Big Data & Cloud Computing

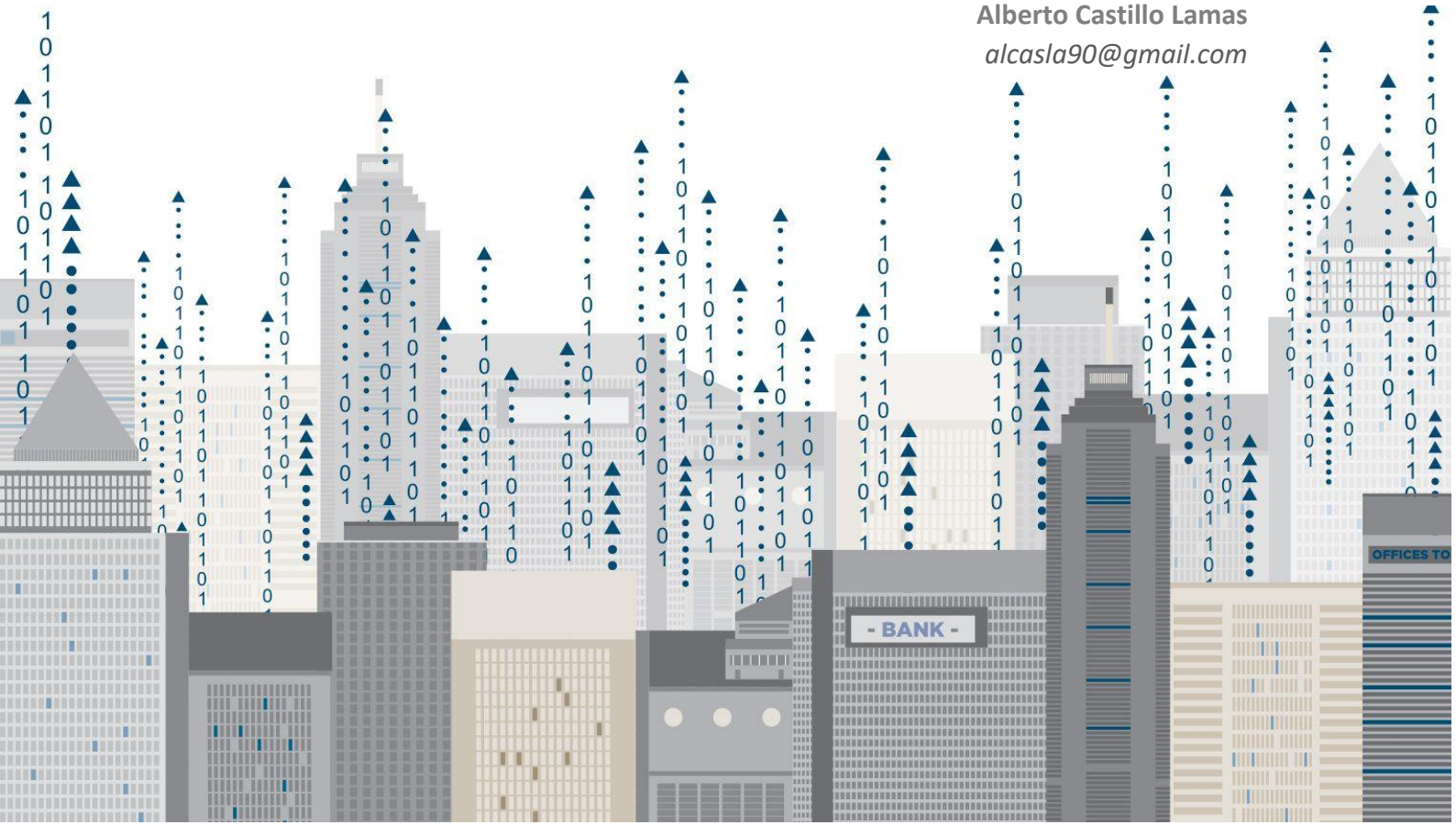
Máster en Ciencia de Datos e Ingeniería de Computadores

Universidad de Granada

5 de junio de 2017

Análisis de datos en Big Data

Alberto Castillo Lamas
alcasla90@gmail.com



1 Introducción

Experimentación realizada para afrontar un problema de predicción sobre una clase binaria en un conjunto de datos desbalanceado donde el 70% de las instancias pertenecen a una de las clases.

