UNIVERSIDAD DE GRANADA

Big Data II

máster ciencia de datos e ingeniería de computadores

Práctica

Análisis de datos en Big Data

Autor
Ignacio Vellido Expósito
ignaciove@correo.ugr.es

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Curso 2020-2021

Índice

1. Introducción

1.1. Conjunto de datos

Para esta práctica tenemos un subconjunto de SUSY Data Set (`https://archive.ics.uci.edu/ml/datasets/SUSY`), un problema de clasificación binaria donde existe un ratio de desbalanceo de 10-90. La tarea consiste en distinguir la señal que produce una partícula supersimétrica frente a una posible señal de fondo que se puede captar.

El dataset cuenta con dos millones de datos (1.000.000 de entrenamiento, 1.000.000 de test) con 18 características numéricas reales, y las siguientes medidas estadísticas:

	Train				**Test**			
Columna	**Max**	**Min**	**Mean**	**Variance**	**Max**	**Min**	**Mean**	**Variance**
1	16.89	0.25	1.23	0.62	15.65	0.25	1.23	0.62
2	2.10	-2.10	-9.92e-4	0.79	2.10	-2.10	-4.19e-4	0.79
3	1.73	-1.73	-5.97e-4	1.00	1.73	-1.73	0.00	1.00
4	17.73	0.42	1.11	0.53	18.18	0.42	1.11	0.53
5	2.05	-2.05	-3.46e-4	0.83	2.05	-2.05	7.25e-5	0.83
6	1.73	-1.73	6.71	1.00	1.73	-1.73	0.00	1.00
7	21.00	9.42	1.34	1.14	21.00	7.19	1.33	1.14
8	1.74	-1.72	0.00	1.00	1.74	-1.72	0.00	1.00
9	23.38	7.69	1.22	1.16	22.56	9.23	1.22	1.16
10	16.93	-15.33	0.06	1.73	17.69	-13.10	0.06	1.74
11	14.93	0.26	1.14	0.43	15.73	0.26	1.14	0.43
12	14.36	0.00	1.21	0.45	14.99	0.00	1.21	0.45
13	5.81	0.00	1.04	0.23	6.03	0.00	1.04	0.23
14	20.68	0.00	1.05	0.92	14.57	0.00	1.06	0.92
15	14.89	0.05	1.14	0.42	15.78	0.05	1.14	0.42
16	15.61	0.00	1.15	0.47	11.33	0.00	1.15	0.47
17	1.59	8.22	1.01	0.18	1.59	2.57	1.01	0.18
18	1.0	3.52	0.27	0.04	1.0	5.89	0.27	0.04

Cuadro 1: Medidas estadísticas del conjunto de datos.

Según la descripción del dataset en la página de la UCI, las primeras 8 características reflejan propiedades de las partículas medidas en un acelerador, y las 10 siguientes indican el resultado de diferentes funciones a partir de estas variables. Estas variables derivadas no aportan información nueva pero se indica que pueden resultar de ayuda en la clasificación de la instancia.

Tal y como vemos en la Tabla 1, las columnas del conjunto de datos se distribuyen en rangos diferentes, aunque de manera similar entre entrenamiento y test. Por ello, normalizaremos los datos antes de pasarlos por los algoritmos, haciendo uso del conjunto de funciones de KeelParser.

1.2. Técnicas aplicadas

Las diferentes técnicas aplicadas en esta práctica son:

- **De aprendizaje**:
  - Árboles de decisión (MLlib.tree.DecisionTree).
  - Random Forest (MLlib.tree.RandomForest).
  - PCARD (MLlib.tree.PCARD).
  - kNN-IS (MLlib.classification.kNN_IS).
- **De preprocesamiento**:
  - **Selección de características**:
    - Principal Component Analysis.
    - Chi-Square Selector.
  - **Ajuste de desbalanceo**:
    - Random Oversampling.
    - Random Undersampling.
  - **Filtrado de ruido**:
    - Homogeneous Ensemble (HME).
    - NCNEdit.
  - **Selección de instancias**:
    - FCNN.
    - SSMA-SFLSDE.

