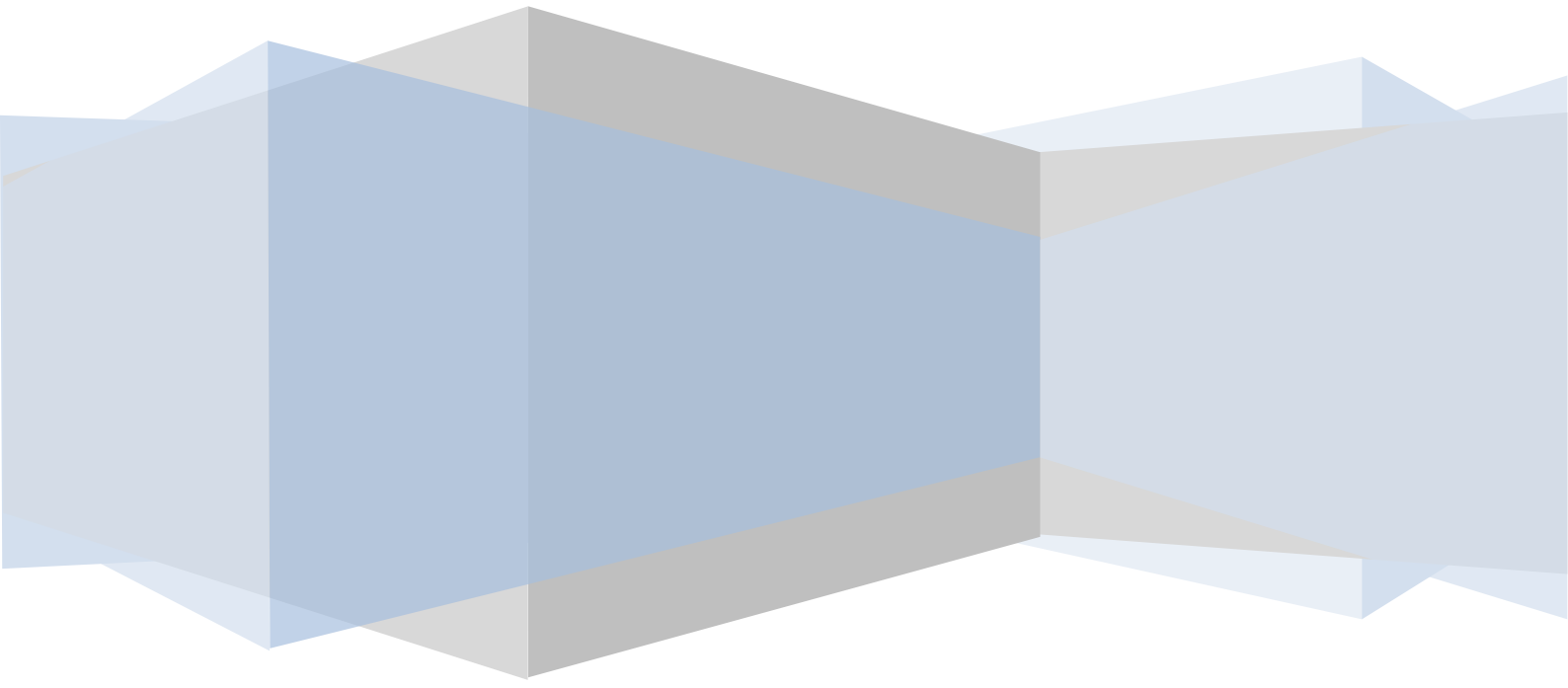
Practica 1

Análisis predictivo empresarial mediante clasificación

David Castro Salazar



Índice

Lista de Ilustraciones

1. Introducción

En esta práctica vamos a tratar la alta proliferación de espacios con noticias en la Web hace cada vez más útil la predicción automática de la popularidad de estas noticias. Utilizaremos unos algoritmos los cuales mostrare los que he elegido posteriormente y el proceso que he usado para tratar los datos. Estos algoritmos los usaremos para que a través de los atributos de cada noticia, poder predecir si va a ser popular o no popular. El conjunto de datos que vamos a tratar es de 39.644 noticias extraídas en 2015 de la web http://mashable.com. A partir de este dato, se puede valorar si la noticia es popular o no. En nuestro caso, fijaremos como umbral 3000 veces, de modo que si la noticia se comparte en un número superior a ese umbral, la noticia es considerada popular.

Primero voy a explicar el tratamiento que le he dado a los datos. En primer lugar he hecho un pre-procesamiento en el que he manipulado el conjunto de datos para que las variables de los días de la semana pasen a ser un valor numérico para que sea más fácil a la hora de normalizar y manejar los datos. Después de ordenar las columnas he normalizado los valores. Y por ultimo he sacado el conjunto de cuatro formas distintas. La primera con los datos sin ningún tipo de tratamiento para ver los resultados aun con los valores perdidos. La segunda quitando las filas que contienen valores perdidos. La tercera usando la media de cada atributo para ponerla sobre los valores perdidos. Y la cuarta usando el valor máximo sobre los valores perdidos. Podemos ver la construcción del nodo en la ilustración 1.

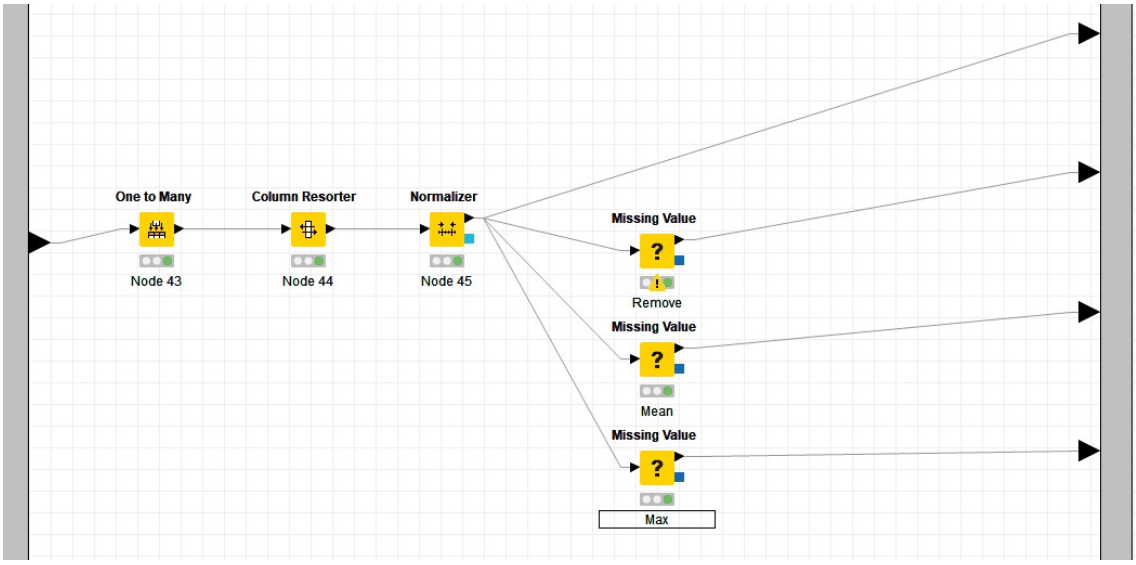


Ilustración 1: Nodo pre-procesamiento

Además del pre-procesamiento también he querido añadir a cada conjunto de datos como funcionaria si tuviera over sampling, under sampling o nada. Como muestro en las ilustraciones 2, 3, 4 y 5. Dentro de cada validación cruzada de cada algoritmo realizo su respectivo procesamiento. En la ilustración 4 con el nodo “equal

size" el mismo te iguala la cantidad de filas que coge eligiendo al atributo que quieres que sea igual en número de filas. Y en la ilustración 5 el proceso de over sampling por el cual dividimos en dos las filas para saber cuáles son populares y cuáles no y aumentamos la cantidad de filas del menor de los atributos.

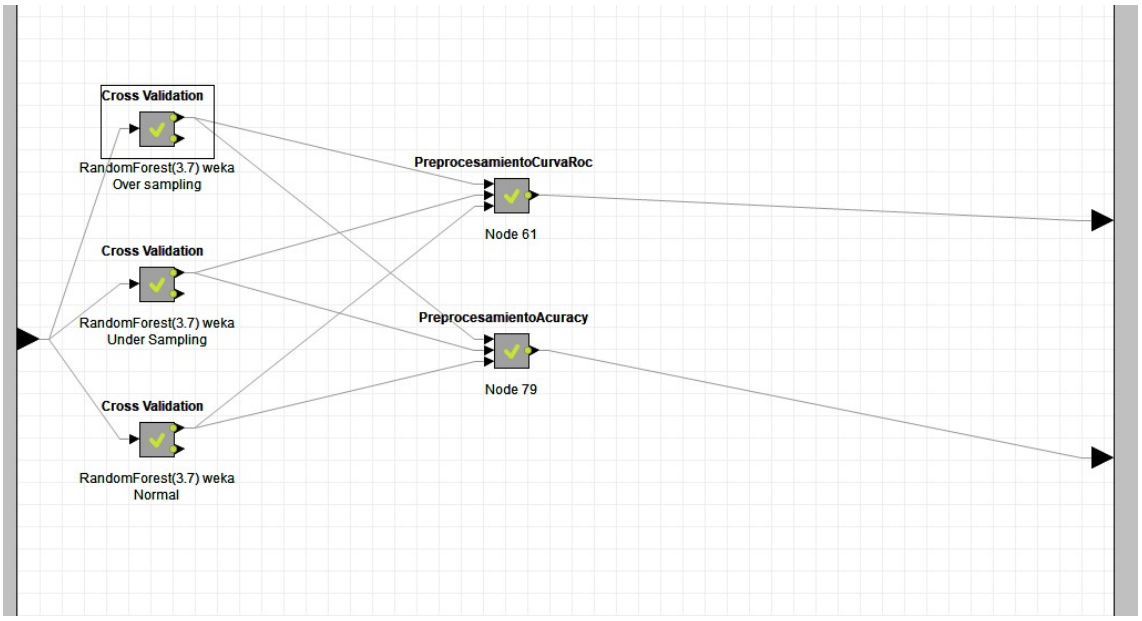


Ilustración 2 Nodo que contiene los distintos tipos de cross validation hechos.

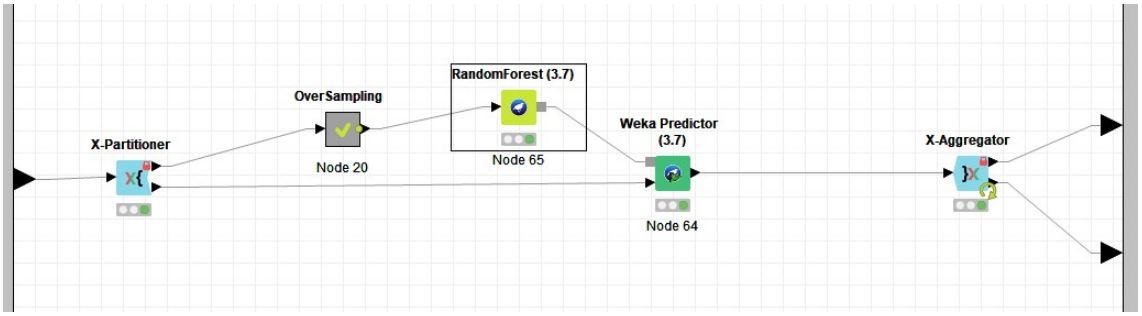


Ilustración 3 Nodo cross validation con over sampling

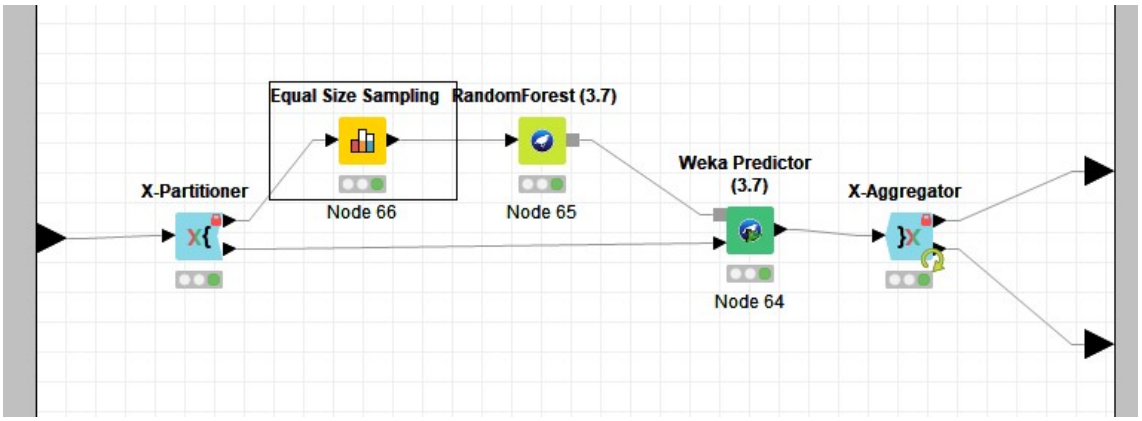


Ilustración 4 Nodo cross validation con under sampling

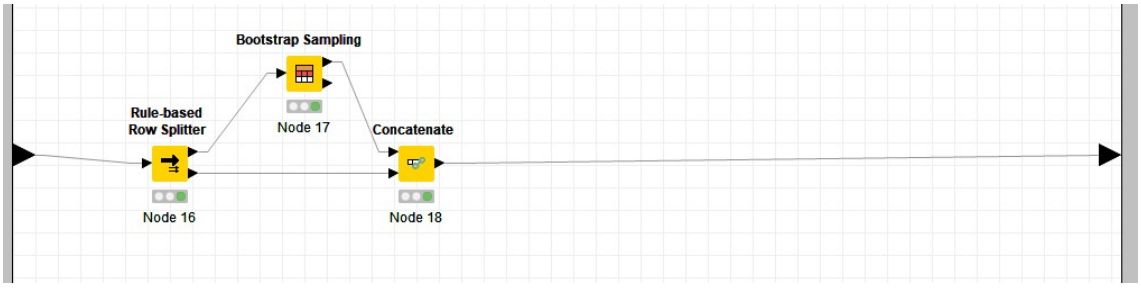


Ilustración 5 Nodo over sampling extendido

Randomforest, lo he escogido ya que parece ser un algoritmo robusto con respecto al ruido por y quería comprobar los resultados que daba. Para ello además he cogido tanto el de la clase de WEKA como el de que ya trae el programa para ver si había alguna diferencia significativa.

