Inteligencia de negocio(2018-2019)
Grado en Ingeniería Informática
Universidad de Granada

Memoria de la práctica 1

David Peinado Perea
Grupo de prácticas 1
davidpeinado@correo.ugr.es

Índice

1 Introducción

En esta práctica se estudiarán una serie de algoritmos de clasificación de aprendizaje supervisado para la resolución, visualización y análisis experimental de problemas de clasificación y preprocesado básico.

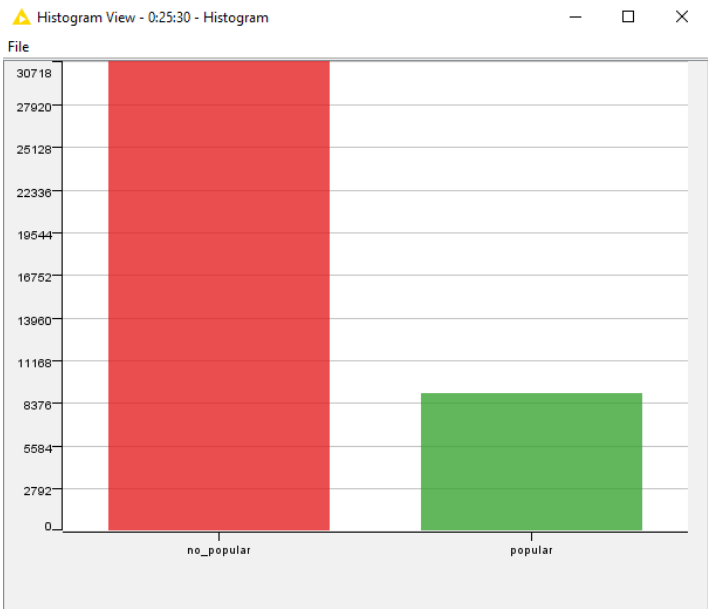
Para hacerlo posible, se hace uso de la herramienta KNIME, además se cuenta con la ayuda de la herramienta Weka, de la que se hará uso en ciertos algoritmos o JavaScript views, utilizada para la visualización de datos.

El método seguido en esta práctica para la comprobación será la validación cruzada de 5 particiones con una semilla fija. Para analizar los resultados se usarán tablas de errores, matrices de confusión, curvas ROC, medidas como el TPR, TNR, FPR, FNR, AUC, Valor-F1(En este caso será F-measure, calculado por KNIME y F1, calculado por mi siguiendo los apuntes de la asignatura, ambas son lo mismo), G-mean...

El conjunto de datos sobre el que se trabajará será onlinenewspopularity.xlsx, que se encuentra en la página de la asignatura, este conjunto es producto de ciertas transformaciones llevadas a cabo por el profesor J.Casillas sobre el original en http://archive.ics.uci.edu/ml/datasets/Online+News+Popularity

Este conjunto de datos estudia la posibilidad de que una noticia, dados unos determinados factores sobre ella, se convierta en popular o no. Se considerará que es popular si tiene mas de 1400 interacciones sociales.
Para ello se han elaborado 59 atributos, 58 de ellos predictivos y 1 objetivo.

El total de instancias del conjunto de datos es de 39644 y el balanceo de clases es de 30718 noticias no populares y 8926 populares. A continuación se muestra el número de instancias de cada clase:

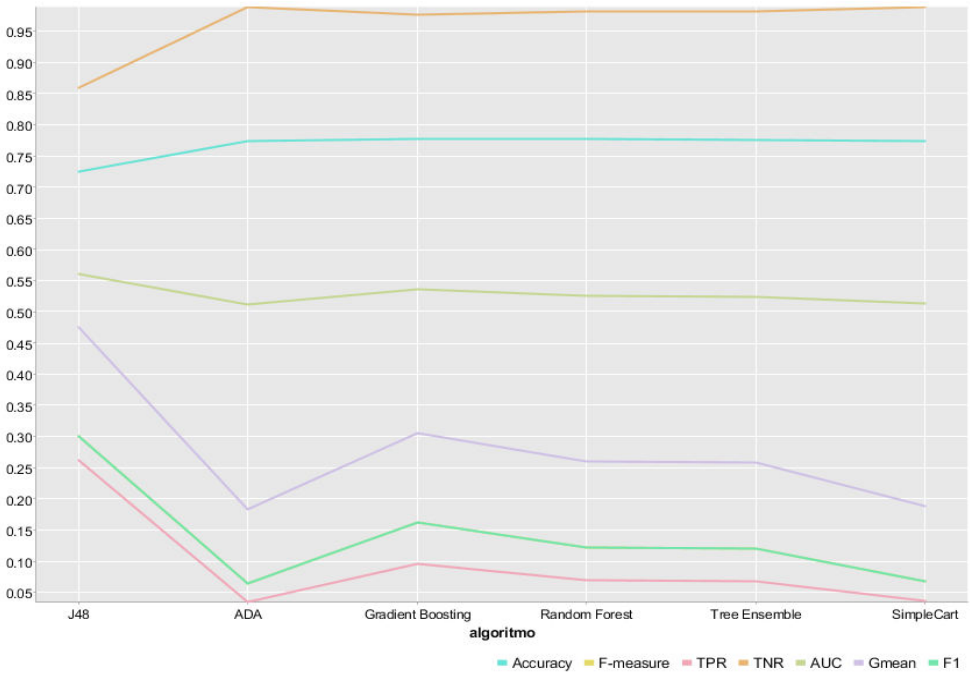


Lo que se pretende hacer es ver si una noticia va a ser popular o no, así que el TPR nos será muy útil en este caso, junto con el TNR.

