Universidad de Granada

PRÁCTICA 1:
Análisis Predictivo Empresarial Mediante Clasificación

Inteligencia de negocio

Realizado por:
Francisco Solano López Rodríguez
DNI: 20100444P
Email: fransol0728@correo.ugr.es

DOBLE GRADO EN INGENIERÍA INFORMÁTICA Y MATEMÁTICAS.
QUINTO CURSO

Índice

1. Introducción

El objetivo de esta práctica consiste en construir un sistema inteligente de apoyo a la decisión que analice artículos que serán publicados y prediga si se volverán populares. El problema al que nos enfrentamos se trata de un problema de clasificación binaria, donde las dos posibles clasificaciones son popular y no popular.

El conjunto de datos utilizado consta de 39644 noticias extraídas en 2015 de la web `http://mashable.com`. El número de características es de 58 y además se dispone de la clase a la que pertenece cada instancia. Hay dos clases posibles a las que puede pertenecer una instancia: popular y no popular. Las etiquetas de la clase han sido obtenidas a partir del numero de veces que la noticia ha sido compartida, para ello se ha fijado como umbral 3000 veces, entonces una noticia es considerada como popular si el número de veces que la noticia ha sido compartida está por encima de ese umbral. De los 39664 artículos de los que disponemos en el conjunto de datos tenemos que 30718 están etiquetados como no populares y 8926 están clasificados como populares. Si hacemos los cálculos veremos que aproximadamente el 77.5 % están etiquetados como no populares y solo un 22.5 % como populares. Así que nos encontramos ante un problema de clasificación donde la distribución de etiquetas está desbalanceada. Además se tiene que el 24 % de los valores de los atributos están perdidos.

A continuación se muestran algunas de las características de las que se disponen:

- Número de palabras en el título. Numérica.
- Número de enlaces. Numérica.
- Número de imágenes. Numérica.
- Día de la semana en la que fue publicado el artículo. Categórica.

Para resolver el problema utilizaremos validación cruzada con k=5 y un muestreo estratificado para mantener la proporción de clases.

Los algoritmos considerados van a ser los siguientes:

- Regresión logística
- Gradient Boosted
- Random Forest
- Naive Bayes
- KNN
- Red neuronal

2. Resultados obtenidos

En este apartado tan solo se ejecutaran los seis algoritmos elegidos con los parámetros por defecto para hacer una comparación entre ellos. Será en el apartado de **Configuración de algoritmos**, donde probaremos distintos parámetros y donde intentaremos mejorar cada algoritmo quedandonos con aquellos parámetros que nos den mejores resultados. Además en esta sección no se realizará procesado de datos, a excepción de algunos algoritmos que necesitaran tratar los valores perdidos para poder funcionar. El procesado de datos será analizado en la sección 5.

A continuación se muestra una imagen en la que podemos ver el flujo de trabajo usado en este apartado. Lo primero que hacemos es la lectura de los datos, después se ha añadido un color a las filas para distinguir la clase a la que pertenece cada instancia. A continuación se ha realizado una validación cruzada la cual explicaremos después y por último hemos obtenido los resultados.

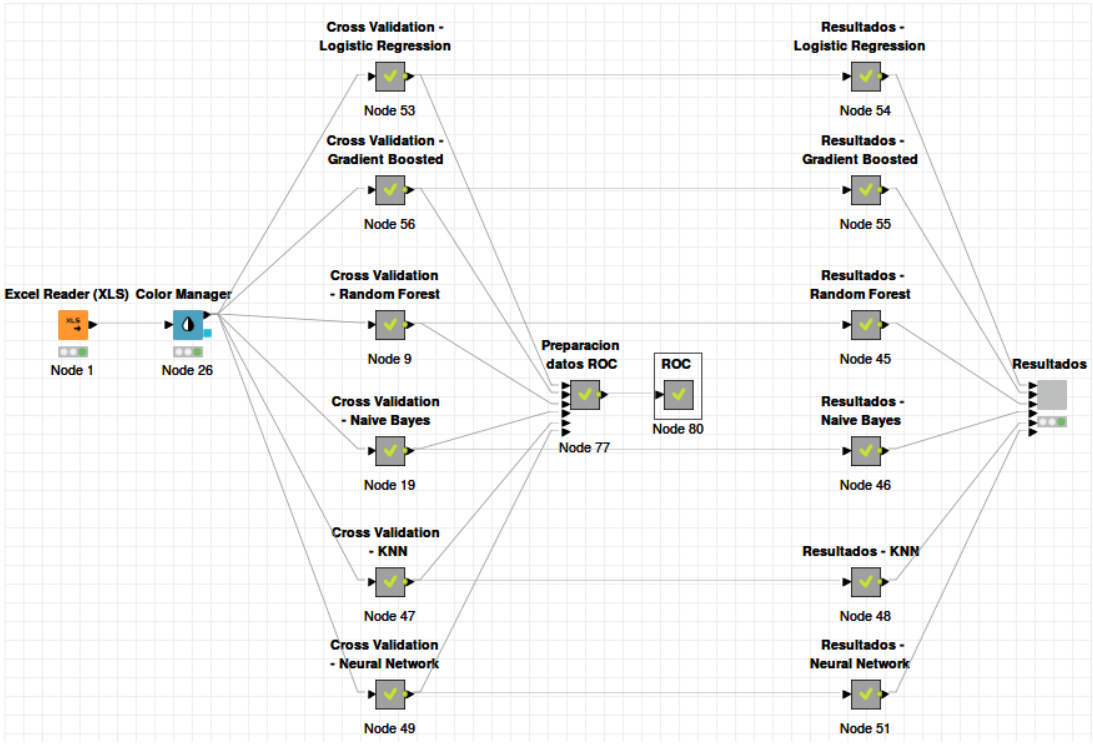


Figura 1: Flujo de trabajo

En la lectura de los datos hemos leído los datos del archivo onlinenewspopularity.xlsx. Después se ha pasado a realizar a validación cruzada, a continuación se muestra una ejemplo de validación cruzada con random forest.

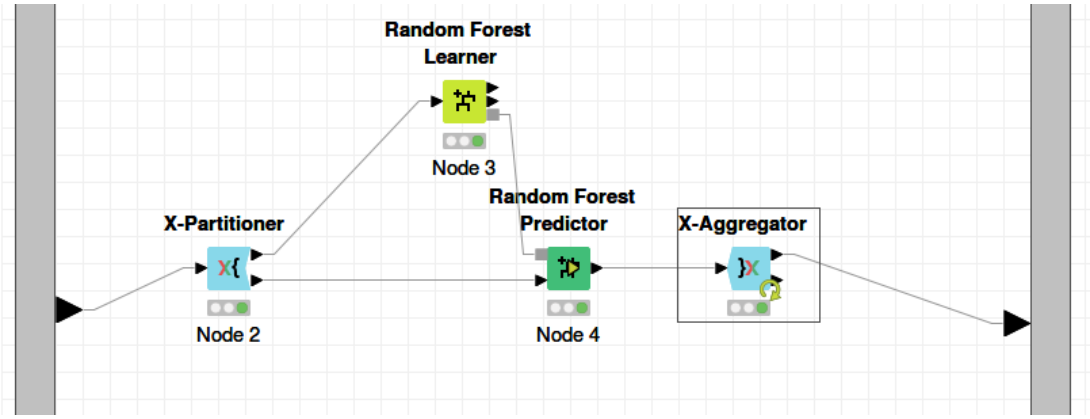


Figura 2: Flujo de trabajo: random forest

En el nodo X-partitioner se han configurado las opciones del cross validation. En la siguiente imagen se muestran las opciones elegidas.

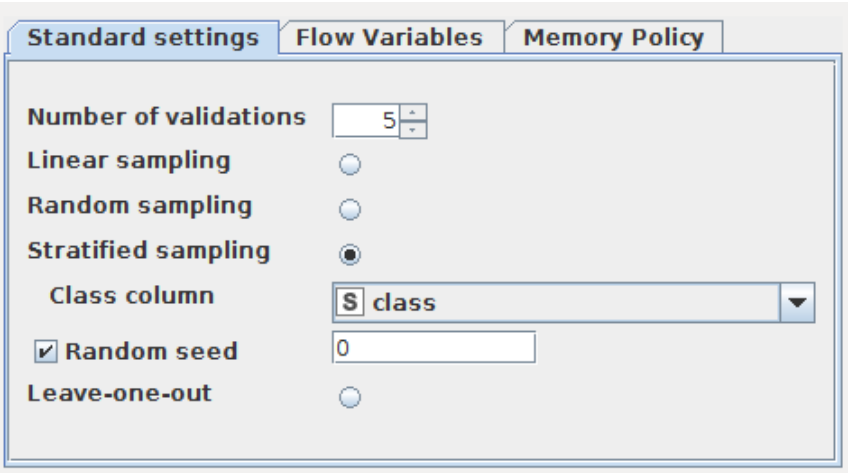


Figura 3: Configuración de la validación cruzada

El numero de particiones para la validación cruzada, como podemos ver, se ha fijado a 5. También se ha elegido la opción stratified sampling, con la cual conseguimos que cada partición realizada mantenga la proporción de clases del conjunto original. Por último se ha fijado el valor de la semilla para que los resultados obtenidos no varien en cada ejecución si los parámetros son los mismos.

2.1. Regresión logística

En la siguiente imagen podemos ver el flujo de trabajo en la validación cruzada para la regresión logística.

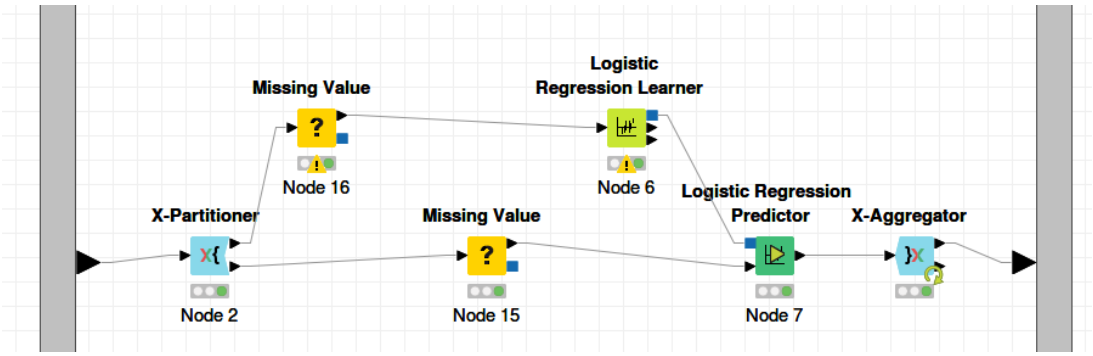


Figura 4: Flujo de trabajo: regresión logística

Primero podemos ver el nodo X-partitioner, en el cual como ya hemos explicado se eligen las opciones para hacer una validación cruzada de 5 particiones y con muestreo estratificado.

Tanto en el conjunto de train como el de test podemos ver que hemos utilizado el nodo missing values. Para el caso del train el nodo missing values elimina las filas con valores perdidos para que solo entrenemos con datos que estén completos. Para el caso del test lo que se ha hecho es que en los valores numéricos perdidos, se ha puesto como valor la media de los valores de esa variable.

Tras el entrenamiento se pasan los datos al metanodo Resultados - Logistic Regression, cuyo flujo de trabajo se muestra en la imagen inferior.