Ensembles of Balanced Nested Dichotomies for Multi-class Problems, este algoritmo lo he escogido para ver la actuación que tiene al tener que diferenciar únicamente entre dos clases en vez de varias, y además comprobar el tiempo que tarda.

K nearest neighbor, Este algoritmo lo he escogido ya que el k-nn sirve para escoger entre los vecinos más cercanos y quería comprobar si se daba el caso que los datos con menor distancia entre ellos eran más parecidos.

Bagging, este algoritmo usa arboles de “clasificación y regresión y la selección de subconjuntos en regresión lineal muestran que el empaquetamiento puede proporcionar ganancias sustanciales en la precisión.”[3]

FURIA weka, este algoritmo está basado en el algoritmo fuzzy. El punto fuerte es que es un algoritmo que tiene una gran precisión así que he decidido ponerlo para compararlo con los distintos resultados que dan los otros algoritmos.

Navie Bayes, en este caso he usado tanto el de weka como el del propio Knime para comprobar sus diferencias. Este algoritmo lo he escogido para poder comprobar si la estimación de parámetros que da es buena.

2. Resultados obtenidos

La ilustraciones 6 y 7 represe el workflow que tiene cada algoritmo

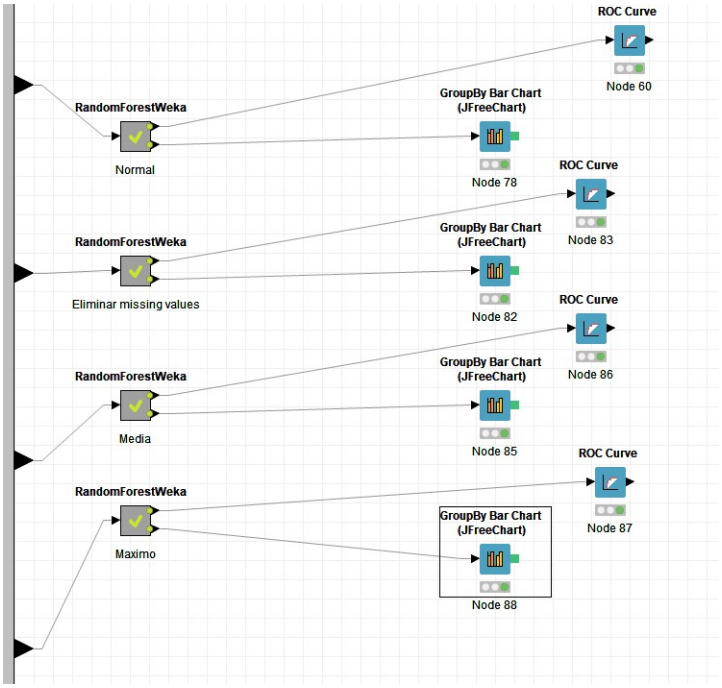


Ilustración 6 nodo con 4 entradas que se usa para recibir los conjuntos de datos y tratarlos con el algoritmo

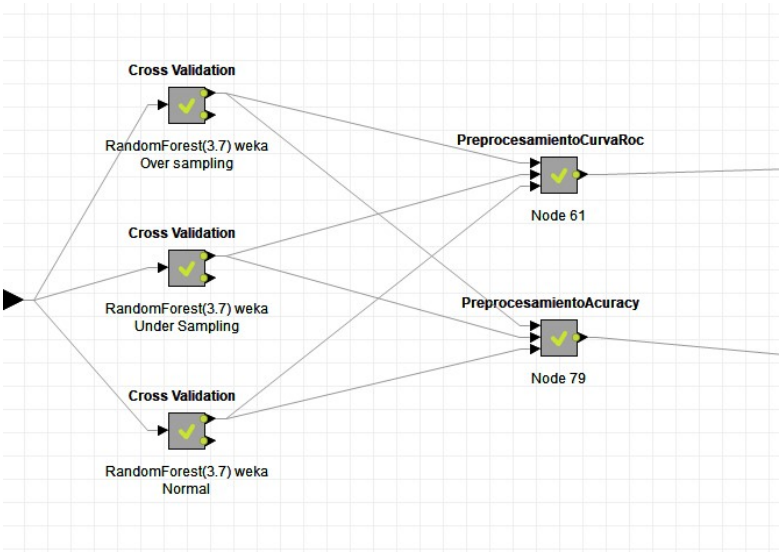


Ilustración 7 nodo para realizar la validación cruzada dividiéndolo en las distintas formas de procesar los datos

Para no repetir ilustraciones, tanto la ilustración 6 como la 7 se repiten en todos los algoritmo así que dejo estas de ejemplo. Al igual que no he puesto los nodos de over sampling y under sampling porque son idénticos a las ilustraciones 3, 4 y 5.

2.1 RandomForest Weka

El workflow de cada cross validation para Randomforest Weka y curva ROC de cada pre-procesamiento de datos.

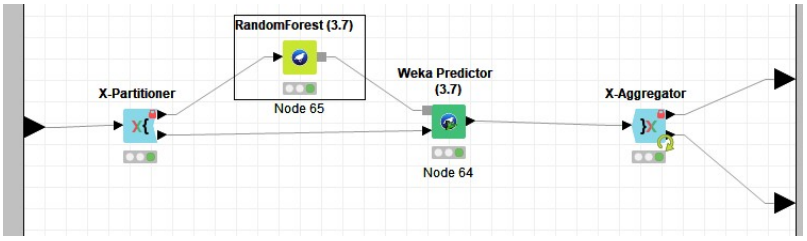


Ilustración 8 nodo cross validation randomforest Weka

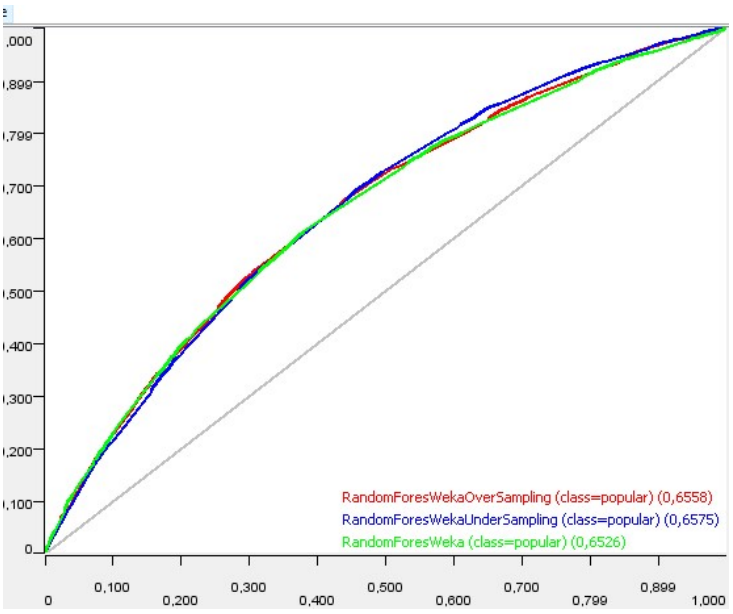


Ilustración 9 Curva ROC de Radonforest Weka

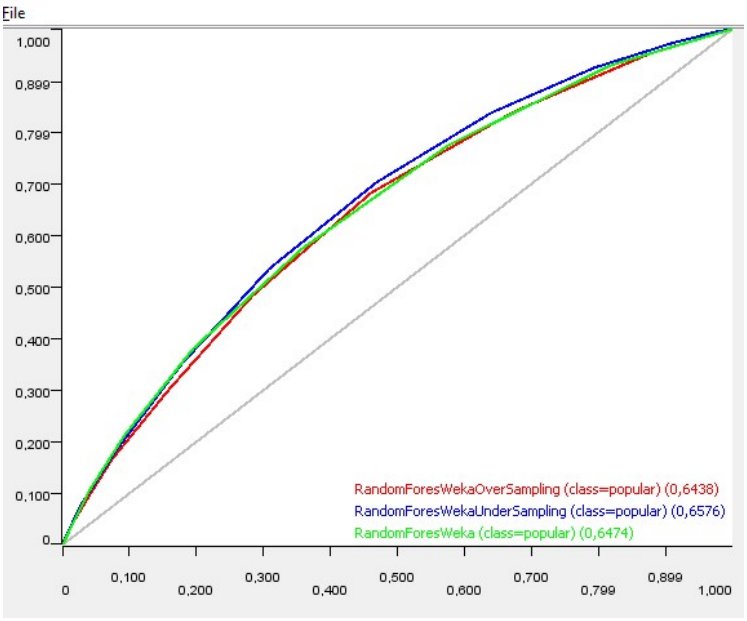


Ilustración 10 Curva ROC de Randonforest Weka sin missing values

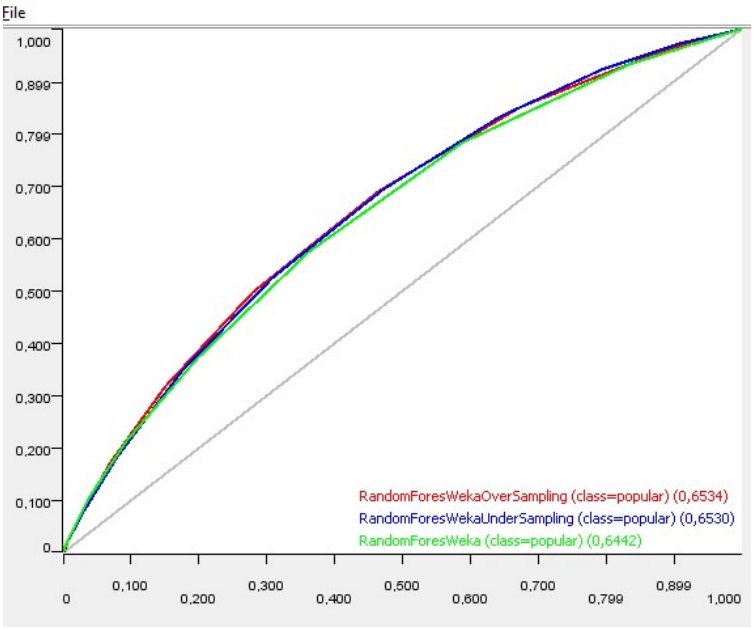


Ilustración 11 Curva ROC de Randonforest Weka con la media

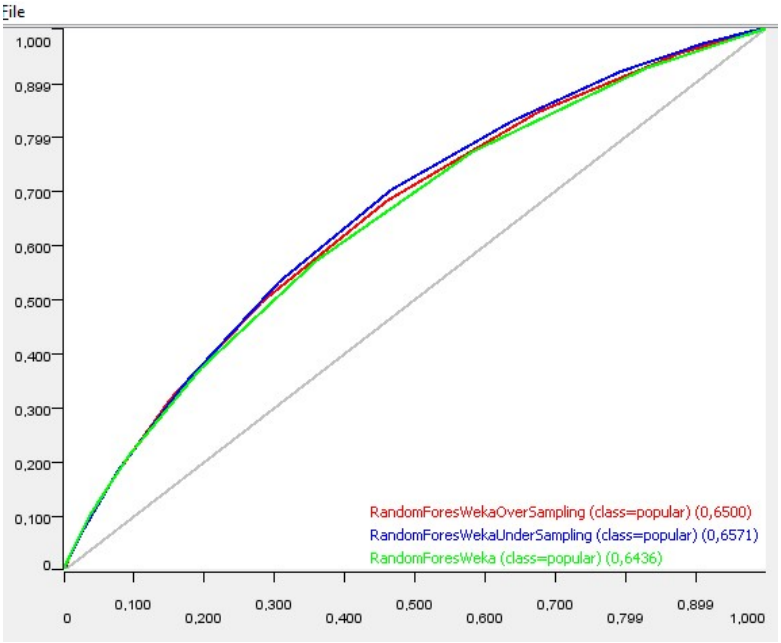


Ilustración 12 Curva ROC de Randonforest Weka con el máximo

2.2. RandomForest

El workflow de cada cross validation para Randomforest y curva ROC de cada pre-procesamiento de datos.