Nos interesa que el TPR sea lo más alto posible, para tener certeza de que cuando un algoritmo nos diga que una noticia es popular, realmente vaya a ser popular.
El TNR, aunque también juega un papel importante, pasa a segundo lugar, ya que al tener muchas mas instancias la clase no popular, los algoritmos difícilmente trabajen mal esta medida.

2 Resultados obtenidos

Los resultados obtenidos con los algoritmos con configuraciones por defecto y con los datos sin modificar han sido estos:



2.1. J48

El algoritmo C4.5 para construir árboles de decisión se implementa en Weka como un clasificador llamado J48. Este algoritmo genera un árbol de decisión mediante el concepto de entropía de información. El algoritmo considera todas las decisiones posibles que pueden dividir el conjunto de datos y selecciona la decisión que aporta mayor ganancia de información para construir el árbol. Permite el manejo de datos con valores desconocidos, ignorándolos, permite trabajar con valores continuos y se basa en el criterio de proporción de ganancia evitando que las variables con mayor numero de categorías se beneficien.

Como se puede comprobar, sin ninguna parametrización este algoritmo nos da unos resultados bastante distintos al resto de algoritmos.
Tiene el mejor resultado de todos los algoritmos probados en la predicción de verdaderos positivos, cosa realmente interesante en este caso concreto.

Si bien es cierto que el TPR es “sólo” del 0.25 aproximadamente, es con diferencia el mayor de este conjunto. Aunque no sea altamente eficaz y determinante, esto lo hace el mejor de este conjunto de algoritmos ya que es el que va a predecir con mayor exactitud si una noticia va a ser popular.
Aunque sólo lo haga 1 de cada 4 veces, su ratio de acierto es muchísimo mayor que el del resto(ratios de 1 a 10 o 1 a 20).

Bien es cierto que no todo son ventajas, el contrapeso de este algoritmo es su menor TNR y su menor accuracy con respecto a sus competidores.

Por lo tanto, si lo que queremos es predecir si una noticia va a ser popular, este algoritmo es el que deberíamos utilizar, que tendrá una probabilidad de acierto algo

superior al 25%.

El AUC es el mayor de todos los algoritmos probados en este análisis.

2.2. AdaBoost

Su nombre viene de Adaptative Boosting, es un algoritmo que combina la salida de otros algoritmos de aprendizaje débil o weak learning, mediante una suma ponderada. Es sensible al ruido y a valores atípicos. A su favor cuenta la gran capacidad de adaptabilidad ya que algunos de los algoritmos que combina pueden ser mejores en distintos tipos de problemas, creando en su conjunto un modelo rápido con resultados decentes. Se ha utilizado la implementación Real AdaBoost de Weka.

Este algoritmo obtiene un TNR y un accuracy mayor que J48 pero su TPR y sus medidas asociadas a éste descienden.
El AUC es inferior al anterior. (Ya se hablara sobre el AUC más adelante, ya que presenta problemas).

2.3. Gradient Boosting

Es un algoritmo que produce un modelo de predicción según un conjunto de modelos de predicción débiles. La construcción del modelo se realiza de manera escalonada, mejora el modelo minimizando el error de dicha función.
Se comporta bien en espacios de varias dimensiones ya que no es susceptible a las dependencias e interacciones lineales entre los atributos. Requiere de una correcta parametrización para un buen funcionamiento y presenta cierta tendencia al sobreentrenamiento.

Este algoritmo, pese a ser el mejor según la curva ROC comparativa que luego veremos, desciende en TPR frente a J48, aunque aumenta frente a AdaBoost.

2.4. Random Forest

Es un algoritmo que genera una gran cantidad de árboles(bosque) de decisión y clasifica el dato según la clasificación que aparezca en el mayor número de árboles. Tiene una excelente capacidad de evitar el sobreentrenamiento, es muy eficiente en grandes conjuntos de datos con gran cantidad de atributos y aporta información sobre la topología de los datos que puede servir de ayuda a la hora de preprocesar este conjunto de datos, como la importancia de los distintos atributos o el error de generalización.

Obtiene menor TPR que J48 y Gradient boosting, aunque su accuracy, TNR y AUC son muy similares al anterior. Obtiene mejores resultados que AdaBoost.