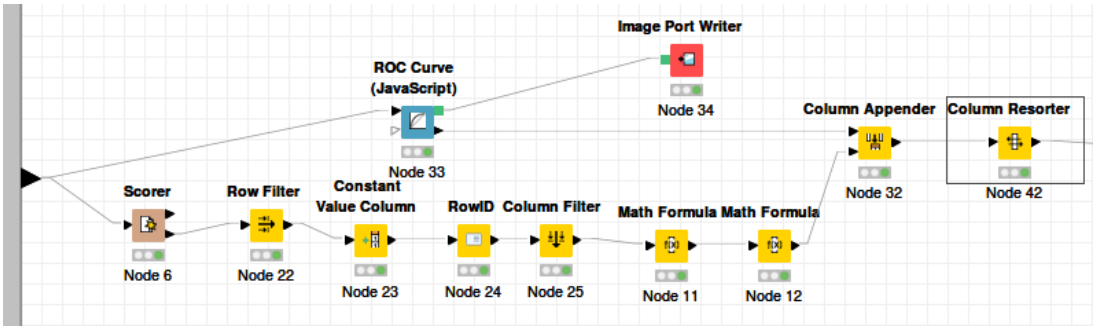


Figura 5: Metanodo Resultados - Logistic Regression

Los datos obtenido en la validación cruzada se pasan al nodo scorer el cual nos proporciona la matriz de confusión y datos estadísticos. Los nodos que siguen a scorer son utilizados para filtrar información y renombrar columnas, en concreto lo que hacemos es quedarnos solo con la fila de popular de scorer, añadir una columna con el nombre del algoritmo y posteriormente cambiar el identificador de la fila por el nombre del algoritmo. A continuación se ejecutan los nodos Math Formula con los cuales calculamos el valor del

accuracy y el de la media geométrica (G-mean).

El accuracy se calcula mediante la fórmula: $\frac{TP+TN}{TP+TN+FP+FN}$.

El valor de G-mean se calcula con la fórmula: $\sqrt{TPR * TNR}$.

El otro nodo que podemos ver es el del cálculo de la curva ROC, el cual nos devuelve una imagen con la curva ROC y el valor de AUC el cual nos da el valor del área que hay bajo la curva ROC. La imagen resultante de la curva ROC es la siguiente:

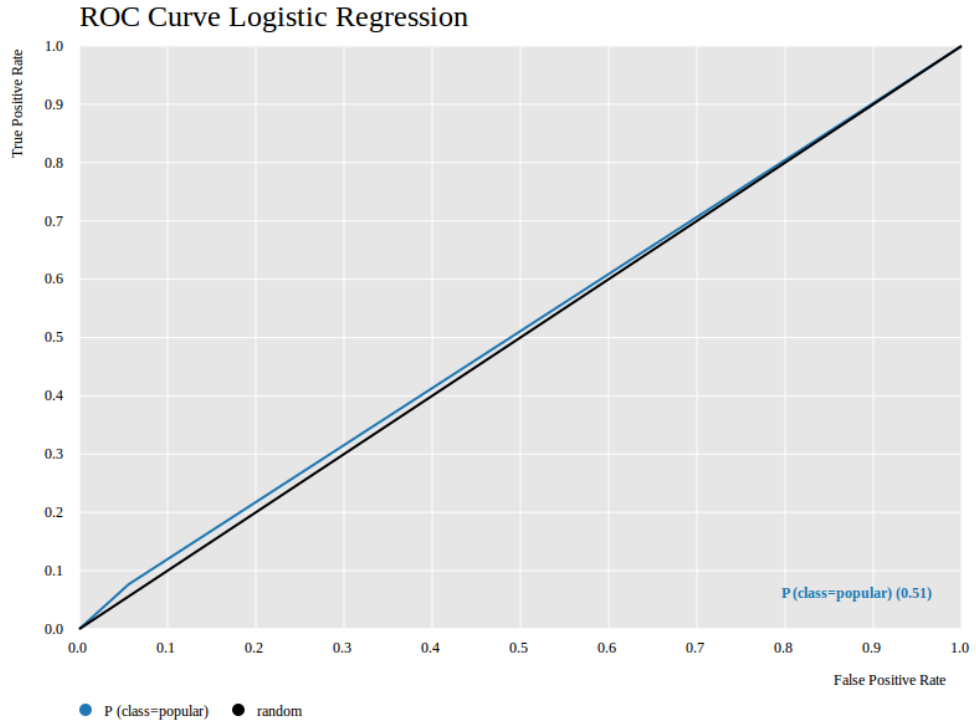


Figura 6: ROC curve Logistic Regression

Finalmente se unen las columnas con Column Appender y ordenandolas obtenemos la siguiente tabla.

ID	Accuracy	TP	FP	TN	FN	TPR	TNR	AUC	PPV	F1-score	G-mean
Logistic Reg	0,749	685	1721	28997	8241	0,077	0,944	0,510	0,285	0,121	0,269

Estos resultados se corresponde con diferentes medidas realizadas sobre la validación cruzada de regresión logística.

En los siguientes apartados no se describirá de nuevo la parte correspondiente al metanodo para obtener los resultados, ya que es igual para todos los algoritmos. Lo que haremos será mostrar directamente los resultados obtenidos.

2.2. Gradient boosted

En la imagen inferior podemos ver el flujo de trabajo del algoritmo gradient boosted.

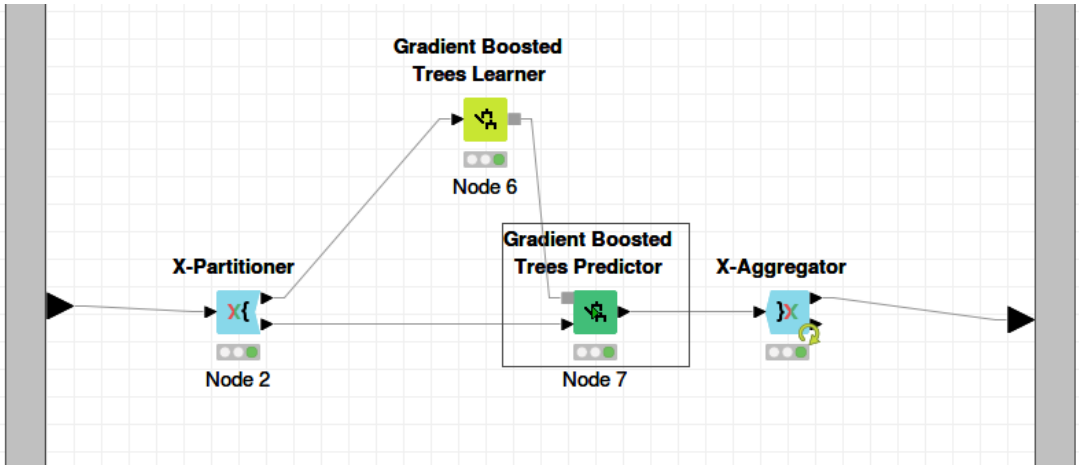


Figura 7: Flujo de trabajo: gradient boosted

Los resultados obtenidos han sido los siguientes:

ID	Accuracy	TP	FP	TN	FN	TPR	TNR	AUC	PPV	F1-score	G-mean
G.Boosted	0,777	835	735	29983	8091	0,094	0,976	0,715	0,532	0,159	0,302

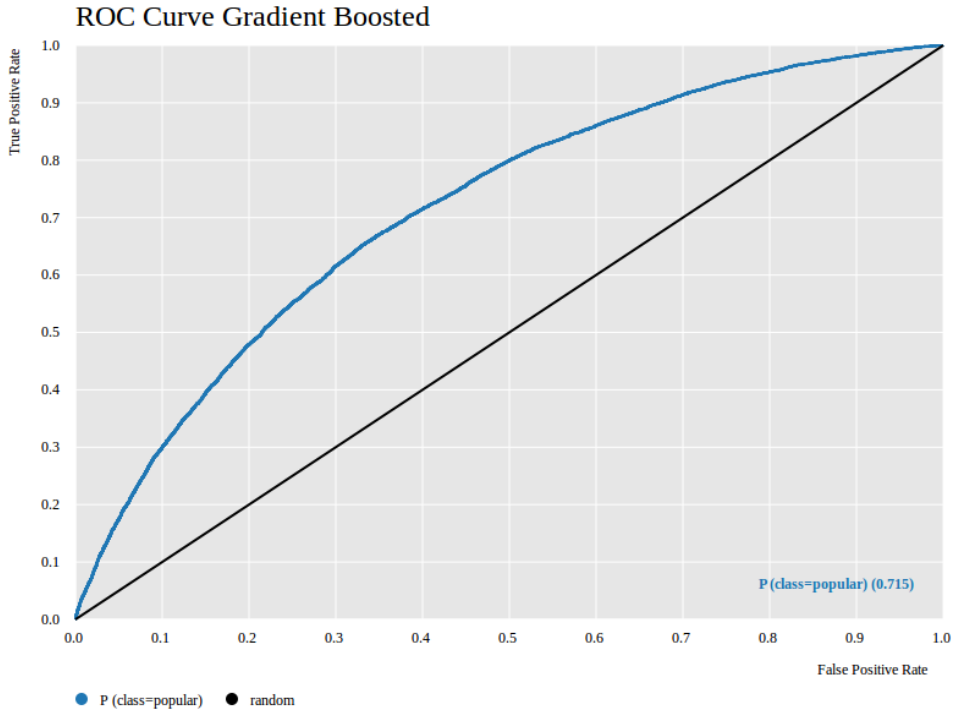


Figura 8: ROC curve Gradient Boosted

2.3. Random Forest

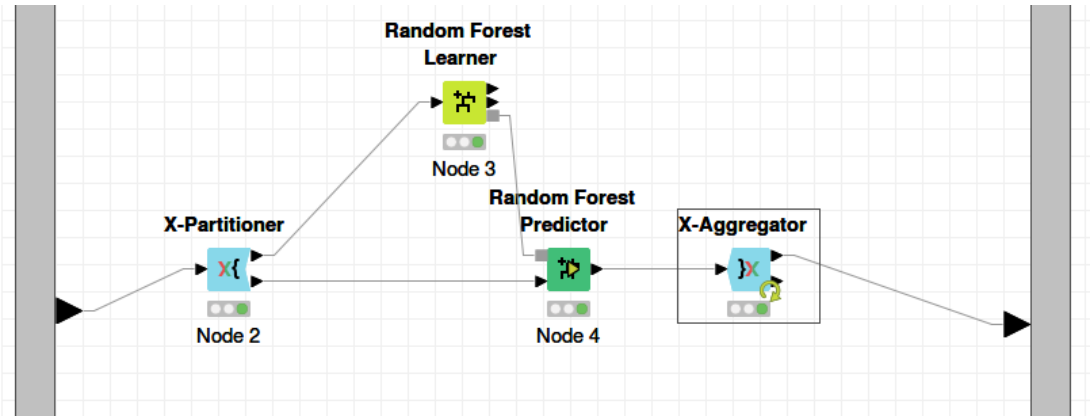


Figura 9: Flujo de trabajo: random forest

Los resultados obtenidos han sido los siguientes:

ID	Accuracy	TP	FP	TN	FN	TPR	TNR	AUC	PPV	F1-score	G-mean
R. Forest	0,777	617	548	30170	8309	0,069	0,982	0,703	0,530	0,122	0,261