La metodología seguida durante la práctica comienza comparando el mayor número de flujos de preprocesamiento posibles, no por ello dejando de reflexionar sobre la coherencia en el uso conjunto de algunas técnicas. Por ejemplo, puesto que el algoritmo PCARD aplica a los datos de entrada PCA, no se ha utilizado ninguna técnica de reducción de características en sus experimentos.

Para hacer esta exploración de flujos los algoritmos fueron entrenados con unos parámetros por defecto. Finalmente, a partir de los mejores resultados obtenidos para cada método se realizó una optimización de los hiperparámetros para alcanzar el mayor valor de TPR x TNR posible.

El flujo de técnicas de preprocesamiento ordenado de acorde a su uso es el siguiente:

1. Selección de características
2. Under|Over-sampling
3. Filtrado de ruido
4. Selección de instancias

La justificación es la siguiente: En base al conjunto de datos de entrenamiento con el que contamos pretendemos reducir la dimensionalidad sin perder excesiva información. Puesto que el dataset cuenta con un ratio de desbalanceo del 90 %, ROS y RUS ajustan los datos para evitar un sesgo en las técnicas de clasificación. Si quitamos características podemos acabar con datos redundantes en el dataset, y el uso de ROS puede generar ruido adicional además del propio ruido inherente que debemos suponer que existe en nuestros datos. Por estos motivos aplicamos técnicas de filtrado de ruido y selección de instancias para reducir el conjunto de datos y, adicionalmente, agilizar el proceso de aprendizaje.

A continuación se indican los parámetros utilizados en cada una de las técnicas:

- **De aprendizaje**:
  - **Árboles de decisión**: Se entrenan árboles con medida GINI, máxima profundidad entre 5 y 8 y número de particiones a 32.
  - **Random Forest**: De igual manera, los árboles se entrenan con medida GINI, máxima profundidad entre 5 y 8 y número de particiones a 32. Se limita el número máximo de árboles entre 100 y 150.
  - **PCARD**: El número de cortes se fija a 5, y el número de árboles entre 10 y 15.
  - **kNN-IS**: Utilizando distancia euclídea, movemos el valor de k entre 5 y 7 y el de particiones a 10.
- **De preprocesamiento**:
  - **Selección de características**:
    - **Principal Component Analysis**: Reducción al 50 % (9 características).
    - **Chi-Square Selector**: Reducción al 50 % (9 características) tras una discretización en 25 intervalos.
  - **Ajuste de desbalanceo**:
    - **Random Oversampling**: Incremento del 50 %.
    - **Random Undersampling**: Decremento hasta alcanzar igualdad en el número de instancias de cada clase.
  - **Filtrado de ruido**:
    - **Homogeneous Ensemble (HME)**: Número de árboles fijado a 100, con máxima profundidad de 10 y 4 particiones.
    - **NCNEdit**: Se consideran los 3 vecinos más cercanos.
  - **Selección de instancias**:
    - **FCNN**: Se consideran los 3 vecinos más cercanos.
    - **SSMA-SFLSDE**.

2. Análisis de resultados

En esta sección se pretenden analizar aquellos resultados más relevantes. Algunos argumentos también se sustentan sobre las tablas completas de experimentos (con información adicional sobre ellos) que se encuentran en la sección *Tablas de resultados*.

Algoritmo	**Selección de características**	**Under/Over sampling**	**Filtrado de ruido**	**Selección de instancias**	**TPR x TNR**
Decision Tree	No	RUS	HME	FCNN	0.606
Random Forest	No	RUS	HME	FCNN	0.607
PCARD	-	RUS	No	No	0.598
kNN-IS	No	RUS	HME	No	0.526

Cuadro 2: Flujos de preprocesamiento para los mejores resultados de cada algoritmo tras la optimización de parámetros.

Todos los valores de TPR x TNR mostrados son calculados sobre el conjunto de test.

2.1. Sobre las técnicas de aprendizaje

En términos de los algoritmos de clasificación, tal y como se muestra en la tabla 2, obtenemos prácticamente la misma calidad con cualquiera de ellos (con variación de milésimas), siendo ligeramente superiores Random Forest (RF) y Árboles de Decisión (DT). Para ambas técnicas el flujo de preprocesamiento coincide, con el que se reduce el tamaño del conjunto de datos a un 10 % del tamaño original.