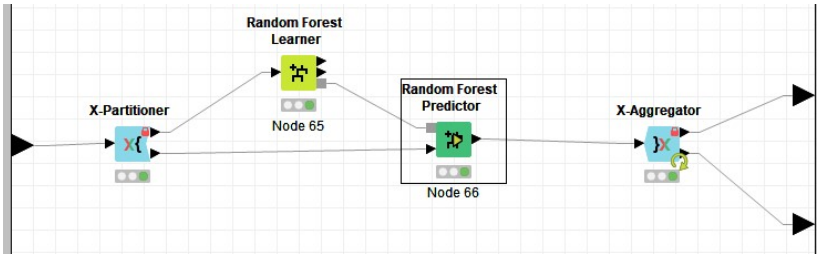


Ilustración 13 nodo cross validation randomforest

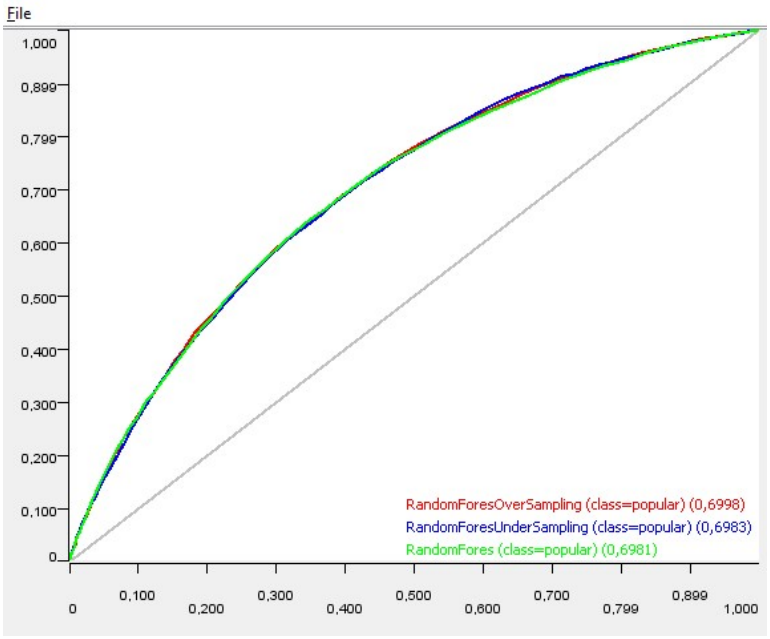


Ilustración 14 Curva ROC de Randonforest

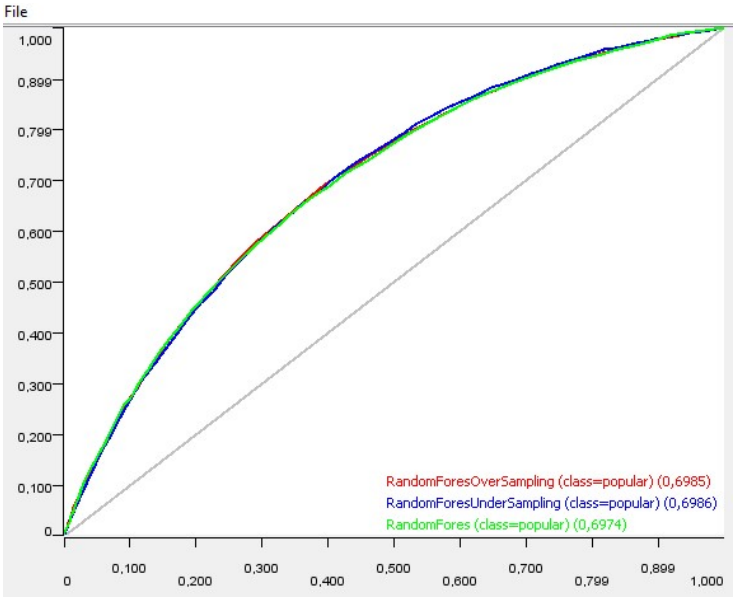


Ilustración 15 Curva ROC de Randonforest sin missing values

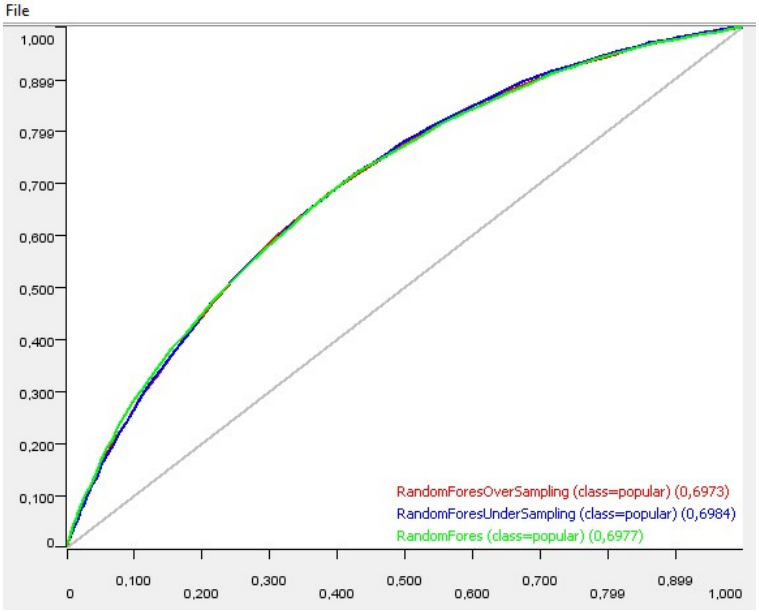


Ilustración 16 Curva ROC de Randonforest con la media

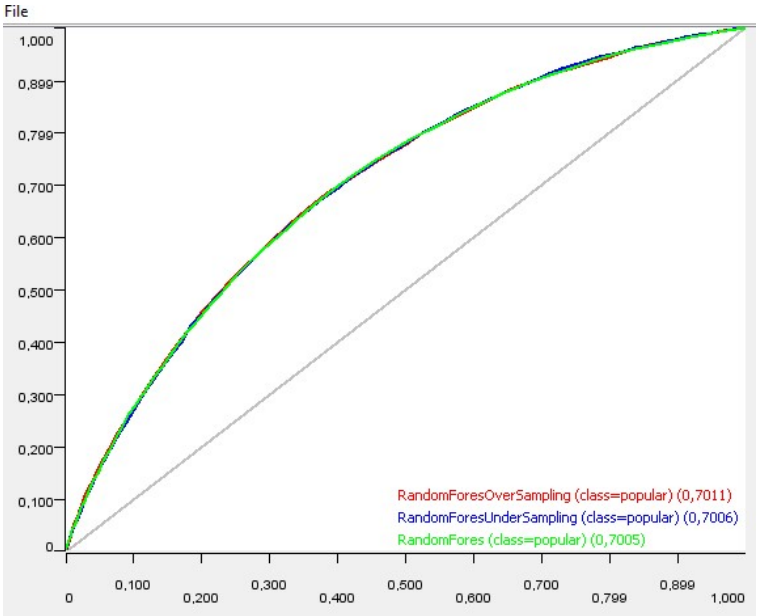


Ilustración 17 Curva ROC de Randonforest con el máximo

2.3. ClassBalancedND

El workflow de cada cross validation para ClassBalancedND y curva ROC de cada pre-procesamiento de datos.

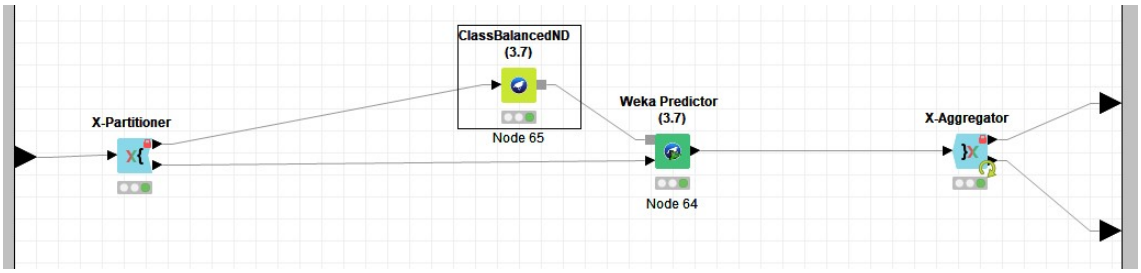


Ilustración 18 nodo cross validation ClassBalancedND

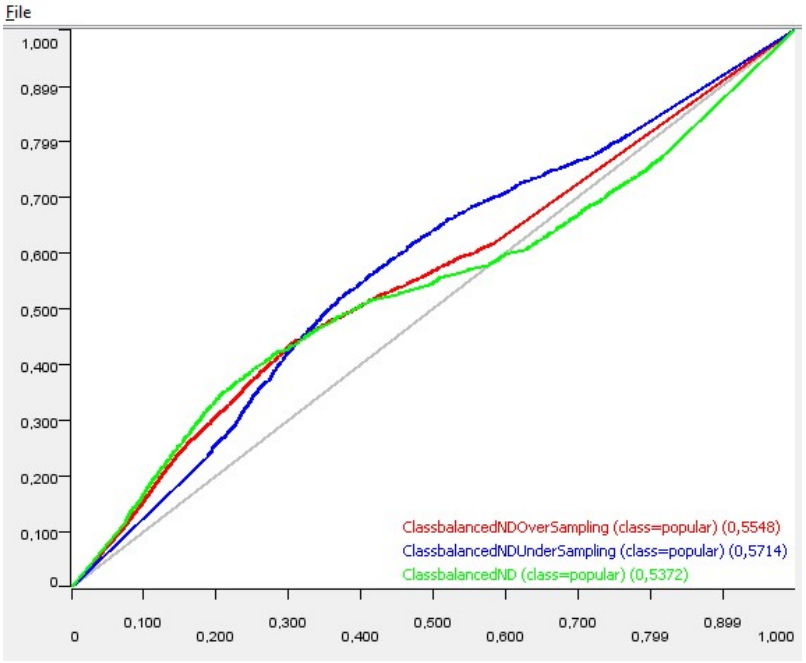


Ilustración 19 Curva ROC de ClassBalancedND

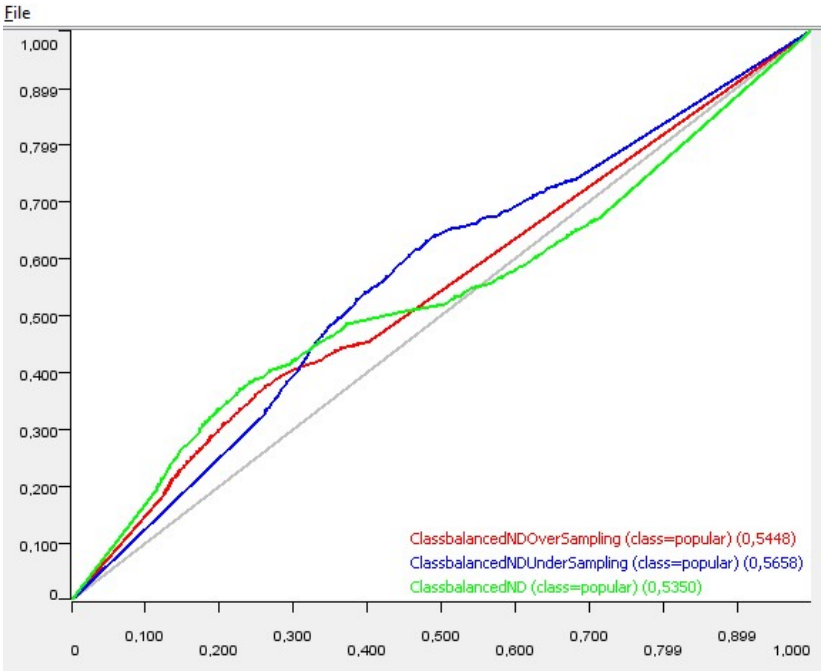


Ilustración 20 Curva ROC de ClassBalancedND sin missing values

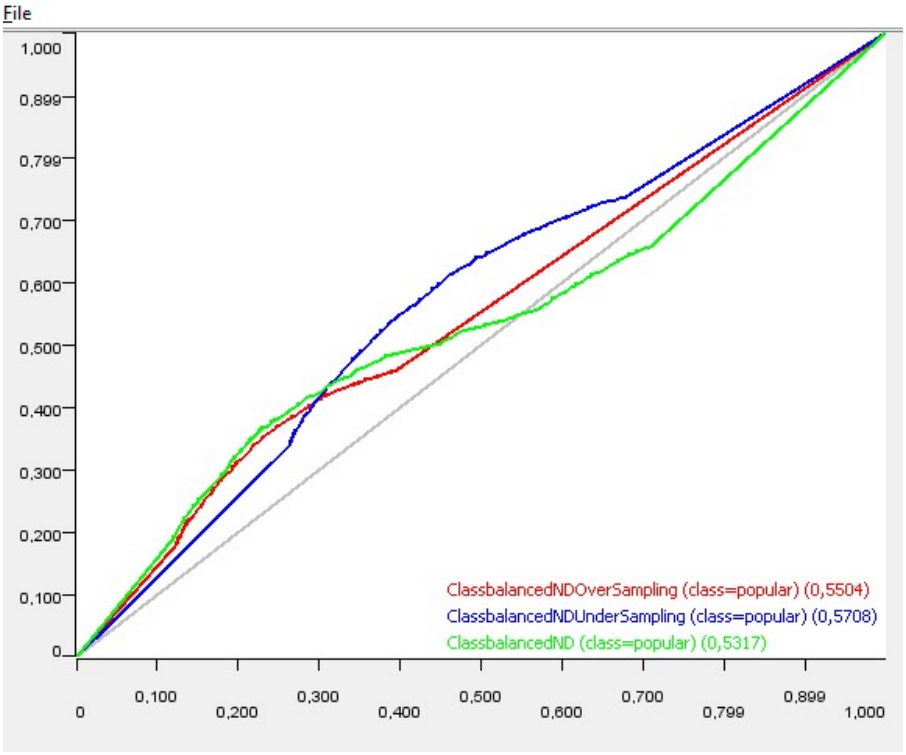


Ilustración 21 Curva ROC de ClassBalancedND con la media

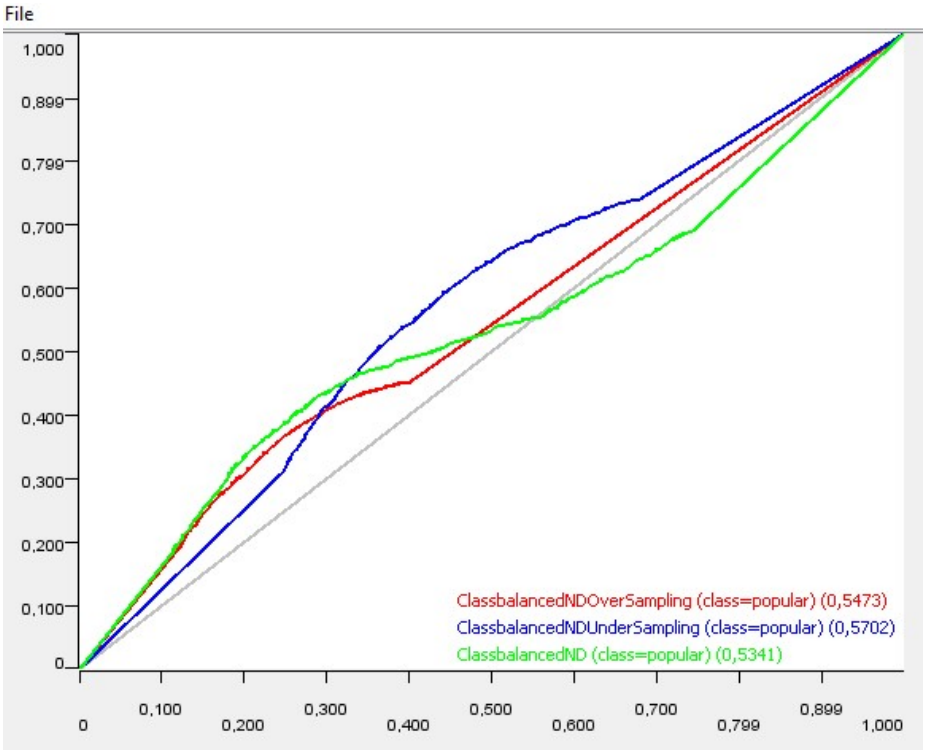


Ilustración 22 Curva ROC de ClassBalancedND con el maximo

2.4 KNN

El workflow de cada cross validation para KNN y curva ROC de cada pre-procesamiento de datos.