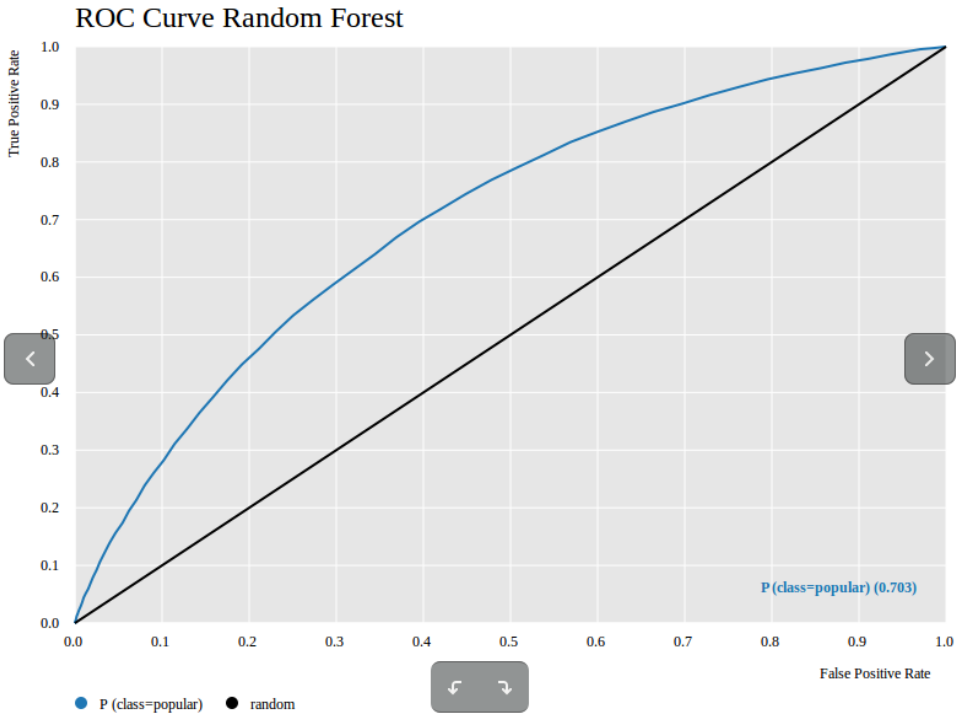


Figura 10: ROC curve Random Forest

2.4. Naive Bayes

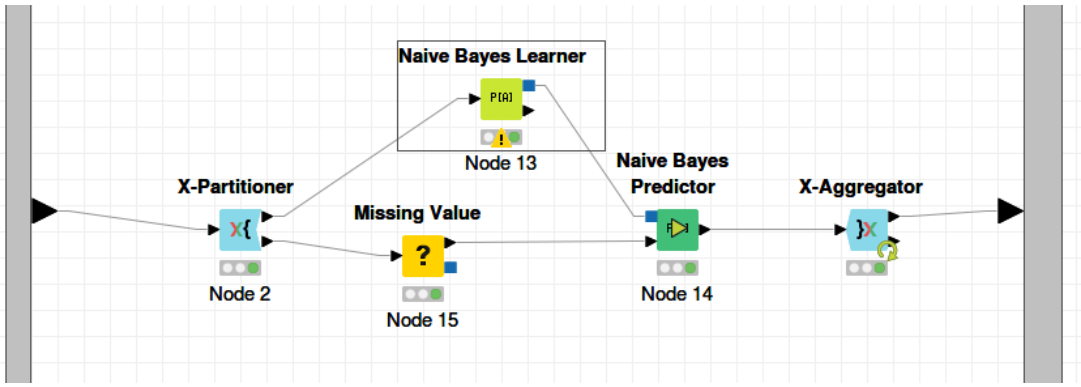


Figura 11: Flujo de trabajo: naive bayes

Los resultados obtenidos han sido los siguientes:

ID	Accuracy	TP	FP	TN	FN	TPR	TNR	AUC	PPV	F1-score	G-mean
N. Bayes	0,565	6279	14601	16117	2647	0,703	0,525	0,646	0,301	0,421	0,608

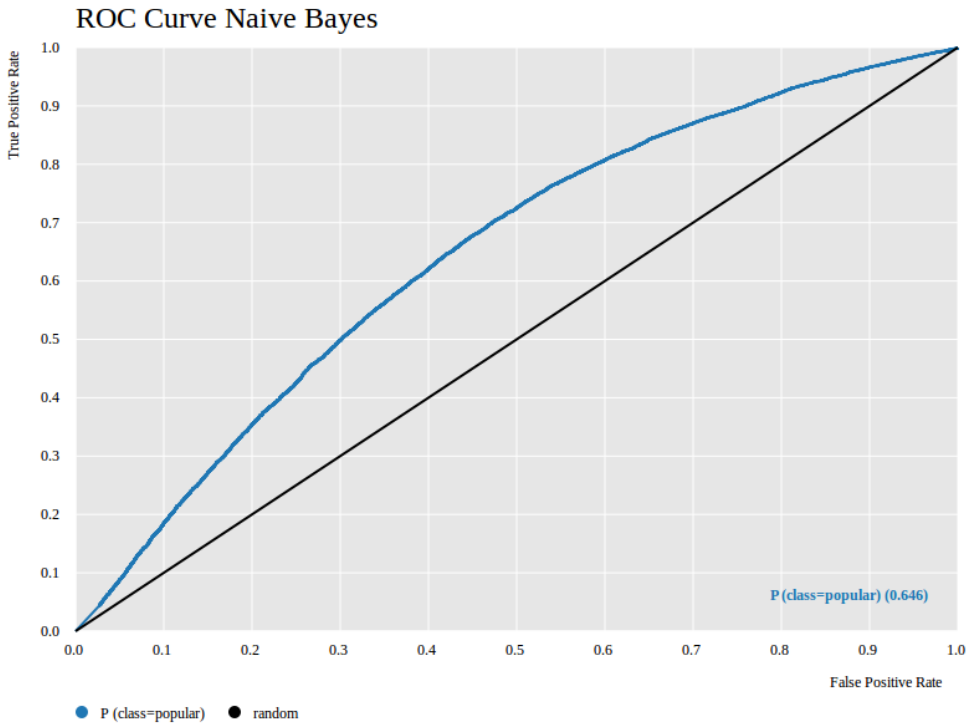


Figura 12: ROC curve Naive Bayes

2.5. KNN

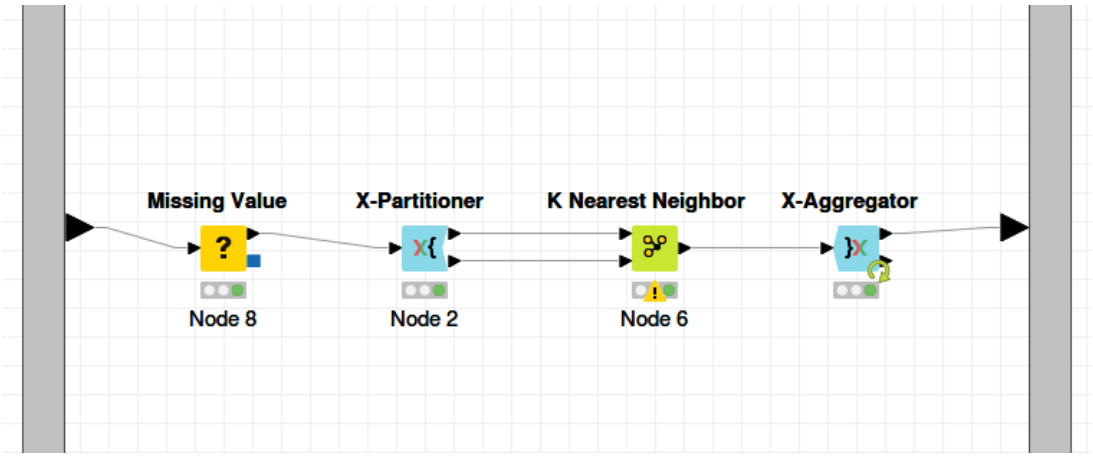


Figura 13: Flujo de trabajo: KNN

Los resultados obtenidos han sido los siguientes:

ID	Accuracy	TP	FP	TN	FN	TPR	TNR	AUC	PPV	F1-score	G-mean
KNN	0,725	1701	3685	27033	7225	0,191	0,880	0,570	0,316	0,238	0,410

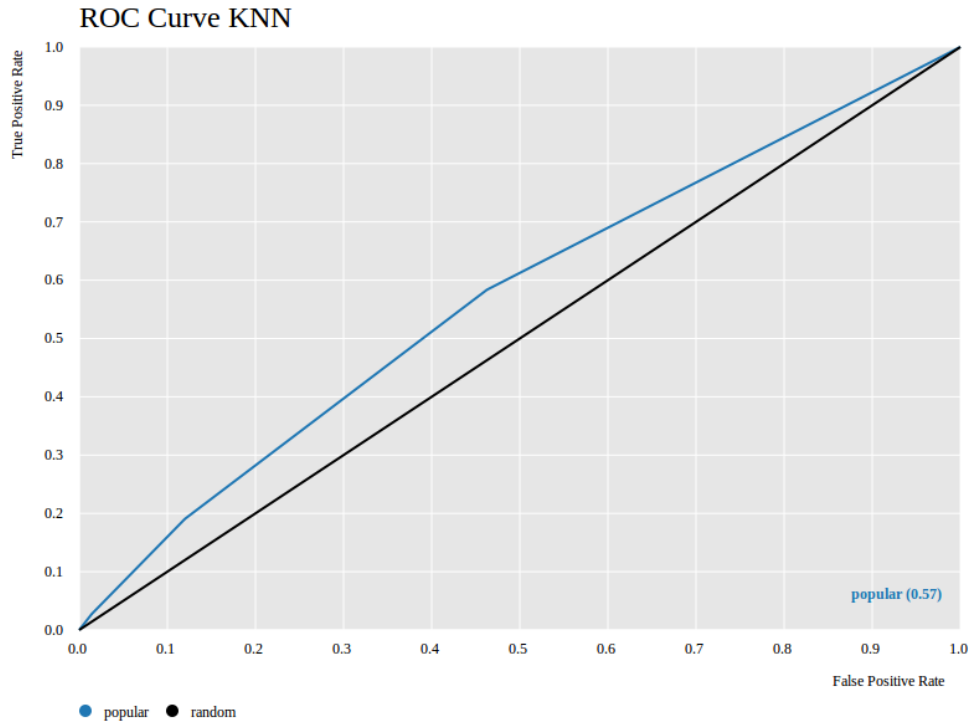


Figura 14: ROC curve KNN

2.6. Red neuronal

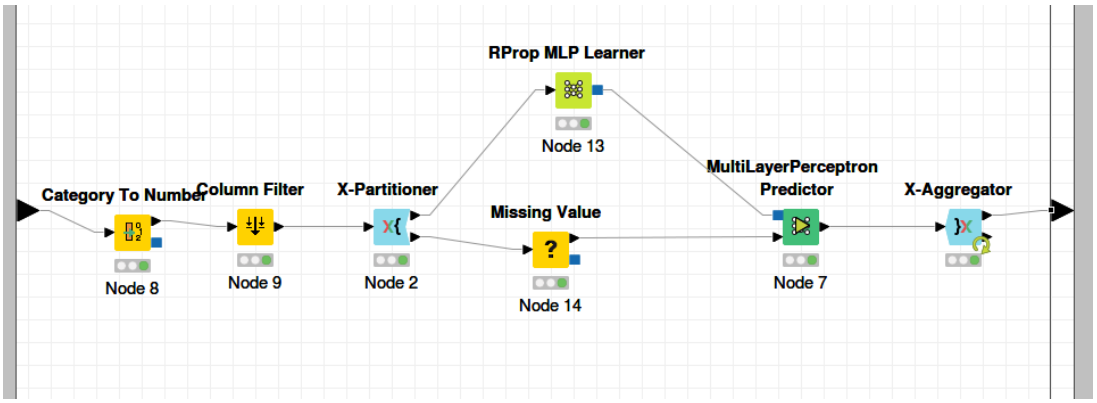


Figura 15: Flujo de trabajo: red neuronal

Los resultados obtenidos han sido los siguientes:

