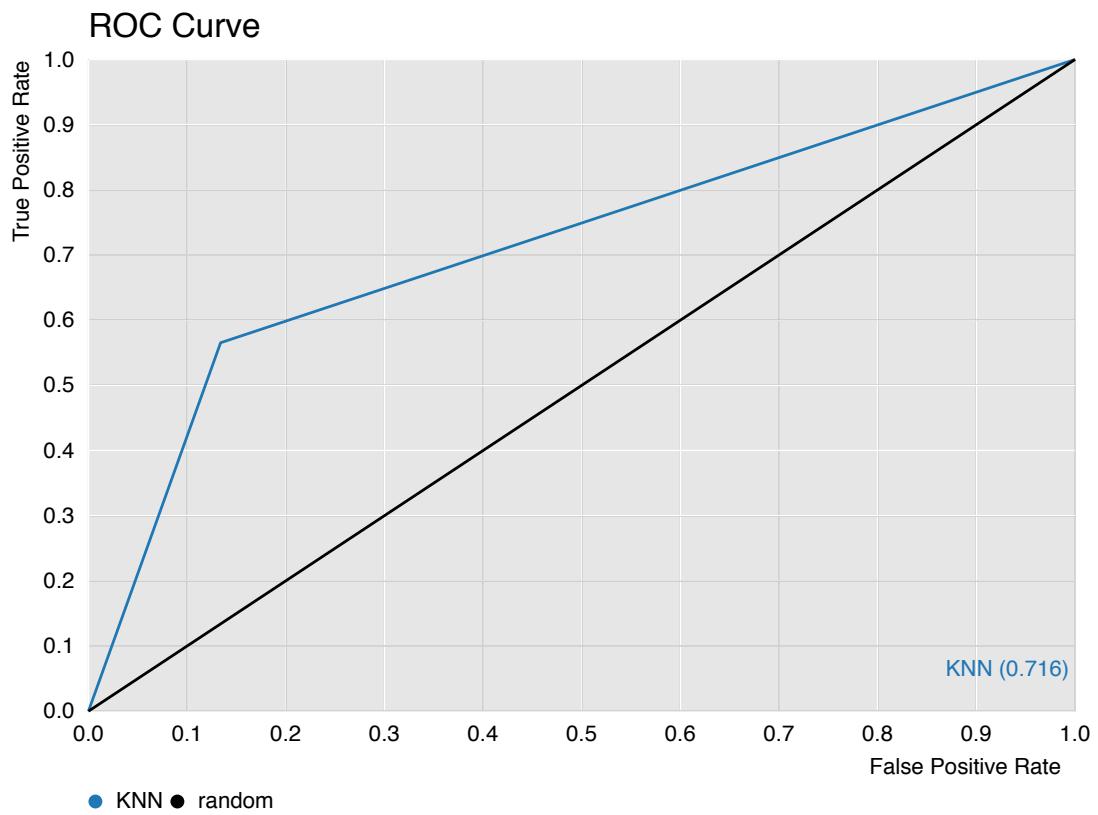


Figura 2.27: Gráfica ROC del algoritmo K-NN.

2.6. Red Neuronal

El tercer algoritmo que usaremos será Red Neuronal[7], un modelo computacional basado en un gran conjunto de unidades neuronales simples. Cada neuronal está conectada a otras, pudiendo incrementar los enlaces entre ellas el estado de activación de las adyacentes.

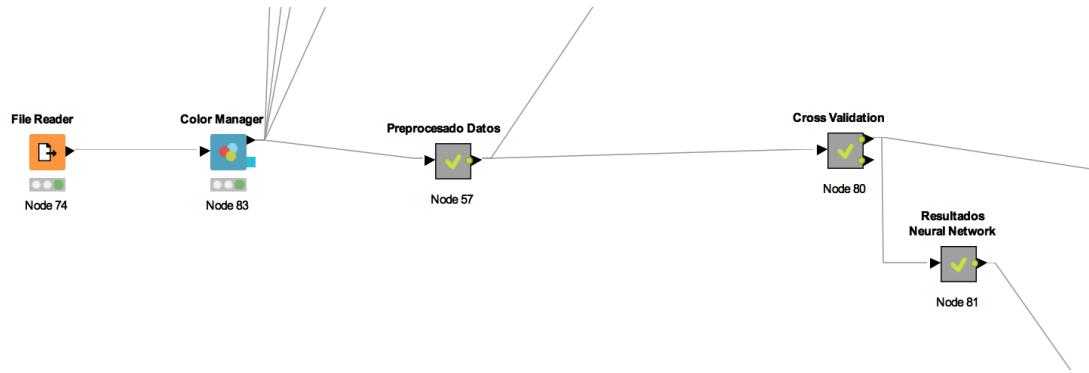


Figura 2.28: Workflow del algoritmo Red Neuronal.

El flujo de trabajo de todos los algoritmos comienza en el nodo File Reader (Lector de fichero) donde se cargan los datos, seguidamente tendremos el nodo Color Manager (Gestor de color) para diferenciar por colores las instancias con número de ingresos mayor o igual a 50K de las instancias con número de ingresos menor que 50K.

Una vez nos encontramos en este punto podemos ver las diferencias significativas entre los distintos algoritmos:

El preprocessado de los datos:

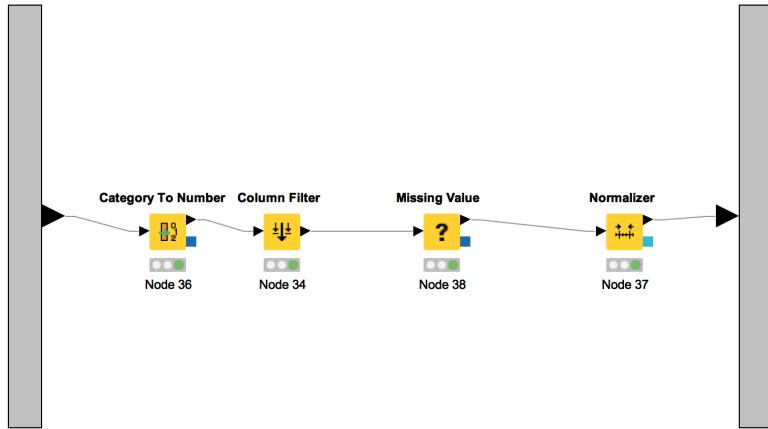


Figura 2.29: Preprocesado de los datos.

La parte de preprocessado es necesaria para el funcionamiento de los algoritmos K-NN y Red Neuronal. Por ejemplo estos algoritmos requieren que las columnas categóricas con datos de tipo cadena (String) sean transformadas a nuevas columnas numéricas. Este papel lo desenvuelve el nodo "Category to Number". Tras este nodo nos encontramos con el nodo "column filter"[9], que nos permite filtrar las columnas de tal forma que podemos eliminar desde este punto en adelante los datos innecesarios. Para poder ejecutar los algoritmos también es necesario utilizar el nodo "Missing Value" que nos permite tratar los valores perdidos de una forma diferente para que no haya errores , en este caso lo que se hace es escoger la media en donde haya valores perdidos. Este comportamiento será modificado en siguientes secciones, en busca de un mejor modelo. Y por último tenemos el nodo "normalizer"[8], el cuál normaliza los valores de todas las columnas numéricas.

El nodo Cross Validation:

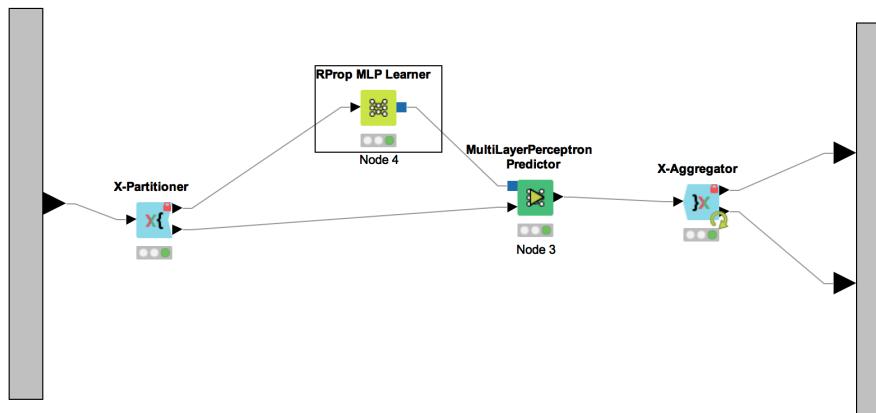


Figura 2.30: Validación cruzada aplicando el algoritmo Red Neuronal.

Dentro de este nodo Cross Validation como podemos ver está el nodo "X-Partitioner", que básicamente nos permite configurar el número de validaciones (En nuestro caso 5.) Después tenemos el nodo Rprop MLP Learner que básicamente implementa el algoritmo Rprop para redes feedforward multicapa. Justo tras este nodo nos encontraríamos con el nodo MultiLayer Perception Predictor que en base a un modelo, calcula los valores de salida. Y por último nos encontramos con el nodo "Y-Aggregator" que se encarga de volver a reproducir los tres nodos anteriores como si de un bucle se tratara, y también se encarga de sacar una tabla con las predicciones.

Posterior al nodo Cross Validation tenemos el nodo Resultados Red Neuronal, que contiene la siguiente información:

En primer lugar nos encontramos con el nodo "Scorer" que nos permite mostrar la matriz de confusión y las estadísticas de precisión del algoritmo. Los nodos posteriores tienen la única función de filtrar estos datos y cambiar el identificador de la clase a predecir por el nombre del algoritmo que le hayamos aplicado, de tal forma que podamos identificar los resultados obtenidos por cualquier algoritmo.

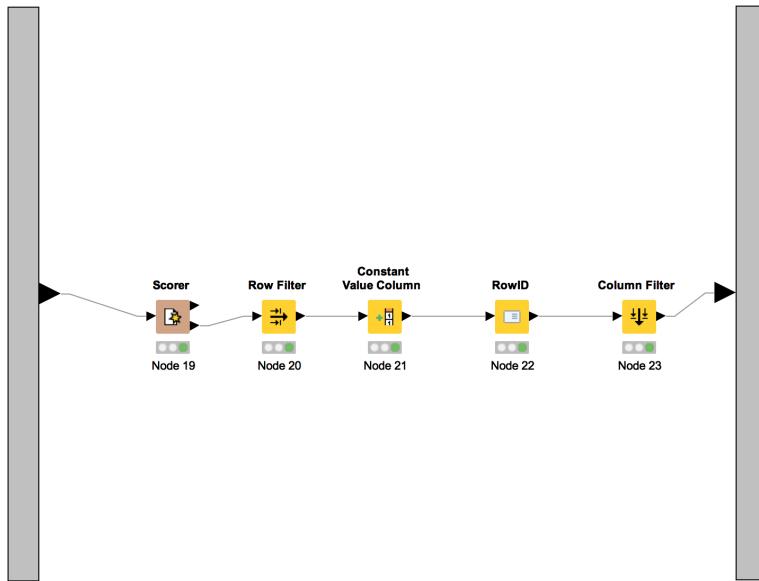


Figura 2.31: Filtrado de los datos de salida del algoritmo Red Neuronal.

Paralelamente a este nodo llamado Resultados Red Neuronal, tenemos otro llamado Join Data, que basicamente une los resultados de 2 algoritmos para poder realizar la Gráfica ROC más adelante de forma comparativa, en donde se refleje en una única gráfica el AUC (Área bajo la curva) de todos los algoritmos. Esto se verá mas adelante, por ello en este apartado simplemente comento el porque de la existencia de tal nodo.

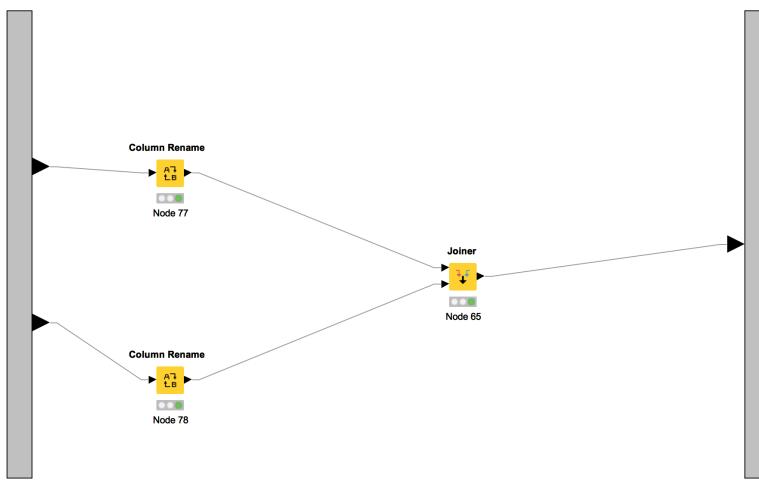


Figura 2.32: Unión de los datos de dos algoritmos, K-NN y Red Neuronal.

Algorithm	TruePositives	FalsePositives	TrueNegatives	FalseNegatives	Recall	Precision	Specificity	F-measure	G-mean	Accuracy	AUC
Neural Network	6672	2464	34691	5015	0.5708	0.7302	0.9336	0.6408	0.7300	0.8468	0.8959

Tabla 2.11: Tabla de datos del algoritmo Neural Network.

	<=50K	>50K
<=50K	34691	2464
>50K	5015	6672

Tabla 2.12: Confusion Matrix del algoritmo Neural Network.

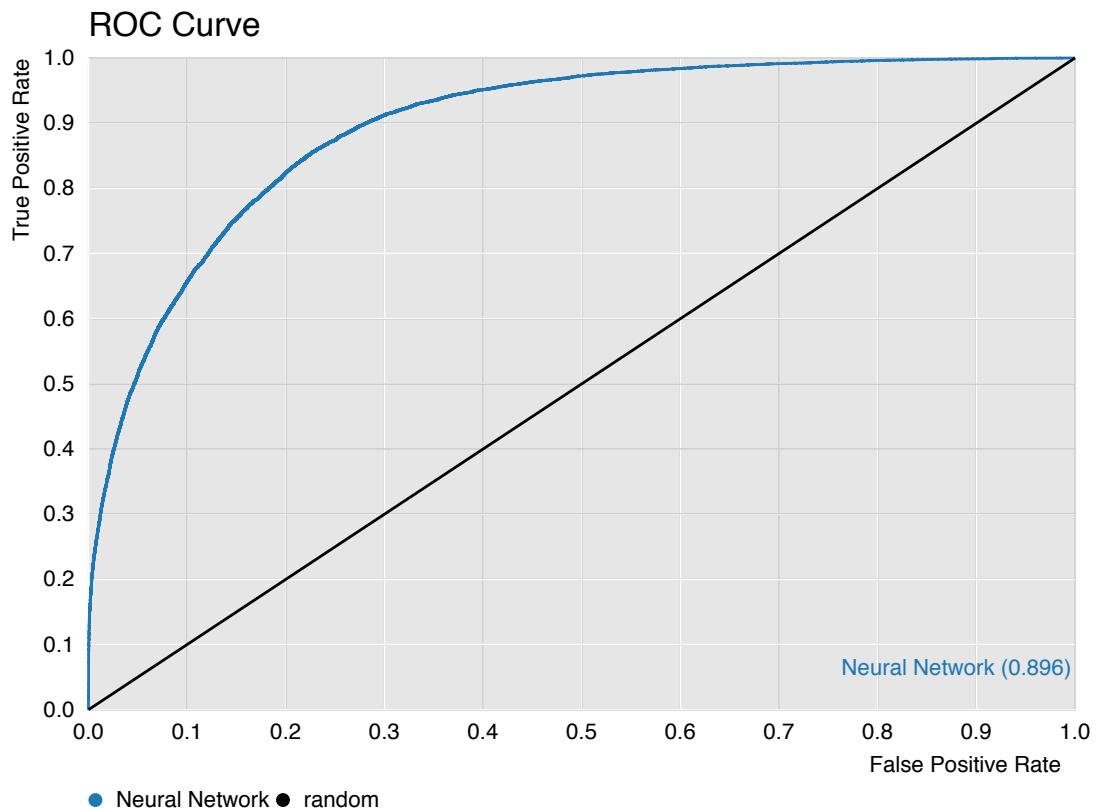


Figura 2.33: Gráfica ROC del algoritmo Red Neuronal.

3. Análisis de los resultados

En este apartado se va a proceder al análisis comparativo de los distintos algoritmos teniendo en cuenta la tabla final de resultados, la gráfica ROC y las distintas gráficas dispuestas para el análisis.

En primer lugar tenemos la tabla final de resultados de todos los modelos juntos:

Algorithm	TruePositives	FalsePositives	TrueNegatives	FalseNegatives	Recall	Precision	Specificity	F-measure	G-mean	Accuracy	AUC
Decision Tree	6969	3744	32991	4541	0.6054	0.6505	0.8980	0.6271	0.7374	0.8282	0.8055
Gradient Boosted	7376	2157	34998	4311	0.6311	0.7737	0.9419	0.6951	0.7710	0.8675	0.9191
Random Forest	7127	1837	35318	4560	0.6098	0.7950	0.9505	0.6902	0.7613	0.8690	0.9096
Naive Bayes	8959	6279	30876	2728	0.7665	0.5879	0.8310	0.6654	0.7981	0.8155	0.891
K-NN	6588	4962	32193	5099	0.5637	0.5703	0.8664	0.8664	0.6988	0.7940	0.7157
Neural Network	6672	2464	34691	5015	0.5708	0.7302	0.9336	0.6408	0.7300	0.8468	0.8959

Tabla 3.1: Tabla de datos de todos los algoritmos.

En segundo lugar tenemos la tabla ROC de todos los algoritmos:

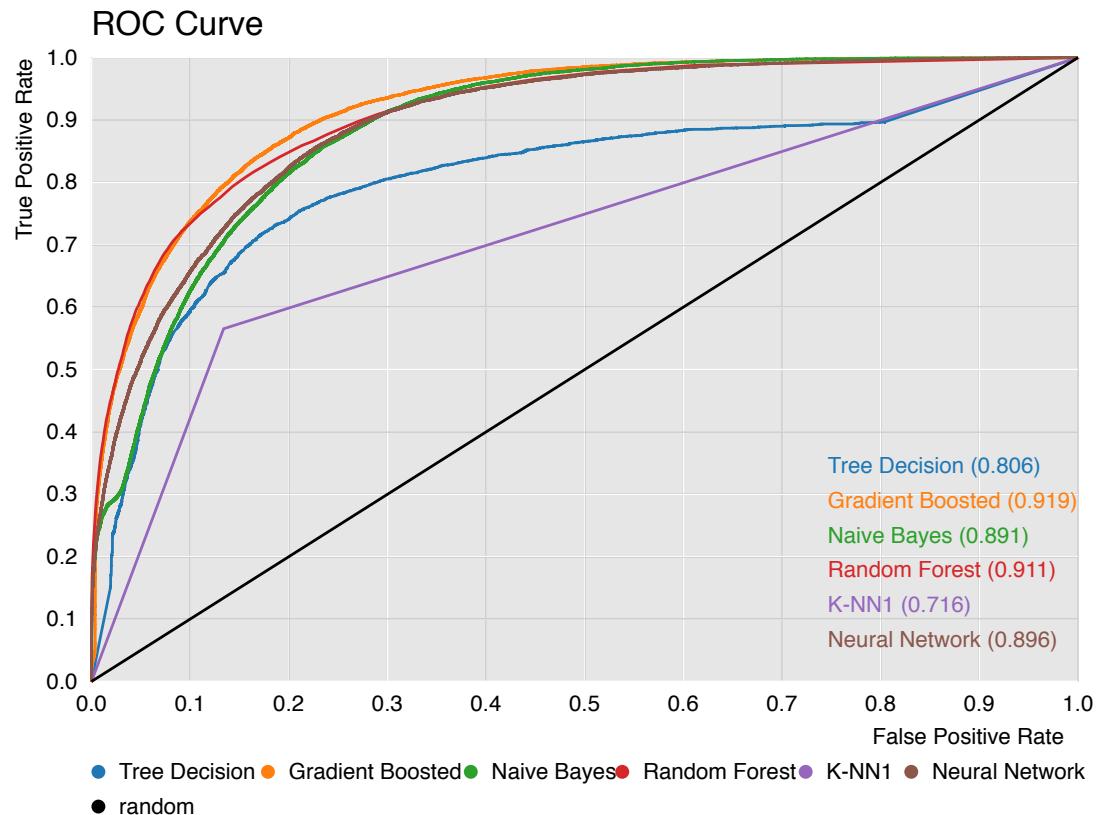


Figura 3.1: Gráfica ROC de todos los algoritmos.