2.5. Tree Ensemble

Aprende un conjunto de árboles de decisión (como variantes de bosque aleatorias). Cada uno de los modelos del árbol de decisión se aprende en un conjunto diferente de filas y/o un conjunto diferente de columnas. El modelo de salida describe un conjunto de modelos de árbol de decisión y se aplica en el nodo de predictor correspondiente utilizando el modo de agregación seleccionado para agregar los votos de los árboles de decisión individuales.

Este algoritmo presenta resultados casi iguales al algoritmo anterior, se puede ver en la gráfica que entre ambos las pendientes de las rectas son casi paralelas al eje X.

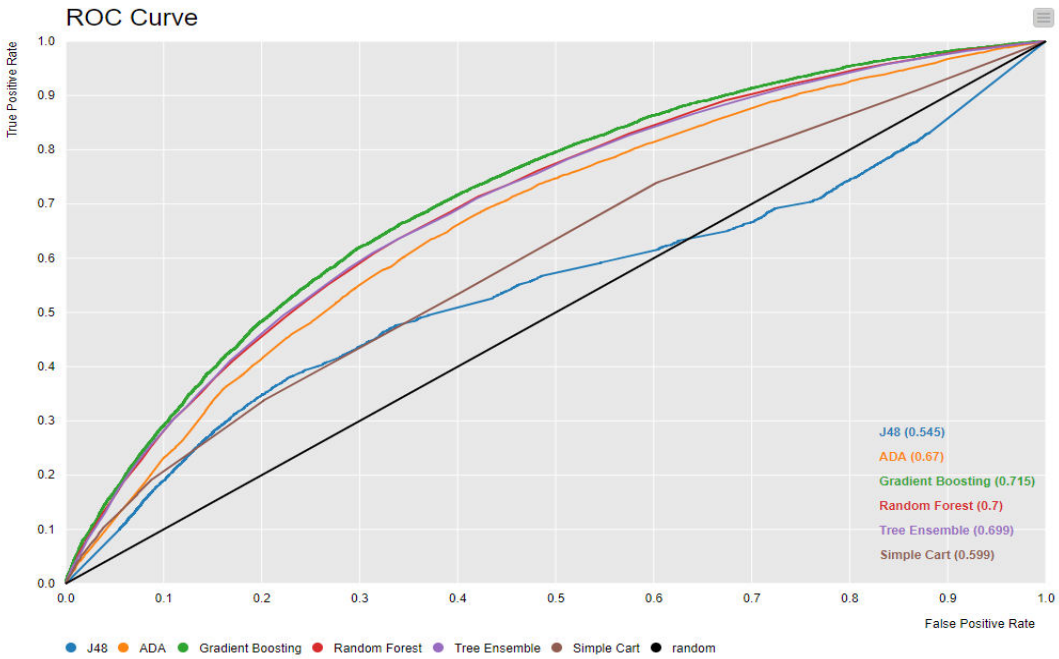
2.6. SimpleCart

Este algoritmo es una versión de CART(árboles de clasificación y regresión). Capaz de procesar atributos continuos y nominales como objetivos y predictores, este algoritmo maneja los datos raw, no se recomienda hacer binning con este algoritmo. Comenzando en el nodo raíz, los datos se dividen en dos hijos, y cada uno de los hijos se divide a su vez en nietos. Los árboles se dejan crecer a un tamaño máximo sin el uso de una regla de detención; el proceso de crecimiento de árboles se detiene cuando no se puede dividir más debido a la falta de datos. El árbol de tamaño máximo luego se poda de nuevo a la raíz a través de un método de poda de complejidad de costos. El mecanismo CART está destinado a producir no un árbol, sino una secuencia de árboles podados anidados, cada uno de los cuales es un candidato para ser el árbol óptimo. El árbol de "tamaño correcto" se identifica mediante la evaluación del rendimiento predictivo de cada árbol en la secuencia de poda en datos de prueba independientes. El mecanismo CART incluye (opcional) el equilibrio automático de clases y el manejo automático del valor perdido, y permite el aprendizaje sensible al coste, la construcción dinámica de características y la estimación del árbol de probabilidad.

Este algoritmo empeora el TPR hasta quedarse a nivel de AdaBoost, mientras el AUC desciende levemente comparado con el algoritmo anterior y su TNR se eleva escasamente.

3 Análisis de resultados

Comparando la imagen anterior con esta otra de las curvas ROC de cada algoritmo:



Se observa claramente como ni la medida AUC ni el accuracy corresponden con lo mostrado en esta gráfica.
Ésta última nos enseña que Gradient Boosting sería el clasificador con mejor performance, mientras que J48 sería el peor.

No obstante hay que interpretar de forma consonante estos datos y nuestro objetivo, si queremos que nuestro clasificador sea el que mas confianza nos de a la hora de creer que una noticia marcada como popular realmente se convertirá en popular, deberíamos optar por el J48, ya que es el que mayor TPR+TRN ha demostrado tener, siendo la suma de ambos 1,121, mientras que la de Gradient Boosting, que es el que le sigue, se sitúa en un 1,072.