ID	Accuracy	TP	FP	TN	FN	TPR	TNR	AUC	PPV	F1-score	G-mean
Neural Net.	0,775	1	3	30715	8925	0,000	1,000	0,502	0,250	0,000	0,011

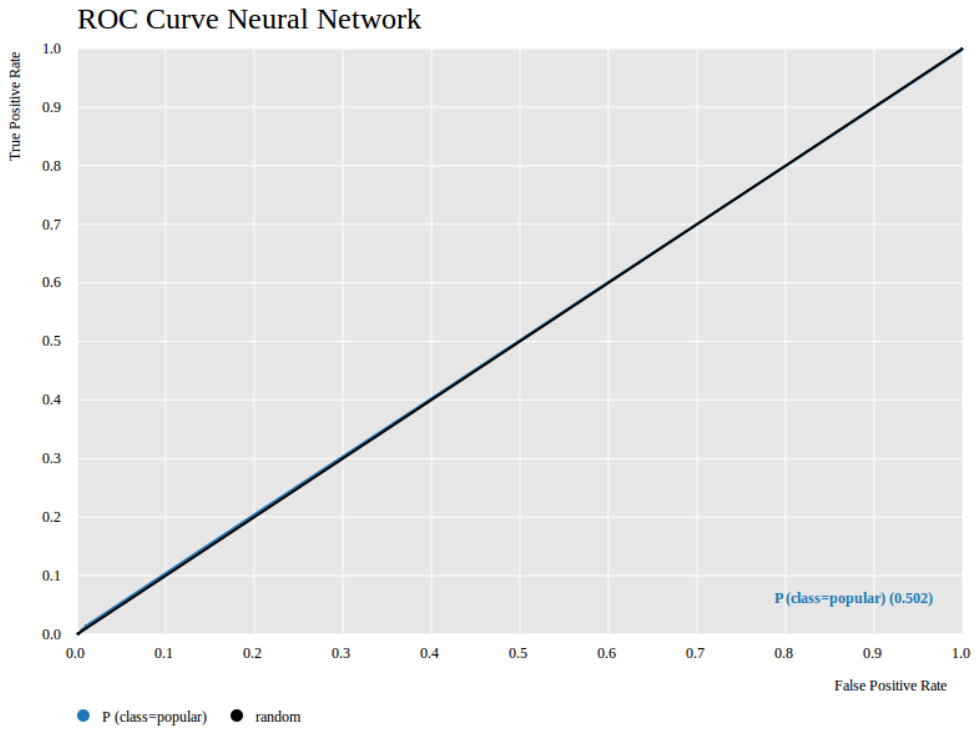
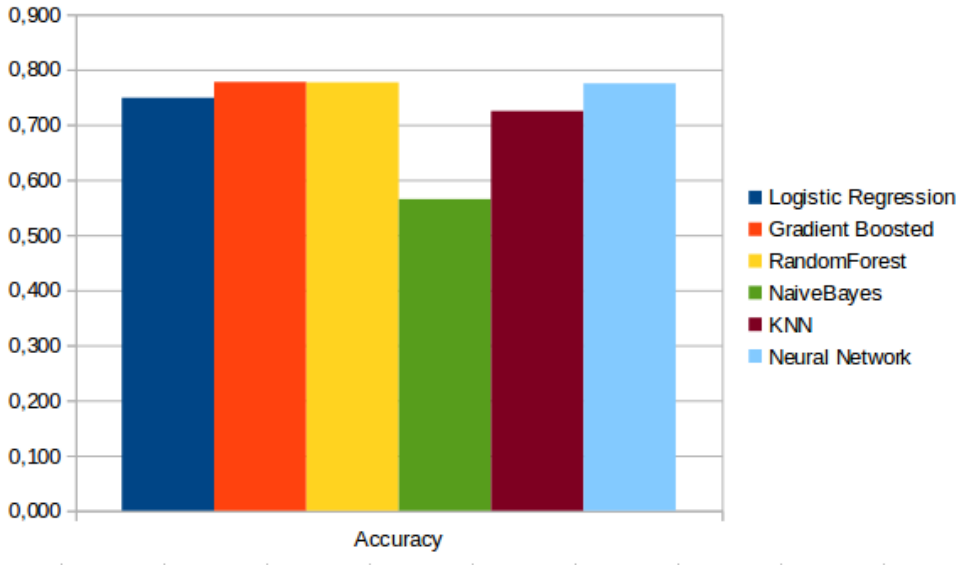


Figura 16: ROC curve Neural Network

3. Análisis de resultados

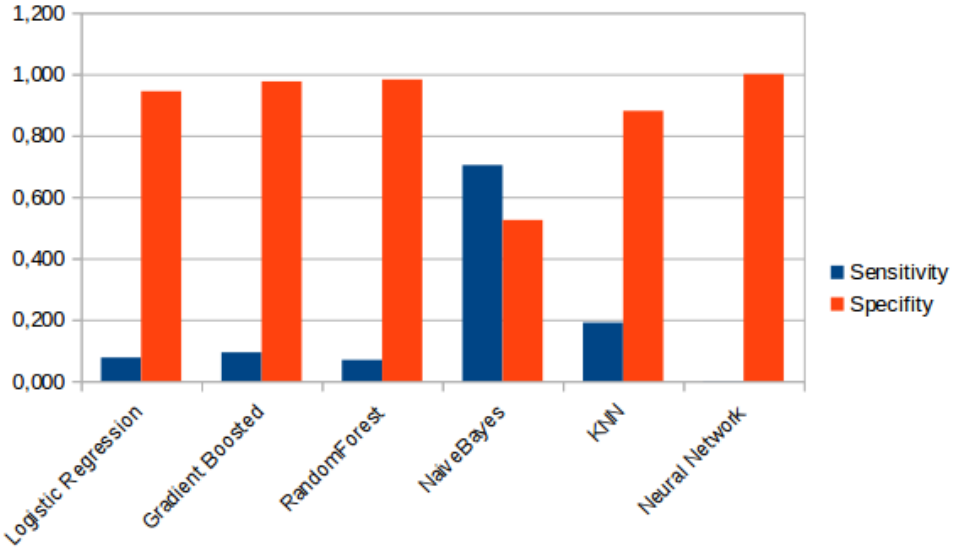
ID	Accuracy	TP	FP	TN	FN	TPR	TNR	AUC	PPV	F1-score	G-mean
Logistic Reg	0,749	685	1721	28997	8241	0,077	0,944	0,510	0,285	0,121	0,269
G. Boosted	0,777	835	735	29983	8091	0,094	0,976	0,715	0,532	0,159	0,302
R. Forest	0,777	617	548	30170	8309	0,069	0,982	0,703	0,530	0,122	0,261
N. Bayes	0,565	6279	14601	16117	2647	0,703	0,525	0,646	0,301	0,421	0,608
KNN	0,725	1701	3685	27033	7225	0,191	0,880	0,570	0,316	0,238	0,410
Neural Net.	0,775	1	3	30715	8925	0,000	1,000	0,502	0,250	0,000	0,011

- **Accuracy:** el valor del accuracy es calculado como el número de predicciones correctas entre el número total de predicciones. Para facilitar la comparación de esta medida se incluye un gráfico de barras.



Podemos ver que el peor es Naive Bayes, lo cual puede explicarse a que este algoritmo supone que las características son independientes y en nuestro caso vemos que bastante correlación entre algunas variables. El siguiente en la cola es el KNN, el cual es posible que mejore los resultados tras normalizar los datos, ya que al ser una algoritmo basado en distancias aquellas características que se muevan en rangos mayores podrían predominar y por eso es importante normalizar los datos, cosa que veremos en el apartado 5. El que le sigue es la regresión logística, este algoritmo nos proporciona un modelo lineal el cual es muy simple, este algoritmo funciona mucho mejor cuando los datos son linealmente separables. y por último los 3 mejores son random forest, gradient boost y neural network.

- **TPR y TNR:** estos se obtienen dividiendo el número de predicciones positivas correctas (respectivamente negativas) entre el número total de positivos (respectivamente negativos). EN la siguiente gráfica se muestra la comparación de los valores de TPR y TNR de los distintos algoritmos.



Lo primero que puede llamar la atención es el valor tan bajo obtenido en el TPR. Esto es debido a que nuestro conjunto de datos tiene la distribución de etiquetas desbalanceada donde solo el 22.5 % son populares y e resto son no populares. Este desbalanceo hace que haya un valor del TNR alto, pero sin embargo el valor del TPR sea bastante bajo. Esto nos sugiere que posteriormente probemos a utilizar un nodo de knime llamado equal size sampling, para hacer un muestreo de forma que se entrene el clasificador con una muestra balanceada. El único que contrasta con los resultados es el clasificador Naive Bayes.

- **AUC:** este valor mide el área bajo la curva ROC. Así que para comparar este valor veamos también las curvas ROC de cada algoritmo.

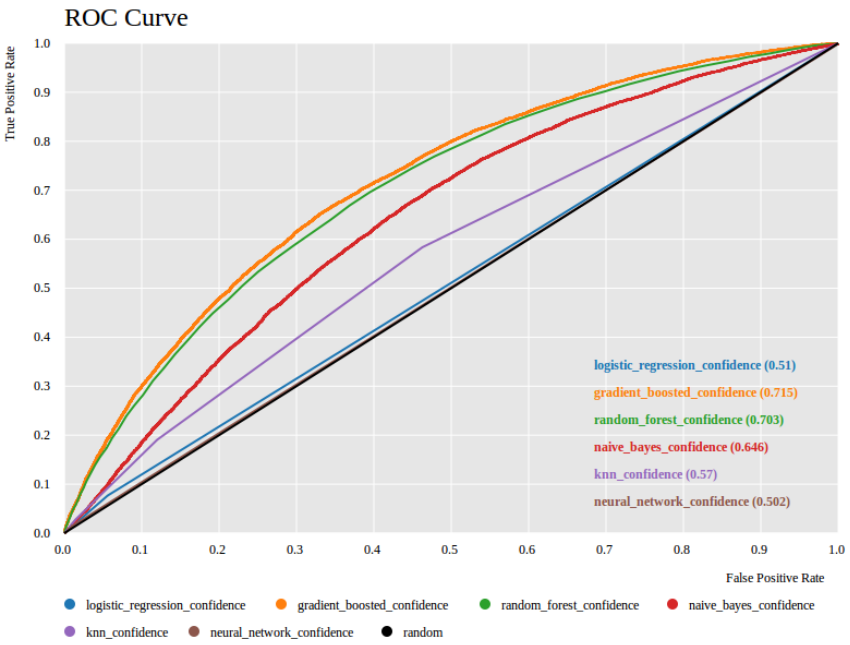
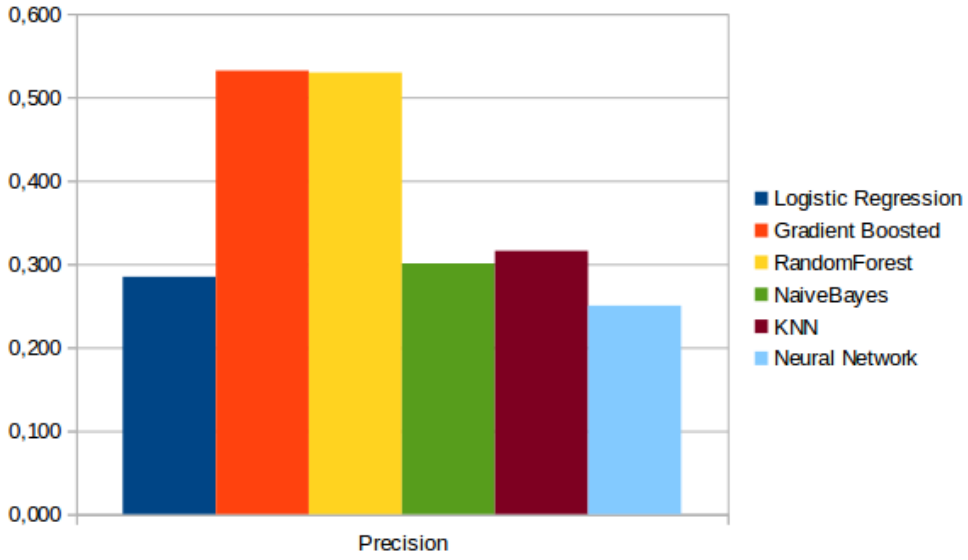