El problema requiere aplicar técnicas para el preprocesamiento de los datos y generación de modelos de forma distribuida, técnicas de BigData utilizando Spark sobre el sistema de archivos distribuido HDFS.

2 Experimentación base

Primero se realizar una prueba sobre los datos en bruto, sin ningún preprocesamiento, sobre los modelos **Decision Tree** y **Random Forest**. Debido a la naturaleza *random* en el segundo modelo las medidas de calidad pueden variar dependiendo de cada ejecución, por tanto durante todo el proceso de experimentación para la práctica las métricas para el algoritmo *RandomForest* serán la media de tres ejecuciones con similares configuraciones entre ellas, tabla 1.

Table 1: Medidas de calidad para cada modelo sobre una muestra del dataset ECBDL_14.

Modelo	TPR	TNR	TPRxTNR
DecisionTree	0.8935	0.4667	0.4169
RandomForest	0.9691	0.1776	0.172

La configuración de *Decision Tree*, con medida de impureza por *entropía*, árbol de profundidad máxima de *10*, y máximo número de particiones *100*. La confiuración de *RandomForest*, número de árboles *10*, medida de impureza por *gini*, máximo de profundidad *4*, y máximo número de particiones *32*.

Aunque el resultado de ambos algoritmos tiene una diferencia significativa a favor de *DecisionTree*, seguramente es debido a la desigualdad de las configuraciones, pues teóricamente un algoritmo que genera un árbol frente a otro que genera más de uno para quedarse con el mejor va a producir mejores resultados.

Si *DecisionTree* ha hecho resultados significativamente superiores a los de *RandomForest* se debe a que el árbol desarrollado aunque solo es uno éste tiene una mayor profundidad. El conjunto de datos al tener una cantidad ingente de características, cada una de ellas con un bajo impacto, hace que necesite crecer árboles de mayor tamaño. Esta hipótesis se estudia en la sección **??**.

Por el momento se va a realizar ejecuciones para *RandomForest* en condiciones distintas aproximándose a la configuración de *DecisionTree* creciendo árboles de mayor profundidad. Esto puede mostrarnos una configuración óptima del algoritmo para este problema que pueda aplicarse en la experimentación futura. Con una medida de impureza por *entropía*, máximo número de particiones *32*, y con profundidad y número de árboles variables, tabla 2.

Table 2: Medidas de calidad para *RandomForest* con distintas configuraciones sobre el conjunto en bruto de datos ECBDL_14. Media sobre tres ejecuciones.

Depth Tree	#trees	TPR	TNR	TPRxTNR
4	10	0.9691	0.1776	0.172
8	10	0.9352	0.3473	0.3257
12	10	0.917	0.421	0.386

Se observa como aumentando la profundidad de los árboles tiene en cuenta más características y su clasificación mejora progresivamente, es un aspecto a explotar más adelante.

3 Random Forest con reducción de dimensionalidad

Hasta el momento se ha trabajado con los datos en bruto, que contienen más de 600 características, pero cuando se aplique preprocesamiento sobre los datos pueden generarse modelos con unos datos de una forma distinta al aplicar una técnica para reducir la dimensionalidad del problema y su comportamiento puede variar.

Como se ha mostrado en el apartado anterior, obviamente, cuando *RandomForest* se ejecuta con árboles incluso un poco menos profundos que el *DecisionTree* obtiene mejores resultados, por tanto se va a aplicar distintas selecciones de características para encontrar el tamaño de árboles adecuado y el conjunto de datos más adecuado al problema (respecto a las características), primero reduciendo progresivamente la dimensión del problema, tabla 3, para posteriormente mantenerla y afinar el modelo basado en *RandomForest*, tomando así este como modelo de referencia en adelante.

Table 3: Medidas de calidad para *RandomForest* con mismas configuraciones sobre distintos conjuntos de datos de ECBDL_14 reduciendo su dimensionalidad. Medidas media sobre tres ejecuciones.