J48 será el que mejor predecirá si una noticia clasificada como popular realmente sea popular. Esta implementación del algoritmo C4.5 corrobora por qué este algoritmo es tan relevante, ya que en este caso ha sido el mejor de todos los analizados para la resolución de este problema.
Este algoritmo es el que mejor se adapta de todos a la clase minoritaria, siendo el que mejor TPR tiene, con diferencia sobre los demás.

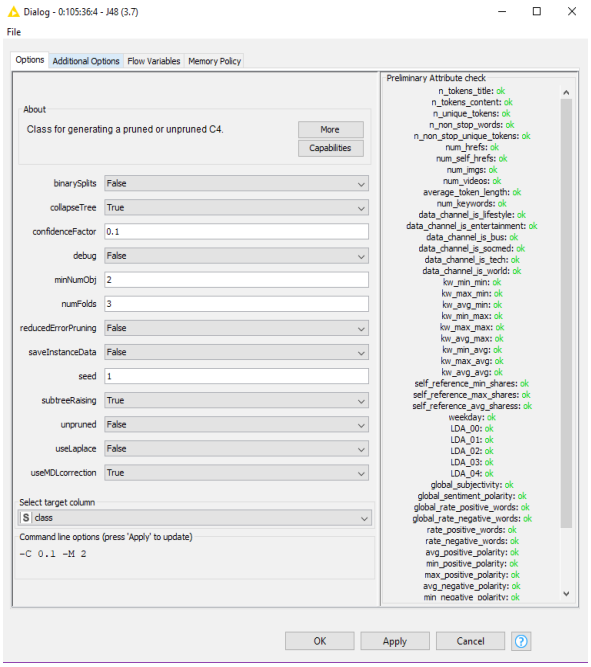
El resto de algoritmos pertenecen a la categoría de ensemble learning, se puede constatar ya que sus medidas en la primera gráfica son similares.

4 Configuración de algoritmos

En este apartado vamos a analizar el comportamiento de 3 algoritmos cuando se cambian sus parámetros, analizando sus resultados al final del mismo.

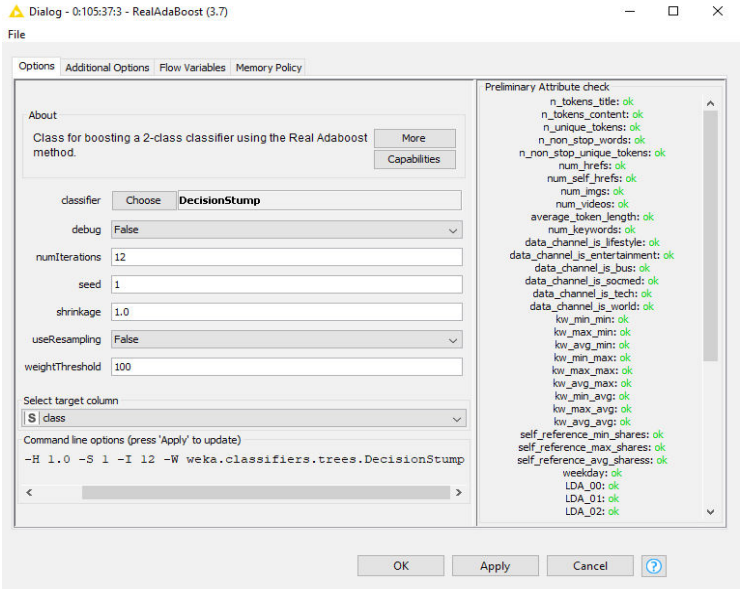
4.1 J48

En este algoritmo modificaremos el factor de confianza usado para la poda, cuanto menor el valor, mayor será la poda.
Lo cambiaremos de 0.2 a 0.1, para hacer una mayor poda.



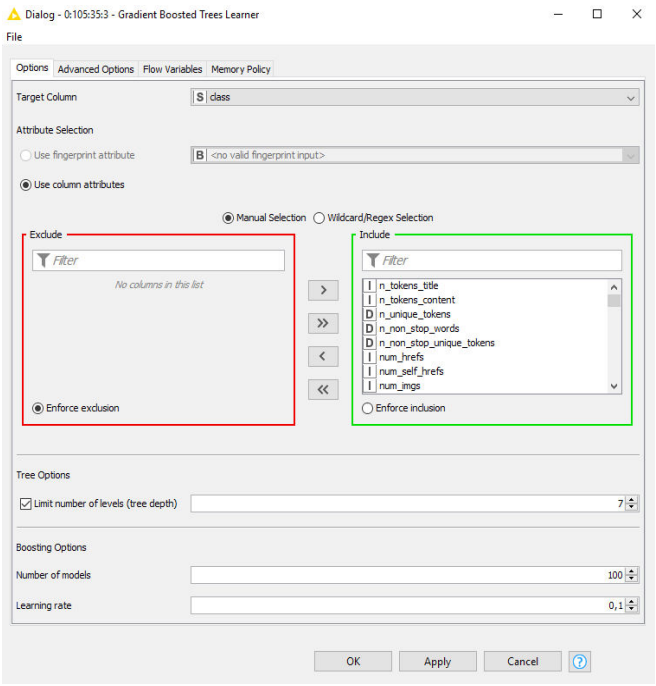
4.2 AdaBoost

Para el algoritmo Real AdaBoost, modificaremos el número de iteraciones que se harán, en este caso, con el clasificador DecisionStump.
Se aumentará el número de iteraciones de 10, por defecto, hasta 12.