Figura 17: Comparación ROC

En esta medida podemos ver como por ejemplo el algoritmo de red neuronal obtiene un resultado muy malo y mucho peor del que cabría esperarse, esto se puede deber a que el algoritmo neural network necesita que los datos estén normalizados para obtener bueno resultados, la mejora producida tras normalizar los datos la veremos como ya se ha comentado anteriormente el el apartado 5. Random forest y gradient boosted vuelven a ser los mejores en esta medida también. Random forest y gradient boosted al ser algoritmos basados en árboles no necesitan normalizar los datos, es por esto que ne esta comparación se encuentran en ventaja con respecto a otros como KNN y neural network.

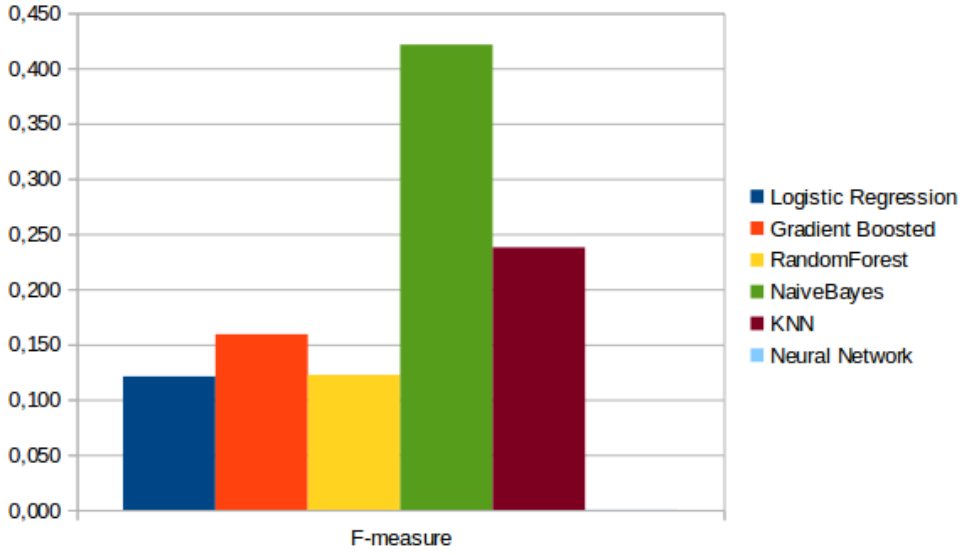
- **PPV:** esta medida se calcula como el número de predicciones positivas correctas entre el número total de predicciones positivas realizadas.

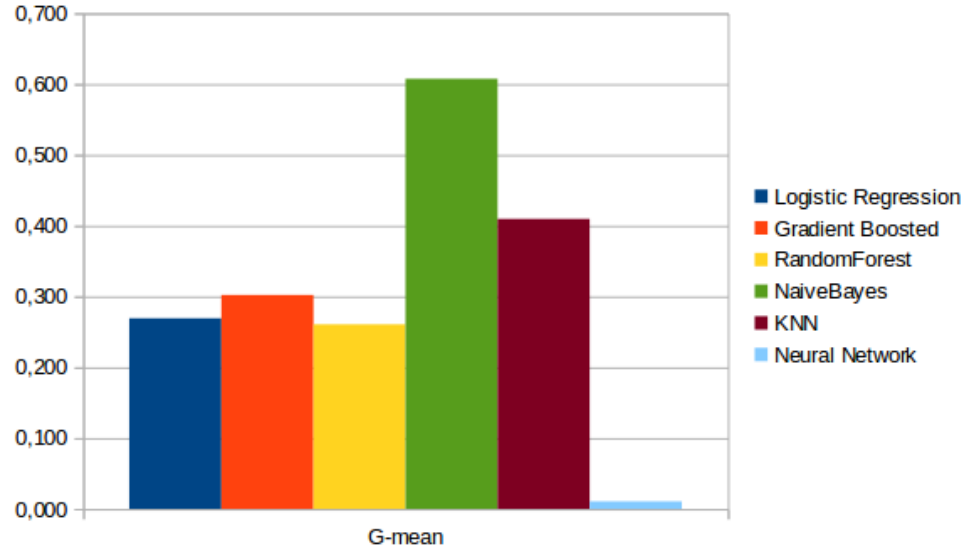


De nuevo los que mejores resultados obtienen son random forest y gradient boosted. Esperemos ver que se igualan algo más los resultados después de hacer el preprocesado de datos en el apartado 5 y algoritmos como neural network mejoren significativamente.

- **F1-score:** el cual se calcula como la media armónica de PPV y TPR.
- **G-mean:** este valor es calculado como la media geométrica de TPR y TNR.

En estas dos últimas medidas vamos a ver que e que obtenía peores resultados en las demás métricas es el que va a sobresalir aquí. Este algoritmo se trata de Naive Bayes y es que este algoritmo es el único que obtuvo unos resultados balanceados en las medidas de TPR y TNR.

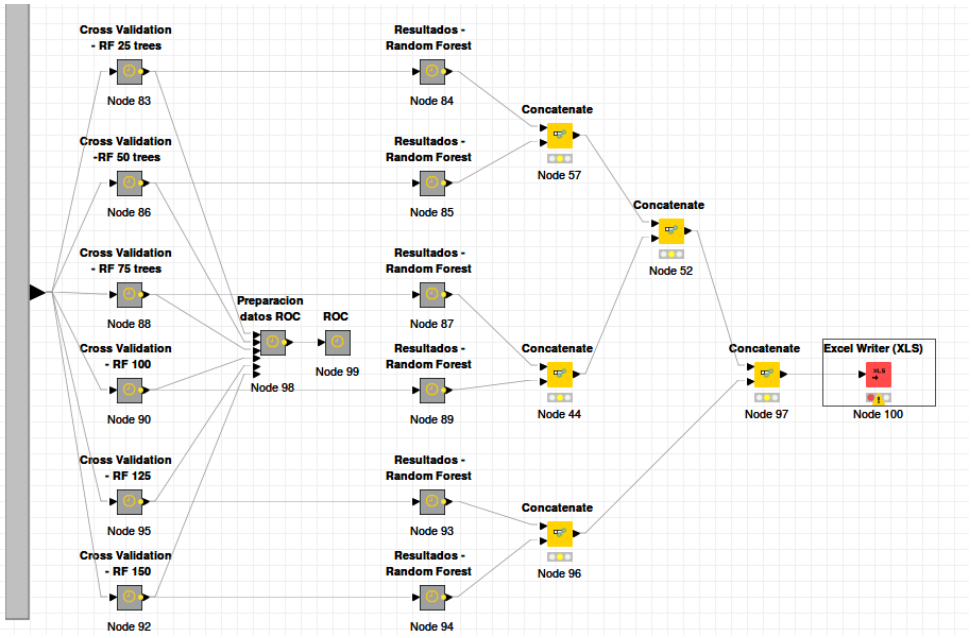

0,450
0,400
0,350
0,300
0,250
0,200
0,150
0,100
0,050
0,000
F-measure
Logistic Regression
Gradient Boosted
RandomForest
NaiveBayes
KNN
Neural Network


0,700
0,600
0,500
0,400
0,300
0,200
0,100
0,000
G-mean
Logistic Regression
Gradient Boosted
RandomForest
NaiveBayes
KNN
Neural Network

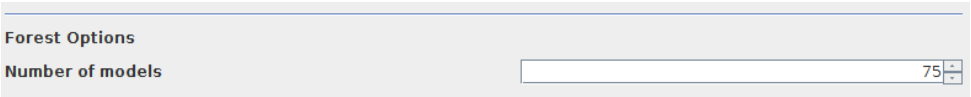
4. Configuración de algoritmos

4.1. Random Forest

Se ha probado a modificar el parámetro que indica el número de árboles que va a utilizar el clasificador. Se ha variado desde 25 árboles hasta 150, con un incremento de 25. En la siguiente imagen se muestra el flujo de trabajo utilizado para obtener la comparación:



A continuación se muestra un ejemplo del parámetro modificado en el random forest:



La siguiente tabla recoge lo valores obtenidos tras la ejecución de todos los random forest con diferentes valores en el parámetro .

ID	Accuracy	TP	FP	TN	FN	TPR	TNR	AUC	PPV	F1-score	G-mean
RF 25	0,772	1007	1129	29589	7919	0,113	0,963	0,678	0,471	0,182	0,330
RF 50	0,775	675	656	30062	8251	0,076	0,979	0,694	0,507	0,132	0,272
RF 75	0,776	711	653	30065	8215	0,080	0,979	0,700	0,521	0,138	0,279
RF 100	0,777	617	548	30170	8309	0,069	0,982	0,703	0,530	0,122	0,261
RF 125	0,777	644	554	30164	8282	0,072	0,982	0,704	0,538	0,127	0,266
RF 150	0,777	586	502	30216	8340	0,066	0,984	0,705	0,539	0,117	0,254