Mediante las tablas 4 y 6 vemos que independientemente del preprocesamiento los resultados siguen siendo muy similares para las dos técnicas, probablemente debido al estar un algoritmo formado como ensamblado del otro. A pesar de esto, vemos que en media RF es más robusto con RUS mientras que DT funciona mejor con ROS.

Sobre PCARD, aunque la calidad máxima se alcanza con las mínimas técnicas de preprocesamiento (ajustar el desbalanceo es imprescindible en este problema, y los resultados lo demuestran) no llega a ser significativamente inferior que el resto.

Además, a partir de la tabla 7 notamos que las técnicas NCNEdit y FCNN empeoran los resultados frente a no usar ninguna reducción de instancias. Creemos que el motivo reside en que al aplicarse PCA antes de entrenar los árboles se acaba perdiendo demasiada información.

Respecto a kNN, notamos peor calidad en comparación con el resto independientemente de la técnica y parámetros con los que se ha probado. Sin hacer un análisis de la distribución de los datos el razonamiento no está claro, pues al ser un algoritmo basado en distancias si existiera alto entremezclado entre las instancias de ambas clases podría ser de esperar que con los pocos valores de k que se han utilizado sea insuficiente.

		Average	STD	Max
Filtrado de ruido	No	0.282	0.234	0.565
	HME	0.339	0.226	0.575
	NCNEdit	0.273	0.249	0.564
Selección de instancias	No	0.321	0.244	0.575
	FCNN	0.277	0.219	0.574
Selección de características	No	0.328	0.214	0.593
	PCA	0.252	0.225	0.584
Balanceo de datos	No	0.070	0.092	0.208
	ROS	0.415	0.229	0.521
	RUS	0.373	0.210	0.575

Cuadro 3: Media de resultados de las diferentes técnicas de preprocesamiento.

		Average	STD	Max
Filtrado de ruido	No	0.288	0.253	0.589
	HME	0.339	0.231	0.593
	NCNEdit	0.292	0.255	0.584
Selección de instancias	No	0.333	0.256	0.597
	FCNN	0.289	0.231	0.593
Selección de características	No	0.341	0.241	0.593
	PCA	0.272	0.242	0.584
Balanceo de datos	No	0.066	0.070	0.215
	ROS	0.426	0.136	0.542
	RUS	0.284	0.273	0.593

Cuadro 4: Efectos de las diferentes técnicas de preprocesamiento para árboles de decisión.

2.2. Sobre las técnicas de selección de características

Como se dijo anteriormente, hemos aplicado PCA únicamente a los algoritmos donde tiene sentido, pero podemos ver a partir de las tablas 3, 4, 6 y 8 que los resultados empeoran tras su uso.

Hacemos notar que PCARD se comporta mejor que los árboles de decisión con PCA, a pesar de acabar teniendo un flujo similar. El razonamiento lo achacamos a la discretización aleatoria (RD) de PCARD, que elige un tamaño de intervalos mejor que el de 32 con el que se han entrenado los árboles.

A pesar de todo, podríamos considerar si la reducción de dimensionalidad y simplicidad obtenida es aceptable a costa de esa cantidad de empeoramiento. En este problema, pasando de una media de 0,593 a 0,584, que corresponde a una clasificación errónea de $593.000 - 584.000 = 9.000$ instancias más, dada la semántica del problema no parece una pérdida substancial. Si por otro caso tratáramos con un problema médico habría que considerar independientemente el TPR y el TNR antes de aceptar esta conclusión.

Por otro lado, hemos aplicado ChiSquareSelector sobre los mejores resultados de DT y RF (sobre kNN no ya que es un método basado en distancias y para aplicar ChiSq era necesario realizar una discretización de las variables continuas).

Algoritmo	Under/Over sampling	Filtrado de ruido	Selección de instancias	TPR x TNR
DT	RUS	HME	FCNN	0.592
	RUS	HME	No	0.595
	RUS	NCNEdit	No	0.586
	RUS	No	No	0.590
RF	RUS	HME	FCNN	0.593
	RUS	HME	No	0.596
	RUS	NCNEdit	No	0.594
	RUS	No	No	0.594

Cuadro 5: Resultandos combinando ChiSquareSelector con diferentes métodos de preprocesamiento en árboles de decisión y random forest.