A apartir de la tabla de resultados y de la gráfica ROC, se pueden intuir varias cosas, la primera y más importante es que los algoritmos que mejores resultados modelan son el Gradient Boosted y Random Forest, y muy pegados a ellos tenemos a Neural Network y Naive Bayes. Ya co resultados bastante peores están Decision Tree y KNN.

El que haya una diferencia entre Gradient Boosted ó Random Forest y Decision Tree es probablemente esperable, ya que estos dos primeros algoritmos son más completos. Más completos desde el punto de vista de que por ejemplo Gradient Boosted crea árboles de uno en uno, en donde cada uno corrige errores cometidos por el aprendido previamente, y esto permite que el modelo se vuelva más expresivo. Y también mirándolo desde el punto de vista de Random Forest, este permite crear un modelo más robusto que un único árbol de decisión. También algo que mejora tanto en Gradient Boosted como en Random Forest es que son menos propensos al sobreajuste en los datos de entrenamiento.

En el caso del algoritmo Neural Network tenemos un algoritmo por lo general ideal para problemas complejos, pero por contra obtenemos unos modelos de caja negra, mayor carga computacional y propenso al sobreajuste. En este caso en particular obtenemos un modelo bastante preciso con Neural Network, estando muy a la par con Naive Bayes. Con Naive Bayes tenemos un modelo simple pero poderoso, con una suposición muy restrictiva de la independencia de los atributos. Tenemos por lo tanto frente a frente a dos algoritmos uno algo más costoso computacionalmente como es Neural Network y otro algo más rápido como Naive Bayes.

Por lo que respecta al mal resultado del algoritmo KNN puede estar sujeto a la configuración por defecto del mismo, en donde solamente se tiene en cuenta a un vecino más cercano. Por ello este algoritmo puede estar más penalizado que los demás de salida. Si bien este hecho parece ser coherente, emplazo al lector a las siguientes secciones para visualizar las posibles mejoras del mismo con otras configuraciones, para así hacer un juicio más completo.

En cuanto a los resultados de los distintos algoritmos con respecto al Recall (Porcentaje de todas las observaciones que deban ser clasificadas, lo sean correctamente.) se tiene la siguiente gráfica. En ella se tienen en cuenta variables como los VerdaderosPositivos y FalsosNegativos que son los que intervienen en el cálculo del Recall. En esta gráfica se puede apreciar perfectamente por que unos algoritmos obtienen mejor Recall que otros.
² En primer lugar vamos a proceder con el algoritmo Naive Bayes, que es el que mejor

Parallel Coordinates Plot

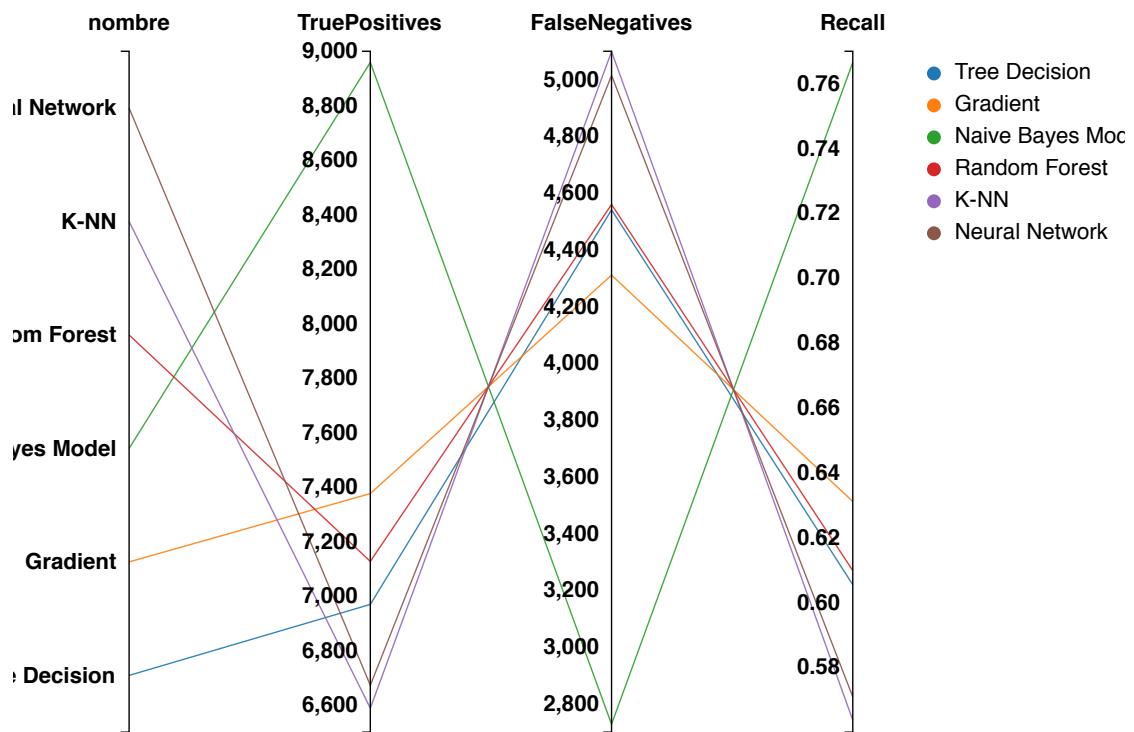


Figura 3.2: Gráfica del Recall de todos los algoritmos, integrando tanto verdaderos positivos como falsos negativos.

Recall obtiene. En la gráfica vemos como obtiene el máximo en verdaderos positivos y el mínimo en falsos negativos.

Más compactados están los demás algoritmos, en donde por ejemplo tenemos el Random Forest que es el tercero que más falsos negativos obtiene, comportándose mejor que otros como el KNN y el Neural Network para obtener mejor Recall. Otro algoritmo que se acerca al Random Forest es el Tree Decision, el cuál obtiene peor Recall por la gran

²Lamento que los nombres a la izquierda no se aprecien bien, pero KNIME la herramienta con la que se han extraído las gráficas, las ha cortado demasiado.

disminución de verdaderos positivos que que obtiene.

Para acabar con el Recall, decir que Neural Network y KNN obtienen el peor porcentaje, y es precisamente por el hecho de que tienen los mayores valores en falsos negativos y los menores en verdaderos positivos. De tal forma que poniendo como analogía a la medicina como es de costumbre en estos casos, podemos decir que estos dos algoritmos tienen el peor comportamiento con esta configuración por defecto para nuestros datos para clasificar correctamente a un individuo enfermo.

En cuanto a los resultados de los distintos algoritmos con respecto al Specificity se tiene la siguiente gráfica. En ella se tienen en cuenta variables como los FalsosPositivos y VerdaderosNegativos que son los que intervienen en el cálculo del Specificity. En esta gráfica se puede apreciar perfectamente por que unos algoritmos obtienen mejor Specificity que otros.³ En primer lugar se aprecia como el Random Forest obtiene el mejor valor se-

Parallel Coordinates Plot

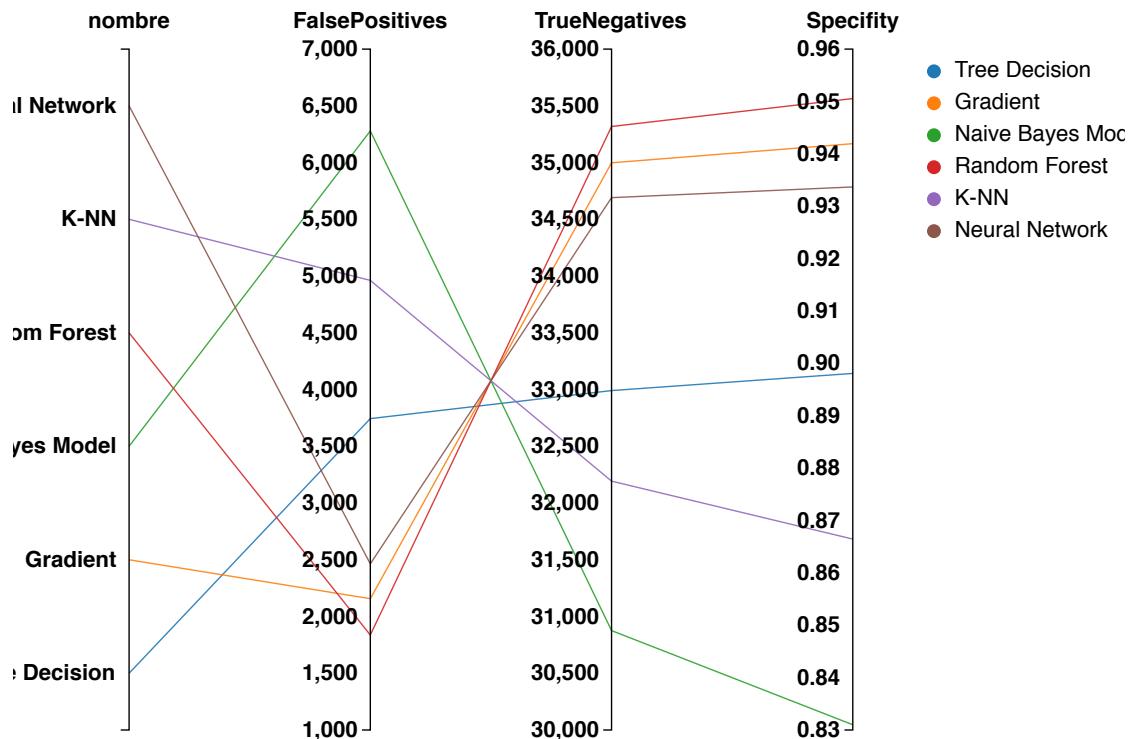


Figura 3.3: Gráfica del Precision de todos los algoritmos, integrando tanto falsos positivos como verdaderos negativos.

guido del Gradient Boosted. Al contrario que ocurría con el Recall, aquí el Naive Bayes se posiciona en último lugar, luego con Naive Bayes obtenemos un modelo totalmente "bipolar", de tal forma que se encuentra con el mejor Recall y el peor Specificity, esto debido al gran aumento de falsos negativos (diagnóstico negativo, y enfermedad presente).

También merece la pena ver como el algoritmo Neural Network en este caso lo vemos en tercera posición, mientras que con el Recall estaba de los últimos.

³Lamento que los nombres a la izquierda no se aprecien bien, pero KNIME la herramienta con la que se han extraído las gráficas, las ha cortado demasiado.

Al final con ambas gráficas lo que podemos ver es que algoritmos son regulares al Recall y Specificity, y por lo tanto sin decaer en ninguno de los dos porcentajes permite ver modelos que se adaptan bastante bien.

En las siguientes gráficas para finalizar nuestro estudio, veremos que algoritmos en cuanto a Accuracy se refiere, tienen un mejor comportamiento.

En primer lugar vemos que el algoritmo Random Forest está en primera posición seguido muy de cerca del Gradient Boosted. Y el algoritmo KNN se mantiene como es de costumbre ya, en última posición, castigado por el bajísimo número de verdaderos positivos.

Naive Bayes tampoco tiene buenos resultados, ya que está muy penalizado por el bajo número de verdaderos negativos.

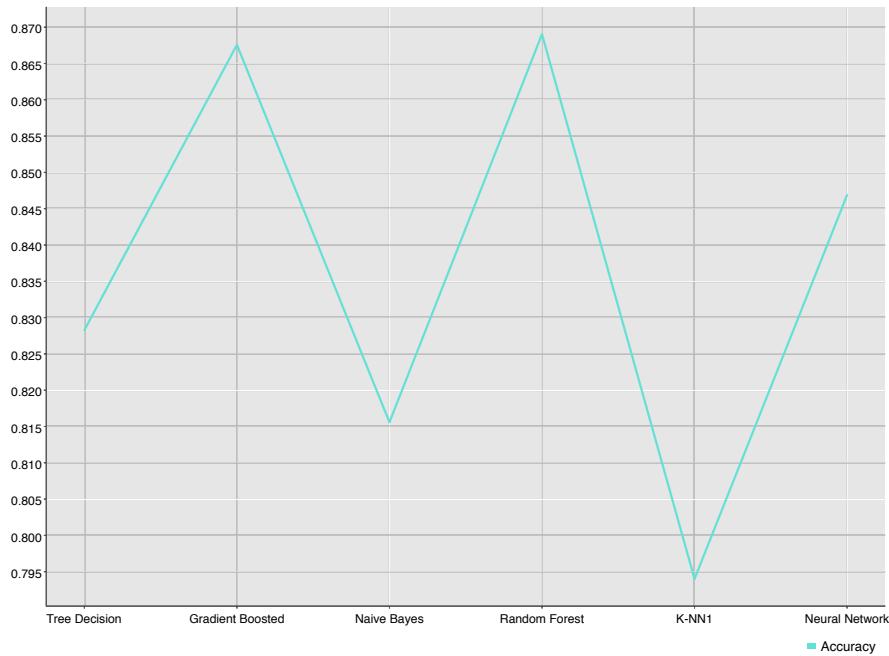


Figura 3.4: Gráfica Accuracy de todos los algoritmos.

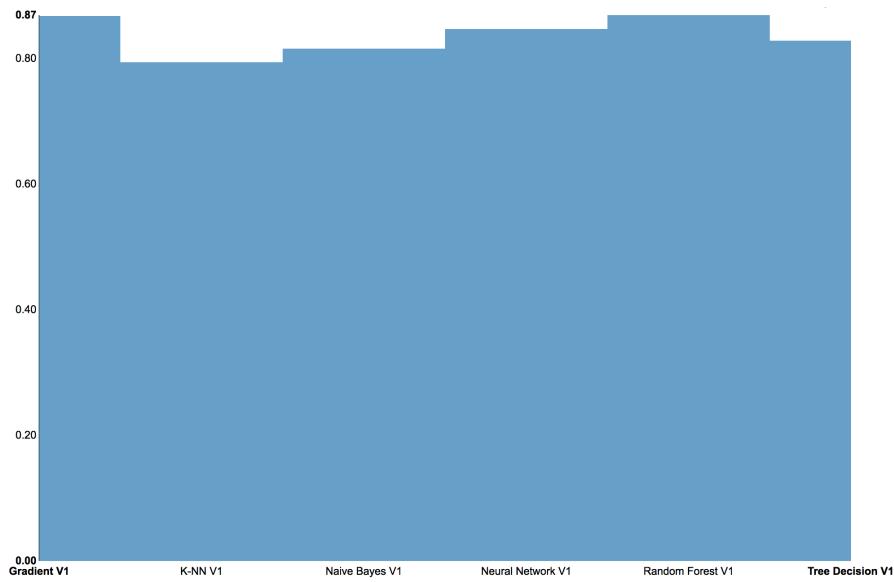


Figura 3.5: Gráfica Accuracy de todos los algoritmos.

En esta gráfica se muestra la misma información que en la anterior de distinta forma, esta gráfica sirve simplemente para visualizar los datos de otra forma.

Para finalizar, podemos decir que los algoritmos Gradient Boosted y sobre todo Random Forest son los que mejor se han comportado en nuestro estudio con las configuraciones por defecto. En distinta sintonía tenemos al KNN, que con esta configuración se adapta muy mal a los datos, obteniendo un modelo muy parco en cuanto a la comparación con los demás.

4. Configuración de algoritmos

En este apartado vamos a comentar los distintos parámetros y la configuración llevada a cabo de los algoritmos usados a lo largo de la práctica. A su vez para algunos algoritmos se mostrará una configuración diferente y se compararán los resultados, en donde podremos ver aspectos como el sobreaprendizaje y la obtención de modelos más simples.

4.1. Decision Tree

En primer lugar tenemos el árbol de decisión con sus valores por defecto, en este caso los parámetros de configuración son del nodo Decision Tree Learner. En primer lugar vemos la columna de la clase, la cuál será inmovilizable como es obvio. Como medida de calidad está indicado el Gini Index y no hay método de poda seleccionado.

En cuanto al mínimo número de registros por nodo viene por defecto en 2 [10]. Y el número de registros para almacenar por vista en 10000.

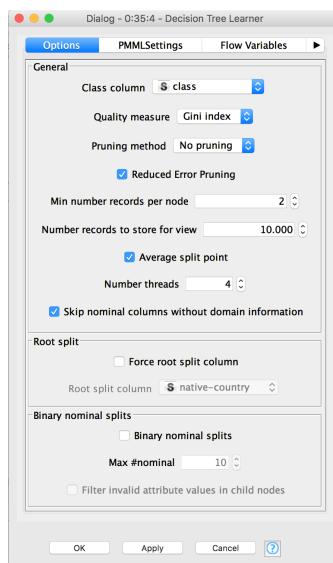


Figura 4.1: Configuración por defecto del algoritmo Decision Tree.

Para intentar sacar un modelo más simple vamos a aumentar el número mínimo de registros por nodo [10], de 2 lo vamos a pasar a 10. Con esto podemos estar ocasionando pérdida de precisión, pero si ocurre que la pérdida es ínfima, podríamos estar ante una mejor solución, por la mayor simplicidad del modelo. En cuanto al número de registros para almacenar por vista lo dejaremos en 10000.

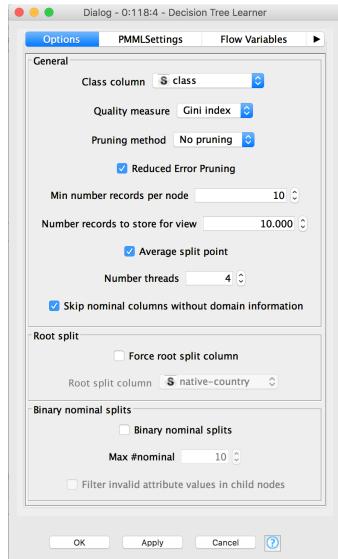


Figura 4.2: Configuración modificada del algoritmo Decision Tree, algoritmo 1).

Después de obtener un modelo simplificado podemos ir más allá, y realizar una prueba con el mínimo número de registros por nodo[10], en 20. Donde podríamos estar ante una mejor solución, por la simplicidad del modelo. En cuanto al mínimo número de registros por nodo viene por defecto en 2. En cuanto al número de registros para almacenar por vista lo dejaremos en 10000.

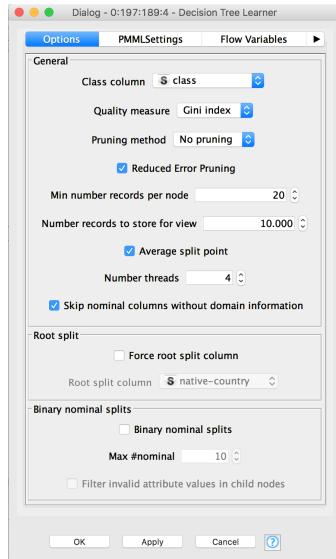


Figura 4.3: Configuración modificada del algoritmo Decision Tree, algoritmo 2.

Para comparar los resultados, extraemos una tabla con los resultados de ambas configuraciones. Y en forma de tabla comparativa podremos extraer conclusiones. A su vez también extraemos una gráfica ROC con las dos curvas pintadas, para apoyar las conclusiones finales sobre ambas configuraciones.

Algorithm	TruePositives	FalsePositives	TrueNegatives	FalseNegatives	Recall	Precision	Specificity	F-measure	G-mean	Accuracy	AUC
Decision Tree V1	6969	3744	32991	4541	0.6054	0.6505	0.8980	0.6271	0.7374	0.8282	0.8055
Decision Tree V2	6963	2995	34070	4688	0.5976	0.6992	0.9191	0.6444	0.7411	0.8422	0.8857
Decision Tree V3	6968	2770	34343	4706	0.5968	0.7155	0.9253	0.6508	0.7431	0.8467	0.8901

Tabla 4.1: Tabla comparativa del algoritmo Decision Tree con las configuraciones por defecto y modificadas.