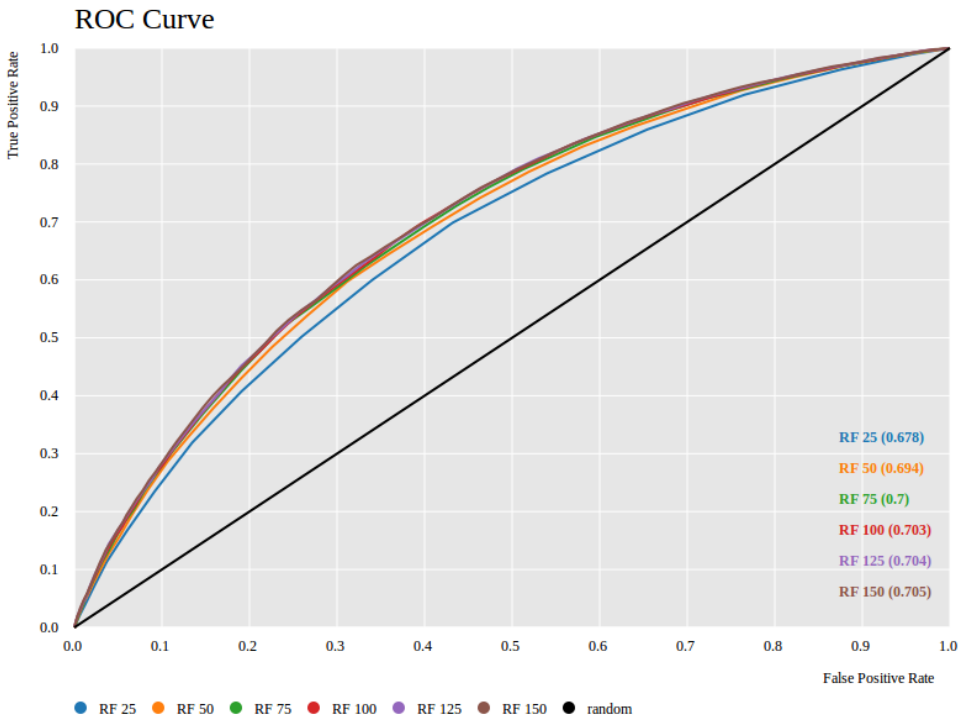


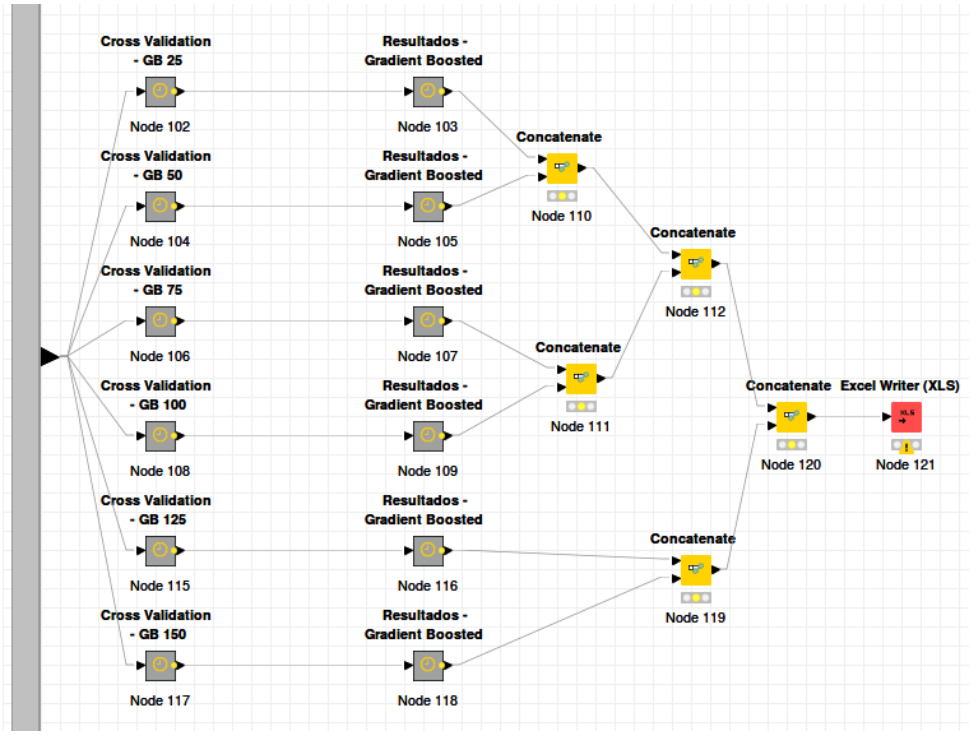
Figura 18: Comparación curva ROC RF

Las mejoras al aumentar el número de árboles son mínimas en el valor de accuracy. Un poco más notable en el valor de AUC y PPV, aunque a partir de los 75 árboles la mejora es insignificante. El TPR empeoran con forme aumenta el número de árboles al igual que le ocurre a los valores de las medidas F1-score y G-mean.

Si tuviera que coger un modelo cogería o bien el de 50 árboles o bien el de 75, ya que las mejoras a partir de ahí son mínimas y en cambio la complejidad de aumenta ya que se aumenta mucho el número de árboles.

4.2. Gradient Boosted

Al igual que en Random Forest se ha probado a modificar el parámetro que indica el número de árboles que va a utilizar el clasificador. Se ha variado desde 25 árboles hasta 150, con un incremento de 25. En la siguiente imagen se muestra el flujo de trabajo utilizado para obtener la comparación:



La siguiente tabla reoge los resultados obtenidos:

ID	Accuracy	TP	FP	TN	FN	TPR	TNR	AUC	PPV	F1-score	G-mean
GB 25	0,777	293	210	30508	8633	0,033	0,993	0,703	0,583	0,062	0,181
GB 50	0,777	588	484	30234	8338	0,066	0,984	0,712	0,549	0,118	0,255
GB 75	0,778	737	628	30090	8189	0,083	0,980	0,715	0,540	0,143	0,284
GB 100	0,777	835	735	29983	8091	0,094	0,976	0,715	0,532	0,159	0,302
GB 125	0,777	904	816	29902	8022	0,101	0,973	0,715	0,526	0,170	0,314
GB 150	0,778	984	853	29865	7942	0,110	0,972	0,716	0,536	0,183	0,327

Con respecto al valor de accuracy no se ven ninguna mejora al aumenta el número de árboles. En el valor de AUC se aprecian ligeras mejoras, mientras que en el PPV empeora ligeramente. Las medidas de F1-score y G-mean mejoran ambas con forme se aumenta el número de árboles, lo cual se debe a una leve mejora en el desbalance que hay entre el TPR y el TNR. Si tengo que elegir me quedaría con el de 50 árboles ya que apenas se mejora el resultado con más árboles y de esta forma no aumentamos la complejidad del modelo excesivamente.

5. Procesado de datos

5.1. Red neuronal

Vamos a volver a probar la red neuronal pero ahora vamos a realizar algunas modificaciones, añadiendo un preprocesado de los datos. El flujo de trabajo que teníamos en el apartado 2 era el siguiente:

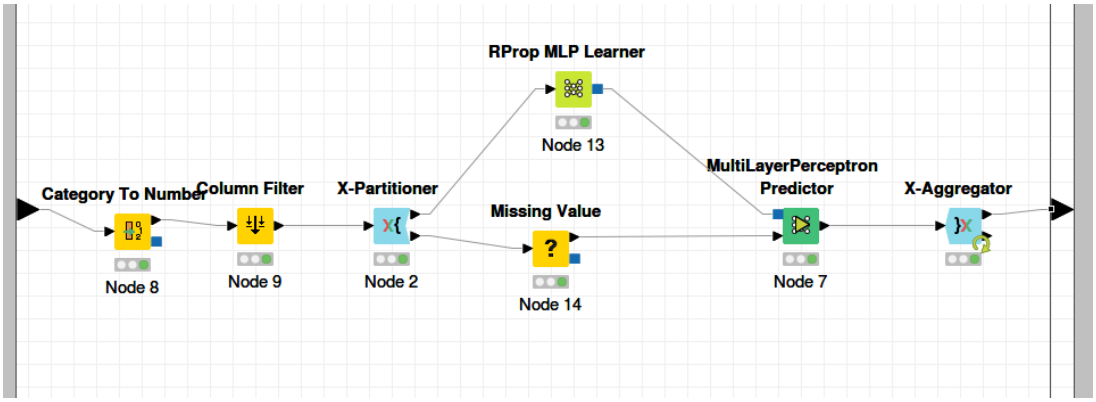


Figura 19: Flujo de trabajo: red neuronal

Comentar que en la red neuronal ya se realizo un procesamiento de los datos. En concreto se realizo una transformación de variables categóricas a numéricas, ya que este algoritmo no puede trabajar con variables categóricas. Para ello se ha usado el nodo category to number.

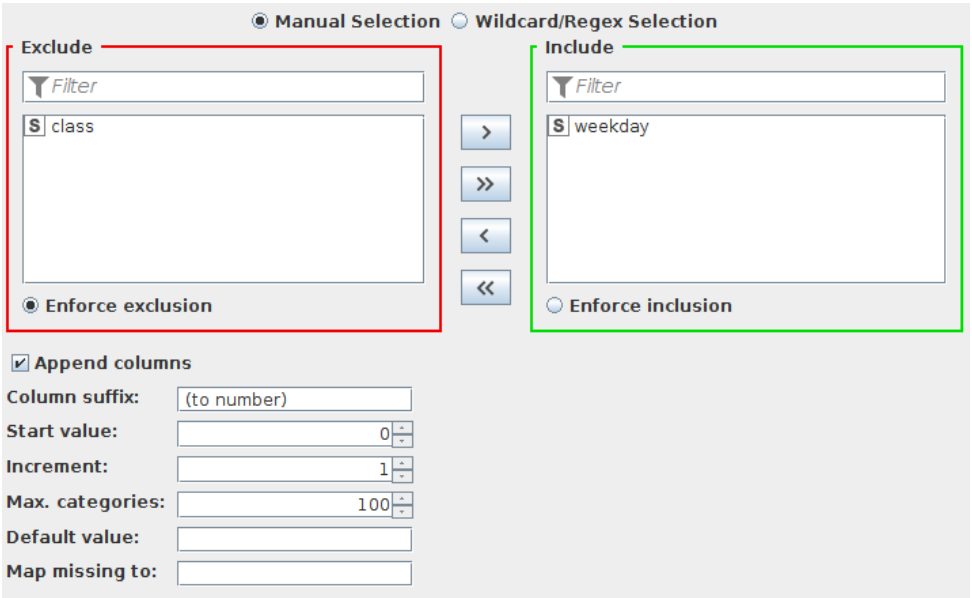


Figura 20: Category To Number

Como podemos ver la única característica categórica es la correspondiente al día de a semana.

Después se ha realizado un filtrado de columnas eliminando las columnas correspondientes a las variables categoricas, de las cuales ya disponemos su columna en versión numérica.

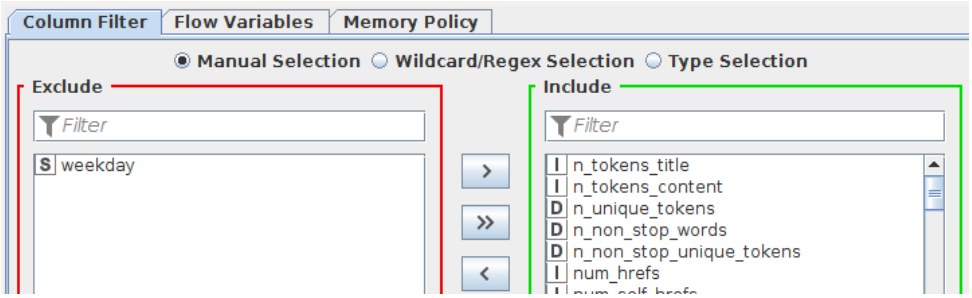


Figura 21: Column Filter

La única columna eliminada es la correspondiente con weekday.

Ademas tratamos el caso de los valores perdidos en el conjunto test tomando como valor en los valores perdidos la media de dicha característica en el conjunto test.

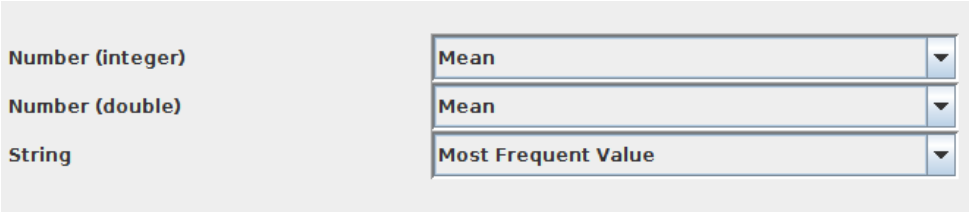


Figura 22: Missing Value

Ahora hemos añadido el nodo Normalizer el cual normaliza los datos en el intervalo [0, 1]. Esto se hace para que aquellas características cuyos valores se muevan en rangos mayores que otras características, no predominen a la hora de obtener el modelo.