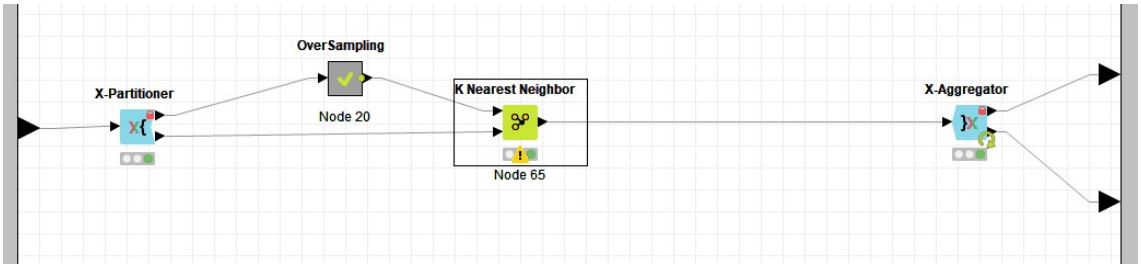


Ilustración 23 nodo cross validation KNN

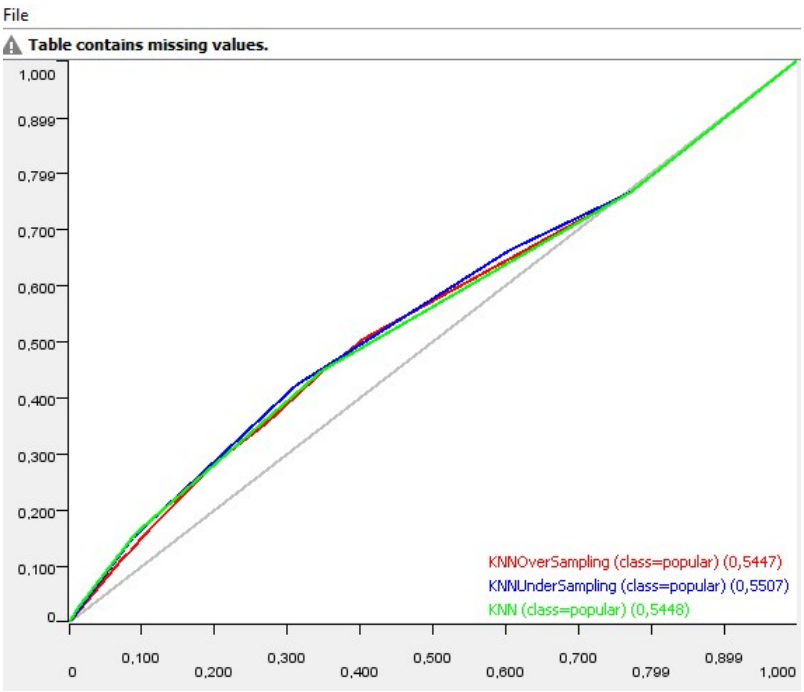


Ilustración 24 Curva ROC de KNN

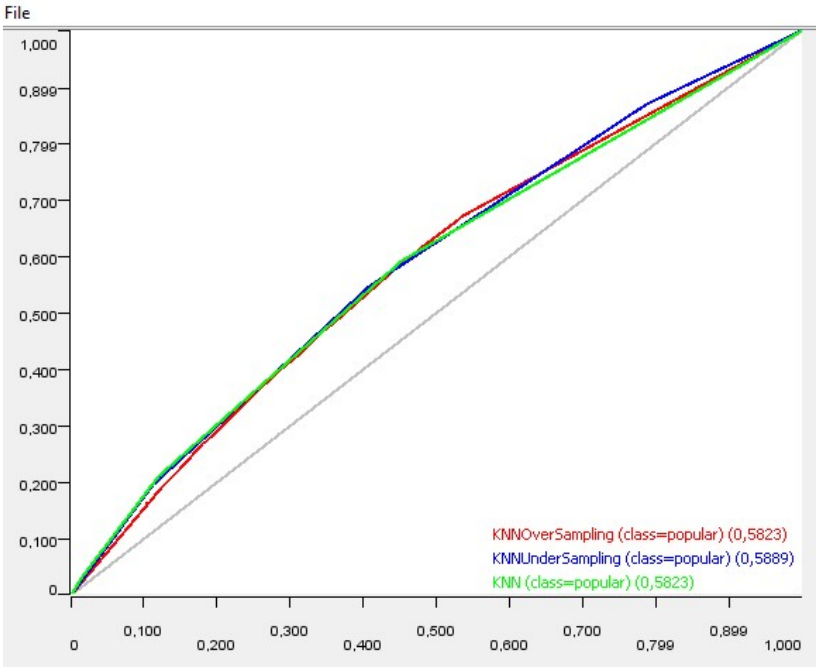


Ilustración 25 Curva ROC de KNN sin missing values

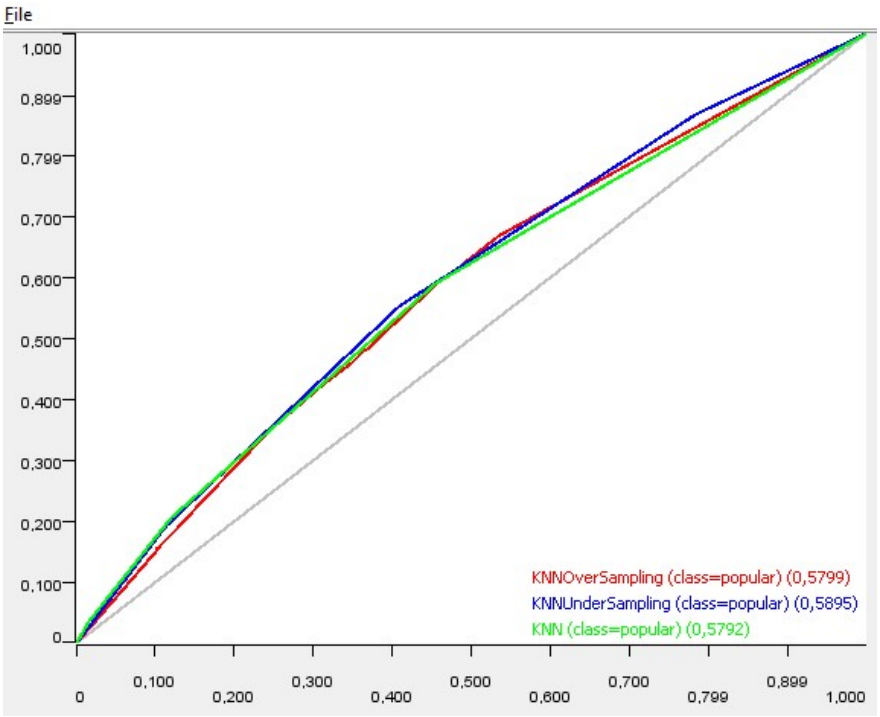


Ilustración 26 Curva ROC de KNN con la media

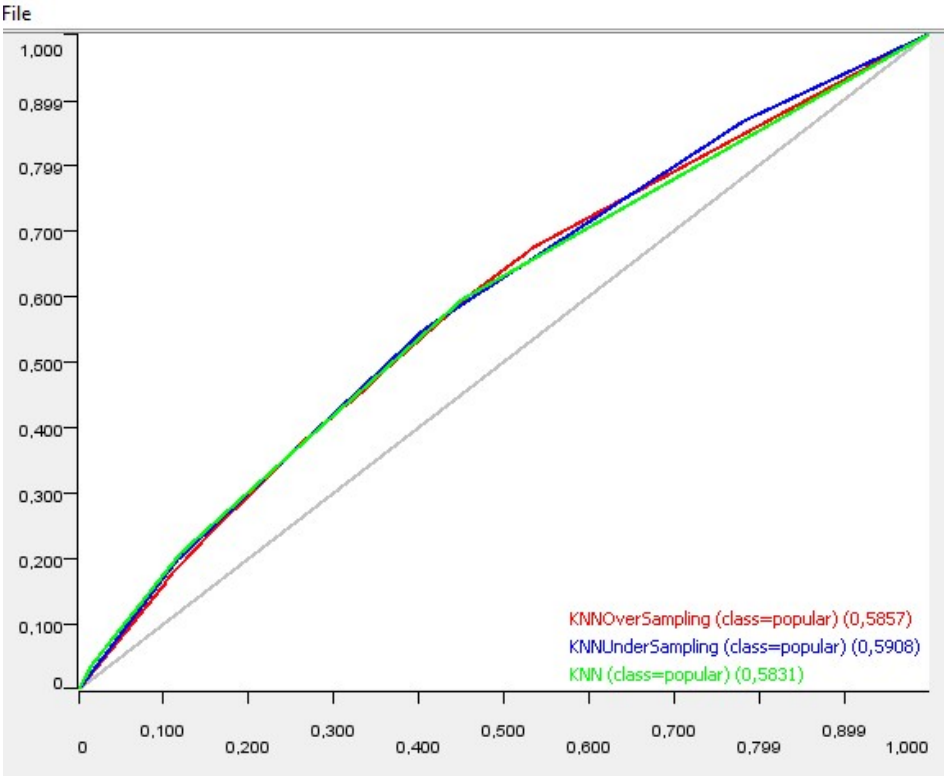


Ilustración 27 Curva ROC de KNN con el máximo

2.5. RProp MLP Learner

El workflow de cada cross validation para RProp MLP Learner y curva ROC de cada pre-procesamiento de datos.

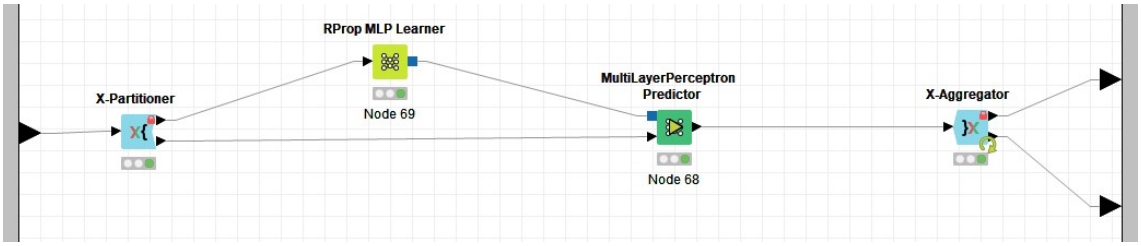


Ilustración 28 nodo cross validation RProp MLP Learner

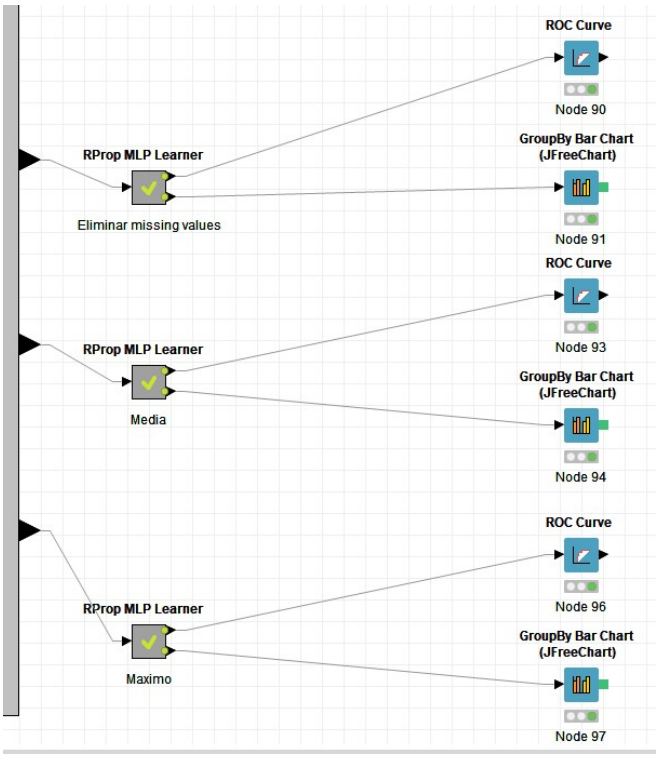


Ilustración 29 nodo de trabajo de RProp MLP Learner al que solo llegan tres entradas