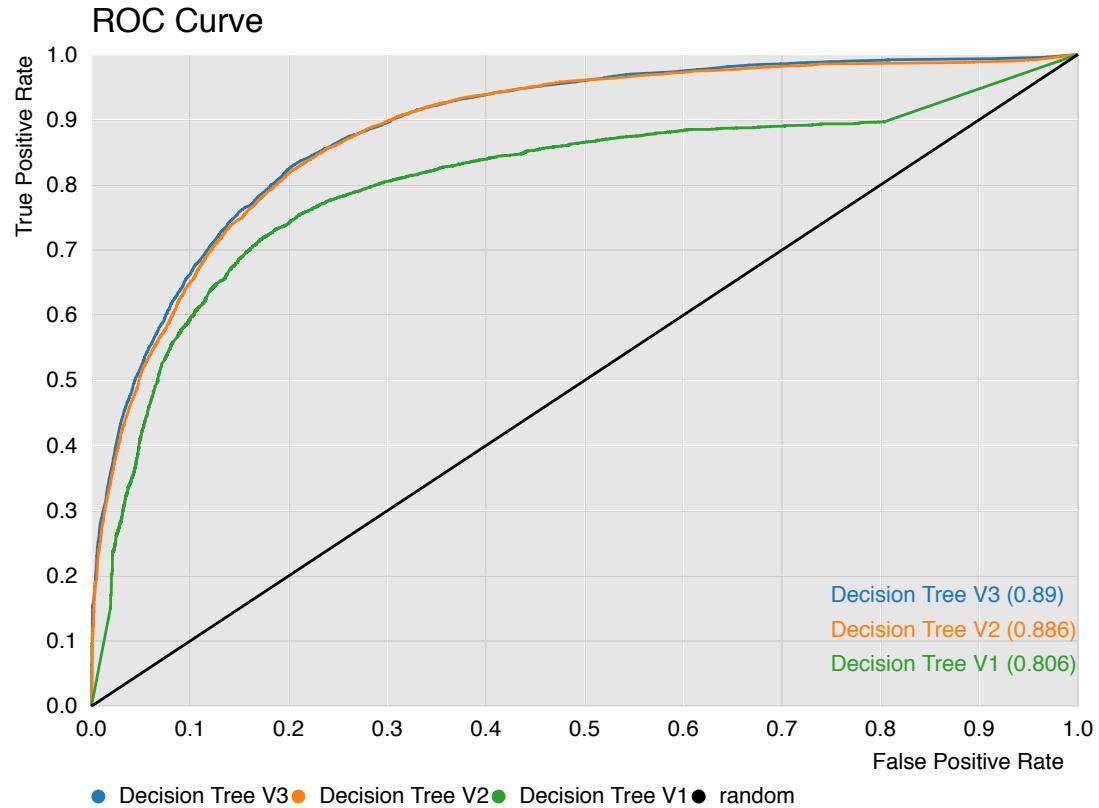


Figura 4.4: Gráfica ROC comparativa del algoritmo Decision Tree con las dos configuraciones.

En este caso podemos ver como el modelo con la configuración por defecto estaba sobreaprendiendo, de tal forma que se adapta demasiado a los datos de entrenamiento y con los datos de test no cumple con las expectativas, una manera clara de demostrarlo es que simplificando el modelo obtenemos mejores resultados. Por lo tanto estaríamos obteniendo un modelo más simplificado y no perdiendo precisión, si no ganándola. Así

nos encontramos con modelos más interpretables.

A su vez podemos observar como el valor del Precision va en aumento conforme simplificamos el modelo, provocado por el aumento de los verdaderos positivos y la disminución de los falsos positivos. En el caso contrario nos encontramos con el Recall, pues aunque los verdaderos positivos van en aumento, los falsos negativos aumentan más rápido, por lo tanto nos encontramos con un Recall inferior. [18]

Para el Area Under Curve vemos perfectamente como va en aumento, para ello podemos apoyarnos en la propia gráfica comparativa. En donde vemos como entre los dos algoritmos posteriormente modificados apenas hay diferencias, pero entre estos y el algoritmo con la configuración por defecto hay una diferencia de 0.08 con el algoritmo de la figura 4.2.

4.2. Gradient Boosted

En esta ocasión nos centramos en el algoritmo Gradient Boosted, en primer lugar vemos los parámetros de configuración por defecto. Parámetros como la selección de atributos (En nuestro caso seleccionaremos todos para todas las pruebas realizadas, la posible modificación de esta parte queda relegada al punto 5), número límite de niveles (La profundidad del árbol); este parámetro será sumamente importante en la posterior modificación del algoritmo para su comparación. También nos encontramos con parámetros como el número de modelos y la tasa de aprendizaje.

En cuanto al límite de profundidad del árbol por defecto viene en 4 [11].

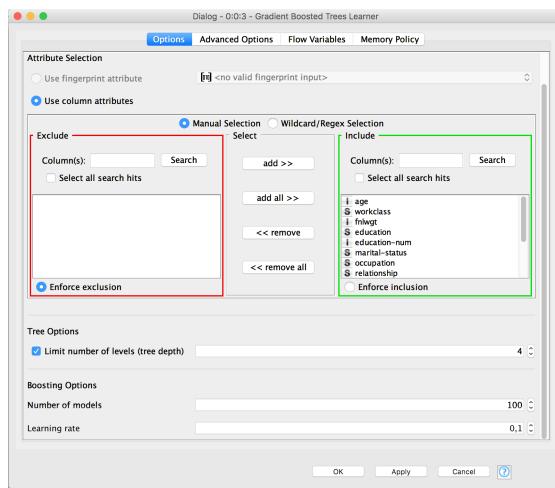


Figura 4.5: Configuración por defecto del algoritmo Gradient Boosted.

Para intentar sacar un modelo más complejo y ver así las posibles diferencias entre ambos modelos, vamos a aumentar el límite de profundidad del árbol, que pasará de 4 a 10.

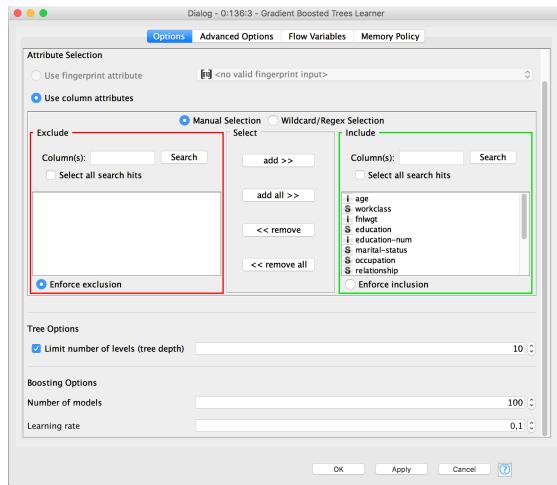


Figura 4.6: Configuración modificada del algoritmo Gradient Boosted, algoritmo 1.

Para intentar sacar un modelo más complejo y ver así las posibles diferencias entre ambos modelos, vamos a aumentar el límite de profundidad del árbol, que pasará de 4 a 10. Aunque en contraposición del aumento de complejidad en la profundidad del árbol, disminuimos la misma, reduciendo el número de modelos.

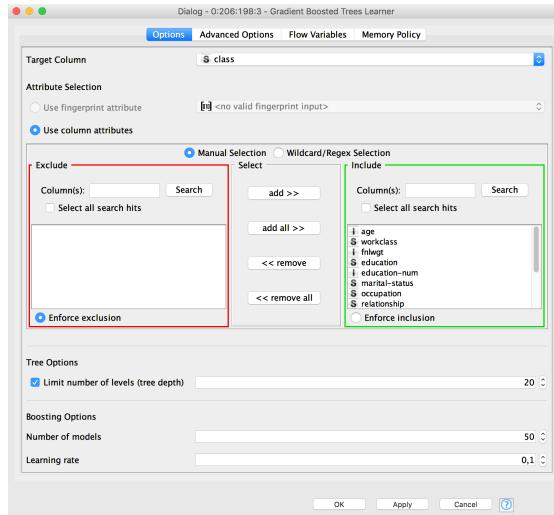


Figura 4.7: Configuración modificada del algoritmo Gradient Boosted, algoritmo 2.

Para comparar los resultados, extraemos una tabla con los resultados de ambas configuraciones. Y en forma de tabla comparativa podremos extraer conclusiones. A su vez también extraemos una gráfica ROC con las dos curvas pintadas, para apoyar las conclusiones finales sobre ambas configuraciones.

Algorithm	TruePositives	FalsePositives	TrueNegatives	FalseNegatives	Recall	Precision	Specificity	F-measure	G-mean	Accuracy	AUC
Gradient Boosted V1	7376	2157	34998	4311	0.6311	0.7737	0.9419	0.6951	0.7710	0.8675	0.9191
Gradient Boosted V2	7611	2451	34704	4076	0.6512	0.7564	0.9340	0.6998	0.7799	0.8663	0.9180
Gradient Boosted V3	7405	2990	34165	4282	0.6336	0.7123	0.9195	0.6706	0.7632	0.8511	0.9038

Tabla 4.2: Tabla comparativa del algoritmo Gradient Boosted con las configuraciones por defecto y modificadas.

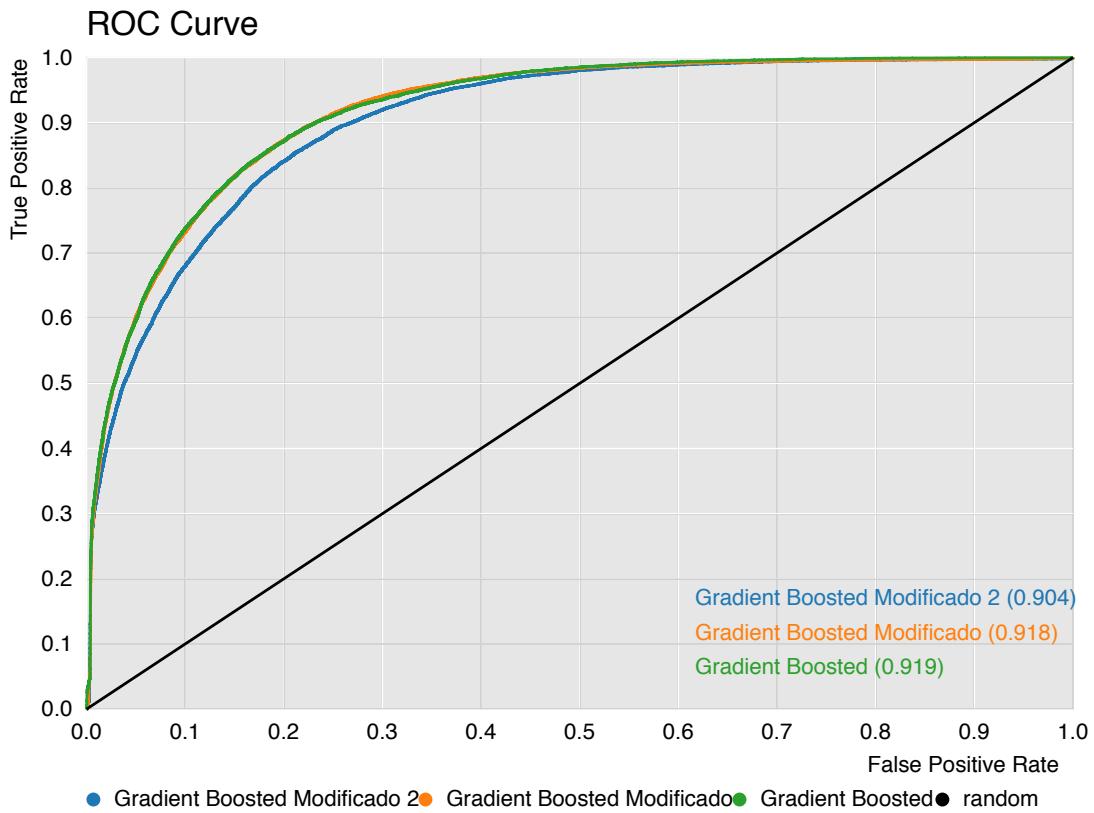


Figura 4.8: Gráfica ROC comparativa del algoritmo Gradient Boosted con las dos configuraciones..

En este caso vemos como aún aumentando la complejidad del modelo, por ejemplo, con el aumento del límite de profundidad del árbol, no conseguimos mejorar sustancialmente los resultados. Incluso por ejemplo en datos como el Precision y Specificity ya no es que no mejoremos sustancialmente los resultados, si no que incluso empeoran, y esto viene, provocado por el aumento de los falsos positivos, que para poner en situación como analogía, nos sirve pensar en la medicina, en donde el diagnóstico de las pruebas nos dicen que el paciente tiene una enfermedad, pero en realidad el paciente no está enfermo. (diagnóstico positivo, enfermedad ausente). Llevándolo a nuestro caso, el modelo nos dice que gana más de 50K dolares, cuando en realidad es falso. [18]

En contraposición nos encontramos con un Recall en aumento en el segundo algoritmo, ya que se distancia con respecto los otros dos en verdaderos positivos, siendo los falsos negativos prácticamente similares a los demás algoritmos.

Con respecto al valor del AUC, apenas notamos diferencias entre las distintas configuraciones, siendo entre los dos primeros diferencias ínfimas.

Como apunte final, podemos pensar que el algoritmo con la configuración de la figura 4.6, podría estar sobreaprendiendo, ya que con la configuración por defecto, obtenemos un modelo más simple y con prácticamente los mismos resultados. Aunque tampoco podríamos aventurarnos a pensar de tal forma con estos datos.

4.3. Naive Bayes

Para el algoritmo Naive Bayes únicamente hemos realizado pruebas con los parámetros por defecto. Para configurar dicho algoritmo, simplemente nos encontramos con la columna de clasificación (class, como siempre), probabilidad por defecto en nuestro caso de 0.0, máximo número de valores nominales únicos por atributo en nuestro caso de 20, ignorar los valores perdidos y crear un modelo compatible PMML 4.2 [12]. Estos dos últimos parámetros sin marcar.

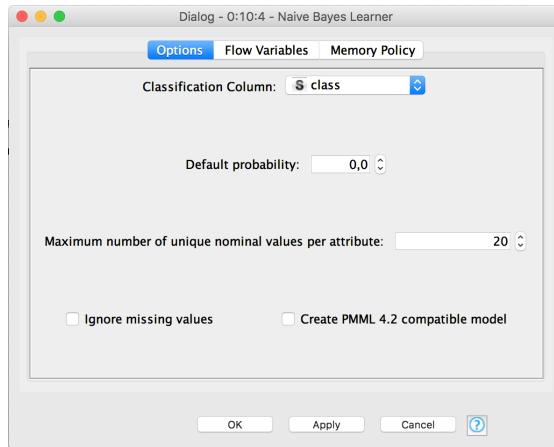


Figura 4.9: Configuración por defecto del algoritmo Naive Bayes.

Para mostrar los resultados simplemente extraemos una tabla con una única fila referente al algoritmo Naive Bayes.

Algorithm	TruePositives	FalsePositives	TrueNegatives	FalseNegatives	Recall	Precision	Specificity	F-measure	G-mean	Accuracy	AUC
Naive Bayes V1	8959	6279	30876	2728	0.7665	0.5879	0.8310	0.6654	0.7981	0.8155	0.891

Tabla 4.3: Tabla de resultados del algoritmo Naive Bayes con la configuración por defecto.

Por último mostramos la gráfica ROC con la única curva del algoritmo Naive Bayes con la configuración por defecto.

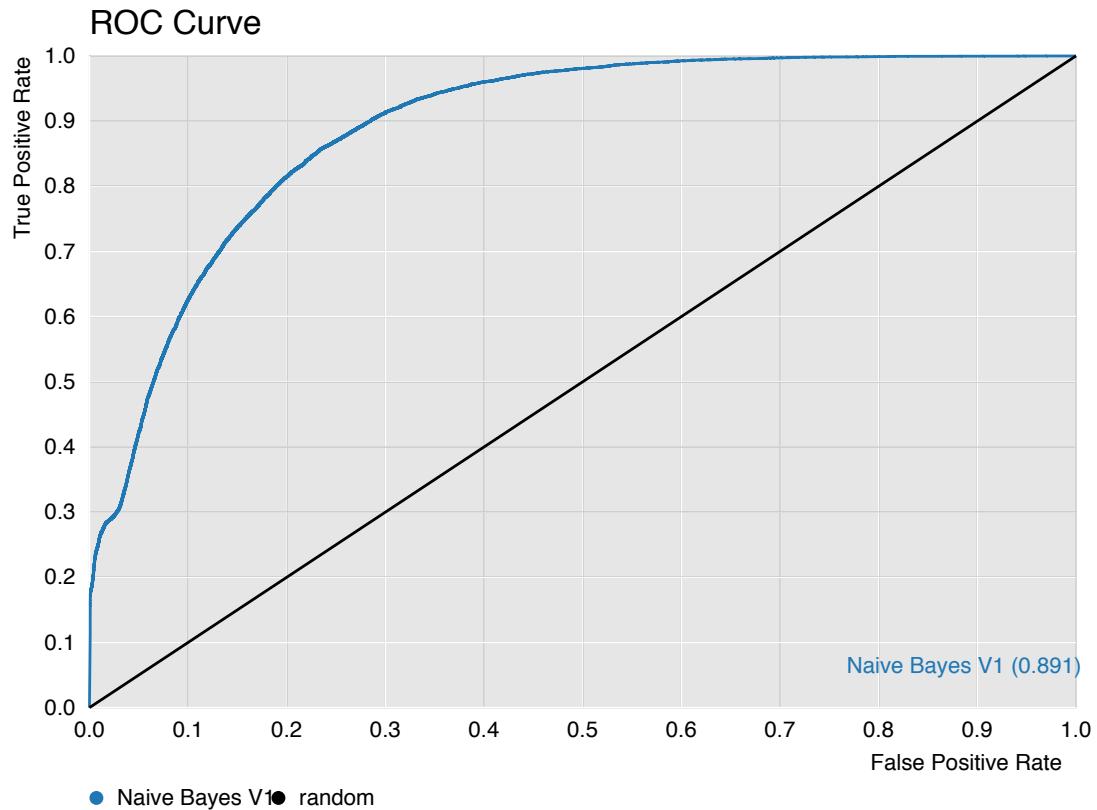


Figura 4.10: Gráfica ROC del algoritmo Naive Bayes con la configuración por defecto.

4.4. Random Forest

En el caso del algoritmo Random Forest como ocurrió con el algoritmo Naive Bayes únicamente se han realizado pruebas con los parámetros por defecto. Los parámetros que nos encontramos son los siguientes, columnas de atributos que añadimos (en nuestro caso todos los posibles, la posible modificación de esta parte queda relegada al punto 5), patrones a almacenar (por defecto 2000), criterio de división (por defecto information gain ratio), profundidad límite del árbol (por defecto 10), mínimo tamaño de nodo (por defecto 1) y por último número de modelos (por defecto 100). En caso de una posible modificación del algoritmo podríamos modificar la profundidad límite del árbol, en caso de aumentarla estaríamos creando un modelo más complejo, y en caso de disminuirla un modelo más simple. O también podríamos crear más modelos con el parámetro "Number Models" crear más modelos o menos.

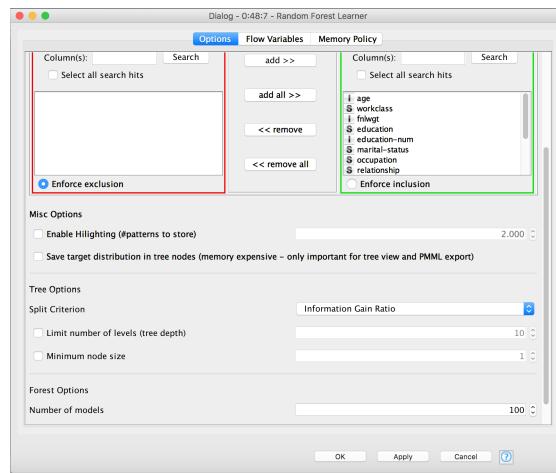


Figura 4.11: Configuración por defecto del algoritmo Random Forest.

Para mostrar los resultados simplemente extraemos una tabla con una única fila referente al algoritmo Random Forest.

Algorithm	TruePositives	FalsePositives	TrueNegatives	FalseNegatives	Recall	Precision	Specificity	F-measure	G-mean	Accuracy	AUC
Random Forest V1	7127	1837	35318	4560	0.6098	0.7950	0.9505	0.6902	0.7613	0.8690	0.9094

Tabla 4.4: Tabla de resultados del algoritmo Random Forest con la configuración por defecto.

Por último mostramos la gráfica ROC con la única curva del algoritmo Random Forest con la configuración por defecto.