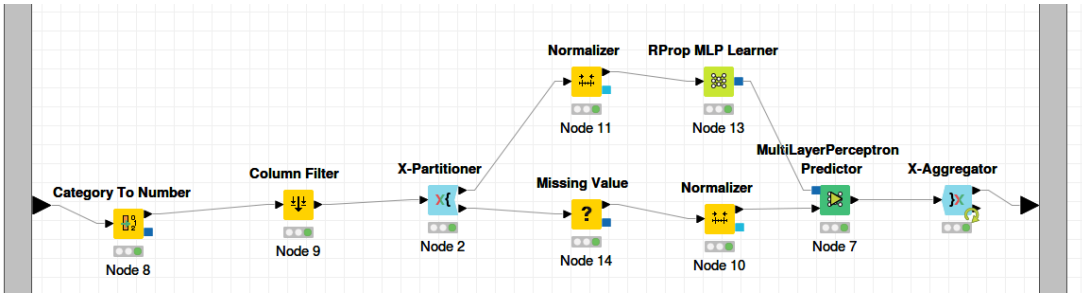
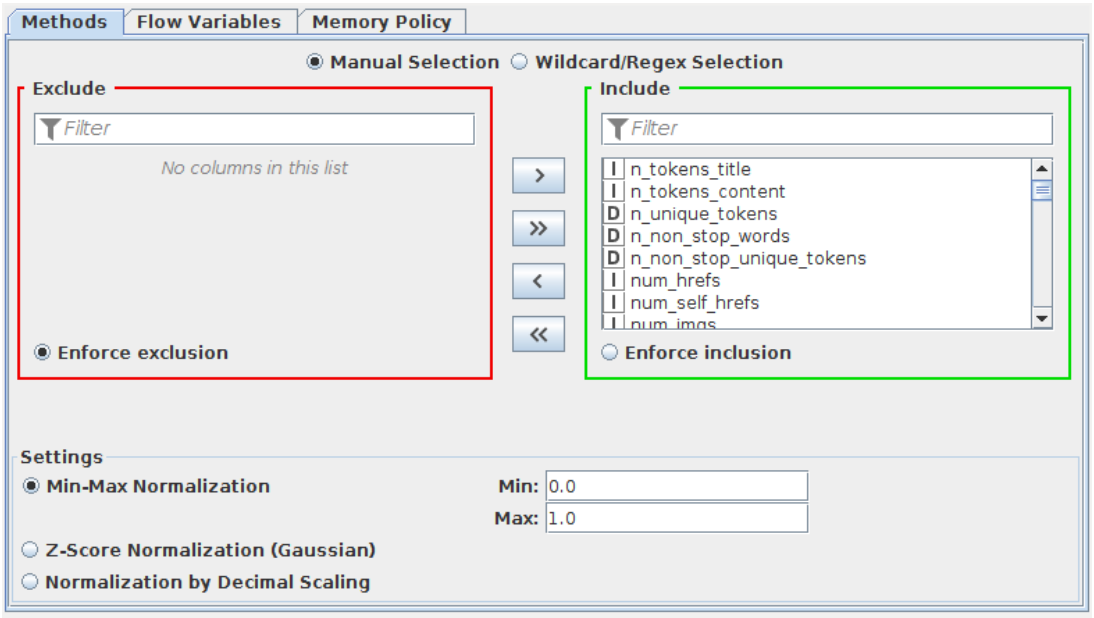


Figura 23: Flujo de trabajo: red neuronal con normalización

Podemos observar que no se han normalizado los datos antes de realizar el particionamiento. Esto es debido a que si normalizaramos todos los datos juntos antes de particionar, tendríamos que en cierta manera los datos del test estaría tomando algo de información del conjunto de entrenamiento. Es por esto que no se deberían hacer modificaciones en los valores de los datos que puedan alterar los resultados positivamente, ya que nos estaríamos engañando al obtener mejores resultados en el test de los que deberían obtenerse y nos llevaríamos posiblemente una decepción al probar nuestro clasificador en el mundo real. Es por esto que primero se realiza el particionamiento y posteriormente se normalizan los datos por separado, por un lado se normalizan los datos del conjunto de entrenamiento y por otro los datos del conjunto test.

En la siguiente imagen se muestran las opciones elegidas para el nodo Normalizer, donde se puede ver que se ha elegido la opción de normalización min-max Normalization, y 0.0 - 1.0 como valores de min-max.



La siguiente tabla muestra los resultados de las dos redes neuronales probadas. La primera fila se corresponde con los resultados obtenidos en el apartado 1 donde no se usó normalización. La segunda fila con los datos de la red neuronal en la que si se normalizan los datos.

ID	Accuracy	TP	FP	TN	FN	TPR	TNR	AUC	PPV	F1-score	G-mean
NN	0,775	1	3	30715	8925	0,000	1,000	0,502	0,250	0,000	0,011
NN normalizer	0,774	736	785	29933	8190	0,082	0,974	0,669	0,484	0,141	0,283

Para poder comparar mejor los datos se muestra el siguiente gráfico en red.

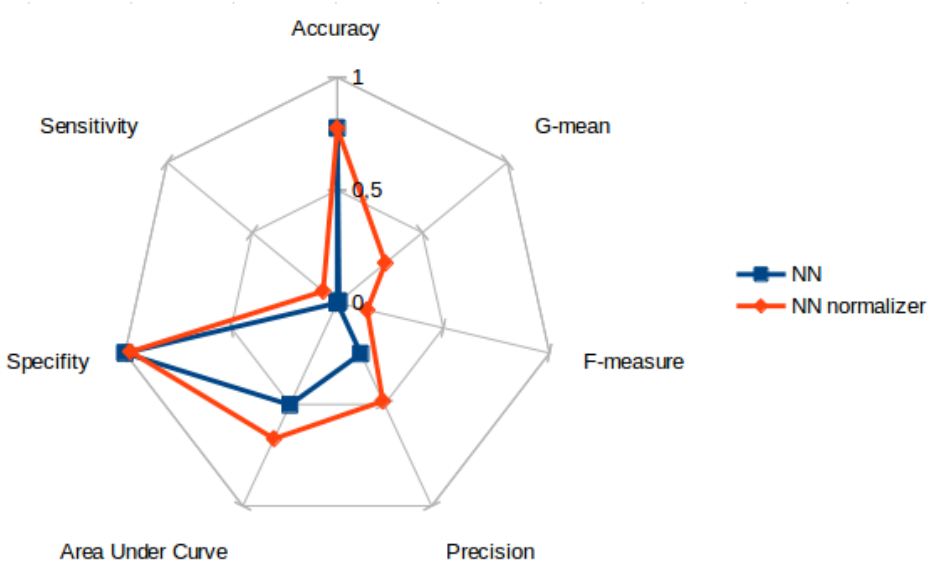


Figura 24: Comparación redes neuronales

Podemos ver que el accuracy y el specifity no ha mejorado, pero las demás medidas si que se han visto mejoradas notablemente. Con esta comparativa podemos ver la importancia de normalizar los datos en aquellos algoritmos que lo requieran para un mejor aprovechamiento de su funcionamiento.

Veamos ahora las curvas ROC de ambos:

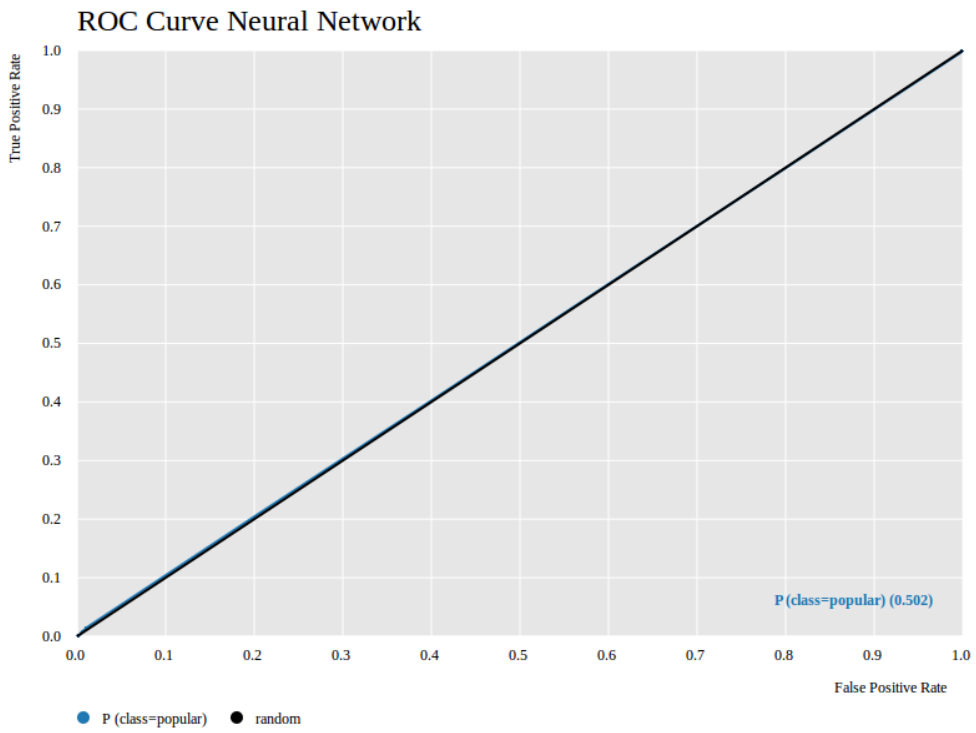


Figura 25: ROC curve Neural Network

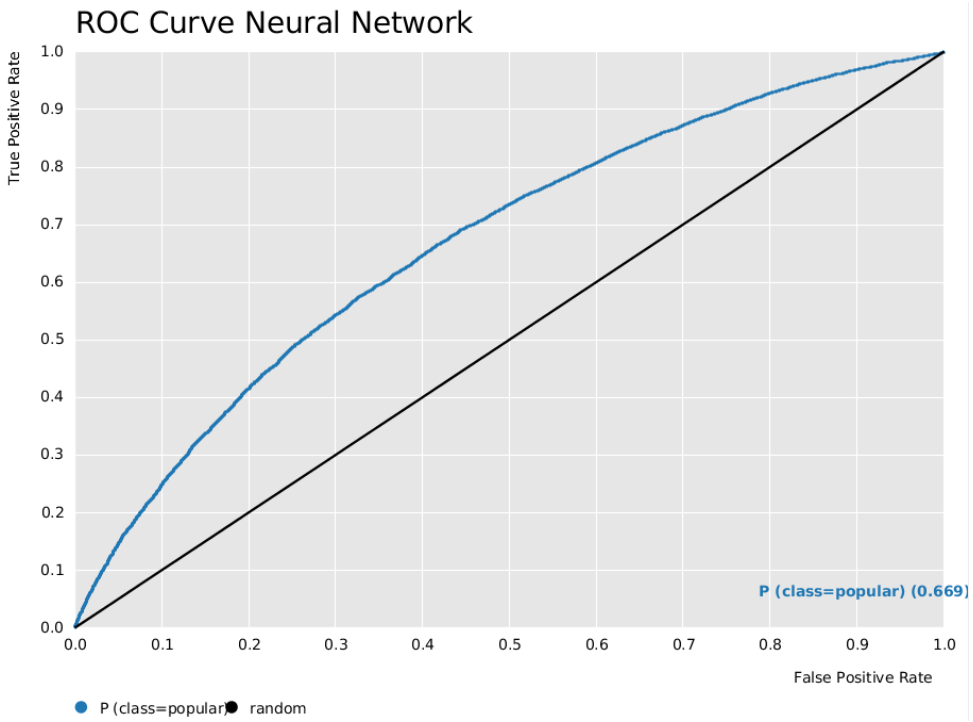


Figura 26: ROC curve Neural Network con normalización

Viendo las dos imagenes podemos apreciar la gran mejora que ha supuesto normalizar los datos en la red neuronal.