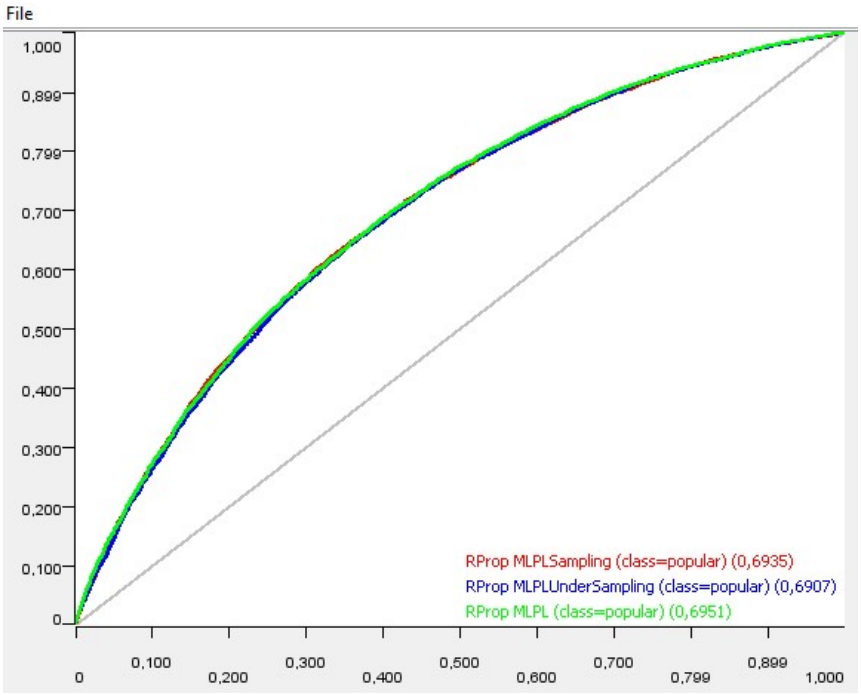


Ilustración 30 Curva ROC de RProp MLP Learner sin missing values

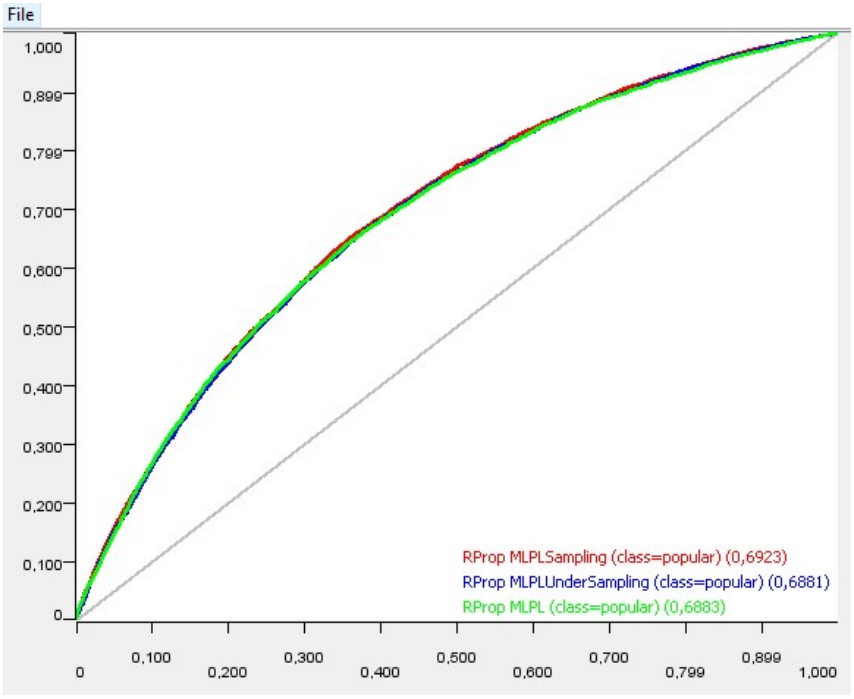


Ilustración 31 Curva ROC de RProp MLP Learner con la media

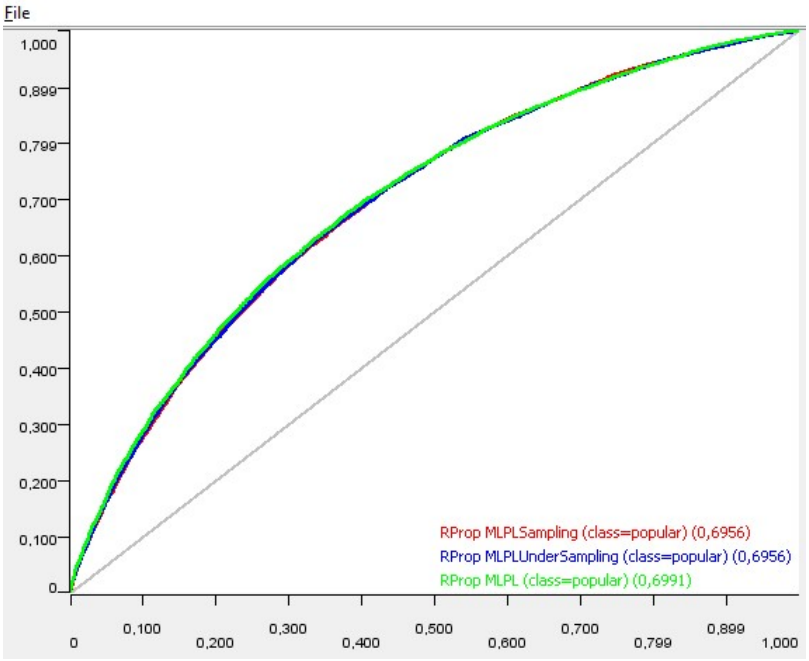


Ilustración 32 Curva ROC de RProp MLP Learner con el máximo

2.6. Bagging

El workflow de cada cross validation para Bagging y curva ROC de cada pre-procesamiento de datos.

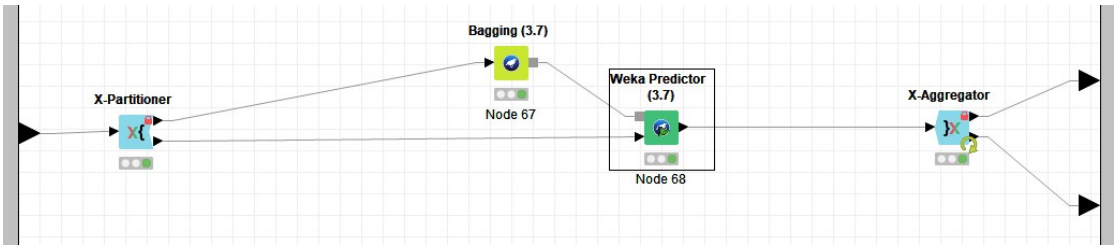


Ilustración 33 nodo cross validation Bagging

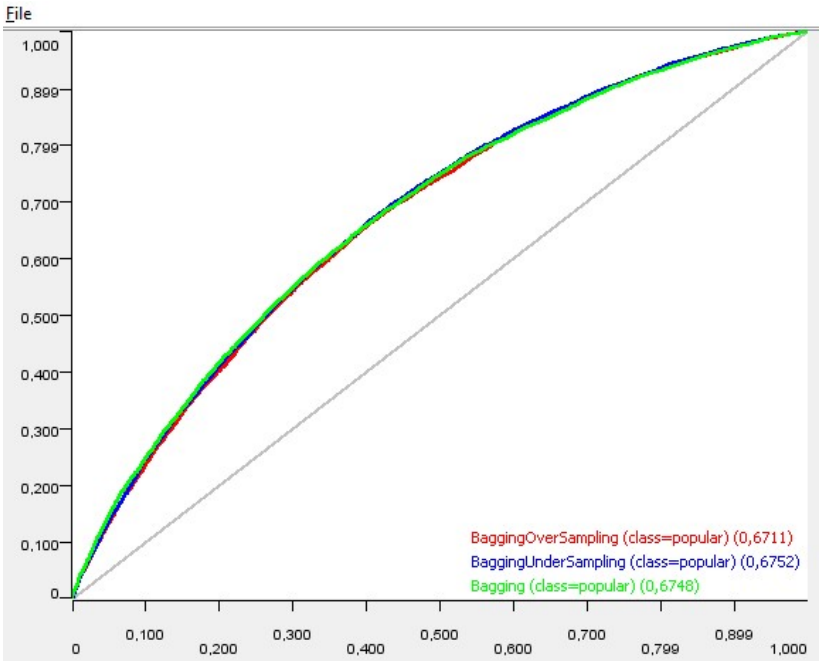


Ilustración 34 Curva ROC de Bagging

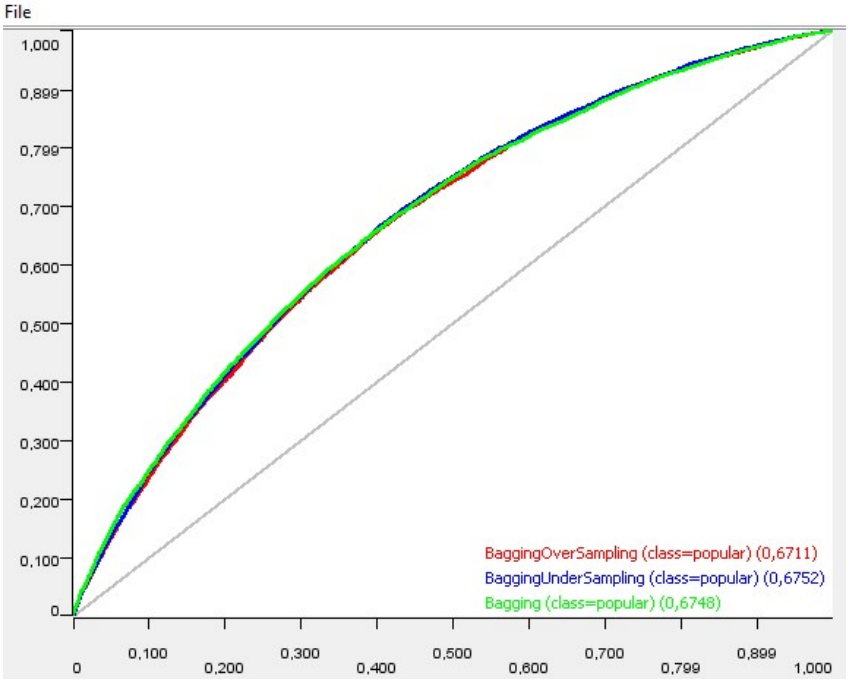


Ilustración 35 Curva ROC de Bagging sin missing values

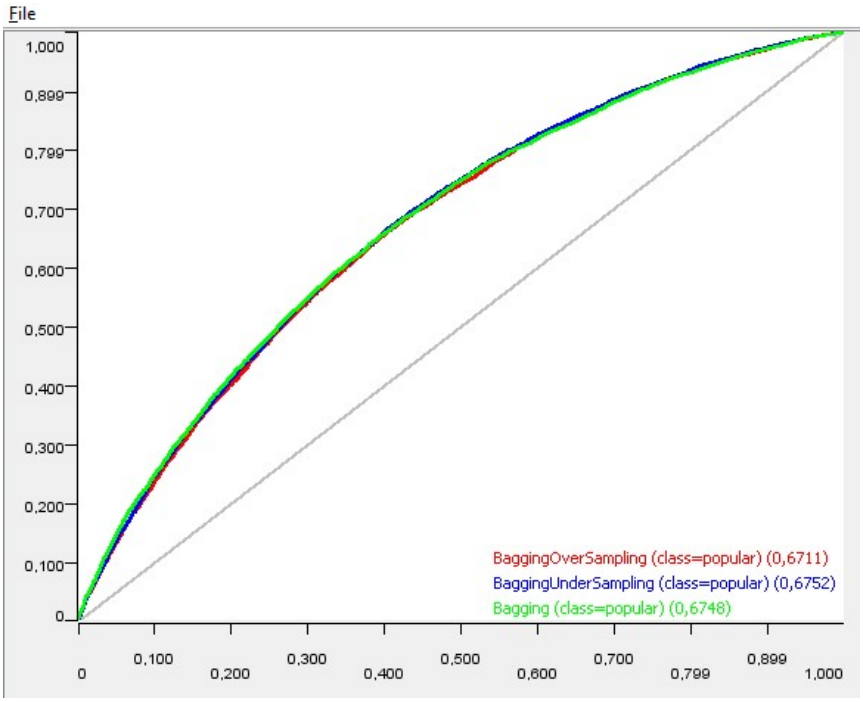


Ilustración 36 Curva ROC de Bagging con la media

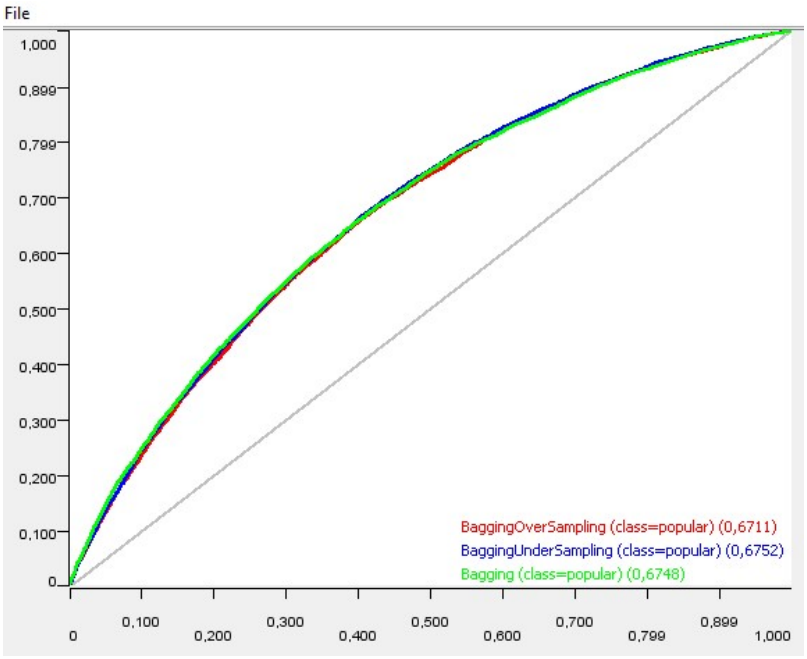


Ilustración 37 Curva ROC de Bagging con el máximo

2.7. FURIA

El workflow de cada cross validation para FURIA y curva ROC de cada pre-procesamiento de datos.