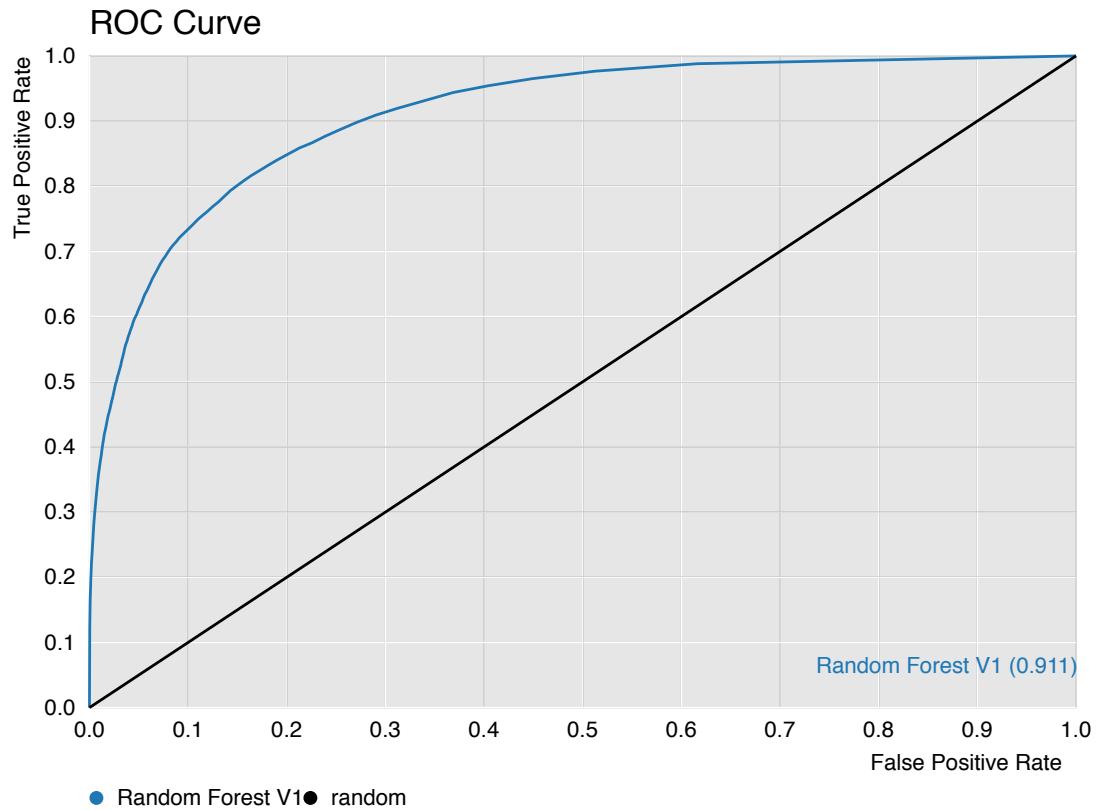


Figura 4.12: Gráfica ROC del algoritmo Random Forest con la configuración por defecto.

4.5. KNN

Para el caso del algoritmo KNN, se han realizado pruebas con dos configuraciones diferentes, en primer lugar tenemos el algoritmo con la configuración por defecto, y en segundo lugar con la configuración modificada.

En cuanto a la configuración por defecto tenemos en primer lugar la columna con la clase (class como siempre), el número de vecinos a considerar (uno por defecto), peso de esos vecinos por distancia (en el caso por defecto no viene activada, y será uno de los parámetros a tener en cuenta en la modificación), y por último salida de las probabilidades de la clase, que lo usaremos para sacar la gráfica ROC.

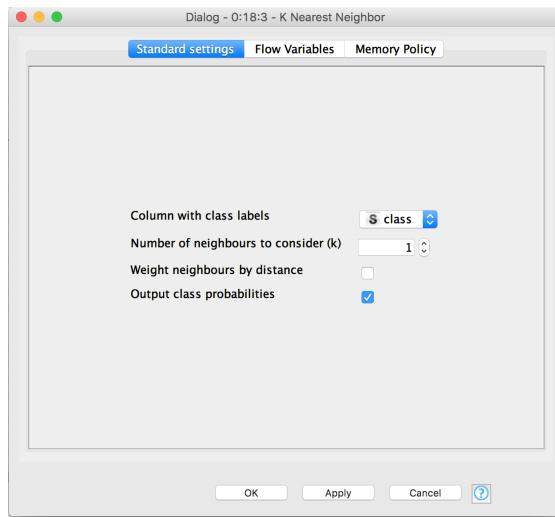


Figura 4.13: Configuración por defecto del algoritmo KNN.

En lo que se refiere al modelo modificado del algoritmo K-NN, se ha modificado el número de vecinos a considerar, que pasa de 1 a 5 [13]. Y se ha considerado oportuno tener en cuenta pesos de vecinos por distancia, incluye la distancia del patrón de consulta a los patrones de entrenamiento, de tal forma que los vecinos más cercanos tendrán una mayor influencia.

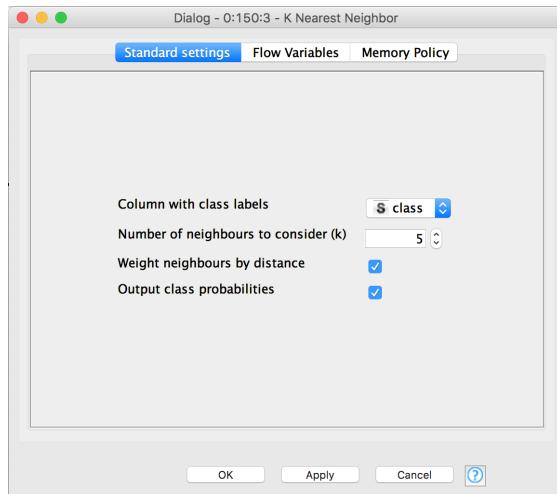


Figura 4.14: Configuración modificada del algoritmo KNN, algoritmo 1.

Para el modelo modificado 2 del algoritmo K-NN, se ha modificado el número de vecinos a considerar, que pasa de 1 a 5 [13]. Y se ha considerado oportuno no tener en cuenta pesos de vecinos por distancia.

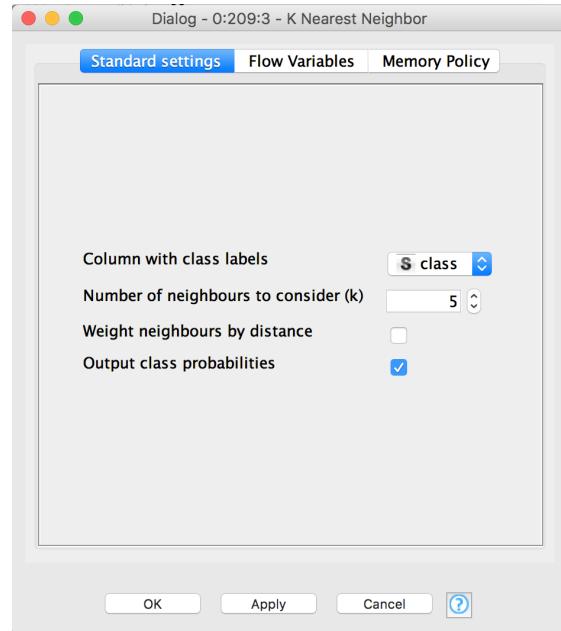


Figura 4.15: Configuración modificada del algoritmo KNN, algoritmo 2.

Y por último en el modelo modificado 3 del algoritmo K-NN, se ha modificado el número de vecinos a considerar, que pasa de 1 a 10 [13]. Y se ha considerado oportuno tener en cuenta pesos de vecinos por distancia, incluye la distancia del patrón de consulta a los patrones de entrenamiento, de tal forma que los vecinos más cercanos tendrán una mayor influencia.

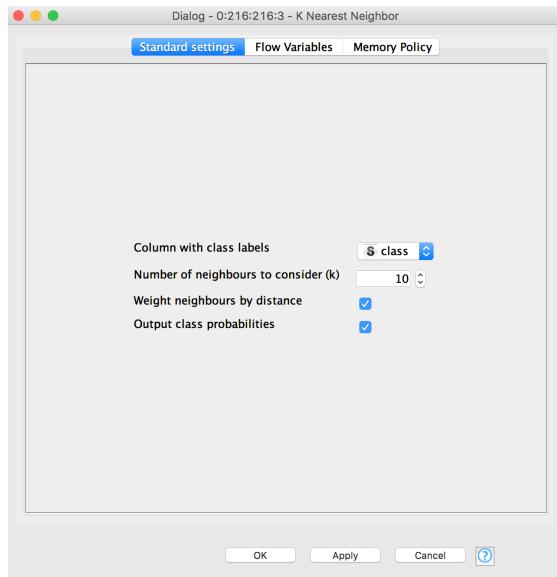


Figura 4.16: Configuración modificada del algoritmo KNN, algoritmo 3.

Para comparar los resultados, extraemos una tabla con los resultados de ambas configuraciones. Y en forma de tabla comparativa podremos extraer conclusiones. A su vez también extraemos una gráfica ROC con las dos curvas pintadas, para apoyar las conclusiones finales sobre ambas configuraciones.

Algorithm	TruePositives	FalsePositives	TrueNegatives	FalseNegatives	Recall	Precision	Specificity	F-measure	G-mean	Accuracy	AUC
K-NN V1	6588	4962	32193	5099	0.5637	0.5703	0.8664	0.8664	0.6988	0.7940	0.7157
K-NN V2	6729	3746	33409	4958	0.5757	0.6423	0.8991	0.6072	0.7195	0.8217	0.8466
K-NN V3	6692	3546	33609	4995	0.5726	0.6536	0.9045	0.6104	0.7196	0.8251	0.8477
K-NN V4	6711	3396	33759	4976	0.5742	0.6639	0.9085	0.6158	0.7223	0.8285	0.8690

Tabla 4.5: Tabla comparativa del algoritmo KNN con las configuraciones por defecto y modificadas.

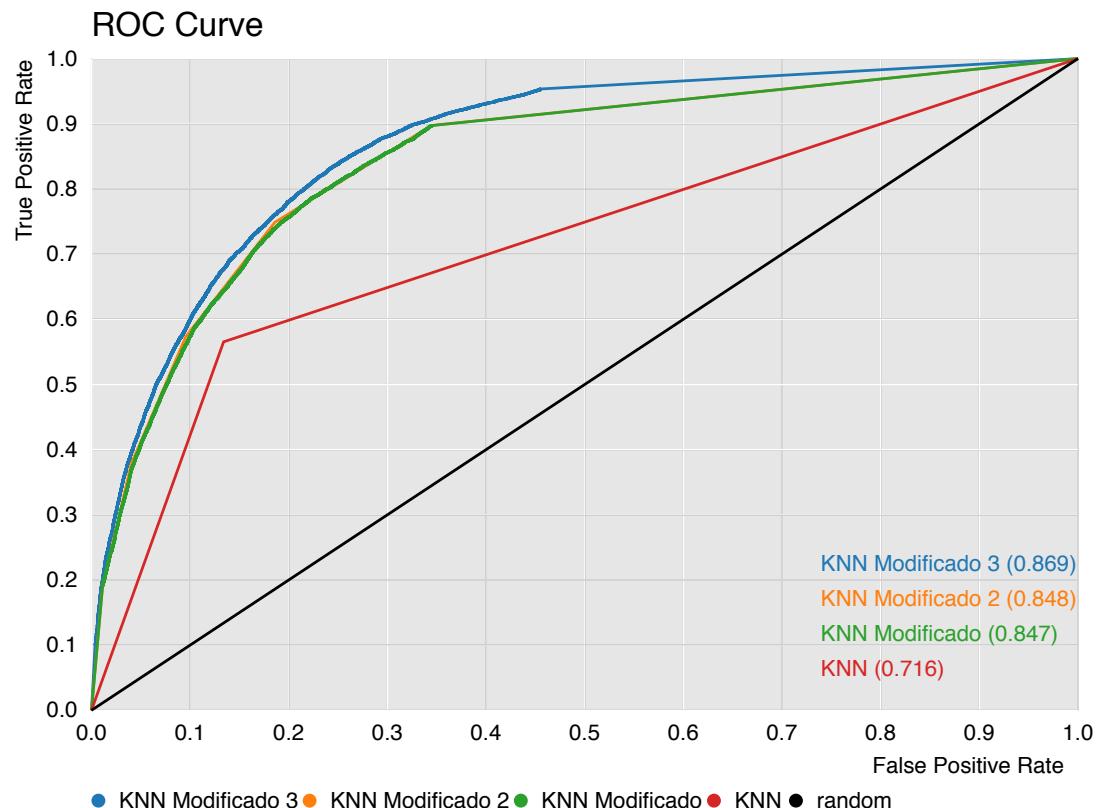


Figura 4.17: Gráfica ROC comparativa del algoritmo KNN con las dos configuraciones..

En el caso del algoritmo KNN se han realizado 3 modificaciones distintas de la configuración por defecto que trae el mismo. Los distintos resultados extraídos nos hablan de que aparentemente no existe sobreaprendizaje, ya que al simplificar el modelo, los resultados no mejoran si no que todo lo contrario, empeoran. Podemos fijarnos como tanto el Recall como el resultado del Precision mejora si comparamos cualquiera de las modificaciones con respecto al algoritmo por defecto. Con sendos aumentos en los verdaderos positivos y reducción en los falsos positivos y falsos negativos.[18] También se muestra un aumento sustancial del Specificity [17], el cuál colocando como analogía de nuevo a la medicina, nos indica la probabilidad de clasificar correctamente a un individuo sano, y por lo tanto aplicado a nuestro caso nos indicaría la probabilidad de clasificar correctamente a los individuos que no ganan más de 50K dólares.

Apoyándonos en la gráfica de la curva ROC,[19] deducimos que existe una diferencia más que sustancial entre la configuración por defecto en donde K es igual a 1 (sin tener en cuenta pesos) y las distintas configuraciones. En este aspecto el KNN Modificado 3 con $K = 10$ y teniendo en cuenta pesos de vecinos por distancia, es el que mejor se comporta. También podemos observar que entre la modificación 1 y 2, apenas hay diferencias, y esto refrenda el hecho de que el tener en cuenta los pesos no es crucial para obtener una mejora en el modelo.

4.6. Red Neuronal

Por último, tenemos el algoritmo Neural Network (Red Neuronal), en donde se ha realizado una comparación como en casos anteriores, modificando ajustes y parámetros de configuración del propio algoritmo.

En cuanto a la configuración por defecto tenemos, parámetro de máximo número de iteraciones (100 por defecto), número de capas ocultas (1 por defecto), número de neuronas ocultas por capa (por defecto 10), ignorar valores perdidos (en este caso no se ignoran) y usar semilla para inicialización aleatoria.

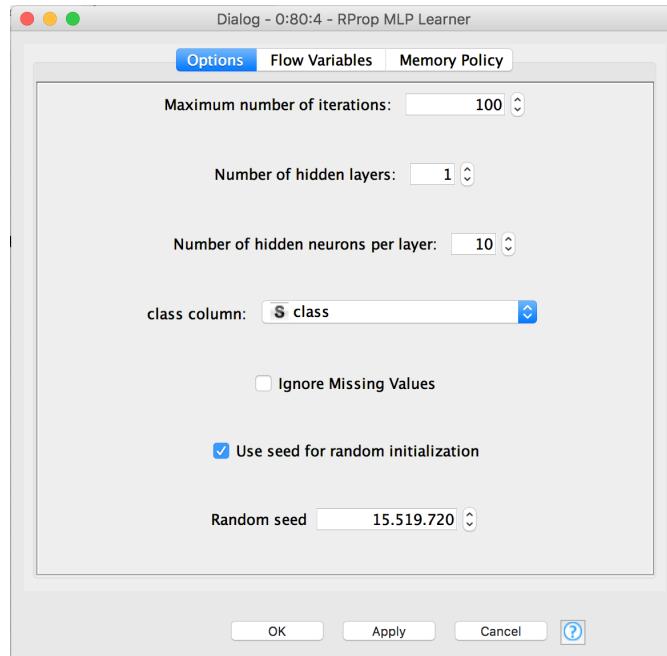


Figura 4.18: Configuración por defecto del algoritmo Red Neuronal.

En lo que se refiere al modelo modificado del algoritmo Neural Network, se ha modificado el número de capas ocultas de 1 a 15, y el número de neuronas ocultas por capa a 20. Esto provocaría un modelo más complejo, aunque si que es verdad que al no ser un modelo que pueda visualizarse y sacar conclusiones claras de el como por ejemplo un árbol de decisión, a priori la complejidad nos podría dar igual siempre y cuando no hubiese diferencias abismales.

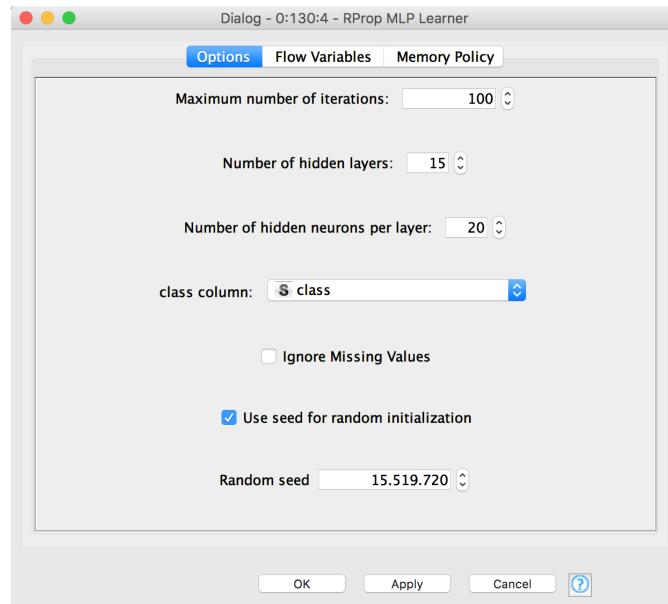


Figura 4.19: Configuración modificada del algoritmo Red Neuronal, algoritmo 1.

En lo que se refiere al modelo modificado del algoritmo Neural Network, se ha modificado el número de capas ocultas de 1 a 15, y el número de neuronas ocultas por capa a 20. Esto provocaría un modelo más complejo, aunque si que es verdad que al no ser un modelo que pueda visualizarse y sacar conclusiones claras de el como por ejemplo un árbol de decisión, a priori la complejidad nos podría dar igual siempre y cuando no hubiese diferencias abismales.

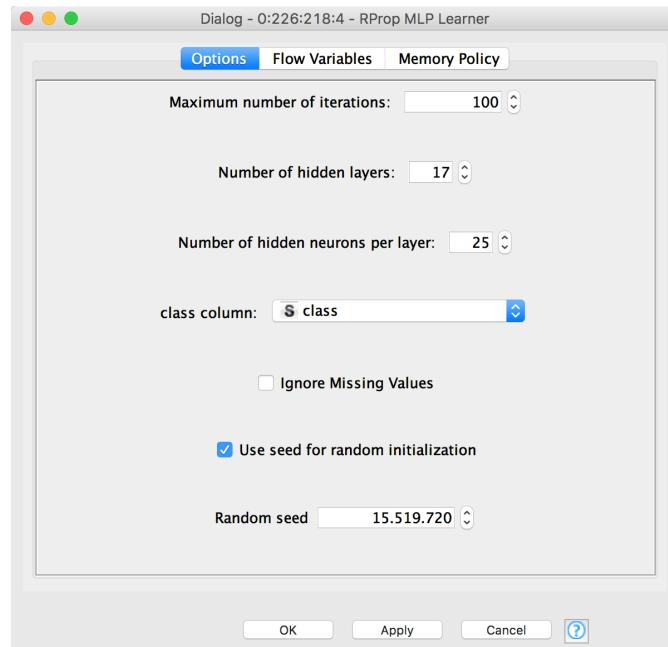


Figura 4.20: Configuración modificada del algoritmo Red Neuronal, algoritmo 2.

En lo que se refiere al modelo modificado del algoritmo Neural Network, se ha modificado el número de capas ocultas de 1 a 15, y el número de neuronas ocultas por capa a 20. Esto provocaría un modelo más complejo, aunque si que es verdad que al no ser un modelo que pueda visualizarse y sacar conclusiones claras de el como por ejemplo un árbol de decisión, a priori la complejidad nos podría dar igual siempre y cuando no hubiese diferencias abismales.