#Features	Depth Tree	#trees	TPR	TNR	TPRxTNR
120	8	10	0.9119	0.3939	0.359
100	8	10	0.9186	0.3803	0.3493
80	8	10	0.9118	0.3991	0.3638
60	8	10	0.9118	0.3887	0.3544
40	8	10	0.9089	0.3999	0.3634
30	8	10	0.9167	0.3814	0.3496

En la tabla 3 se observa como se reduce la dimensión del dataset y no se produce una merma de la calidad del modelo, pero si podría producirse un sobre-ajuste y prefiero mantener una cantidad de variables suficientes para que el modelo generalice adecuadamente. Se selecciona la configuración de selección de características con *80*, ya de por si reduce suficiente la dimensión a menos de un 15% de la original.

A continuación utilizando este subconjunto de *80 características* se va a afinar un modelo *RandomForest* como muestra la tabla 4. Así, como se menciona anteriormente, el modelo no se ve perjudicado por el cambio en la forma de los datos.

Table 4: Medidas de calidad para *RandomForest* con mismas configuraciones sobre distintos conjuntos de datos de ECBDL_14 reduciendo su dimensionalidad. Medidas media sobre tres ejecuciones.

#Features	Depth Tree	#trees	TPR	TNR	TPRxTNR
80	8	10	0.9118	0.3991	0.3638
	10	10	0.9153	0.4056	0.3714
	12	10	0.9079	0.4298	0.3902
	14	10	0.9047	0.4444	0.402
	16	10	0.8997	0.4591	0.413
	18	10	0.8936	0.476	0.4252
	20	10	0.886	0.4878	0.4322

Se observa como aumentando la profundidad de los árboles el modelo mejora progresivamente, en este caso no voy a utilizar árboles con mayor profundidad porque se considera que utilizando un 25% de las características es suficiente para la clasificación. Además durante la experimentación realizada con *RandomForest* se utiliza como medida de impureza la *entropía*, junto a los parámetros especificados en las tablas 3 y 4.

4 No desbalanceo. RUS o ROS

Una vez se ha seleccionado el modelo *RandomForest* y obtenido una configuración óptima para el conjunto de datos con las 80 características más importantes se procede a preprocesar el conjunto de datos mediante técnicas de manipulación de instancias pero solo en cuestión de presencia, así durante esta sección se estudia como se comporta el modelo ante la reducción y aumento de instancias para potenciar la clasificación de la clase minoritaria igualando las instancias presentes en el dataset ya sea por eliminación de instancias de la clase mayoritaria o inserción de instancias duplicadas de la clase minoritaria. En ambos casos el porcentaje final de instancias de ambas clases es el mismo, un 50% de cada clase, pero varía en número.

Table 5: Medidas de calidad para *RandomForest* aplicando ROS y RUS, equilibrando las clases a un 50% de instancias, sobre el conjuntos de datos de ECBDL_14 reduciendo su dimensionalidad con las 80 características de mayor importancia. Medidas media sobre tres ejecuciones.

Modelo	Técnica	TPR	TNR	TPRxTNR
Random Forest	RUS	0.6802	0.7562	0.5144
Random Forest	ROS	0.7533	0.6919	0.5212

La configuración utilizada en todas las ejecuciones de la tabla 5 se realizan con similar configuración del modelo *RandomForest*, con medida de impureza por entropía, número de árboles 10 y profundidad de estos 20.

Los resultados muestran como igualando ambas clases del problema el modelo mejora más al insertar instancias existentes, por tanto repitiendo información en algunos casos, en vez de eliminando instancias de la clase mayoritaria. Por tanto *Random Oversampling* es preferible para seguir aplicándolo como medio para obtener mejores resultados.

5 ROS

Se ha demostrado como esta técnica de preprocesamiento se ajusta mejor a este problema. En esta sección a través de esta técnica se muestra la experimentación levada a cabo para obtener el mejor resultado posible.

Si se observa la tabla 5 los resultados nos muestran una información fundamental y diferencial entre ambas técnicas, cada una hace que una de las clases se prediga con un ajuste del 75%. Si se piensa en la forma de funcionar de cada técnica, RUS y ROS, la primera solo puede optar a eliminar más información y por tanto deteriorar el modelo a partir de lo que ya ha hecho con un equilibrio de las dos clases. Pero en el caso de ROS se puede mejorar el resultado de la clase minoritaria sin perjudicar a la otra haciendo así que el estadístico *TPRxTNR* suba.