		Average	STD	Max
Filtrado de ruido	No	0.239	0.250	0.583
	HME	0.294	0.238	0.587
	NCNEdit	0.222	0.252	0.585
Selección de instancias	No	0.278	0.257	0.585
	FCNN	0.224	0.229	0.587
Selección de características	No	0.316	0.258	0.587
	PCA	0.187	0.211	0.511
Balanceo de datos	No	0.039	0.184	0.196
	ROS	0.353	0.273	0.532
	RUS	0.362	0.179	0.587

Cuadro 6: Efectos de las diferentes técnicas de preprocesamiento para random forest.

Los resultados, mostrados en la tabla 5, nos indican una variación tanto por arriba como por abajo únicamente de milésimas, pero en media superiores a su equivalente con PCA (creemos que por la discretización). Aun así, puesto que la alteración en TPR x TNR es mínima reincidimos en las conclusiones anteriores, añadiendo que además al haber reducido dimensionalidad hemos formado árboles más simples y, por tanto, más interpretables.

2.3. Sobre las técnicas de balanceo de datos

Tal y como se mencionó anteriormente, a falta de tener unos algoritmos donde se les puedan indicar unos pesos para cada clase de forma que evitemos un sesgo hacia la mayoritaria, el ajuste del desbalanceo es vital en nuestro problema.

En concreto apreciamos que no solo RUS ayuda a acelerar la tarea de aprendizaje, también nos da los mejores resultados. Aún así, en media vemos que se comporta peor que ROS, debido probablemente a que su combinación con técnicas de selección de instancias reduce en algunos casos de manera excesiva el conjunto de datos.

		Average	STD	Max
Filtrado de ruido	No	0.309	0.273	0.597
	HME	0.369	0.241	0.595
	NCNEdit	0.286	0.290	0.593
Selección de instancias	No	0.362	0.268	0.597
	FCNN	0.281	0.252	0.595
Balanceo de datos	No	0.072	0.076	0.186
	ROS	0.496	0.325	0.542
	RUS	0.397	0.220	0.597

Cuadro 7: Efectos de las diferentes técnicas de preprocesamiento para PCARD.

		Average	STD	Max
Filtrado de ruido	No	0.292	0.159	0.491
	HME	0.354	0.193	0.525
	NCNEdit	0.294	0.200	0.492
Selección de instancias	No	0.313	0.195	0.525
	FCNN	0.313	0.165	0.521
Selección de características	No	0.328	0.177	0.525
	PCA	0.299	0.188	0.516
Balanceo de datos	No	0.103	0.038	0.235
	ROS	0.386	0.180	0.432
	RUS	0.448	0.167	0.525

Cuadro 8: Efectos de las diferentes técnicas de preprocesamiento para kNN.

2.4. Sobre las técnicas de reducción de ruido

Los resultados dan a entender de que el dataset no es de por sí bastante ruidoso, y el posible ruido introducido por las otras técnicas no resulta influyente. No por ello dejamos de apreciar que HME ayuda en la obtención de los mejores valores de TPR x TNR y funciona mejor en este dataset que NCNEdit.

2.5. Sobre las técnicas de reducción de instancias

Vemos que el uso de FCNN apenas altera los resultados, pero no por ello deja de ser útil, pues aplica una reducción en torno al 50 % del conjunto de datos. En una situación de big data como la que nos encontramos esto es totalmente deseable, ya que reducimos tiempo de cómputo y carga en el sistema.

Finalmente, indicamos que aunque la técnica SSMA sobrepasa el límite de 4GB de memoria impuesto en la práctica, en base a las dos ejecuciones con las que contamos (de los primeros días cuando el límite estaba en 46GB) vemos que la reducción en el número de instancias es extremadamente grande, llegando a obtener subconjuntos de 12.000 y 20.000 instancias. A pesar de ello en este caso los resultados sí son bastante inferiores respecto a FCNN o a no aplicar nada, por lo que concluímos que su uso no es positivo en este conjunto de datos.