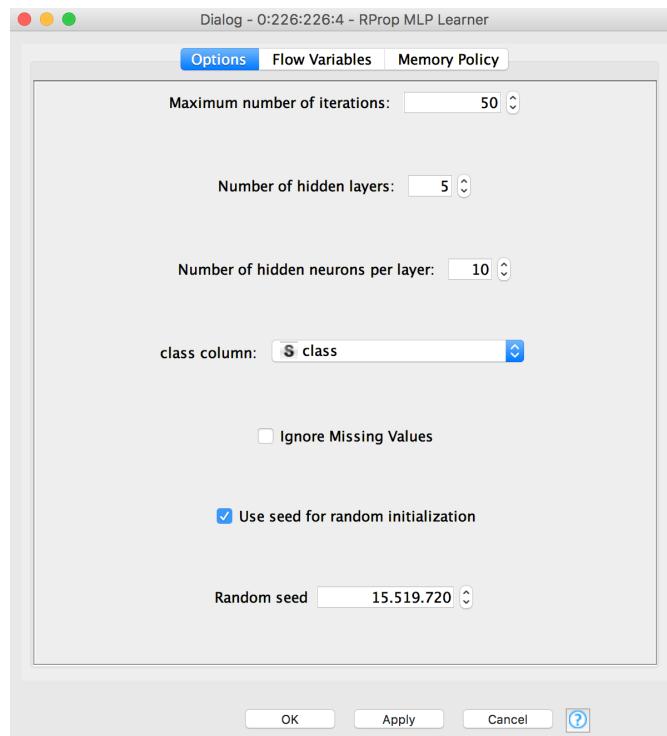


Figura 4.21: Configuración modificada del algoritmo Red Neuronal, algoritmo 3.

Para comparar los resultados, extraemos una tabla con los resultados de ambas configuraciones. Y en forma de tabla comparativa podremos extraer conclusiones. A su vez también extraemos una gráfica ROC con las dos curvas pintadas, para apoyar las conclusiones finales sobre ambas configuraciones.

Algorithm	TruePositives	FalsePositives	TrueNegatives	FalseNegatives	Recall	Precision	Specificity	F-measure	G-mean	Accuracy	AUC
Neural Network V1	6672	2464	34691	5015	0.5708	0.7302	0.9336	0.6408	0.7300	0.8468	0.8959
Neural Network V2	3618	3385	33770	8069	0.3095	0.5166	0.9088	0.3871	0.5304	0.7654	0.7818
Neural Network V3	5511	4699	32456	6176	0.4715	0.5397	0.8735	0.5033	0.6418	0.7773	0.7918
Neural Network V4	5786	2297	34858	5901	0.4950	0.7158	0.9381	0.5853	0.6815	0.8321	0.8419

Tabla 4.6: Tabla comparativa del algoritmo Red Neuronal con las configuraciones por defecto y modificadas.

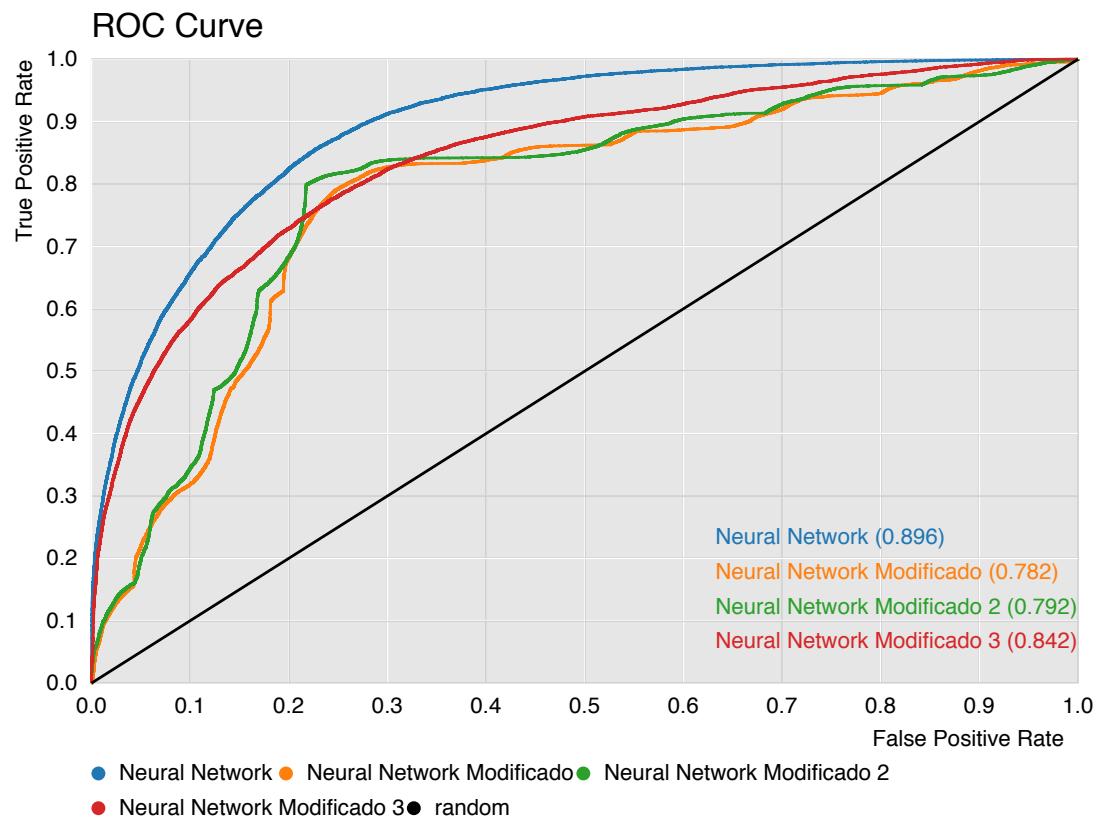


Figura 4.22: Gráfica ROC comparativa del algoritmo Red Neuronal con las dos configuraciones.

Por último hemos realizado ejecuciones comparativas del algoritmo Neural Network con diferentes configuraciones, en concreto han sido 4 las distintas configuraciones que se han seguido.

En los resultados obtenidos de los distintos modelos, en principio puede parecer verse la existencia del sobreaprendizaje, ya que en el primer caso con la configuración por defecto y el último caso serían los más simples, mientras que en los otros dos casos obtendríamos modelos más complejos, con mayores números de capas y a su vez mayores números de neuronas por capa.

En el caso de la segunda configuración con un número de capas de 15 y 20 neuronas por capa, se aprecia una reducción drástica con respecto a las demás configuraciones en el resultado de los verdaderos positivos, incidiendo sobre todo en el Recall y Precision. Sin embargo, donde por parte de esta configuración se produce un aumento mas que sustancial en comparación con el resto es en los falsos negativos (recordemos con la analogía de la medicina, diagnóstico negativo, enfermedad presente).

Mientras esta configuración comentada se aleja en cuanto al modelo perfecto, estaría la configuración por defecto en primer lugar de todas estas con el mejor Recall, Precision, Specificity y AUC. Esto último puede verse claramente en la gráfica ROC.

Este estudio de los resultados realizado puede verse afectado por el preprocesamiento que se ha realizado a los datos previo paso por el algoritmo. Preprocesamiento que se explicó en secciones anteriores, y que se modificará en distintas versiones posteriormente.

5. Procesado de datos

En esta sección se hablará de los distintos preprocesados [22] que se han realizado para algunos de los algoritmos con la intención de mejorar los resultados que hasta ahora hemos obtenido. En algunos casos no es posible conseguir el objetivo de mejorar los modelos, pero aún así, no viene mal tener información acerca de los distintos comportamientos de los algoritmos. Para mejorar los modelos, en KNIME tenemos distintos nodos de preprocesamiento, como pueden ser: Normalizar, Discretizar, filtrar, selección de variables, etc.

En nuestro caso nos vamos a focalizar en 3 algoritmos: Decision Tree, Naive Bayes y Random Forest. Estos dos últimos han sido escogidos por el hecho de que no se realizó un estudio en el anterior apartado para las distintas posibles configuraciones, por lo que para poder obtener unos modelos lo más estudiados posibles, se hará en este apartado un estudio más profundo en cuanto a preprocesamiento se refiere.

En primer lugar tenemos el algoritmo Decision Tree.

5.1. Decision Tree

Para el algoritmo Decision Tree (árbol de decisión) hemos considerado 2 preprocesamientos diferentes, uno más restrictivo que el otro.

En primer lugar hemos considerado para las 3 versiones obtenidas del apartado anterior 4 las siguientes medidas de preprocesamiento:

- One to Many: Transforma todos los posibles valores de una columna en una nueva columna. De tal forma que en las nuevas columnas sale 1 cuando la fila pertenece a ese valor, y 0 cuando no. El nodo agrega tantas columnas como posibles valores. En nuestro caso lo hacemos con las siguientes columnas: workclass (String), education (String), marital-status (String), occupation (String), relationship (String), race (String), sex (String), native-country (String). [20]
- Column Filter: Elimina las columnas que previamente han sido transformadas en nuevas columnas por el nodo One to Many. En nuestro caso eliminamos las siguientes columnas: workclass (String), education (String), marital-status (String), occupation (String), relationship (String), race (String), sex (String), native-country (String). [20]
- Missing Values: Maneja los valores perdidos de las distintas celdas, de tal forma que se puede actuar reponiendo valores en forma de media, más frecuente ó eliminar fila. En nuestro caso lo usaremos para aquellas columnas de tipo numérico escoger en donde ocurra un valor perdido la media. [21]
- Normalizer: Normaliza los valores de las columnas numéricas entre un rango indicado previamente, en nuestro caso [0.0 , 1.0] [20]

A continuación se muestran imágenes del flujo del preprocesamiento y de los distintos aspectos de configuración previamente explicados:

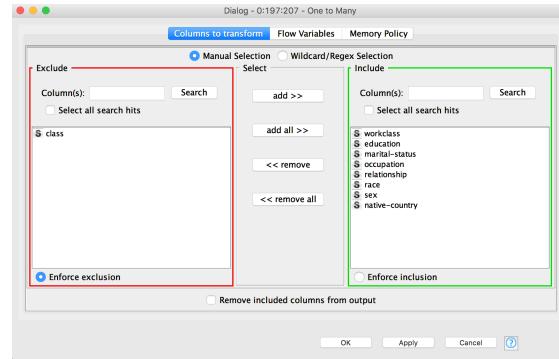


Figura 5.1: Configuración del nodo One to Many.

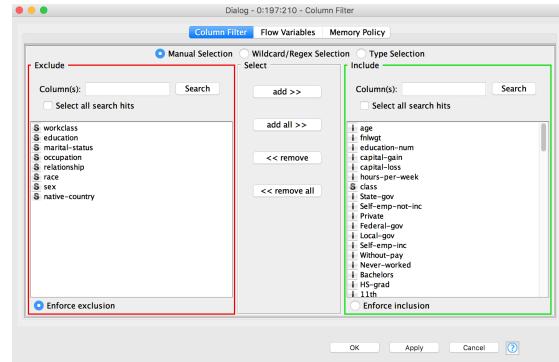


Figura 5.2: Configuración del nodo Column Filter.

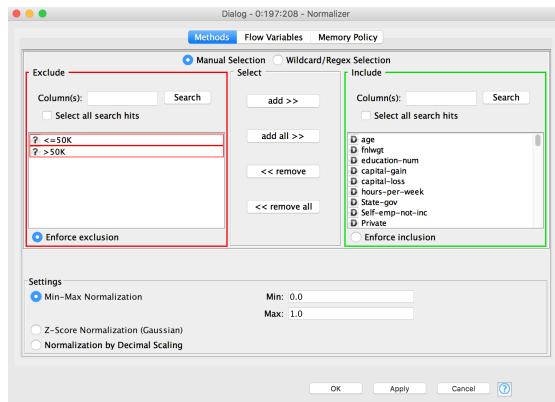


Figura 5.3: Configuración del nodo Normalizer.

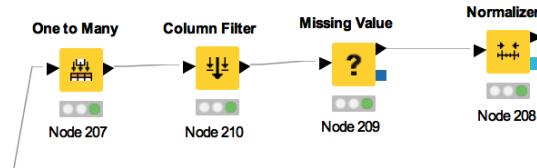


Figura 5.4: Workflow de preprocessamiento 1, del algoritmo Decision Tree para las tres versiones del apartado anterior 4.

Los datos obtenidos por los modelos explicados en el anterior apartado 4 son los siguientes:⁴

Algorithm	TruePositives	FalsePositives	TrueNegatives	FalseNegatives	Recall	Precision	Specificity	F-measure	G-mean	Accuracy	AUC
Decision Tree V1 PP1	7353	4017	33138	4334	0.6291	0.6467	0.8918	0.6378	0.7490	0.8290	0.7777
Decision Tree V2 PP1	7238	3166	33989	4449	0.6193	0.6956	0.9147	0.6552	0.7526	0.8440	0.8999
Decision Tree V3 PP1	7041	2606	34549	4646	0.6024	0.7298	0.9298	0.6600	0.7484	0.8515	0.9023

Tabla 5.1: Tabla comparativa del algoritmo Decision Tree con las configuraciones del apartado anterior 4 y el primer tipo de preprocessamiento.

⁴Las iniciales PP se refieren a PreProcesamiento. Y el número que le sigue, a la versión del preprocessamiento.

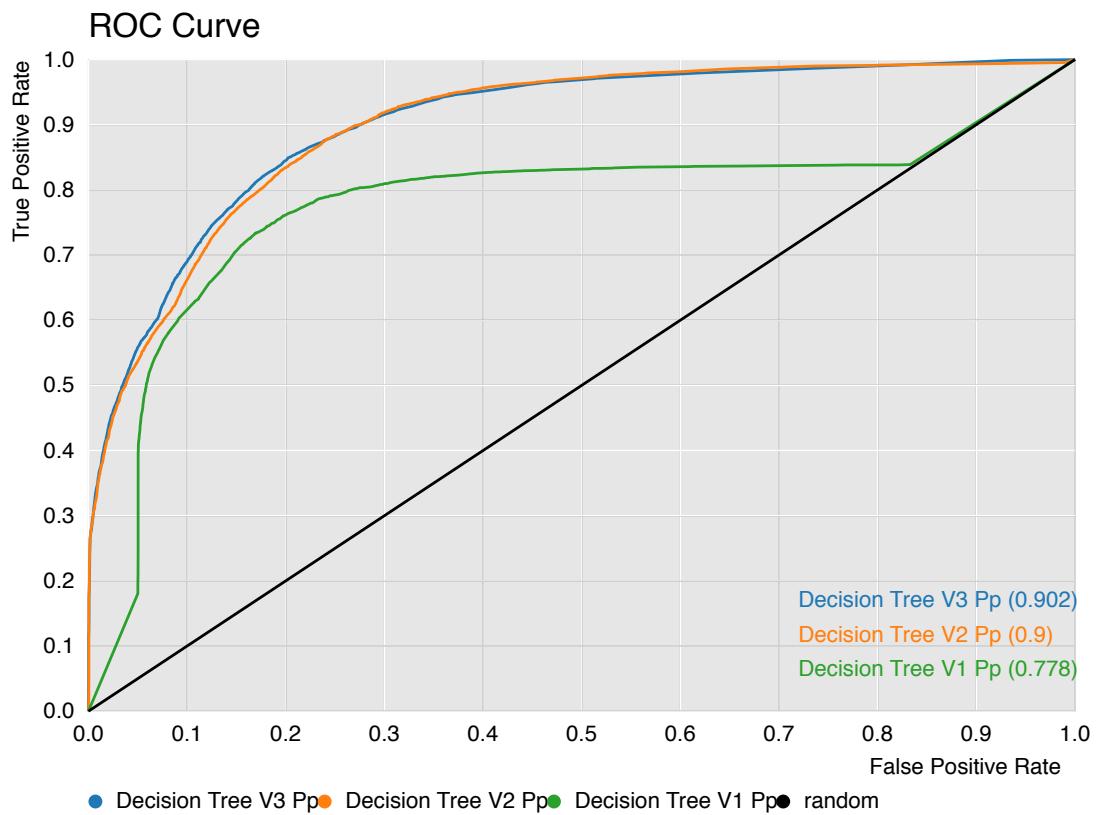


Figura 5.5: Gráfica ROC comparativa de las anteriores versiones del Decision Tree con el preprocesamiento 1.

En segundo lugar hemos considerado para las 3 versiones obtenidas del apartado anterior 4 las siguientes medidas de preprocesamiento:

- One to Many: Transforma todos los posibles valores de una columna en una nueva columna. De tal forma que en las nuevas columnas sale 1 cuando la fila pertenece a ese valor, y 0 cuando no. El nodo agrega tantas columnas como posibles valores. En nuestro caso lo hacemos con las siguientes columnas: workclass (String), education (String), marital-status (String), occupation (String), relationship (String), race (String), sex (String), native-country (String). [20]
- Column Filter: Elimina las columnas que previamente han sido transformadas en nuevas columnas por el nodo One to Many. En nuestro caso eliminamos las siguientes columnas: workclass (String), education (String), marital-status (String), occupation (String), relationship (String), race (String), sex (String), native-country (String). [20]
- Normalizer: Normaliza los valores de las columnas numéricas entre un rango indicado previamente, en nuestro caso [0.0 , 1.0] [20]

En cuanto a la configuración de los tres nodos es exactamente la misma que para el preprocesamiento 1. Por ello no se incluyen las imágenes de los aspectos de configuración.

Los datos obtenidos por los modelos explicados en el anterior apartado 4 son los siguientes:⁵

Algorithm	TruePositives	FalsePositives	TrueNegatives	FalseNegatives	Recall	Precision	Specificity	F-measure	G-mean	Accuracy	AUC
Decision Tree V1 PP2	7353	4017	33138	4334	0.6291	0.6467	0.8918	0.6378	0.7490	0.8290	0.7777
Decision Tree V2 PP2	7238	3166	33989	4449	0.6193	0.6956	0.9147	0.6552	0.7526	0.8440	0.8999
Decision Tree V3 PP2	7041	2606	34549	4646	0.6024	0.7298	0.9298	0.6600	0.7484	0.8515	0.9023

Tabla 5.2: Tabla comparativa del algoritmo Decision Tree con las configuraciones del apartado anterior 4 y el primer tipo de preprocesamiento.

⁵Las iniciales PP se refieren a PreProcesamiento. Y el número que le sigue, a la versión del preprocesamiento.

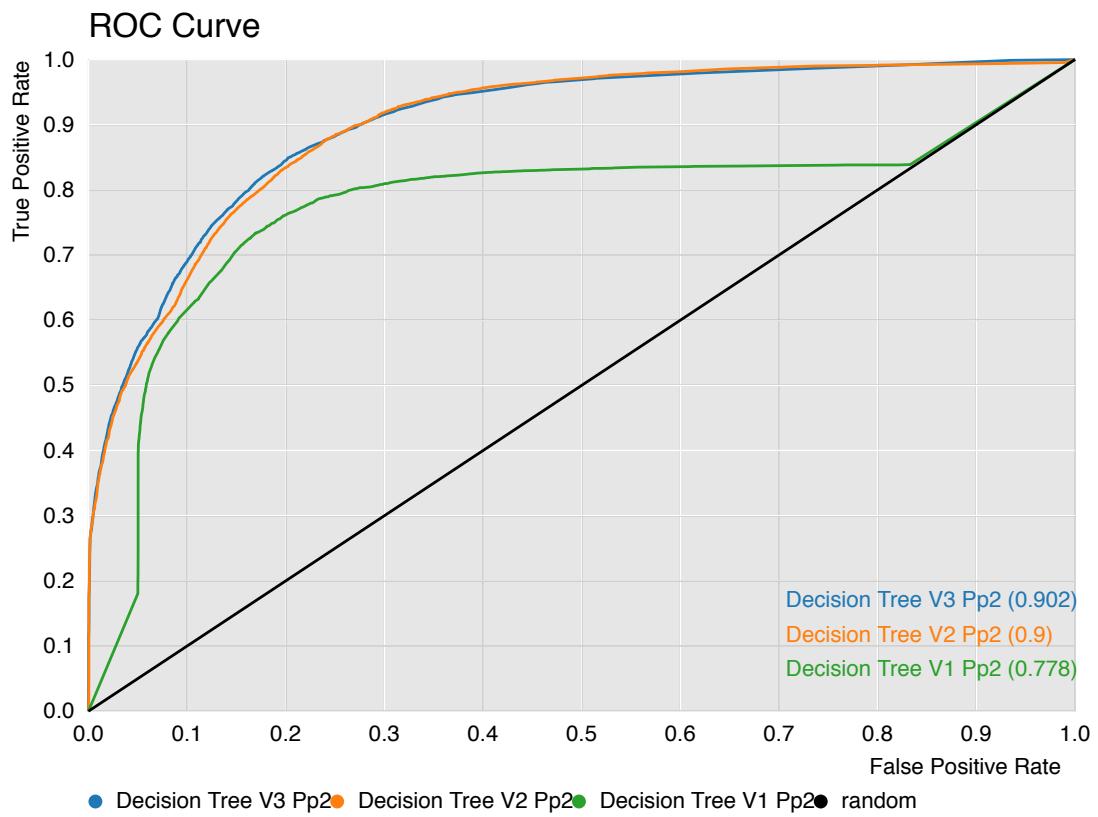


Figura 5.6: Gráfica ROC comparativa de las anteriores versiones del Decision Tree con el preprocesamiento 2.