5.2. KNN

Veamos primero el flujo de trabajo de KNN que utilizamos para el apartado 2.

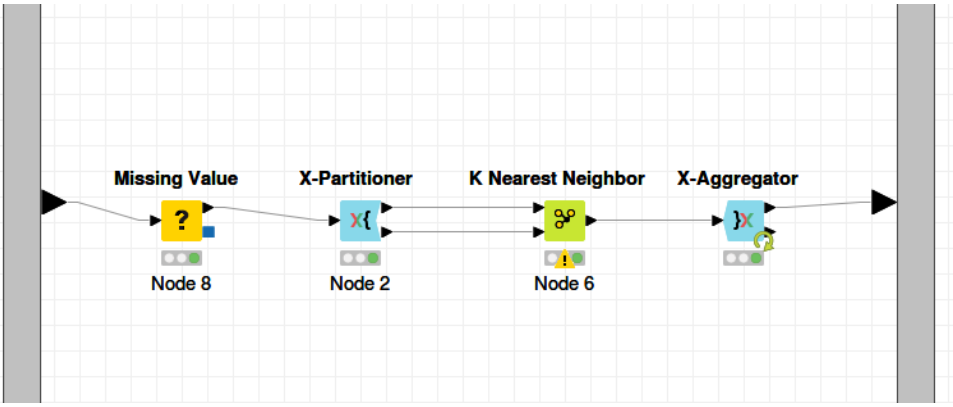


Figura 27: Flujo de trabajo de KNN

En el siguiente se muestra el modificado en el que se han normalizado los datos y además se han transformado las variables categoricas a numéricas.

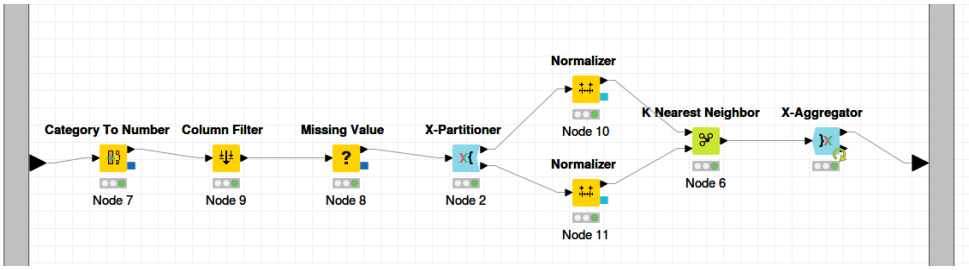


Figura 28: Flujo de trabajo de KNN con preprocesado de datos

Tabla comparativa:

ID	Accuracy	TP	FP	TN	FN	TPR	TNR	AUC	PPV	F1-score	G-mean
KNN	0,725	1701	3685	27033	7225	0,191	0,880	0,570	0,316	0,238	0,410
KNN 2	0,724	1920	3942	26776	7006	0,215	0,872	0,579	0,328	0,260	0,433

Al igual que en redes neuronales vemos que el acurracy no ha mejorado nada. De nuevo se ha vuelto a mejorar en las demás medidas pero esta vez la mejora ha sido mínima, a diferencia de la que se obtuvo en redes neuronales.

6. Interpretación de resultados

En el siguiente gráfico en red se muestran los resultados de los algoritmos utilizados. Se corresponden con los resultados usando los parámetros que mejores resultados me han dado y utilizando preprocesamiento de datos.

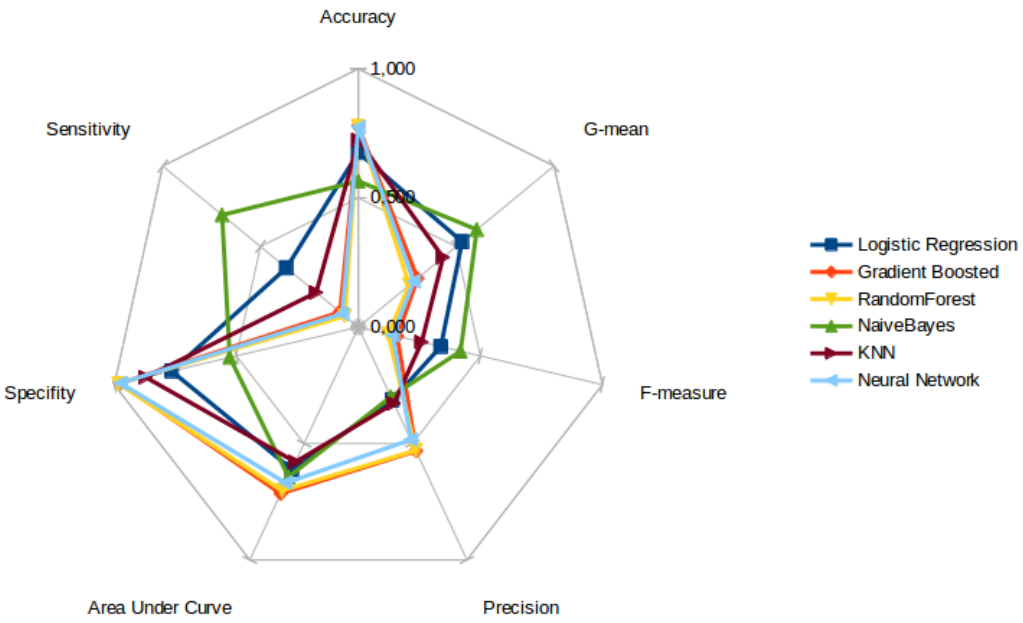


Figura 29: Comparación algoritmos

Viendo estos resultados el algoritmo final que yo cogería sería Gradient Boosted. ya que es de los que obtiene mejor puntuación en el accuracy junto con random forest y neural network, además es el que mejor puntuación tiene en el valor de AUC.

Con el siguiente flujo de trabajo podemos obtener la imagen del árbol obtenido con decission tree learner.

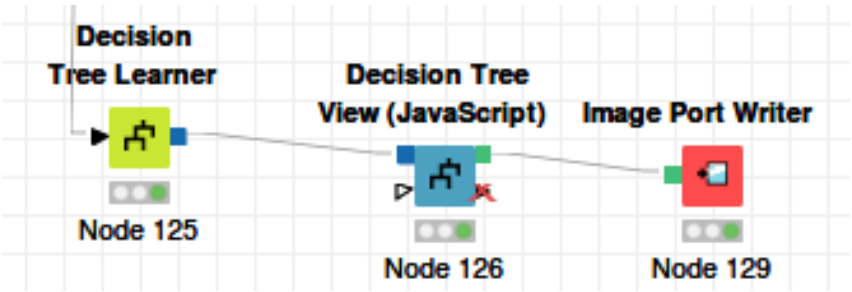


Figura 30: Obtener árbol

El árbol obtenido es el siguiente.

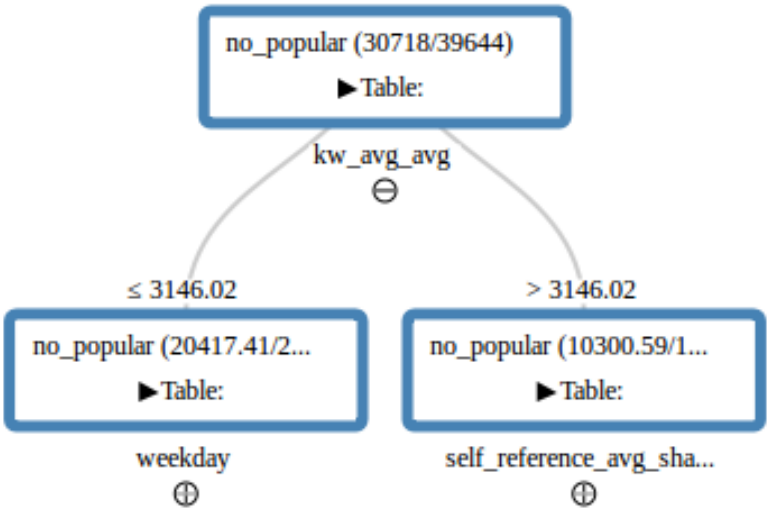


Figura 31: Tree

Como podemos ver la primera característica que utiliza para separar es la llamada kw_avg_avg, la cual se corresponde con el número medio de palabras clave. Es decir esta característica es la que considera más importante para poder determinar si un artículo se volverá popular o no.

Si observamos la siguiente imagen que se corresponde con la matriz de correlación podemos ver que efectivamente en la columna class que está a la derecha, la característica que tiene un mayor valor es kw_avg_avg.

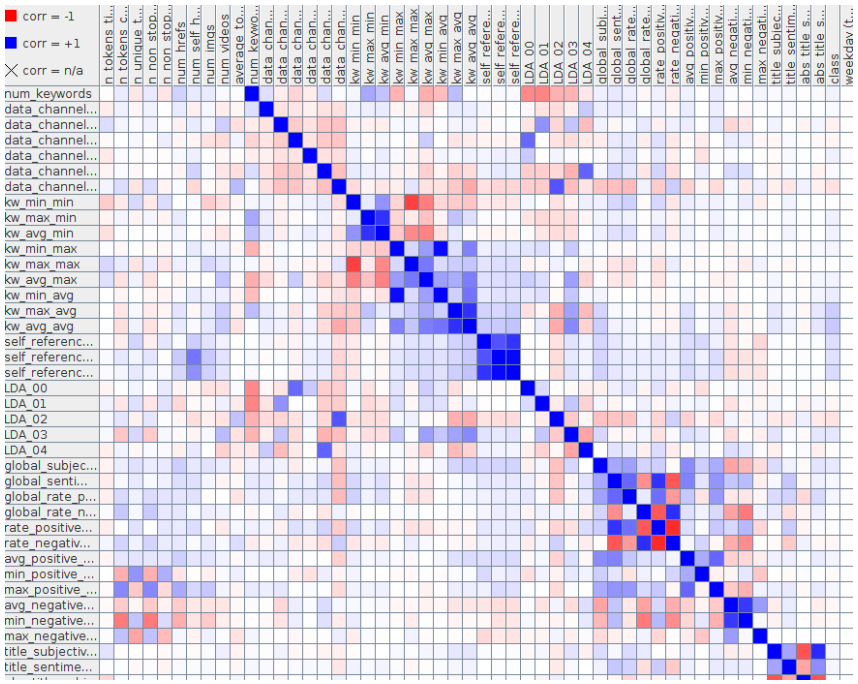


Figura 32: Tree

Las siguientes características más importantes son el día de la semana y self_reference_avg_sharess

(la cual indica el número medio de)Avg. shares of referenced articles in Mashable).

7. Bibliografía

Transparencias de clase.