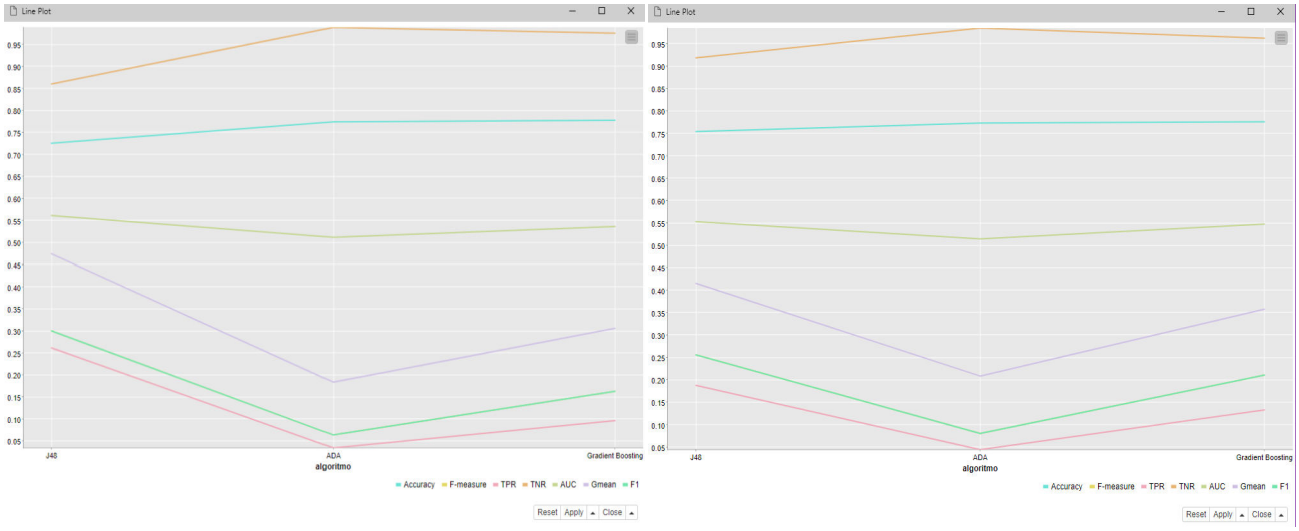
4.3 Gradient Boosting

En este algoritmo se modificará la profundidad del árbol, pasando de 4 a 7 niveles.



4.4 Resultados

La comparativa entre ambos:



Algoritmos por defecto

Algoritmos con parámetros cambiados

Se puede observar que son optimizables según que parámetros se usen, con los cambios dichos anteriormente:

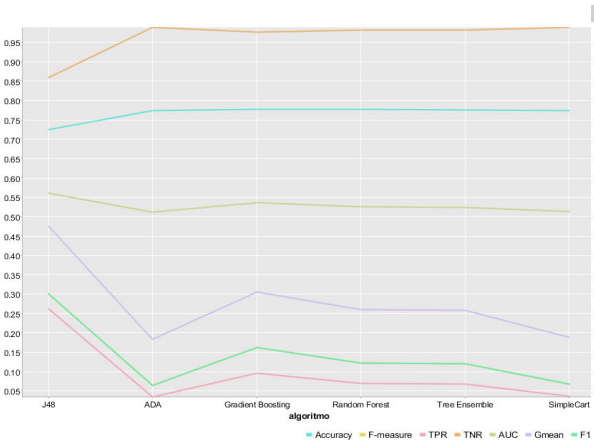
-J48 aumenta su TNR a costa de bajar su TPR, obtiene mejor precisión, ya que la poda es mayor, a costa de unos cuantos TP de menos.

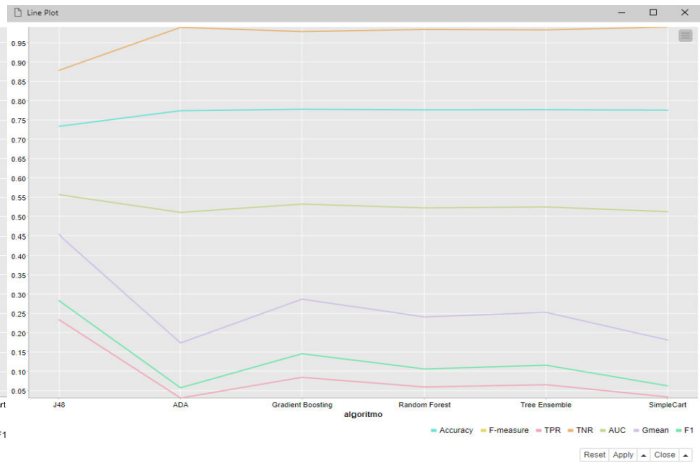
-Ada logra subir ligeramente su TPR al hacer más iteraciones, sin sacrificar TNR.