2.6. Algunas conclusiones generales

- La reducción de datos en problemas de big data es esencial. Debemos transformar el dataset en un conjunto manejable y representativo de forma que aprovechemos al máximo el tiempo y los recursos disponibles.
- Lo mismo es aplicable a la selección de características, y debemos tener en cuenta que una mejora no es solo útil por el valor en la métrica que alcance, sino además por la reducción de complejidad que puede obtener.

  También debemos tener en cuenta la simplificación de los modelos y resultados obtenidos, que suele ser indicio de una mayor generalización.
- No existe un único flujo de preprocesamiento para todos los algoritmos, dependerá del funcionamiento de cada técnica de aprendizaje para sacar el máximo potencial. Por ejemplo, el ruido es más influyente en kNN (donde en base al valor de k puede cambiar radicalmente la clase predicha) que en una técnica basada en árboles (donde la poda ayudará a generalizar).
- En algunos casos es más interesante reconsiderar y mejorar el flujo de preprocesamiento que optimizar una y otra vez los hipérparametros del algoritmo. Hemos visto mejoras de 2 o 3 milésimas cambiando los parámetros frente a mejoras en las decenas con flujos diferentes.

  No por ello hay que ignorar la etapa de optimización, pero resulta más eficiente aplicarla en último lugar.

3. Tablas de resultados

Algorithm	Noise Filtering	Instance selection	Under/Oversampling	Instance selection	Final training instances	Accuracy	TPR x TNR
DT	No	No	No	No	1,000,000	0.900	0.000
				ROS	1,350,901	0.788	0.578
				RUS	200,160	0.753	0.589
			PCA	No	1,000,000	0.900	0.000
				ROS	1,350,854	0.805	0.442
				RUS	200,171	0.644	0.503
		FCNN	No	No	327,863	0.890	0.215
				ROS	754,343	0.800	0.563
				RUS	10	0.900	0.000
			PCA	No	327,947	0.887	0.127
				ROS	758,771	0.807	0.438
				RUS	10	0.900	0.000
	HME	No	No	No	903,094	0.902	0.049
				ROS	836,576	0.841	0.502
				RUS	157,627	0.727	0.592
			PCA	No	901,966	0.901	0.041
				ROS	807,345	0.896	0.072
				RUS	151,299	0.690	0.519
		FCNN	No	No	30,577	0.898	0.142
				ROS	450,797	0.831	0.520
				RUS	105,534	0.736	0.593
			PCA	No	23,870	0.892	0.122
				ROS	431,957	0.826	0.399
				RUS	101,441	0.698	0.519
		SSMA	No	ROS	22,395	0.865	0.260
				RUS	12,104	0.603	0.516
	NCNEdit	No	No	No	878,496	0.901	0.015
				ROS	981,405	0.816	0.548
				RUS	200,047	0.758	0.584
			PCA	No	878,286	0.900	0.005
				ROS	977,415	0.784	0.452
				RUS	200,235	0.634	0.496
		FCNN	No	No	80,104	0.901	0.052
				ROS	340,780	0.817	0.551
				RUS	10	0.900	0.000
			PCA	No	78,203	0.900	0.018
				ROS	340,699	0.822	0.424
				RUS	10	0.900	0.000
							0.593

Figura 1: Tabla de resultados del algoritmo Decision Tree.

Algorithm	Noise Filtering	Instance selection	Under/Oversampling	Instance selection	Final training instances	Accuracy	TPR x TNR
RF	No	No	No	No	1,000,000	0.900	0.000
				ROS	1,350,214	0.819	0.530
				RUS	199,746	0.736	0.583
			PCA	No	1,000,000	0.900	0.000
				ROS	1,349,043	0.890	0.129
				RUS	200,103	0.653	0.503
		FCNN	No	No	327,863	0.894	0.182
				ROS	754,464	0.821	0.521
				RUS	10	0.900	0.000
			PCA	No	327,947	0.900	0.000
				ROS	759,247	0.825	0.419
				RUS	10	0.900	0.000
	HME	No	No	No	903,094	0.901	0.011
				ROS	813,512	0.871	0.347
				RUS	158,309	0.731	0.585
			PCA	No	901,966	0.900	0.000
				ROS	817,594	0.900	0.007
				RUS	151,891	0.649	0.511
		FCNN	No	No	30,577	0.895	0.196
				ROS	451,715	0.817	0.532
				RUS	105,913	0.739	**0.587**
			PCA	No	23,875	0.899	0.083
				ROS	427,801	0.891	0.188
				RUS	101,747	0.619	0.477
	NCNEdit	No	No	No	878,496	0.900	0.000
				ROS	981,334	0.831	0.512
				RUS	199,074	0.725	0.585
			PCA	No	878,286	0.900	0.000
				ROS	979,265	0.884	0.200
				RUS	199,605	0.658	0.509
		FCNN	No	No	80,104	0.900	0.000
				ROS	340,931	0.824	0.523
				RUS	10	0.900	0.000
			PCA	No	78,203	0.900	0.000
				ROS	342,209	0.854	0.333
				RUS	10	0.900	0.000
							0.587