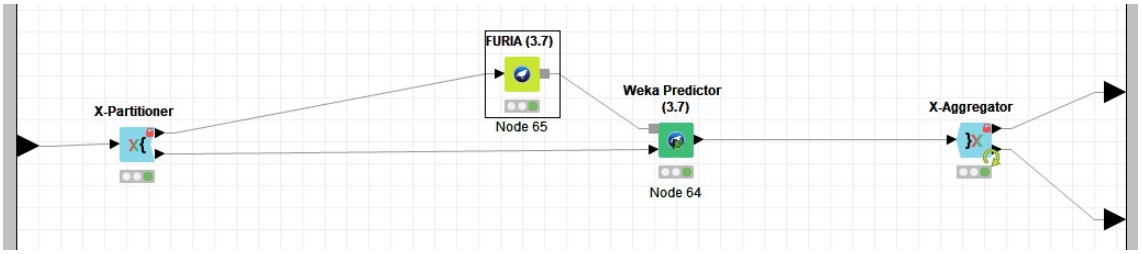


Ilustración 38 nodo cross validation FURIA

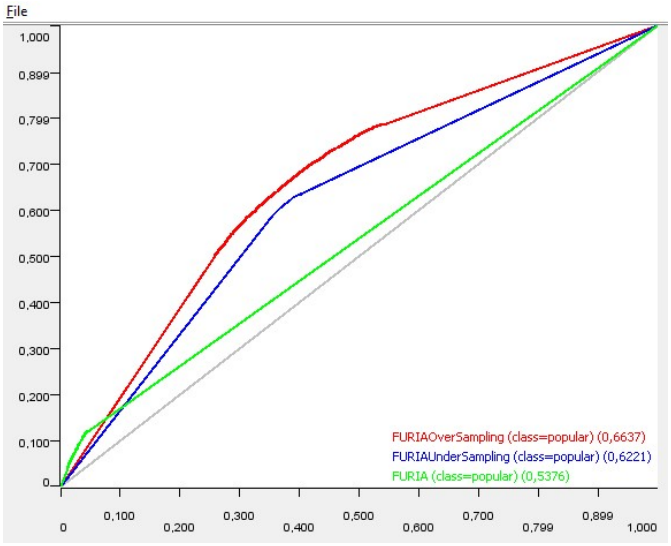


Ilustración 39 Curva ROC de FURIA

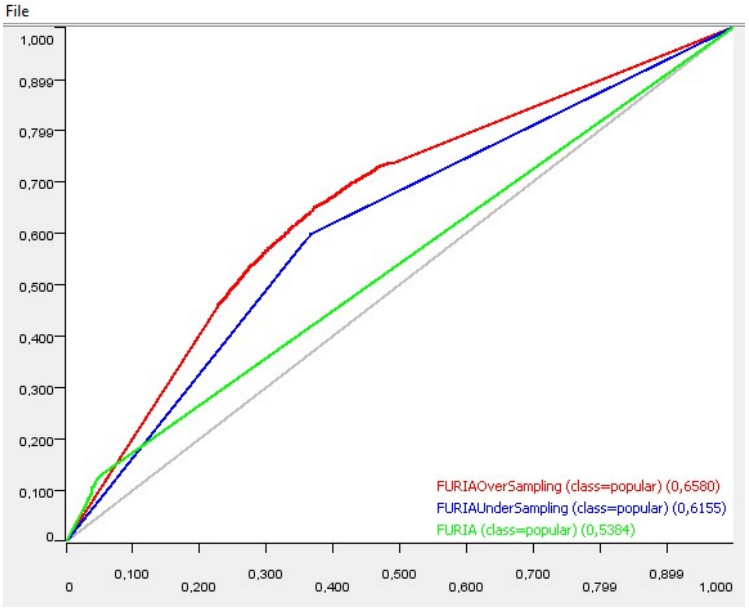


Ilustración 40 Curva ROC de FURIA sin missing values

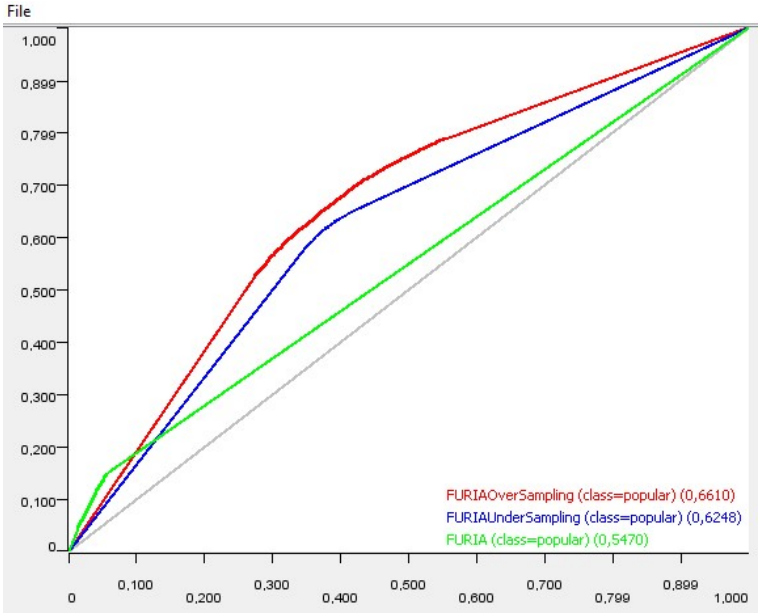


Ilustración 41 Curva ROC de FURIA con la media

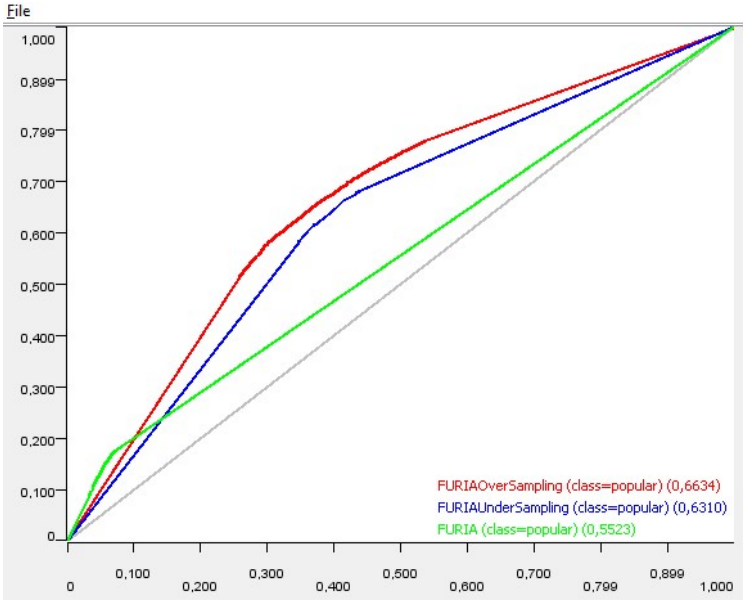


Ilustración 42 Curva ROC de FURIA con el máximo

2.8. NaiveBayes Weka

El workflow de cada cross validation para NaiveBayes Weka y curva ROC de cada pre-procesamiento de datos.

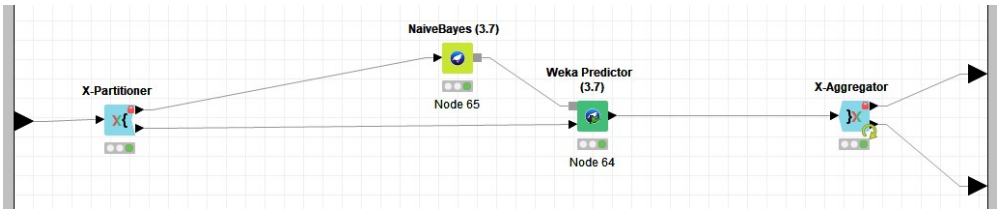


Ilustración 43 nodo cross validation NaiveBayes Weka

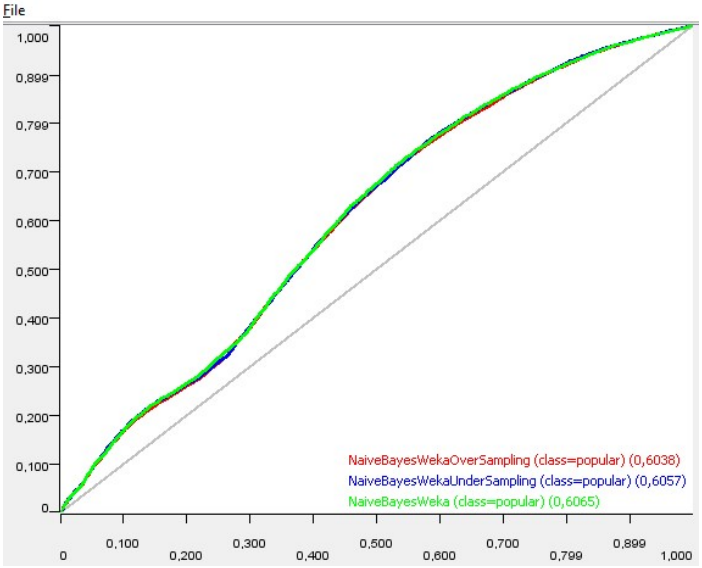


Ilustración 44 Curva ROC de NaiveBayes Weka

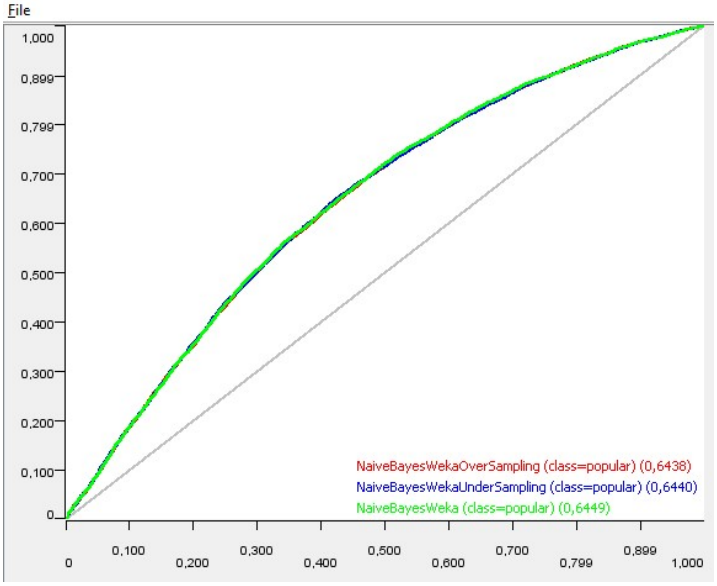


Ilustración 45 Curva ROC de NaiveBayes Weka sin missing values

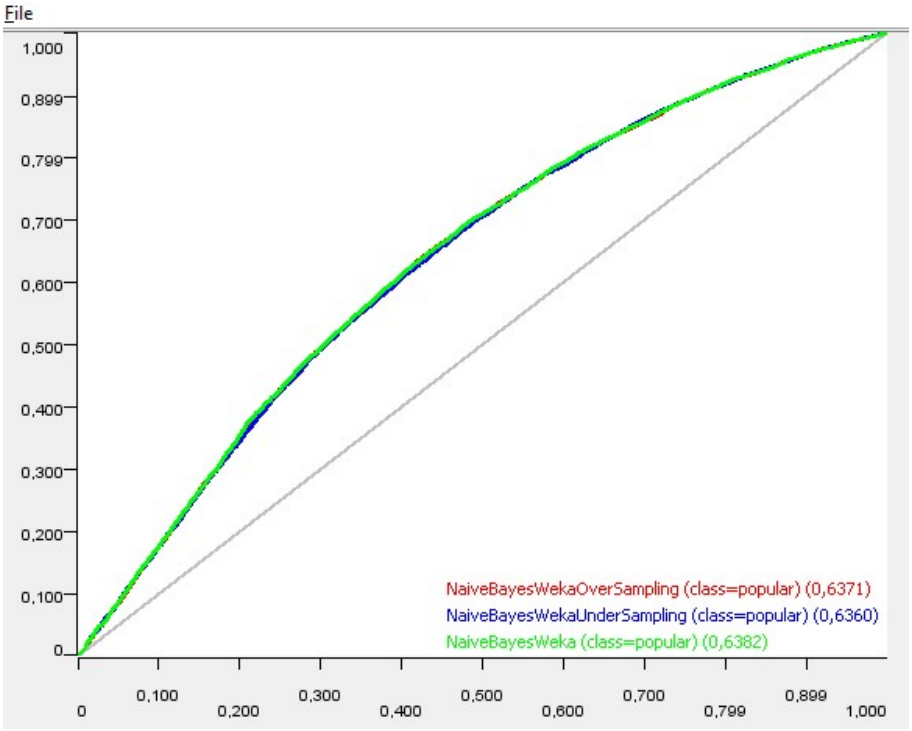


Ilustración 46 Curva ROC de NaiveBayes Weka con la media

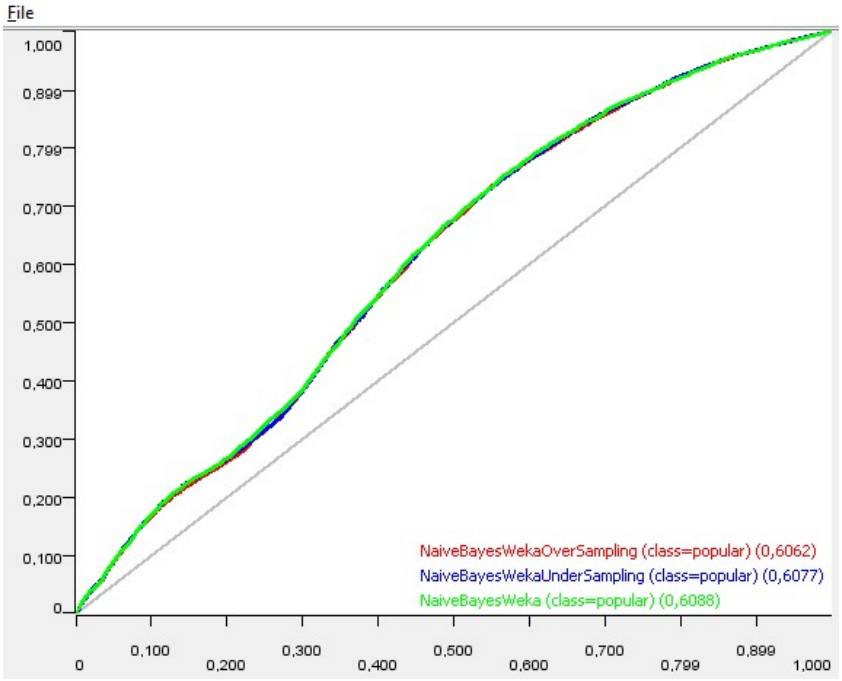


Ilustración 47 Curva ROC de NaiveBayes Weka con el máximo

2.9. NaiveBayes

El workflow de cada cross validation para NaiveBayes y curva ROC de cada pre-procesamiento de datos.