A parte de las comparaciones de las distintas versiones del algoritmo Decision Tree con sus respectivos preprocesados, también se ha considerado obtener la gráfica ROC comparativa por versiones de la configuración del algoritmo con los distintos preprocesados, así como sus tablas.

Para la versión 1:⁶

Algorithm	TruePositives	FalsePositives	TrueNegatives	FalseNegatives	Recall	Precision	Specificity	F-measure	G-mean	Accuracy	AUC
Decision Tree V1	6969	3744	32991	4541	0.6054	0.6505	0.8980	0.6271	0.7374	0.8282	0.8055
Decision Tree V1 PP1	7353	4017	33138	4334	0.6291	0.6467	0.8918	0.6378	0.7490	0.8290	0.7777
Decision Tree V1 PP2	7353	4017	33138	4334	0.6291	0.6467	0.8918	0.6378	0.7490	0.8290	0.7777

Tabla 5.3: Tabla comparativa del algoritmo Decision Tree con las configuraciones del apartado anterior 4 y el primer tipo de preprocesamiento.

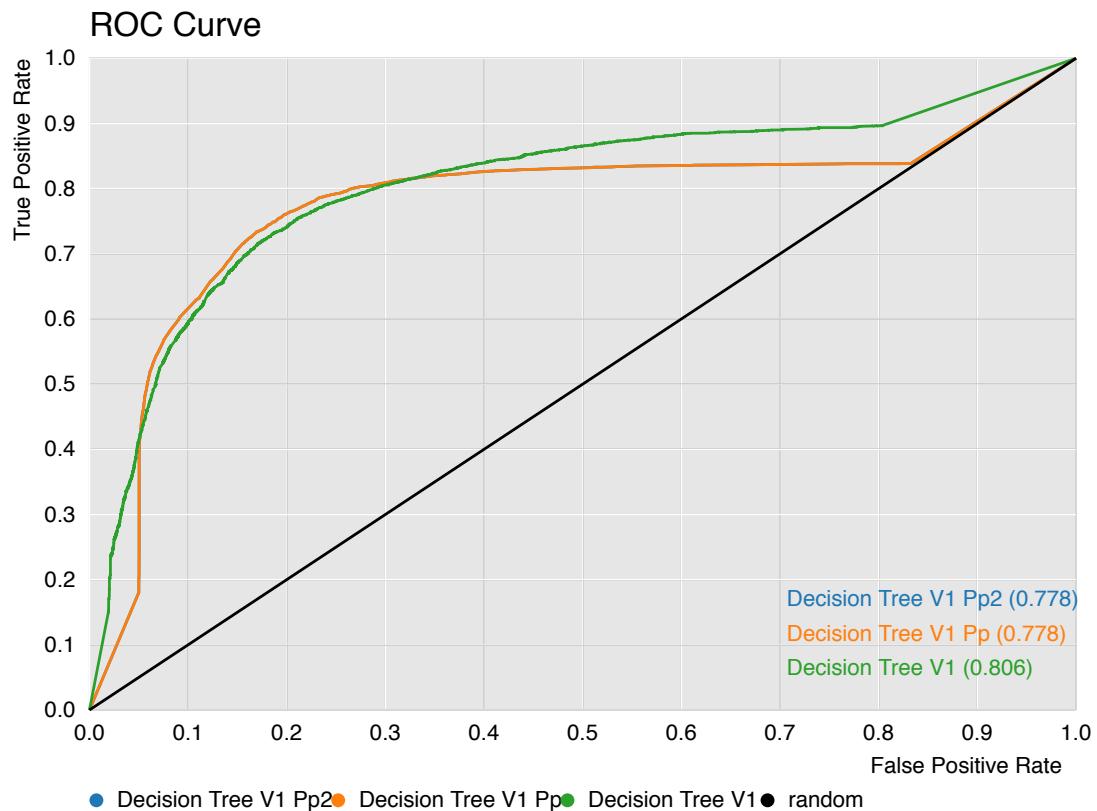


Figura 5.7: Gráfica ROC comparativa teniendo en cuenta la misma versión (1) y distintos preprocesados.

⁶En los casos en donde no hay iniciales PP es por que no se produce preprocesamiento.

Para la versión 2:

Algorithm	TruePositives	FalsePositives	TrueNegatives	FalseNegatives	Recall	Precision	Specificity	F-measure	G-mean	Accuracy	AUC
Decision Tree V2	6963	2995	34070	4688	0.5976	0.6992	0.9191	0.6444	0.7411	0.8422	0.8857
Decision Tree V2 PP1	7238	3166	33989	4449	0.6193	0.6956	0.9147	0.6552	0.7526	0.8440	0.8999
Decision Tree V2 PP2	7238	3166	33989	4449	0.6193	0.6956	0.9147	0.6552	0.7526	0.8440	0.8999

Tabla 5.4: Tabla comparativa del algoritmo Decision Tree con las configuraciones del apartado anterior 4 y el primer tipo de preprocesamiento.

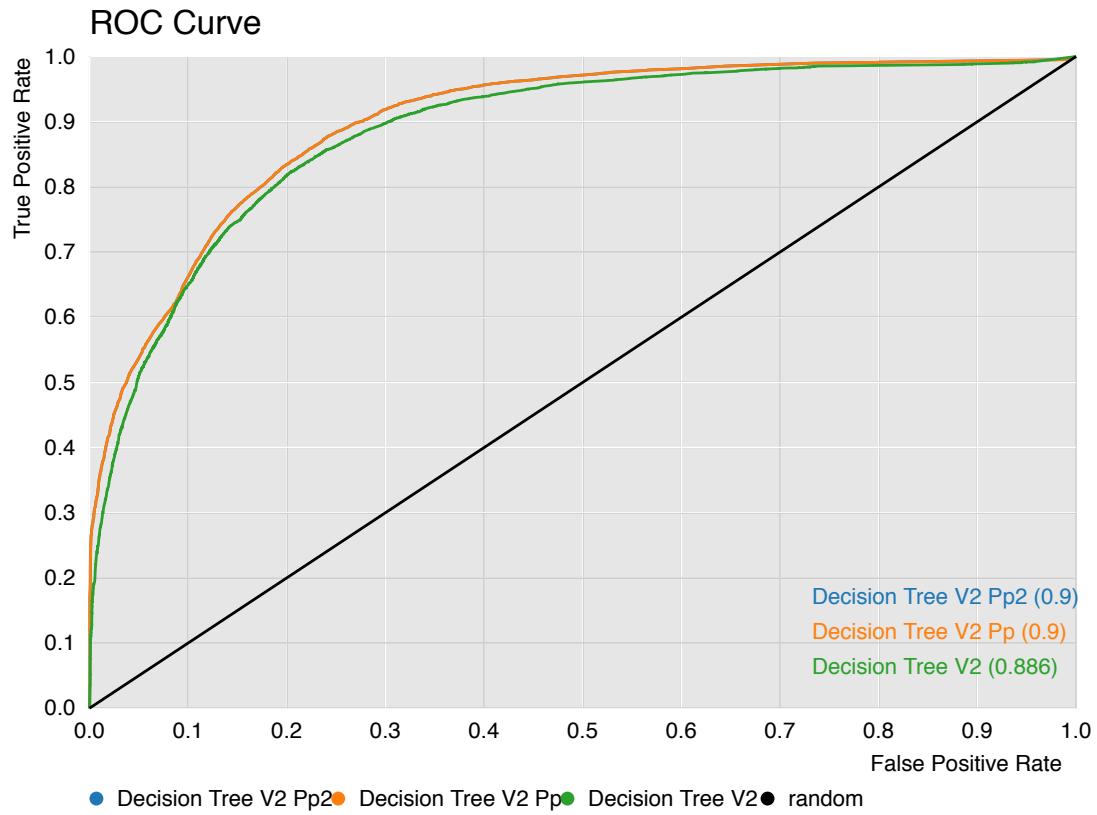


Figura 5.8: Gráfica ROC comparativa teniendo en cuenta la misma versión (2) y distintos preprocesados.

Para la versión 3:

Algorithm	TruePositives	FalsePositives	TrueNegatives	FalseNegatives	Recall	Precision	Specificity	F-measure	G-mean	Accuracy	AUC
Decision Tree V3	6968	2770	34343	4706	0.5968	0.7155	0.9253	0.6508	0.7431	0.8467	0.8901
Decision Tree V3 PP1	7041	2606	34549	4646	0.6024	0.7298	0.9298	0.6600	0.7484	0.8515	0.9023
Decision Tree V3 PP2	7041	2606	34549	4646	0.6024	0.7298	0.9298	0.6600	0.7484	0.8515	0.9023

Tabla 5.5: Tabla comparativa del algoritmo Decision Tree con las configuraciones del apartado anterior 4 y el primer tipo de preprocesamiento.

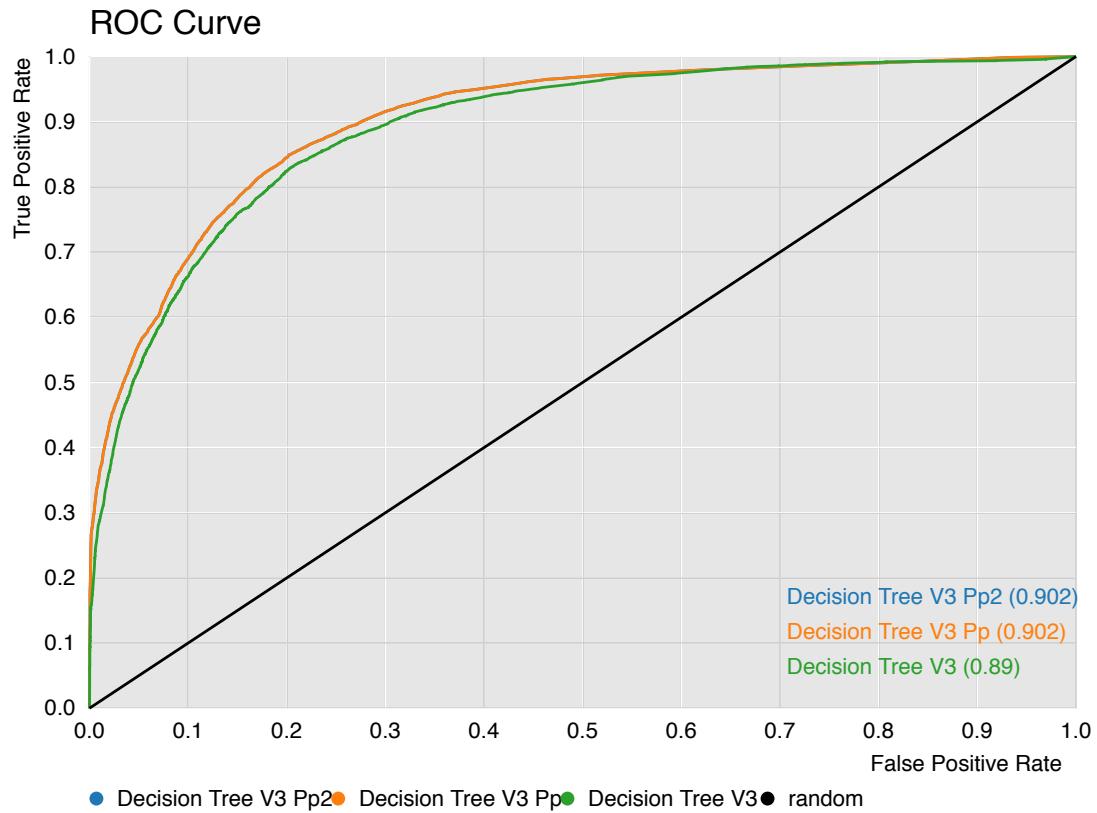


Figura 5.9: Gráfica ROC comparativa teniendo en cuenta la misma versión (3) y distintos preprocesados.

Tabla final de todas las versiones tanto con las distintas configuraciones como con los distintos preprocesados: Como punto final y para argumentar los distintos preprocesamientos realizados.

Algorithm	TruePositives	FalsePositives	TrueNegatives	FalseNegatives	Recall	Precision	Specificity	F-measure	G-mean	Accuracy	AUC
Decision Tree V1	6969	3744	32991	4541	0.6054	0.6505	0.8980	0.6271	0.7374	0.8282	0.8055
Decision Tree V2	6963	2995	34070	4688	0.5976	0.6992	0.9191	0.6444	0.7411	0.8422	0.8857
Decision Tree V3	6968	2770	34343	4706	0.5968	0.7155	0.9253	0.6508	0.7431	0.8467	0.8901
Decision Tree V1 PP1	7353	4017	33138	4334	0.6291	0.6467	0.8918	0.6378	0.7490	0.8290	0.7777
Decision Tree V2 PP1	7238	3166	33989	4449	0.6193	0.6956	0.9147	0.6552	0.7526	0.8440	0.8999
Decision Tree V3 PP1	7041	2606	34549	4646	0.6024	0.7298	0.9298	0.6600	0.7484	0.8515	0.9023
Decision Tree V1 PP2	7353	4017	33138	4334	0.6291	0.6467	0.8918	0.6378	0.7490	0.8290	0.7777
Decision Tree V2 PP2	7238	3166	33989	4449	0.6193	0.6956	0.9147	0.6552	0.7526	0.8440	0.8999
Decision Tree V3 PP2	7041	2606	34549	4646	0.6024	0.7298	0.9298	0.6600	0.7484	0.8515	0.9023

Tabla 5.6: Tabla comparativa del algoritmo Decision Tree con las configuraciones del apartado anterior 4 y todos los tipos de preprocesamiento.

mientos realizados a lo largo de la práctica en el algoritmo Decision Tree, procedemos a la interpretación de los resultados.

En primer lugar, para el primer preprocesamiento se ha intentado que el algoritmo trabaje sin columnas categóricas, eliminando las mismas y convirtiéndolas en nuevas columnas numéricas, y sin valores perdidos, y a todo esto se finaliza normalizando el conjunto de valores a una escala apropiada.

Para el segundo preprocesamiento, se vuelve a trabajar sin columnas categóricas, pero esta vez no se intentan resolver los valores perdidos. También se realiza el normalizado de los datos.

A través de las distintas tablas y gráficas ROC podemos observar que entre las mismas versiones no hay grandes diferencias aún realizando distintos preprocesados. La mayor diferencia se produce cuando el algoritmo trae la configuración por defecto, en tal caso vemos como el AUC es mejor para el algoritmo al que no se le aplica preprocesado. Sin embargo para las versiones 2 y 3 se produce el efecto contrario, mejora el modelo cuando introducimos un preprocesamiento previo. Anque entre no preprocesar y preprocesar hay diferencias, cabe decir que entre el preprocesamiento 1 y el 2 no las hay. Se puede apreciar como los resultados son identicos entre tratar los valores perdidos y no tratarlos.

A todo esto con la versión 3 y los preprocesados 1 y 2, obtenemos los mejores resultados para el Accuracy , Precision y el AUC. Siendo igual de buenos ambos modelos. Esto se produce entre otras cosas por la gran disminución de FalsosPositivos (con la analogía médica: diagnóstico positivo, enfermedad ausente), ya que en cuanto a verdaderos positivos se mantienen en cotas bastante respetables, no siendo las mejores.

5.2. Random Forest

Para el algoritmo Random Forest hemos considerado 3 preprocesamientos diferentes:

En primer lugar hemos considerado para la versión obtenida del apartado anterior 4 con las siguientes medidas de preprocesamiento:

- One to Many: Transforma todos los posibles valores de una columna en una nueva columna. De tal forma que en las nuevas columnas sale 1 cuando la fila pertenece a ese valor, y 0 cuando no. El nodo agrega tantas columnas como posibles valores. En nuestro caso lo hacemos con las siguientes columnas: workclass (String), education (String), marital-status (String), occupation (String), relationship (String), race (String), sex (String), native-country (String). [20]
- Column Filter: Elimina las columnas que previamente han sido transformadas en nuevas columnas por el nodo One to Many. En nuestro caso eliminamos las siguientes columnas: workclass (String), education (String), marital-status (String), occupation (String), relationship (String), race (String), sex (String), native-country (String). [20]
- Missing Values: Maneja los valores perdidos de las distintas celdas, de tal forma que se puede actuar reponiendo valores en forma de media, más frecuente ó eliminar fila. En nuestro caso lo usaremos para aquellas columnas de tipo numérico escoger en donde ocurra un valor perdido la media. [23]
- Normalizer: Normaliza los valores de las columnas numéricas entre un rango indicado previamente, en nuestro caso [0.0 , 1.0] [20]

Puesto que la configuración de los nodos de preprocesamiento es idéntica a las del apartado del algoritmo Decision Tree 5.1 no se van a mostrar las imágenes. Para más información por lo tanto volver al anterior subapartado.

La siguiente captura corresponde al flujo de trabajo del preprocesamiento previo al algoritmo Random Forest.

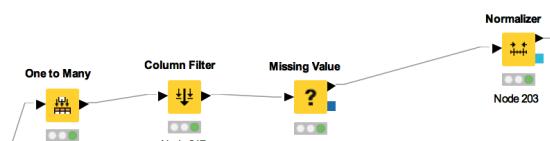


Figura 5.10: Workflow de preprocesamiento 1, del algoritmo Random Forest para la versión del apartado anterior 4.

Los datos obtenidos por los modelos explicados en el anterior apartado 4 son los siguientes:⁷

Algorithm	TruePositives	FalsePositives	TrueNegatives	FalseNegatives	Recall	Precision	Specificity	F-measure	G-mean	Accuracy	AUC
Random Forest V1 PP1	7187	1943	35212	4500	0.6149	0.7871	0.9477	0.6904	0.7634	0.8290	0.8680

Tabla 5.7: Tabla del algoritmo Random Forest con la configuración del apartado anterior 4 y el segundo tipo de preprocesamiento.

⁷Las iniciales PP se refieren a PreProcesamiento. Y el número que le sigue, a la versión del preprocesamiento.

En segundo lugar hemos considerado un segundo preprocesamiento para la versión obtenida del apartado anterior 4 las siguientes medidas de preprocesamiento:

- Equal Size Sampling: Elimina filas aleatoriamente pertenecientes a la clase de la mayoría, de tal forma que devuelve todas las filas de la clase minoritaria y una muestra aleatoria de la clase mayoritaria. De tal forma que realiza un balanceo entre clases. Más conocido como UnderSampling. [24]

La siguiente captura corresponde a la configuración escogida para el nodo Equal Size Sampling:

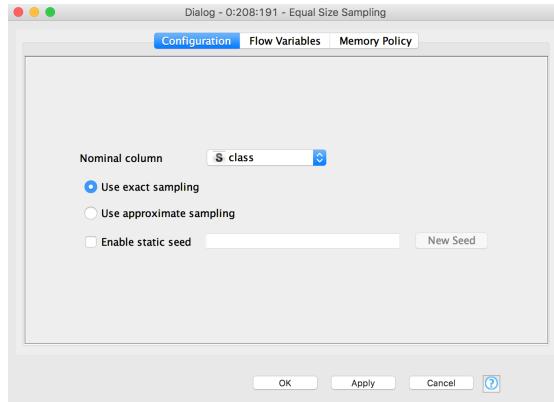


Figura 5.11: Configuración del nodo equal size sampling.

Los datos obtenidos por los modelos explicados en el anterior apartado 4 son los siguientes:

Algorithm	TruePositives	FalsePositives	TrueNegatives	FalseNegatives	Recall	Precision	Specificity	F-measure	G-mean	Accuracy	AUC
Random Forest V1 PP2	9892	2036	9651	1795	0.8464	0.8293	0.8257	0.8377	0.8360	0.8360	0.9132

Tabla 5.8: Tabla del algoritmo Random Forest con la configuración del apartado anterior 4 y el segundo tipo de preprocesamiento.

En tercer lugar hemos considerado un tercer y último preprocesamiento para la versión obtenida del apartado anterior 4 las siguientes medidas de preprocesamiento:

- Equal Size Sampling: Elimina filas aleatoriamente pertenecientes a la clase de la mayoría, de tal forma que devuelve todas las filas de la clase minoritaria y una muestra aleatoria de la clase mayoritaria. De tal forma que realiza un balanceo entre clases. Más conocido como UnderSampling. [24]
- Missing Values: Maneja los valores perdidos de las distintas celdas, de tal forma que se puede actuar reponiendo valores en forma de media, más frecuente ó eliminar fila. En nuestro caso lo usaremos para aquellas columnas de tipo numérico escoger en donde ocurra un valor perdido la media. [23]
- Normalizer: Normaliza los valores de las columnas numéricas entre un rango indicado previamente, en nuestro caso [0.0 , 1.0] [20]

Puesto que la configuración de los nodos de preprocesamiento es idéntica a las del apartado del algoritmo Decision Tree 5.1 no se van a mostrar las imágenes. Para más información por lo tanto volver al anterior subapartado.