Figura 2: Tabla de resultados del algoritmo Random Forest.

Algorithm	Noise Filtering	Instance selection	Under/Oversampling	Instance selection	Final training instances	Accuracy	TPR x TNR
PCARD	No	No	No	No	1,000,000	0.901	0.018
				ROS	1,349,105	0.838	0.515
				RUS	199,843	0.738	0.597
			PCA	No	X	X	X
				ROS	X	X	X
				RUS	X	X	X
		FCNN	No	No	327,863	0.897	0.186
				ROS	754,526	0.826	0.539
				RUS	10	0.900	0.000
			PCA	No	X	X	X
				ROS	X	X	X
				RUS	X	X	X
	HME	No	No	No	903,094	0.902	0.037
				ROS	800,941	0.879	0.358
				RUS	158,588	0.718	0.594
			PCA	No	X	X	X
				ROS	X	X	X
				RUS	X	X	X
		FCNN	No	No	30,577	0.901	0.124
				ROS	438,282	0.843	0.507
				RUS	106,099	0.730	0.595
			PCA	No	X	X	X
				ROS	X	X	X
				RUS	X	X	X
	NCNEdit	No	No	No	878,496	0.900	0.001
				ROS	982,131	0.831	0.542
				RUS	199,965	0.726	0.593
			PCA	No	X	X	X
				ROS	X	X	X
				RUS	X	X	X
		FCNN	No	No	80,104	0.902	0.069
				ROS	342,103	0.838	0.513
				RUS	10	0.900	0.000
			PCA	No	X	X	X
				ROS	X	X	X
				RUS	X	X	X
							0.597

Figura 3: Tabla de resultados del algoritmo PCARD.

Algorithm	Noise Filtering	Instance selection	Under/Oversampling	Instance selection	Final training instances	Accuracy	TPR x TNR
KNN	No	No	No	No	1,000,000	0.887	0.134
				ROS	1,351,028	0.775	0.389
				RUS	199,961	0.652	0.491
			PCA	No	1,000,000	0.887	0.125
				ROS	1,350,241	0.776	0.379
				RUS	200,157	0.657	0.490
		FCNN	No	No	327,863	0.849	0.235
				ROS	754,740	0.735	0.374
				RUS	10		
			PCA	No	327,947	0.850	0.231
				ROS	760,288	0.734	0.366
				RUS	10	0.900	0.000
	HME	No	No	No	903,094	0.901	0.024
				ROS	849,438	0.805	0.342
				RUS	158,009	0.635	0.525
			PCA	No	901,966	0.900	0.017
				ROS	818,533	0.818	0.310
				RUS	151,560	0.633	0.516
		FCNN	No	No			
				ROS	449,224	0.757	0.392
				RUS	106,229	0.636	0.521
			PCA	No			
				ROS	436,610	0.763	0.382
				RUS	101,019	0.634	0.512
	NCNEdit	No	No	No	878,496	0.900	0.030
				ROS	981,398	0.810	0.432
				RUS	199,834	0.652	0.492
			PCA	No	878,286	0.900	0.027
				ROS	976,648	0.813	0.416
				RUS	199,712	0.656	0.490
		FCNN	No	No	80,104	0.891	0.107
				ROS	340,985	0.784	0.432
				RUS	10		
			PCA	No	78,203	0.891	0.099
				ROS	342,429	0.785	0.420
				RUS	10		
							0.525

Figura 4: Tabla de resultados del algoritmo kNN-IS.