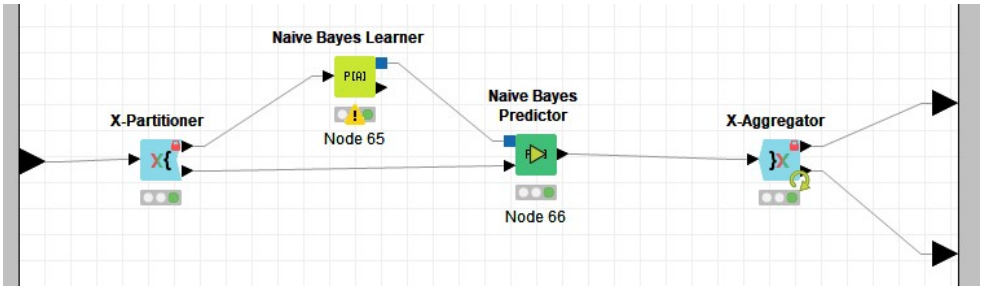


Ilustración 48 nodo cross validation NaiveBayes

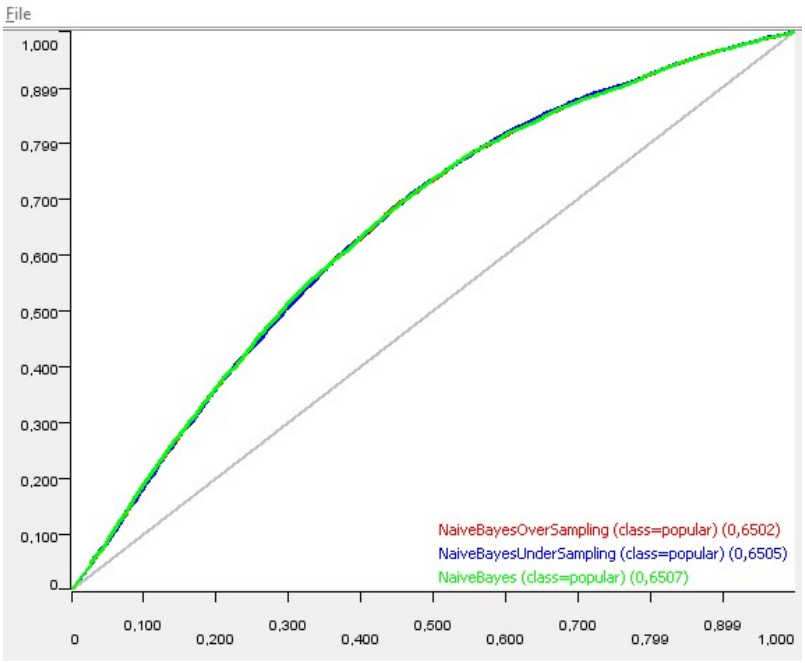


Ilustración 49 Curva ROC de NaiveBayes

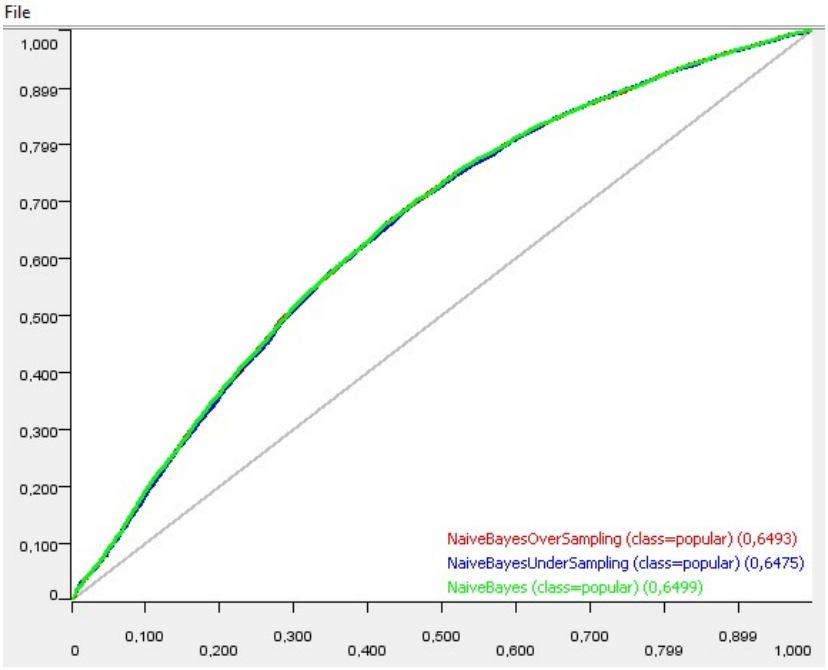


Ilustración 50 Curva ROC de NaiveBayes sin missing values

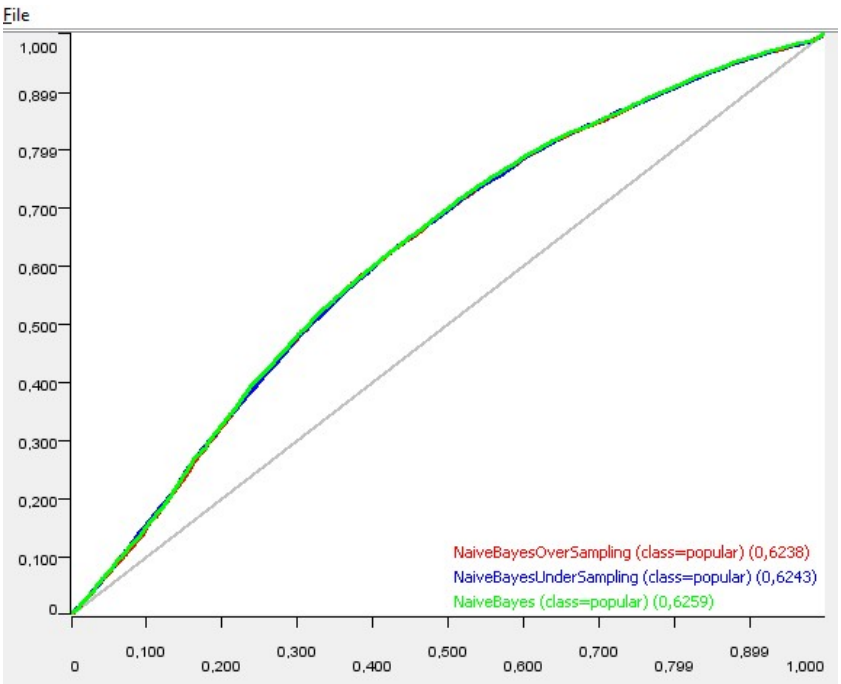


Ilustración 51 Curva ROC de NaiveBayes con la media

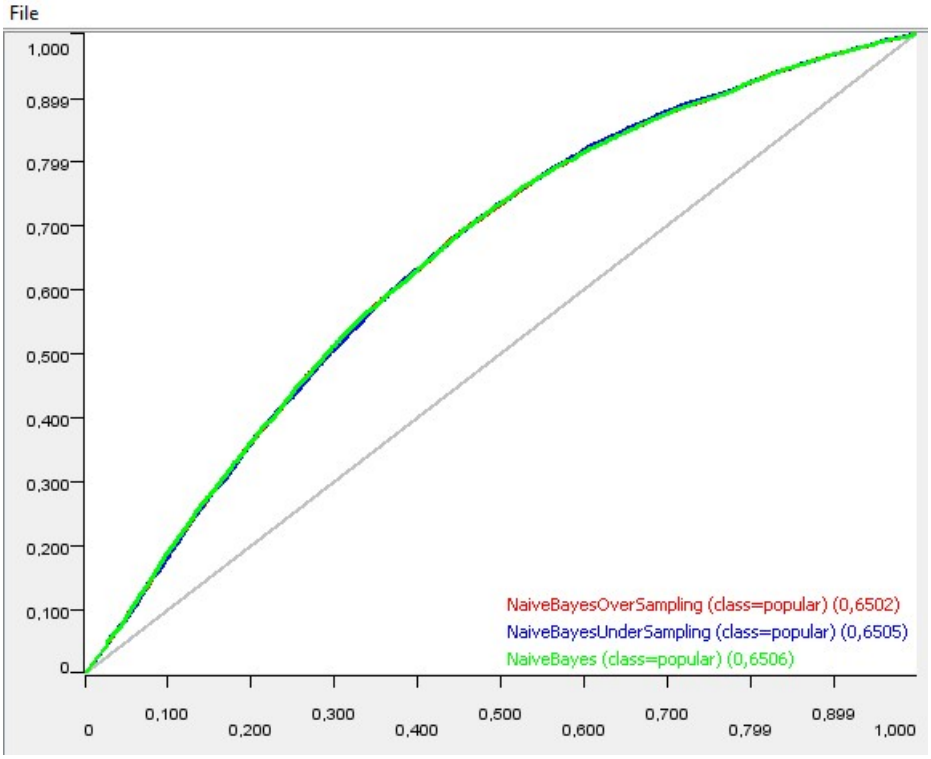


Ilustración 52 Curva ROC de NaiveBayes con el máximo

3. Análisis de resultados

Para el análisis de datos vamos a tener en cuenta las curvas ROC mostradas en las ilustraciones del anterior punto.

Como se puede ver he trabajado con distintos grupos de datos a través del pre-procesamiento, y al menos una de las variantes de cada algoritmo dan una superficie bajo la curva superior al 0.55. Por una lado al algoritmos que en todas sus variantes den resultados muy similares y muy buenos, pero también hay otros que hay una gran diferencia dependiendo de cómo se traten los datos. Voy a hablar de cada uno por separado y al final los comparare.

3.1. RandomForest Weka

Ha dado muy buen resultado prácticamente en todas sus variantes en todas superiores al 0.64, el algoritmo en si es muy bueno con el pre-procesamiento de datos no parece que haya mucha diferencia, lo único que se puede notar es que l bloque en el que los valores perdidos se cambian por el máximo si hay una pequeña perdida. Esto se puede deber a que el randomforest trata los valores de forma individual.

3.2 RandomForest

Este ha sido el mejor algoritmo he general todas sus variantes han dado los mejores resultado pero en especial hay que destacar el conjunto de datos en el que están los valores perdidos tratados como el máximo de cada clase. Es un poco raro ya que sale al contrario que el randomForest de Weka. Creo que esto se puede deber a que las noticias populares suelen tener mayor cantidad de valores máximos.

3.3. ClassBalancedND

Al contrario que el anterior este ha sido el peor algoritmo. Además tiene en algún caso una gran diferencia dependiendo del tipo de pre-procesamiento hecho. Se puede ver con las graficas que cuando las clases están muy desiguales salir con peor curva como se muestra en las ilustraciones 19 y 21. También se puede apreciar que es peor porque es un algoritmo que funciona mejor con una mayor cantidad de clases.

3.4. KNN

En este caso el KNN no me da especialmente un buen resultado, entiendo con ello que los vecinos mas cercanos no sirven especialmente a la hora de acertar las clases.

3.5. RProp MLP Learner

En este caso el comportamiento de las rede neuronales también da uno de los mejores resultados de la práctica, muy parecido al del RandomForest. Esto se debe a que en general las redes neuronales resulven este tipo de problemas con mayor acierto.

3.6. Bagging

Para este algoritmo da unos resultados bastantes buenos por el algoritmo en sí. También se puede ver que no hay mucha diferencia entre los distintos pre-procesamientos. No entiendo porque pero da el mismo resultado de la curva en todos.

3.7. FURIA

Da unos resultados intermedios. Aunque aquí sí que se pueden presenciar una gran diferencia entre los pre-procesamientos. El FURIA trabaja especialmente mal con clases que estén muy diferenciadas en el peso. Y mejor cuanto mayor sea la cantidad de datos con los que se trabaja. Si se puede decir que la forma de tratar los valores perdidos no afectas mucho.

3.8. NaiveBayes Weka

No da especialmente buenos resultados, se nota que funciona peor cuando no se tratan los valores perdidos. Además funciona mal cuando se sustituyen por el máximo. Aun así es llamativo que aunque la clase este desbalanceada los resultados son muy similares.

3.9. NaiveBayes

En este caso da unos resultados buenos, a excepción del que tratamos los valores perdidos como la media. Los demás dan entorno al 0.65. Es llamativo que este algoritmo funcione mejor que el respectivo de weka.

3.10 general

En primer lugar nombrar que los algoritmos que trae el propio KNIME son aparentemente mejores que sus respectivos en Weka. También quiero añadir que el algoritmo FURIA ha tardado un tiempo llamativamente grande en ejecutarse con respecto a los demás, y no ha dado unos resultados especialmente buenos.

Los algoritmos de de multiclases como era de esperar no han dado unas soluciones muy buenas.

Redes neuronales ha funcionado muy bien con respecto a los demás y no han tardado un gran tiempo en ejecutarse.

Por último los algoritmos relacionados con arboles han dado en general buenos resultados y entre ellos el mejor. Aunque no sean especialmente rápidos sí que creo que son los que merecerían la pena.

4. Configuración de algoritmos

Voy a variar algunos de los valores predeterminados de los algoritmos para ver que tal seria su rendimiento. Para ello voy a escoger tres algoritmos el KNN, el MLP y END. De cada uno voy a coger la hebra que mejor resultados he obtenido.

4.1. KNN

Para este he escogido la hebra que pone los valores perdidos como máximos y que además tiene under sampling. Como se muestra en las siguientes ilustraciones voy a probar a los 15 vecinos más próximos y para el vecino más próximo y compararemos.

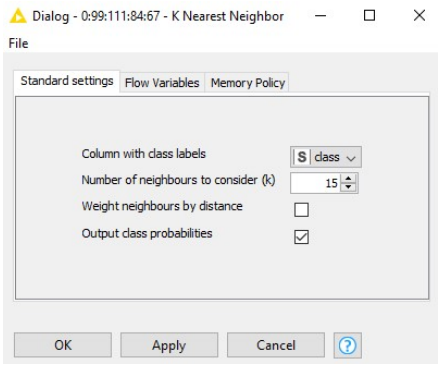


Ilustración 53 KNN con 15 vecinos cercanos

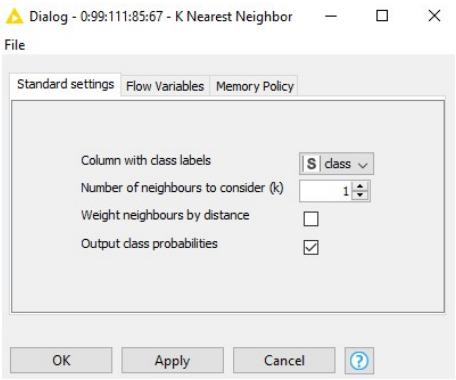


Ilustración 54 KNN con el vecino más cercano

Como era de esperar el KNN cuantos más vecinos se tengan en cuentan mejor soluciones dará. Se puede observar tanto en la curva ROC como en el accuracy.

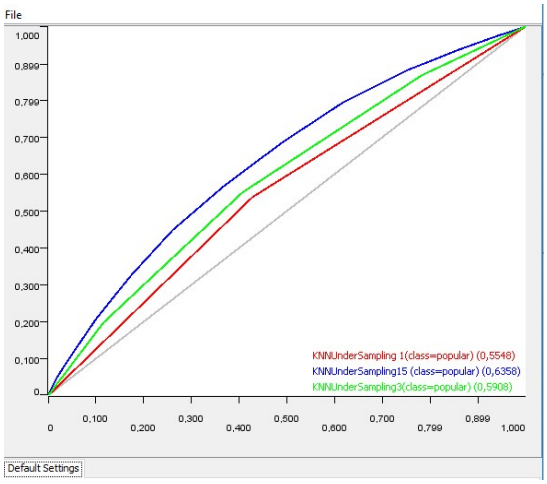


Ilustración 55 curva ROC comparativa

Table "default" - Rows: 3 | Spec - Columns: 2 | Properties | Flow Variables

Row ID	D Accuracy	S Algoritmo
Overall	0.563	RandomForesWekaOverSampling
Overall_dup	0.617	RandomForesWekaUnderSampl...
Overall_dup_...	0.584	RandomForesWeka

Ilustración 56 tabla de valores

4.2 MLP

En este caso vamos escoger el mismo que el anterior pero esta vez sin under sampling. Vamos a poner dos redes neuronales en vez de una que era la que estaba pre determinada.