Los datos obtenidos por los modelos explicados en el anterior apartado 4 son los siguientes:

Algorithm	TruePositives	FalsePositives	TrueNegatives	FalseNegatives	Recall	Precision	Specificity	F-measure	G-mean	Accuracy	AUC
Random Forest V1 PP3	9885	2071	9616	1802	0.8458	0.8267	0.8227	0.8361	0.8342	0.8343	0.9117

Tabla 5.9: Tabla del algoritmo Random Forest con la configuración del apartado anterior 4 y el tercer tipo de preprocesamiento.

Para finalizar con la visualización de los distintos resultados, juntamos las distintas tablas y mostramos la gráfica ROC conjunta.

Algorithm	TruePositives	FalsePositives	TrueNegatives	FalseNegatives	Recall	Precision	Specificity	F-measure	G-mean	Accuracy	AUC
Random Forest V1	7127	1837	35318	4560	0.6098	0.7950	0.9505	0.6902	0.7613	0.8690	0.9094
Random Forest V1 PP1	7187	1943	35212	4500	0.6149	0.7871	0.9477	0.6904	0.7634	0.8290	0.8920
Random Forest V1 PP2	9892	2036	9651	1795	0.8464	0.8293	0.8257	0.8377	0.8360	0.8360	0.9132
Random Forest V1 PP3	9885	2071	9616	1802	0.8458	0.8267	0.8227	0.8361	0.8342	0.8343	0.9117

Tabla 5.10: Tabla comparativa del algoritmo Random Forest con la configuración del apartado anterior 4 y todos los tipos de preprocesamiento.

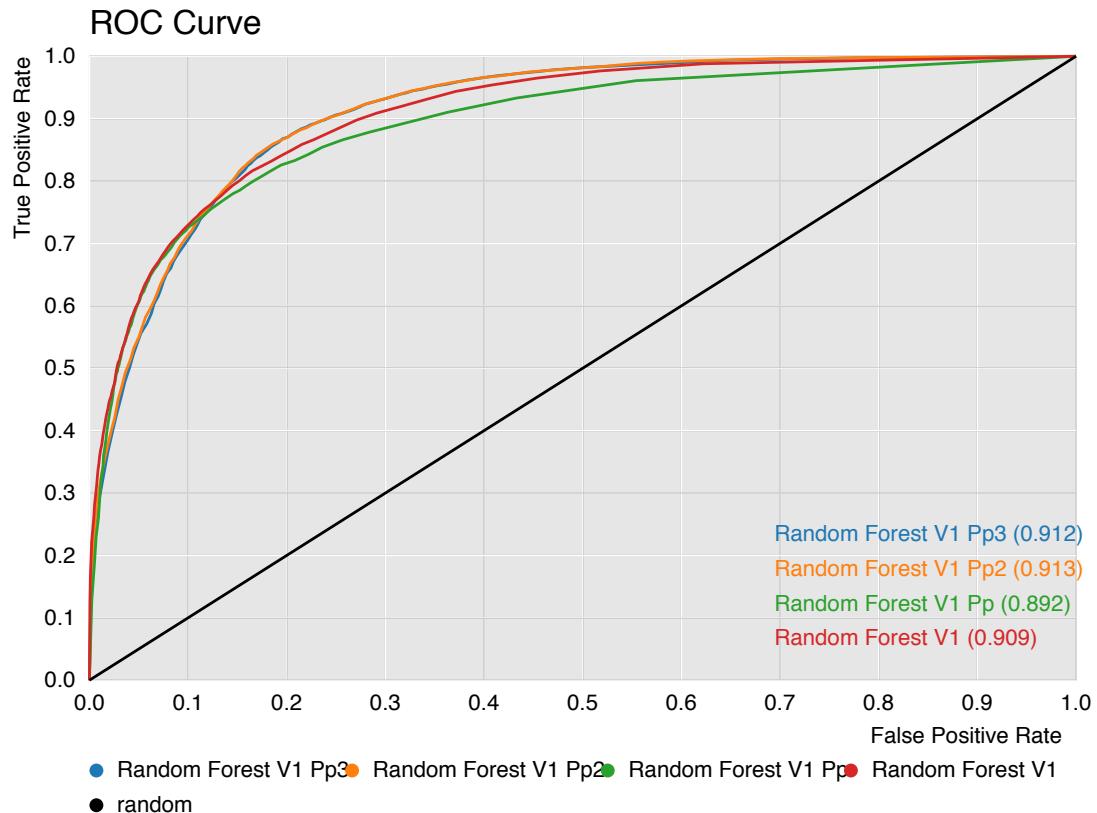


Figura 5.12: Gráfica comparativa del algoritmo Random Forest con la configuración del apartado anterior 4 y todos los tipos de preprocesamiento.

Antes de continuar con el siguiente algoritmo cabe analizar varias cosas:

En primer lugar con los distintos preprocesamientos se ha intentado en primer lugar con el primer preprocesado, eliminar columnas categóricas, tratamiento de valores perdidos y normalización de valores numéricos. En el segundo preprocesamiento se ha intentado contrarrestar el posible desbalanceo de clases, y en el segundo preprocesamiento se ha

intentado tanto contrarrestar el desbalanceo de clases como el tratamiento de los valores perdidos y la normalización.

También hay que mencionar que al realizar el balanceo entre clases con los preprocesamiento 2 y 3 el conjunto de datos como ya se ha comentado disminuye, por lo tanto los resultados posteriores difícilmente podrán compararse con los resultados de los modelos con todo el conjunto de datos. Aún así en la tabla y gráfica anterior se han metido en común todos estos datos.

En particular entre el Random Forest sin preprocesamiento y con el primer preprocesamiento hay poca diferencia, pero la que hay nos habla de un empeoramiento del modelo final cuando introducimos preprocesamiento. Empeora el AUC, Precision, Accuracy y Specificity. Provocado entre otras cosas por el aumento de falsos positivos cuando realizamos el primer preprocesamiento. Entre otras cosas obtenemos un modelo con menor precisión.

Entre los otros dos modelos, se mejoran los resultados considerablemente, aumentando bastante los verdaderos positivos por ejemplo, y el AUC. Aunque nos habla también de dos modelos con un bajo Accuracy con respecto al algoritmo base.

5.3. Naive Bayes

Para el algoritmo Naive Bayes hemos considerado 4 preprocesamientos diferentes, uno más restrictivo que el otro.

En primer lugar hemos considerado para la versión obtenida del apartado anterior 4 las siguientes medidas de preprocesamiento:

- One to Many: Transforma todos los posibles valores de una columna en una nueva columna. De tal forma que en las nuevas columnas sale 1 cuando la fila pertenece a ese valor, y 0 cuando no. El nodo agrega tantas columnas como posibles valores. En nuestro caso lo hacemos con las siguientes columnas: workclass (String), education (String), marital-status (String), occupation (String), relationship (String), race (String), sex (String), native-country (String). [20]
- Column Filter: Elimina las columnas que previamente han sido transformadas en nuevas columnas por el nodo One to Many. En nuestro caso eliminamos las siguientes columnas: workclass (String), education (String), marital-status (String), occupation (String), relationship (String), race (String), sex (String), native-country (String). [20]
- Missing Values: Maneja los valores perdidos de las distintas celdas, de tal forma que se puede actuar reponiendo valores en forma de media, más frecuente ó eliminar fila. En nuestro caso lo usaremos para aquellas columnas de tipo numérico escoger en donde ocurra un valor perdido la media. [23]

Puesto que la configuración de los nodos de preprocesamiento es idéntica a las del apartado del algoritmo Decision Tree 5.1 no se van a mostrar las imágenes. Para más información por lo tanto volver al anterior subapartado.

Los datos obtenidos por los modelos explicados en el anterior apartado 4 son los siguientes:⁸

Algorithm	TruePositives	FalsePositives	TrueNegatives	FalseNegatives	Recall	Precision	Specificity	F-measure	G-mean	Accuracy	AUC
Naive Bayes V1 PP	8903	6547	30608	2784	0.76178	0.5762	0.8237	0.6561	0.7921	0.8089	0.8601

Tabla 5.11: Tabla del algoritmo Naive Bayes con las configuraciones del apartado anterior 4 y el primer tipo de preprocesamiento.

⁸Las iniciales PP se refieren a PreProcesamiento. Y el número que le sigue, a la versión del preprocesamiento.

En segundo lugar hemos considerado un segundo preprocesamiento para la versión obtenida del apartado anterior 4 con las siguientes medidas de preprocesamiento:

- One to Many: Transforma todos los posibles valores de una columna en una nueva columna. De tal forma que en las nuevas columnas sale 1 cuando la fila pertenece a ese valor, y 0 cuando no. El nodo agrega tantas columnas como posibles valores. En nuestro caso lo hacemos con las siguientes columnas: workclass (String), education (String), marital-status (String), occupation (String), relationship (String), race (String), sex (String), native-country (String). [20]
- Column Filter: Elimina las columnas que previamente han sido transformadas en nuevas columnas por el nodo One to Many. En nuestro caso eliminamos las siguientes columnas: workclass (String), education (String), marital-status (String), occupation (String), relationship (String), race (String), sex (String), native-country (String). [20]
- Missing Values: Maneja los valores perdidos de las distintas celdas, de tal forma que se puede actuar reponiendo valores en forma de media, más frecuente ó eliminar fila. En nuestro caso lo usaremos para aquellas columnas de tipo numérico escoger en donde ocurra un valor perdido la media. [23]
- Normalizer: Normaliza los valores de las columnas numéricas entre un rango indicado previamente, en nuestro caso [0.0 , 1.0] [20]

Puesto que la configuración de los nodos de preprocesamiento es idéntica a las del apartado del algoritmo Decision Tree 5.1 no se van a mostrar las imágenes. Para más información por lo tanto volver al anterior subapartado.

Los datos obtenidos por los modelos explicados en el anterior apartado 4 son los siguientes:

Algorithm	TruePositives	FalsePositives	TrueNegatives	FalseNegatives	Recall	Precision	Specificity	F-measure	G-mean	Accuracy	AUC
Naive Bayes V1 PP2	8903	6547	30608	2784	0.7617	0.5762	0.8237	0.6561	0.7921	0.8089	0.8601

Tabla 5.12: Tabla del algoritmo Naive Bayes con las configuraciones del apartado anterior 4 y el segundo tipo de preprocesamiento.

En tercer lugar hemos considerado un tercer preprocesamiento para la versión obtenida del apartado anterior 4 con las siguientes medidas de preprocesamiento:

- Normalizer: Normaliza los valores de las columnas numéricas entre un rango indicado previamente, en nuestro caso [0.0 , 1.0] [20]

Puesto que la configuración de los nodos de preprocesamiento es idéntica a las del apartado del algoritmo Decision Tree 5.1 no se van a mostrar las imágenes. Para más información por lo tanto volver al anterior subapartado.

Los datos obtenidos por los modelos explicados en el anterior apartado 4 son los siguientes:

Algorithm	TruePositives	FalsePositives	TrueNegatives	FalseNegatives	Recall	Precision	Specificity	F-measure	G-mean	Accuracy	AUC
Naive Bayes V1 PP3	8959	6279	30876	2728	0.7665	0.5879	0.8310	0.6654	0.7981	0.8155	0.8907

Tabla 5.13: Tabla del algoritmo Naive Bayes con las configuraciones del apartado anterior 4 y el tercer tipo de preprocesamiento.

En cuarto lugar hemos considerado un cuarto preprocesamiento para la versión obtenida del apartado anterior 4 con las siguientes medidas de preprocesamiento:

- One to Many: Transforma todos los posibles valores de una columna en una nueva columna. De tal forma que en las nuevas columnas sale 1 cuando la fila pertenece a ese valor, y 0 cuando no. El nodo agrega tantas columnas como posibles valores. En nuestro caso lo hacemos con las siguientes columnas: workclass (String), education (String), marital-status (String), occupation (String), relationship (String), race (String), sex (String), native-country (String). [20]
- Column Filter: Elimina las columnas que previamente han sido transformadas en nuevas columnas por el nodo One to Many. En nuestro caso eliminamos las siguientes columnas: workclass (String), education (String), marital-status (String), occupation (String), relationship (String), race (String), sex (String), native-country (String). [20]
- Normalizer: Normaliza los valores de las columnas numéricas entre un rango indicado previamente, en nuestro caso [0.0 , 1.0] [20]

Puesto que la configuración de los nodos de preprocesamiento es idéntica a las del apartado del algoritmo Decision Tree 5.1 no se van a mostrar las imágenes. Para más información por lo tanto volver al anterior subapartado.

Los datos obtenidos por los modelos explicados en el anterior apartado 4 son los siguientes:

Algorithm	TruePositives	FalsePositives	TrueNegatives	FalseNegatives	Recall	Precision	Specificity	F-measure	G-mean	Accuracy	AUC
Naive Bayes V1 PP4	8903	6547	30608	2784	0.7617	0.5762	0.8237	0.6561	0.7921	0.8089	0.8601

Tabla 5.14: Tabla del algoritmo Naive Bayes con las configuraciones del apartado anterior 4 y el cuarto tipo de preprocesamiento.

Para finalizar obtenemos una tabla con todos los datos juntos y la gráfica ROC para poder comparar los distintos preprocesamientos:

Algorithm	TruePositives	FalsePositives	TrueNegatives	FalseNegatives	Recall	Precision	Specificity	F-measure	G-mean	Accuracy	AUC
Naive Bayes V1	8959	6279	30876	2728	0.7665	0.5879	0.8310	0.6654	0.7981	0.8155	0.891
Naive Bayes V1 PP	8903	6547	30608	2784	0.76178	0.5762	0.8237	0.6561	0.7921	0.8089	0.8601
Naive Bayes V1 PP2	8903	6547	30608	2784	0.7617	0.5762	0.8237	0.6561	0.7921	0.8089	0.8601
Naive Bayes V1 PP3	8959	6279	30876	2728	0.7665	0.5879	0.8310	0.6654	0.7981	0.8155	0.8907
Naive Bayes V1 PP4	8903	6547	30608	2784	0.7617	0.5762	0.8237	0.6561	0.7921	0.8089	0.8601

Tabla 5.15: Tabla del algoritmo Naive Bayes con las configuraciones del apartado anterior 4 y el primer tipo de preprocesamiento.

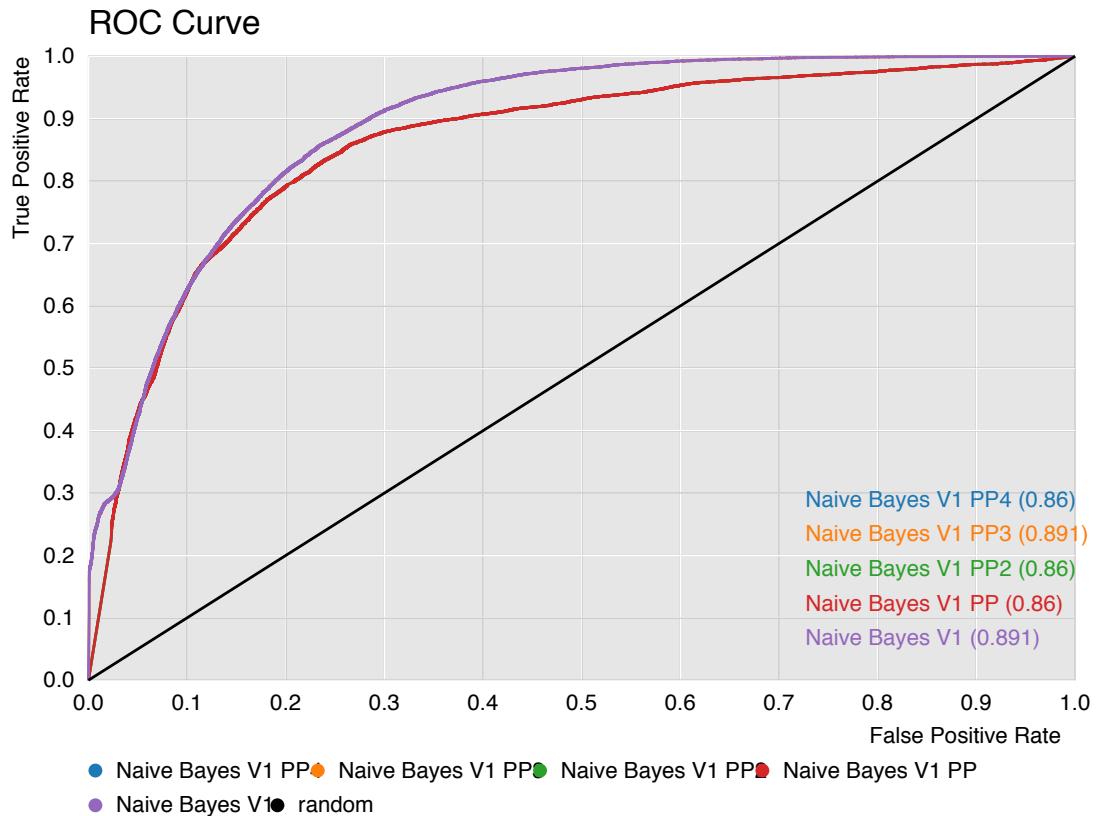


Figura 5.13: Gráfica comparativa del algoritmo Naive Bayes con las configuraciones del apartado anterior y el primer tipo de preprocesamiento.

Para finalizar se procede a analizar los distintos preprocesamientos y los resultados de los mismo:

En primer lugar con el primer preprocesamiento se tratan las variables categóricas y los valores perdidos. Para el segundo preprocesamiento se hace el mismo preprocesamiento que en el primero con la introducción del normalizado de los datos numéricos. En

el tercer preprocesamiento se reduce el mismo a normalizar únicamente los datos numéricos. Y por último en el cuarto preprocesamiento se tratan las variables categóricas y se normalizan los datos.

Con respecto en primer lugar a la gráfica ROC comparativa, se ve perfectamente como el mejor modelo es el que no tiene nada de preprocesamiento y el algoritmo que hace uso del tercer tipo de preprocesamiento. Es curioso, que cuando se empiezan a tratar más profundamente los datos como lo hacen los otros tipos de preprocesamiento, los resultados empeoran. Por ejemplo el AUC. Ocurre lo mismo con los valores de Precision, Accuracy y Recall. De tal forma que decaen en los preprocesamientos 1, 2 y 3 los verdaderos positivos y verdaderos negativos.

De tal forma, y observando los datos, se aprecia perfectamente que obtenemos peores modelos con la entrada de un preprocesamiento no muy exhaustivo pero si notorio.

6. Interpretación de resultados

En este apartado se extraerán las distintas conclusiones sobre los factores que determinan que una persona pueda ganar más de 50K dólares anuales o no. Para llegar a estas conclusiones hemos tenido en cuenta los siguientes aspectos.

Esta gráfica muestra el total de cada clase en el conjunto de datos que tenemos. Se ve perfectamente como la clase mayoritaria es $\leq 50K$ (menos o igual a 50K dólares anuales) con un total de 37155 instancias. Mientras, la clase minoritaria $>50K$ (más de 50K dólares anuales) tiene un total de 11687 instancias.