-Gradient Boosting aumenta su TPR con un descenso leve en el TNR, no obstante el cambio es a mejor, ya que el TPR sube más de lo que baja el TNR. Al tener más profundidad el árbol, se obtiene más conocimiento sobre la clase minoritaria, lo que incrementa el TPR.

5 Procesado de datos

Se quitan diversos atributos del conjunto de datos original, los resultados que se obtienen son prácticamente los mismos.





Resultado original Resultado quitando atributos

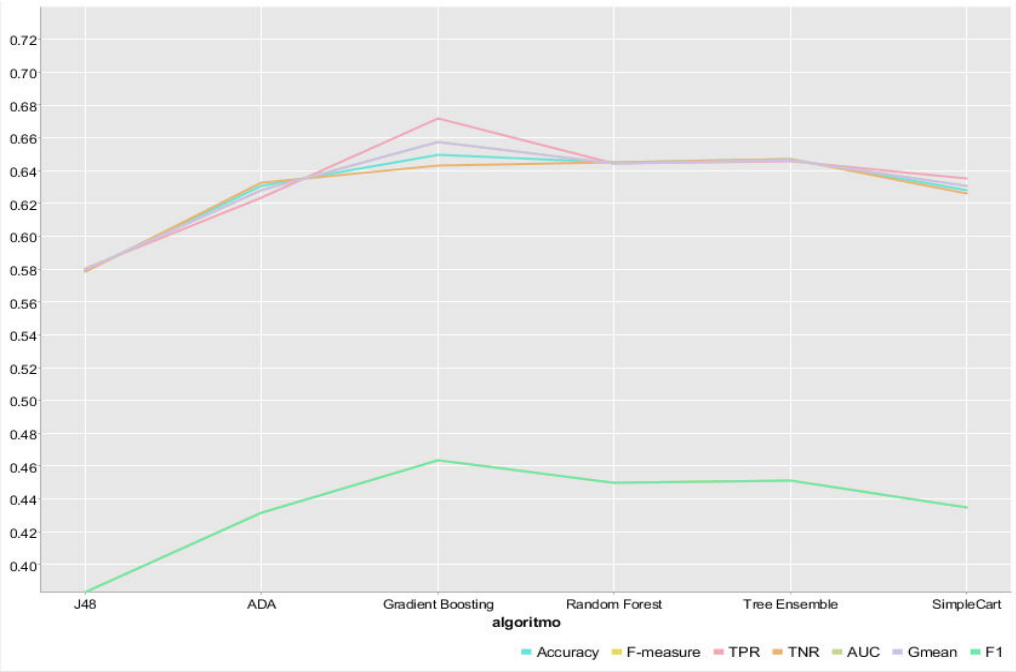
No obstante, al ser los resultados los mismos, el tiempo de computación y la interpretación de los modelos generados es mucho más facil.

Para todos los algoritmos se hace undersampling y oversampling.

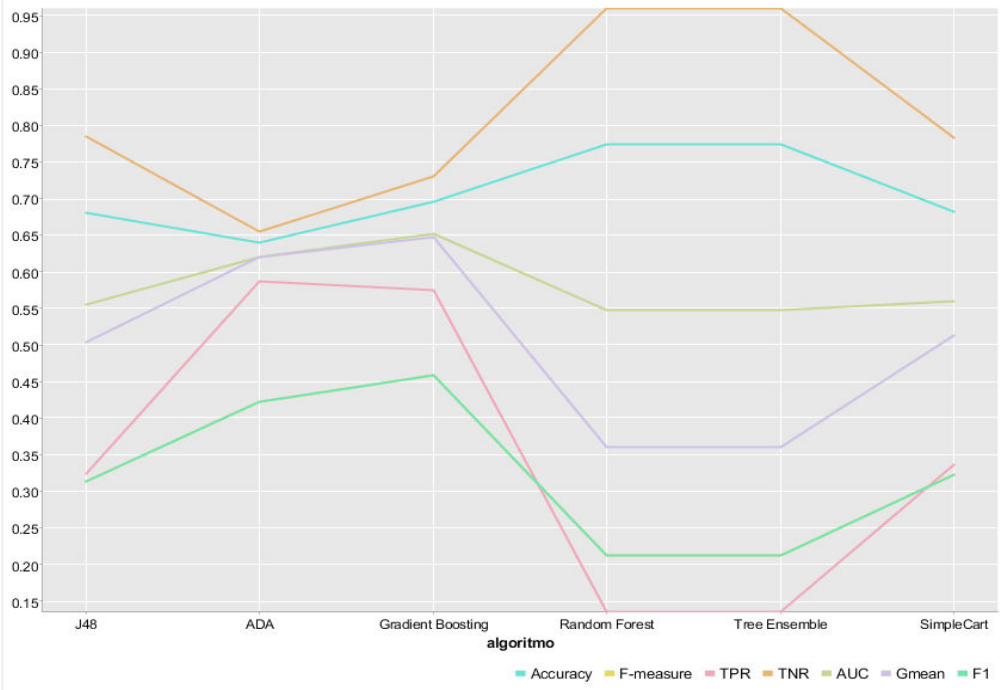
Para el undersampling se le añade a los metanodos de validación cruzada el nuevo nodo Equal Size Sampling, que nos reducirá el número de instancias de la clase mayoritaria al número de instancias de la minoritaria, aleatoriamente, y en todos los algoritmos con el mismo seed.