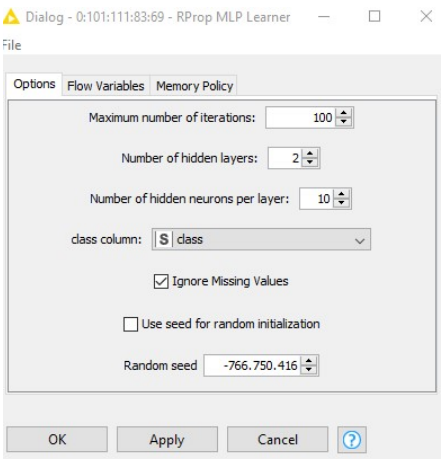


Ilustración 57 Opciones del MLP para poenr dos layers

El cambien en el numero de Layers no produce un gran cambio en este caso con la semilla puesta a hecho incluso que sea peor.

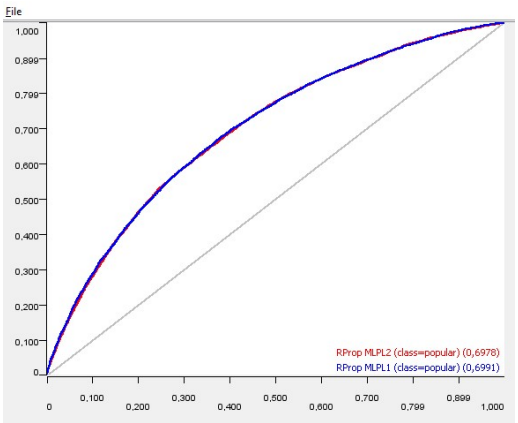


Ilustración 58 Curva ROC de MLP

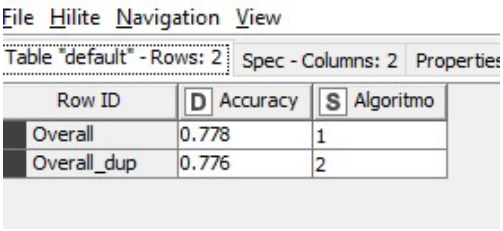


Ilustración 59 Tabla de valores de MLP

4.3 NaiveBayes

Para este caso vamos a escoger el caso en el que no se trata de ninguna manera los datos que tenemos, es decir con el valores perdidos y sin equilibrar la clase. Y lo que vamos a cambiar es la probabilidad. En el primero será el estándar que es 0 y pondré uno con 0,5 y otro como 1 como se muestra en las siguientes imágenes.

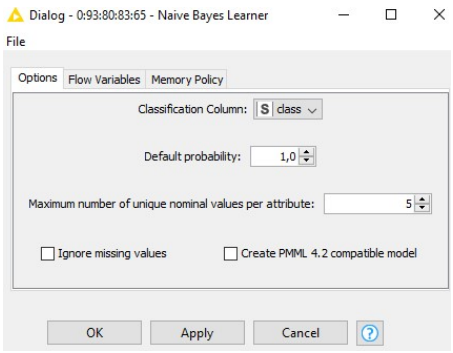


Ilustración 60 Naive bayes con 1 de probabilidad

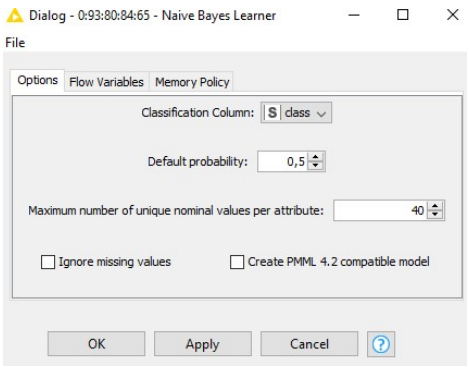


Ilustración 61 Naive bayes con 0.5 de probabilidad

Como se puede comprobar no ha habido ninguna mejora de hecho cuanto más subas la probabilidad pero salen los resultados como se muestra en la curva y en la tabla.

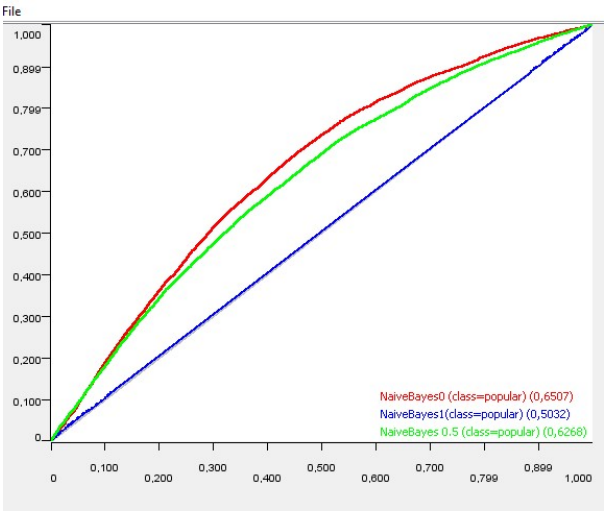


Ilustración 62 Curva ROC de Naive bayes

File Hilite Navigation View

Table "default" - Rows: 3 | Spec - Columns: 2 | Properties | Flo

Row ID	D Accuracy	S Algoritmo
Overall	0.564	0
Overall_dup	0.775	1
Overall_dup_...	0.681	0.5

Ilustración 63 Tabla de varoles de Naive bayes

5. Procesado de datos

Como podemos observar en al siguiente imagen es un problema que está muy desigualados. Siendo la clase no popular la más numerosa. Para tratar como puede afectar esto a los resultados he decidido dividirlo en 12 formas de interpretarlos por algoritmo. Para dejarlo más claro lo divido en dos fases.

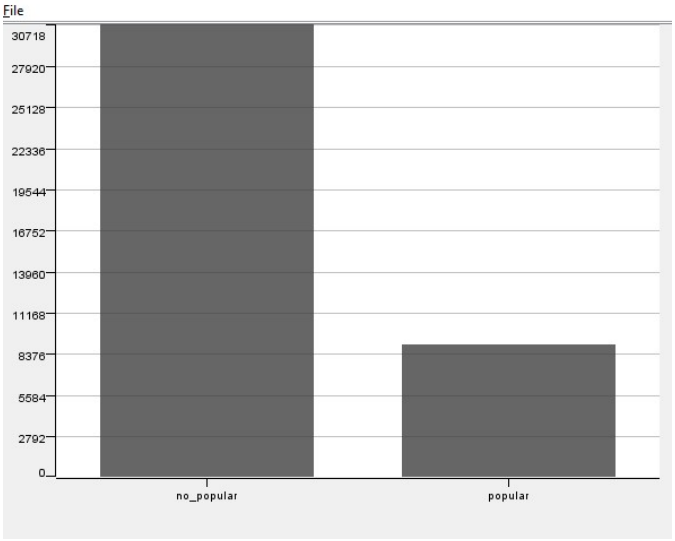


Ilustración 64 Historigrama de los datos normales

La primera fase seria considera que hacer con los valores perdidos que como muestro en la ilustración 1 seria o dejarlos sin modificar y que el algoritmo haga lo que vea convenientes con ellos. O borrar aquellas filas que contengan algún valor perdido. O tratarlos de forma que se sustituya ese valor por la media de ese atributo o en el otro caso que he puesto por el máximo.

La segunda fase consiste en que dentro de cada algoritmo antes de usarlo se le aplica un over samplin, o un under sampling, o nada.

En general quedaría como se muestra en la siguiente ilustración. Del pre-procesamiento salen 4 líneas que representan la primera fase la segunda fase se representa dentro de cada nodo como se muestra en la ilustración 7.

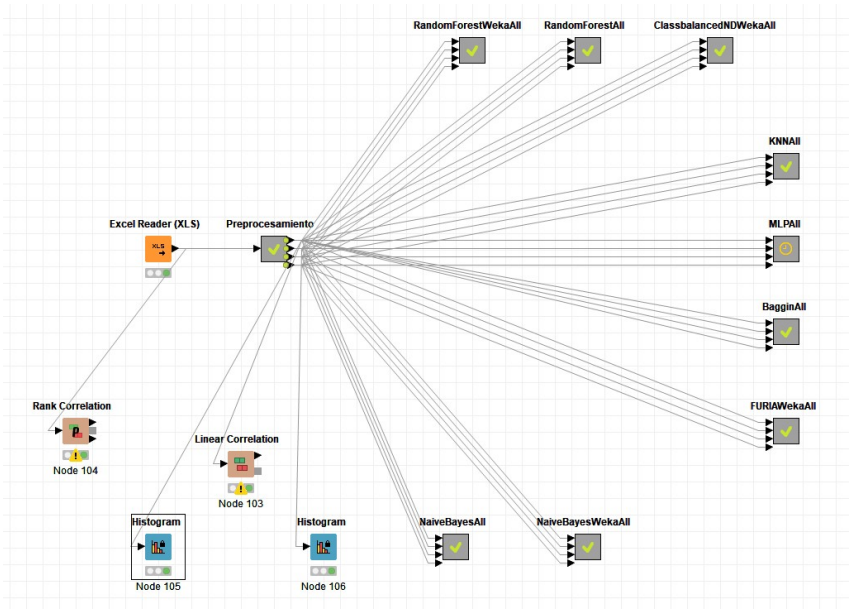


Ilustración 65 Nodo general que contiene todo

Por otro los nodos que se usan para ser el over sampling y el under sampling son, para el under sampling el nodo equal size que reduce el número hasta que estén equilibrados como se muestra en la ilustración 66. Y por otro lado para el over sampling se usan los nodos que se muestran en la ilustración 5. Con esos nodos simplemente divido en dos las filas (populares y no populares) cojo las que son menores y les aplico el bootstrap para aumentar el número de filas de la clase menor hasta igualarla. Se quedaría como en la ilustración 67. Como se puede ver en mi caso no las he puesto exactas.

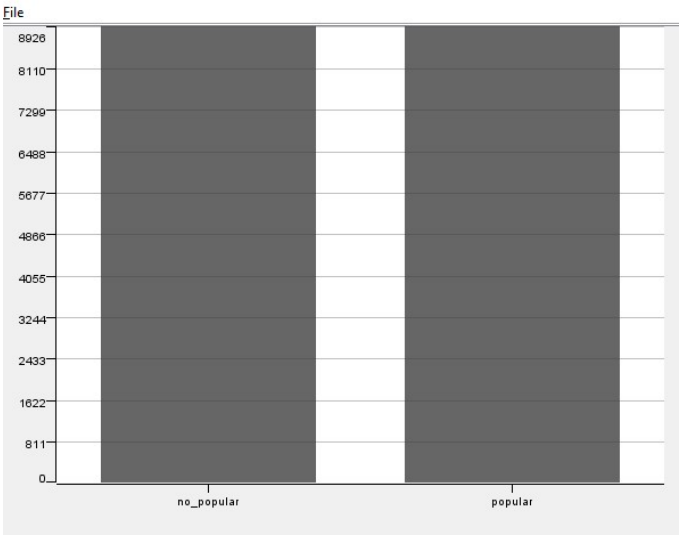


Ilustración 66 historigrama con under sampling

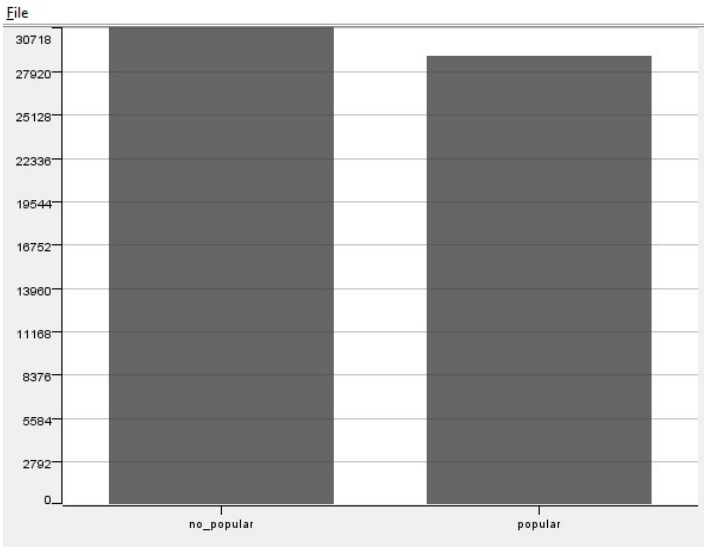


Ilustración 67 historigrama con over sampling

Como he estado explicando anterior mente hay en casos en el que el tratamiento de datos marca una gran diferencia en algunos algoritmos. Pero también hay varios casos como en el del ramdonForest que apenas se diferencia el tratamiento

de datos de no haberlos tratado. En general al haber tratado las 12 casuísticas para cada algoritmo hacen que quede claro que cada algoritmo funciona mejor con un tipo de pre-procesamiento concreto. Aunque sí que se podría decir que hay una mayor cantidad de algoritmos que han reaccionado mejor cuando los valores perdidos se han tratado como máximos.

6. Interpretación de resultados

A la vista de los resultados obtenidos podemos decir que tanto las redes neuronales como los arboles son buenos algoritmos para resolver este problema. El que mejor comportamiento ha tenido ha sido el randomforest seguido muy de cerca del MLP.

También se puede observar que los algoritmos de weka no han dado resultados mejores que sus respectivos algoritmos en KNIME. Es algo extraño ya que tanto en el randomforest como en el navie bayes hay diferencias significativas.

Mencionar que como era de esperar los algoritmos que trabajan mejor con multiclases no ha dado buenos resultados.

Si se podría destacar que en varios casos los hay algoritmos que mejoran mucho cuando procesas los datos antes de usarlos.

7. Referencias

[1] https://www.stat.berkeley.edu/~breiman/randomforest2001.pdf

[2] https://www.cs.waikato.ac.nz/ml/publications/2005/dong_et_al_cr.pdf

[3] https://www.stat.berkeley.edu/~breiman/bagging.pdf

[4] https://dl.acm.org/citation.cfm?id=1657091

[5] https://en.wikipedia.org/wiki/Naive_Bayes_classifier