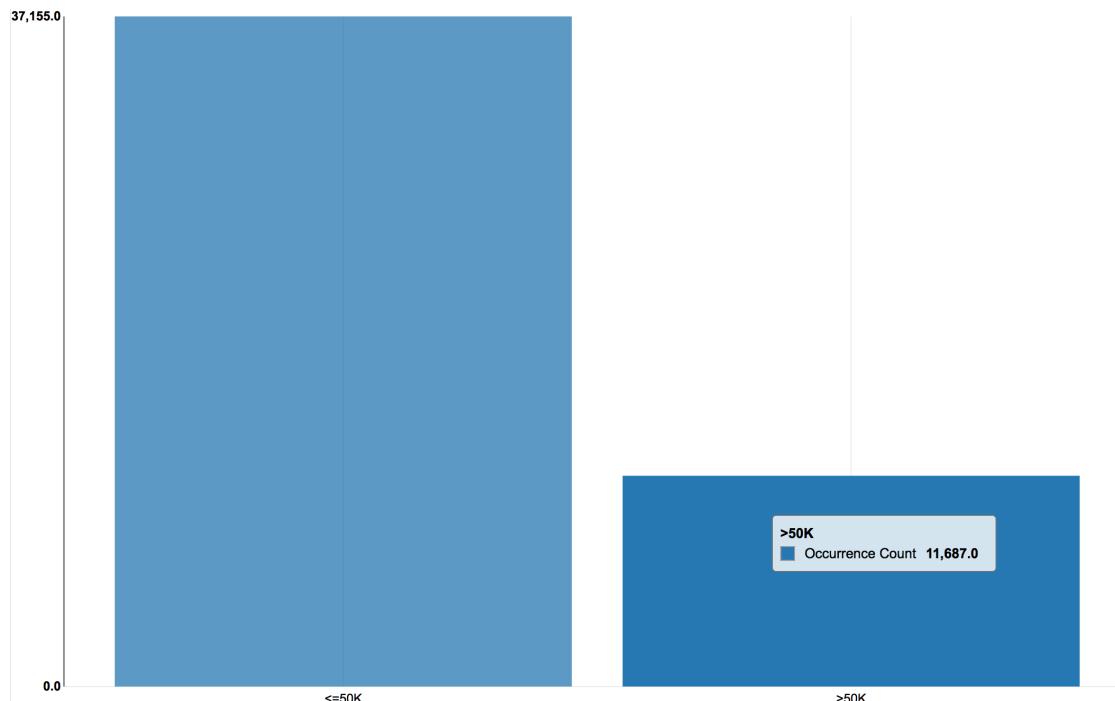


Figura 6.1: Gráfica de barras con total de ocurrencias de cada clase.

En la siguiente gráfica se muestra la clase de trabajo en donde se gana mas de 50K dolares anuales. En este caso despejamos pocas dudas, porque tal y como es de prever, las clases de trabajo que no permiten obtener unos ingresos anuales mayores a 50K dolares son la gente sin empleo y la gente sin paga. Todos los demás al menos un individuo con esa clase de trabajo ha conseguido obtener 50K dolares.

Parallel Coordinates Plot

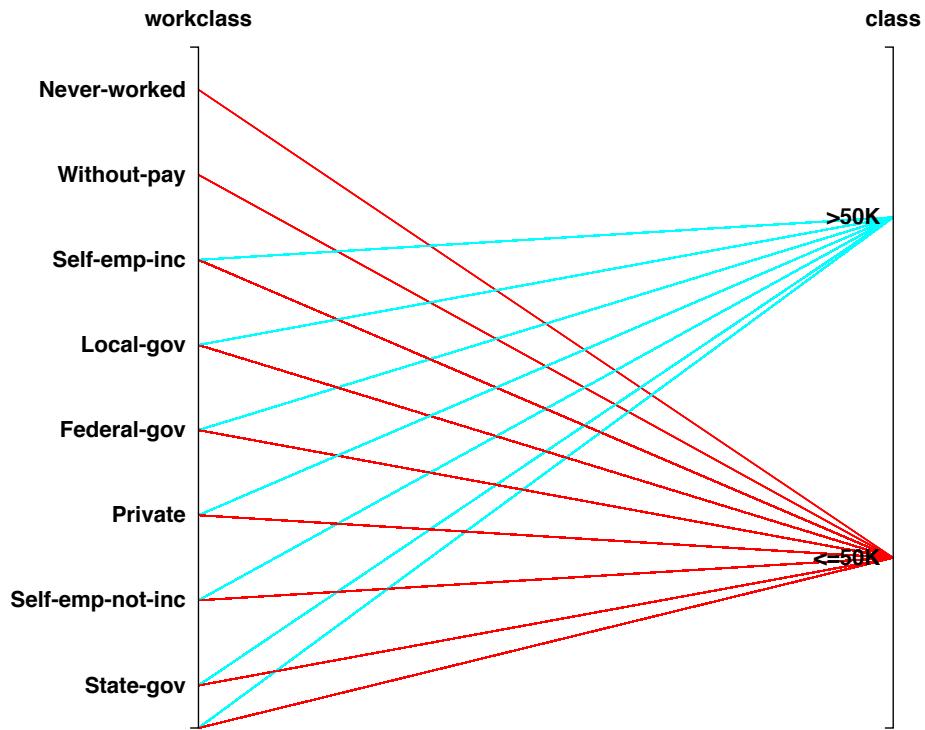


Figura 6.2: Gráfica que compara las clases con las clases de trabajo.

En consonancia con esta gráfica anterior, se muestra a continuación otra gráfica con las clases de trabajo que aparecen en más instancias del conjunto de datos. En la siguiente gráfica vemos como la clase de trabajo mayoritaria es por lo privado. De tal forma que abarca más del 50 % del total, lo que nos hace prever que alguno de estos trabajadores puedan estar relacionados con la clase $>50K$.

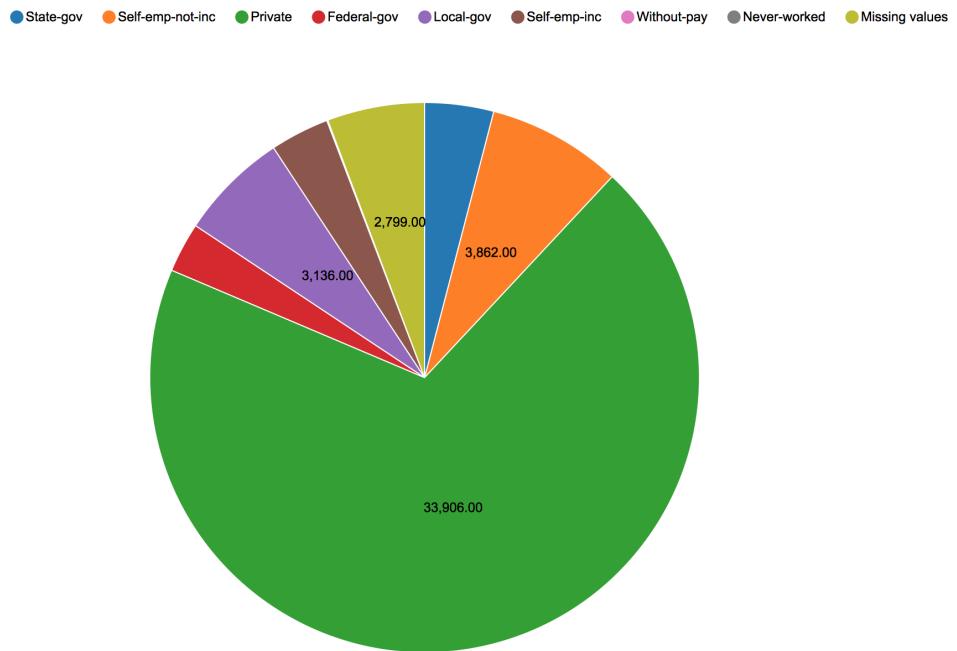


Figura 6.3: Gráfica con número de instancias de cada clase de trabajo.

En la siguiente gráfica es donde se puede ratificar nuestros pensamientos, en donde la clase >50K por mayoría se encuentra situada con la clase de trabajo "Private"(Privado). También vemos como por detrás se sitúan "local govz "self-emp-not-inc" que curiosamente también son los tipos de trabajos que más aparecen en nuestro conjunto de datos.

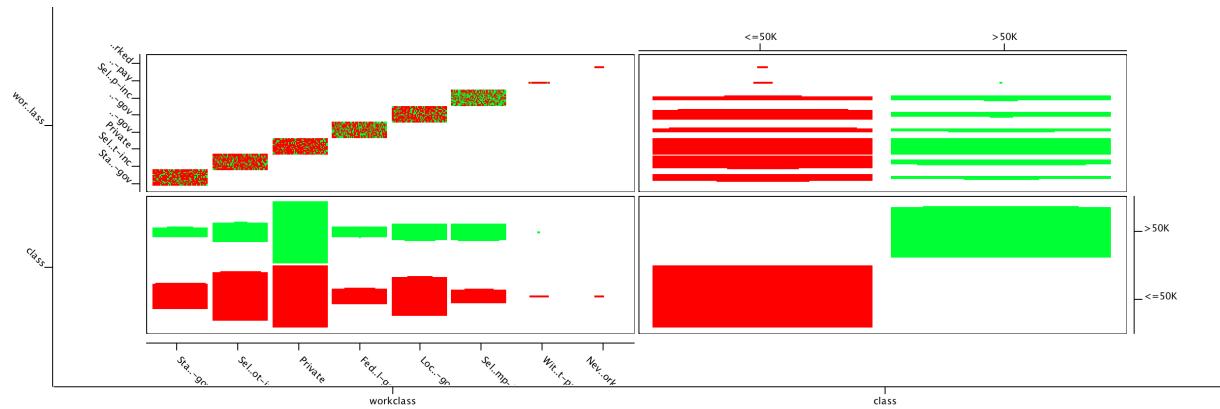


Figura 6.4: Gráfica comparativa de la clase de trabajo y la clase del conjunto de datos.

Para la siguiente gráfica se puede ver las distintas distribuciones que tienen las clases sobre las edades. De tal forma que sacamos conclusiones como que hay instancias en donde las edades están por encima de los 73 años pero no son habituales. Estos datos son llamados outliers (valores atípicos).

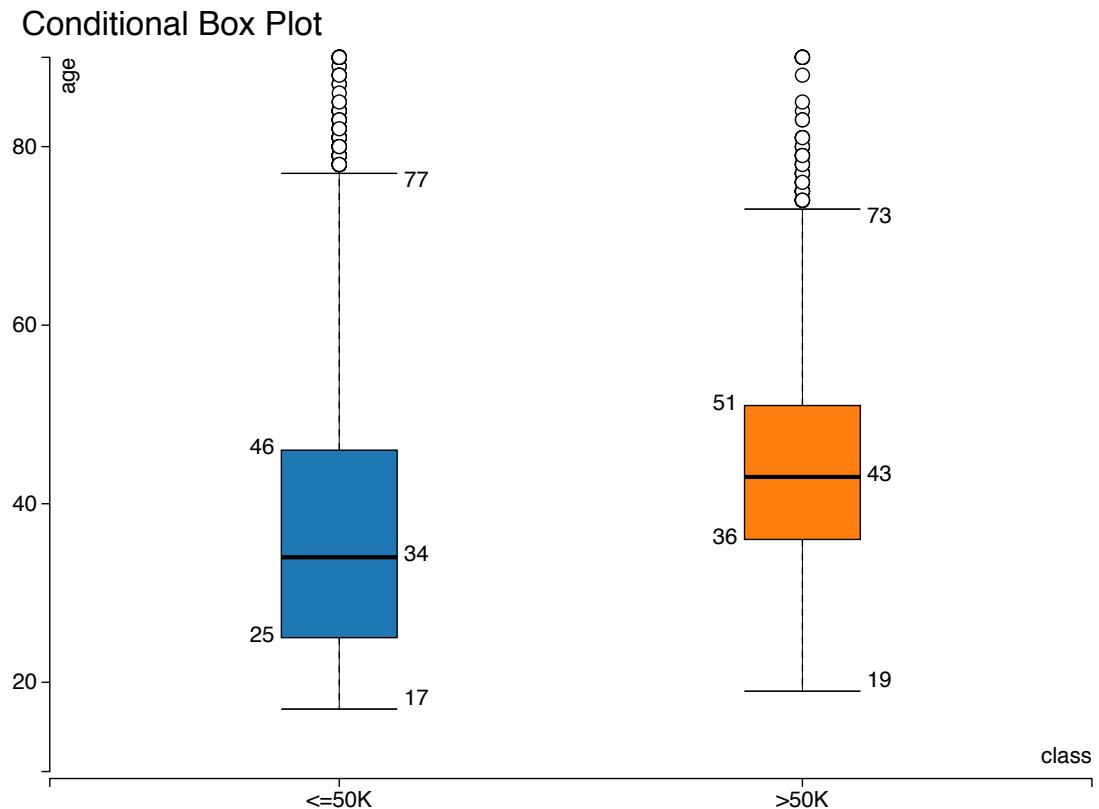


Figura 6.5: Gráfica Box comparando las dos clases con la edad.

Para sustentar este análisis tenemos la siguiente gráfica en donde se ve claramente como los datos de ganar mas de 50K dolares al año se distribuyen mayormente entre los 35 y 53 años. En este intervalo parece recogerse la mayoría de las edades e instancias que ganan más de 50K dolares anuales.

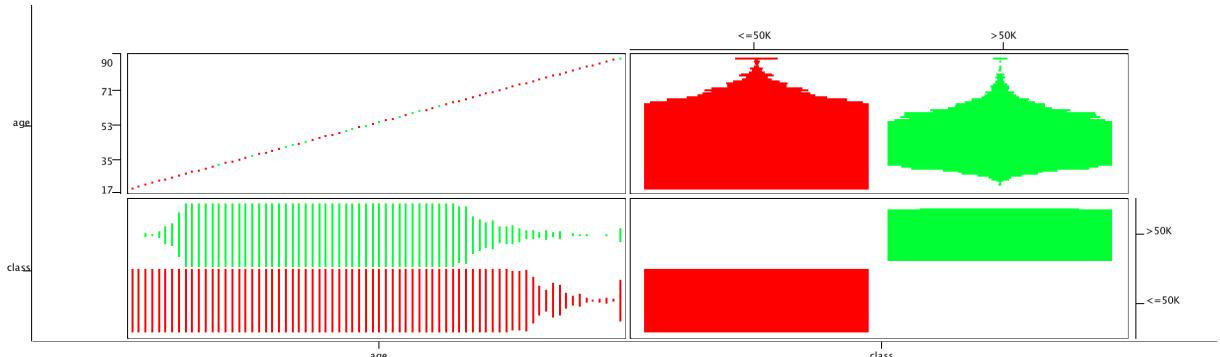


Figura 6.6: Gráfica comparando las dos clases con la edad.

Para finalizar con este análisis, haremos un pequeño recorrido por el árbol de decisión extraido. En el se ve que la forma más rápida de acceder a las instancias que han ganado más de 50K dolares anuales es fijándose en las que no tienen familia y tienen unas ganancias de mas de 8296 dolares.

En contraposición de lo anterior tenemos a los que no tienen familia, tienen unas ganancias iguales o menores a 8296 dolares y una educación de preescolar, que ganan $\leq 50K$ dolares anuales. Aunque parezca que la educación es un aspecto crucial, fíjémonos en que más de la mitad de los que no tienen familia, tienen unas ganancias igual o menor a 8296 dolares y una educación "master", siguen ganando igual o menos de 50K dolares anuales.

Si nos vamos a otros aspectos en donde la clase mayoritaria es ganar más de 50K dólares anuales, como por ejemplo tener una relación de marido, ser doctorado y capital ganado mayor a 5095 dólares, vemos como el 100 % de las instancias son de la clase $>50K$. En este caso pocas, porque data de 55 instancias.

Aún más descarado es para las instancias con relación de mujer y educación con doctorado, en donde 18 de las 19 instancias son de la clase $>50K$. Lo cuál nos indica una alta probabilidad de ganar más de 50K dólares anuales con estos dos aspectos, sin tener en cuenta si quiera el capital ganado.

Otras instancias condenadas mayoritariamente a no ganar mas de 50K dólares anuales son las que tienen como relación mujer y educación entre séptimo y octavo, donde las 24 instancias con estos valores son de la clase $\leq 50K$. Igual ocurre con los no casados,

capital ganado menor o igual a 7139 dólares y educación noveno, en donde 81 de las 82 instancias son de la clase $<=50K$.

En el caso de no estar casado, tener unas ganancias mayores a 7139 dólares y una ocupación de ventas, se tiene asegurado ganar más de 50K dólares anuales. Además en este caso hay un gran número de instancias que lo confirman, 2033.

Cabe reseñar que en el apartado de relaciones, los dos valores con más atributos de la clase $>50K$ dólares son marido y mujer, junto con sin familia en menor grado. En el caso de marido hay un total de 8709 que son más de la mitad del total de instancias de la clase. Y en el caso de mujer 967 instancias. Si que es verdad que el atributo marido es el que más instancias lo poseen con un total de 15800, por delante de sin familia que tiene un total de 10046 instancias, mientras que mujer tiene 1856 instancias, luego más del doble de las instancias en el conjunto de datos con relación igual a mujer ganan más de 50K dólares anuales, dato más que interesante.

Estos últimos apuntes están sacados del Decision Tree.

7. Contenido adicional

Referencias

- [1] <https://archive.ics.uci.edu/ml/datasets/adult>, consultado el 1 de octubre de 2017.
- [2] <https://www.knime.com/nodeguide/analytics/classification-and-predictive-modelling/gradient-boosted-trees>, consultado el 1 de octubre de 2017.
- [3] <https://www.knime.com/nodeguide/analytics/classification-and-predictive-modelling/example-for-learning-a-decision-tree>, consultado el 1 de octubre de 2017.
- [4] <https://www.knime.com/nodeguide/analytics/meta-learning/learning-a-random-forest>, consultado el 1 de octubre de 2017.
- [5] <https://www.knime.com/nodeguide/analytics/classification-and-predictive-modelling/example-for-learning-a-naive-bayes-model>, consultado el 1 de octubre de 2017.
- [6] https://www.ibm.com/support/knowledgecenter/es/SSLVMB_23.0.0/spss/base/idh_idd_knn_variables.html, consultado el 1 de octubre de 2017.
- [7] <https://www.knime.com/nodeguide/analytics/classification-and-predictive-modelling/example-for-learning-a-neural-network>, consultado el 1 de octubre de 2017.
- [8] https://www.ibm.com/support/knowledgecenter/es/SSGU8G_11.50.0/com.ibm.ddi.doc/ids_ddi_066.htm, consultado el 1 de octubre de 2017.
- [9] <https://www.knime.com/nodeguide/etl-data-manipulation/filtering/column-filter>, consultado el 1 de octubre de 2017.
- [10] https://www.ibm.com/support/knowledgecenter/en/SSEPGG_9.7.0/com.ibm.im.model.doc/c_minimum_number_of_records_per_internal_node.html, consultado el 1 de octubre de 2017.
- [11] https://www.ibm.com/support/knowledgecenter/en/SSC5CS/com.ibm.pm.doc/using/t_configure_model_settings.html, consultado el 1 de octubre de 2017.
- [12] <http://dmg.org/pmmml/v4-2-1/NaiveBayes.html>, consultado el 1 de octubre de 2017.
- [13] https://www.ibm.com/support/knowledgecenter/en/SSLVMB_22.0.0/com.ibm.spss.statistics.help/spss/base/idh_idd_knn_neighbors.htm, consultado el 1 de octubre de 2017.
- [14] <https://www.knime.com/nodeguide/etl-data-manipulation/joining-and-concatenating/concatenate>, consultado el 1 de octubre de 2017.
- [15] <http://gim.unmc.edu/dxtests/roc3.htm>, consultado el 1 de octubre de 2017.

- [16] <http://gim.unmc.edu/dxtests/roc3.htm>, consultado el 1 de octubre de 2017.
- [17] [https://es.wikipedia.org/wiki/Especificidad_\(epidemiolog%C3%ADA\)](https://es.wikipedia.org/wiki/Especificidad_(epidemiolog%C3%ADA)), consultado el 1 de octubre de 2017.
- [18] [https://es.wikipedia.org/wiki/Verdadero_negativo_\(medicina\)](https://es.wikipedia.org/wiki/Verdadero_negativo_(medicina)), consultado el 1 de octubre de 2017.
- [19] https://www.google.es/url?sa=t&rct=j&q=&esrc=s&source=web&cd=3&ved=0ahUKEwj38Mqxr53XAhVQGsAKHUXSDaAQFgg3MAI&url=http%3A%2F%2Fwww.springer.com%2Fcda%2Fcontent%2Fddocument%2Fcda_downloaddocument%2F9783319629889-c2.pdf%3FSGWID%3D0-0-45-1618772-p181018669&usg=A0vVaw0wj43tix-3t42x4N-SpB3t, consultado el 1 de octubre de 2017.
- [20] <https://www.knime.com/knime-online-self-training-lesson-3>, consultado el 1 de octubre de 2017.
- [21] https://www.ibm.com/support/knowledgecenter/es/SSLVMB_22.0.0/com.ibm.spss.statistics.help/spss/mva/idh_miss.htm, consultado el 1 de octubre de 2017.
- [22] <https://www.knime.com/nodeguide/etl-data-manipulation>, consultado el 1 de octubre de 2017.
- [23] <https://www.knime.com/blog/how-to-deal-with-missing-values>, consultado el 1 de octubre de 2017.
- [24] http://eio.usc.es/pub/mte/descargas/ProyectosFinMaster/Proyecto_1469.pdf, consultado el 1 de octubre de 2017.