En el caso del oversampling, se añaden los 3 nodos necesarios para que se repitan aleatoriamente instancias de la clase minoritaria hasta, en este caso, un 300% y se añaden al conjunto de datos.

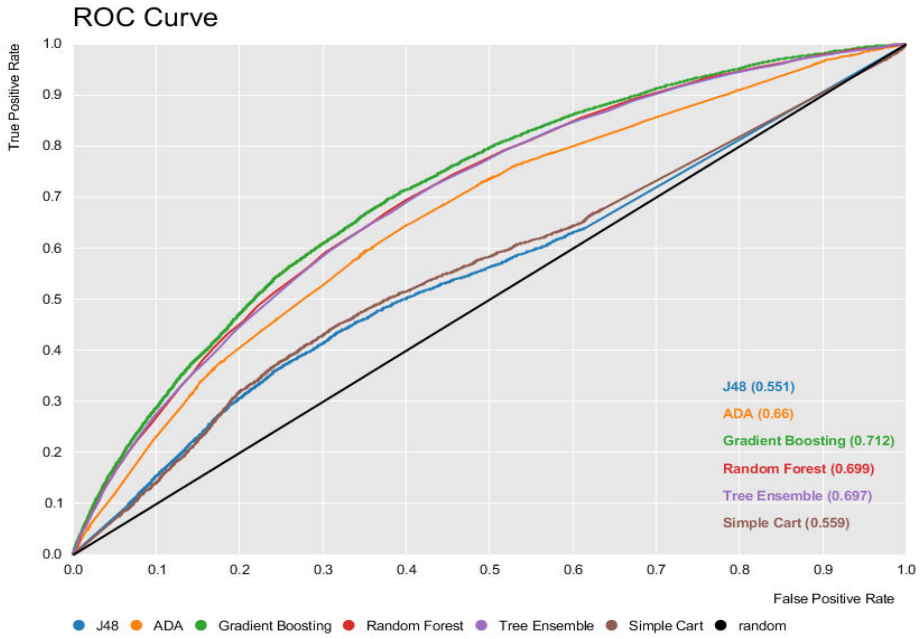
Resultados con undersampling:



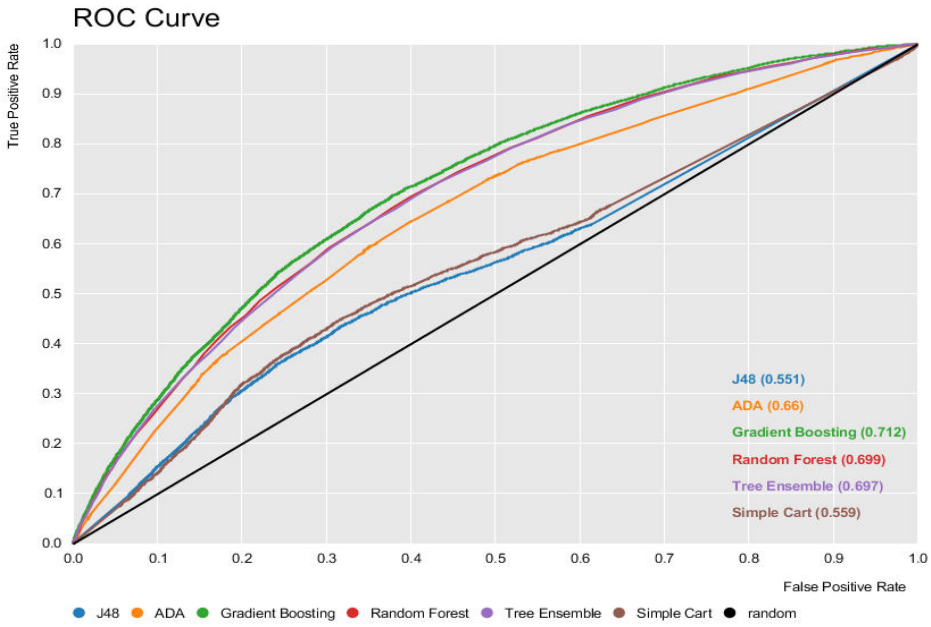
Resultados con oversampling:



Curva ROC con undersampling:



Curva ROC con oversampling:



A continuación se exponen los resultados de cada caso:

5.1 J48

En este caso, con undersampling obtiene mejor performance según la curva ROC, y todas las mediciones salvo F1 pasan a valer 0.58 aprox. El TPR al haber aumentado significativamente hace que el algoritmo valga mucho mas para predecir correctamente si una noticia se puede convertir en popular.