En la tabla 6 se experimenta con distintas configuraciones de ROS con las que se pretende mejorar los resultados de la clase minoritaria, y compararlos. Ambas salidas de ambas técnicas de preprocesamiento apoyadas en un modelo *RandomForest* con una configuración similar a la utilizada en la sección 4.

Table 6: Medidas de calidad para *RandomForest* aplicando distintas configuraciones ROS sobre el conjuntos de datos de ECBDL_14 reduciendo su dimensionalidad con las 80 características de mayor importancia. Medidas media sobre tres ejecuciones.

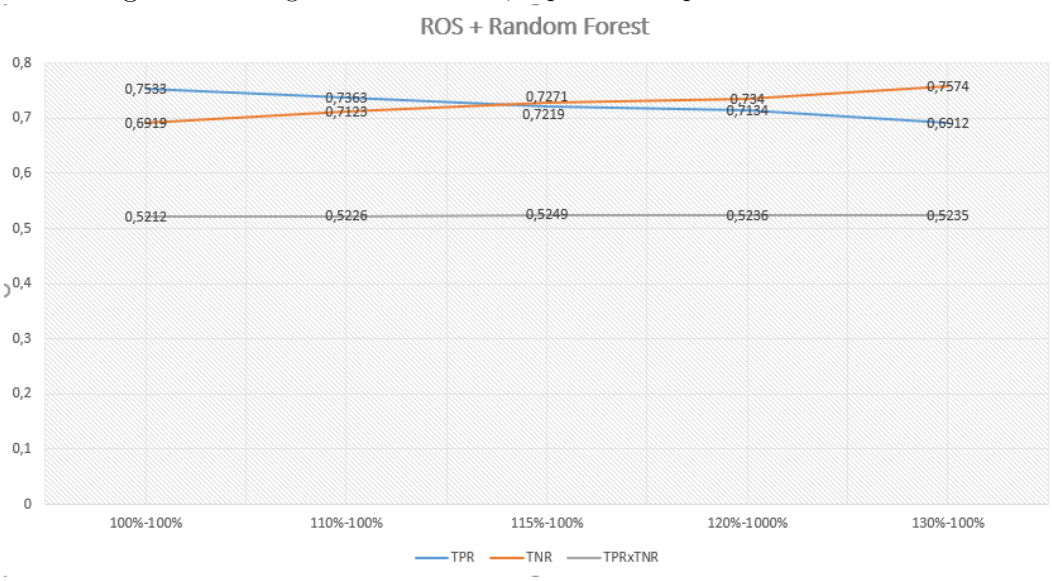
% Clase minoritaria	% Clase mayoritaria	TPR	TNR	TPRxTNR
100%	100%	0.7533	0.6919	0.5212
110%	100%	0.7363	0.7123	0.5226
120%	100%	0.7134	0.734	0.5236
130%	100%	0.6912	0.7574	0.5235
115%	**100%**	0.7219	0.7271	0.5249

Los porcentajes de cada clase se toman con referencia a la clase mayoritaria. Se establece el número

de instancias de la clase mayoritaria como el 100%, se aumenta el número de instancias incrementalmente de la clase minoritaria hasta igualar y superar el número de instancias de la mayoritaria.

Si comparamos con los resultados iniciales, sin preprocesamiento (tabla 4), y la progresión de la tabla 6 con procesamiento, se observa como con cada incremento del número de instancias de la clase minoritaria el clasificador mejora para esta y empeora para la clase mayoritaria. Esto es un beneficio al medir ratios de las clases, en particular *TPRxTNR* con el equilibrio de en el ratio de ambas clases.

Figure 1: Configuraciones de ROS, experimentos plasmados en la tabla 6.



No hay diferencias significativas en *TPRxTNR* pero si obtenemos el mejor resultado cuando se equilibran y cruzan las tendencias de *TPR* y *TNR*.

Finalmente mencionar que se obtienen unos mejores resultados, algo superiores al 0.54, sin realizar selección de características. Esto se debe a la reducción del tamaño del dataset por facilidad para trabajar sobre la práctica, por tanto en la experimentación mantengo el proceso de *selección de características* simulando el entorno real en el que se desarrollaría este problema.