Con oversampling los resultados son muy satisfactorios, el TPR asciende a un 0.33, con lo que predeciría correctamente 1 de cada 3 veces cuando una noticia sera popular. Sólo se pierde un 0.05 de TNR con este ascenso del TPR, así que si nuestro

objetivo es predecir con efectividad qué noticias pueden llegar a ser populares, este algoritmo es relativamente decente, pero no el mejor, como veremos ahora.

Este algoritmo funciona bien con undersampling, al tener mayor conocimiento relativo sobre la clase positiva, y no funciona tan bien con oversampling, aunque lo hace considerablemente mejor que sin tocar el conjunto de datos.

5.2 AdaBoost

Si se le aplica undersampling, las mediciones subirán con respecto a J48, este algoritmo será mejor que el anterior, ya que su TPR será mayor.

En oversampling se comporta muy bien, aunque no tanto como con undersampling, llegando al 0.58 de TPR.

A este algoritmo le sienta muy bien tanto undersampling como oversampling en este caso concreto, con ambos mejorara una barbaridad el TPR, lo cual es el objetivo, eso si, disminuyendo, aunque en menor proporción, el TNR.

5.3 Gradient Boosting

A este algoritmo es al que mejor le sienta el undersampling, ya que presenta cierta tendencia al sobreentrenamiento.
Como se ve en la gráfica, el TPR y TNR son los mejores en undersampling.

En oversampling lo hace algo peor que Ada y mucho mejor que J48, no le sienta tan bien como a otros algoritmos pero si lo hace mejor que de normal, con respecto al TPR.

5.4 Random Forest

Para undersampling este algoritmo se porta como el resto, está en la media de resultados del conjunto.

Para oversampling se disparan los resultados con respecto a J48, Ada o Gradient, tiende a subir mucho el TNR, lo que es bueno, pero el TPR baja muchísimo, este sobreentrenamiento es malo de cara a la precisión con la que se evalúa si una noticia es popular, bajando al 14%.

Este algoritmo sin correcta parametrización lo hace muy mal con oversampling.

5.5 Tree Ensemble

Este caso es prácticamente idéntico al anterior, como se puede constatar en las gráficas, los resultados son prácticamente los mismo en Random Forest y Tree Ensemble. Son algoritmos que se comportan en estos 3 casos vistos anteriormente

con mucha similitud y casi exactitud.

5.6 SimpleCart

Este algoritmo se comporta casi igual que Ada en el caso del undersampling y muy parecido a J48 en el caso del oversampling. Si bien es cierto que el TNR es mayor en este.

6 Interpretación de los resultados

Se ha observado el problema que existe con el desbalanceo de clases, equilibrando las clases haciendo undersampling se obtienen resultados muy satisfactorios en cuanto a la confianza que podemos depositar en los algoritmos a la hora de creer que una noticia marcada como popular sea realmente popular.
Todos los algoritmos se comportan mejor con este cambio en el conjunto de datos respecto al conjunto original.
En el caso de hacer oversampling, los resultados también son superiores a los que se obtienen con el conjunto de datos original, ya que los TPR de los algoritmos aumentan considerablemente(excepto Random Forest y Tree Ensemble) y aún bajando su TNR, en algunos de manera considerable, el cambio es positivo.

Estos dos algoritmos antes mencionados, Random Forest y Tree Ensemble no se ven prácticamente afectados por el oversampling, que les viene de maravilla, ya que sin sacrificar TNR, obtienen una subida en su TPR.

Se ha visto como la preparación de los datos tiene vital importancia de cara al funcionamiento de los algoritmos y resultados obtenidos.

También se ha visto como la eliminación de diversos atributos mejora la eficiencia, la interpretación de los modelos generados y en algunos casos, los resultados.

7 Contenido adicional

8 Bibliografía

https://en.wikipedia.org/wiki/AdaBoost
https://en.wikipedia.org/wiki/C4.5_algorithm
https://en.wikipedia.org/wiki/Ensemble_learning
https://en.wikipedia.org/wiki/Gradient_boosting

https://en.wikipedia.org/wiki/Weka_(machine_learning)
https://www.researchgate.net/profile/Dan_Steinberg2/publication/265031802_Chapter_10_CART_Classification_and_Regression_Trees/links/567dcf8408ae051f9ae493fe/Chapter-10-CART-Classification-and-Regression-Trees.pdf
http://stp.lingfil.uu.se/~santinim/ml/2016/Lect_03/Lab02_DecisionTrees.pdf
https://es.wikipedia.org/wiki/Random_forest
https://sci2s.ugr.es/sites/default/files/files/Teaching/GraduatesCourses/InteligenciaDeNegocio/Curso18-19/Evaluaci%C3%B3n%20de%20Clasificadores.pdf