Análisis de datos con Spark

Big Data 2

Alberto Armijo Ruiz

Introducción

En esta práctica se procederá a estudiar el comportamiento de diferentes modelos y métodos de preprocesamiento para un problema de Big Data de clasificación. Como herramienta se utilizará Spark para ejecutar los modelos y los métodos de preprocesamiento. El dataset que se va a utilizar es una partición del dataset Susy; el dataset original contiene 5 millones de instancias y 18 columnas, para la práctica el subconjunto de datos de entrenamiento contiene 10 columnas y 1 millón de datos para train y otro subconjunto del mismo tamaño para test.

Resultados iniciales

Lo primero que se va a hacer es probar el rendimiento de algunos modelos clásicos y otros enfocados a problemas Big Data (PCARD) sin realizar ningún preprocesamiento a los datos. Las métricas que se van a utilizar son la Precisión y AUC. Los resultados obtenidos son los siguientes:

Model	**Class Balance**	**Noise Reduction**	**Instance Selection**	**Precision**	**AUC**
Decision Tree	None	None	None	0.9	1
GBT	None	None	None	0.901093	0.78878783
SVM	None	None	None	0.9	0.5
PCARD	None	None	None	0.900847	0.5080261
Random Forest	None	None	None	0.902313	0.79059665

Como se puede ver, los resultados son muy dispares entre los diferentes algoritmos, en todos los casos se puede apreciar que la precisión siempre es 0.9; pero la métrica AUC varía mucho y en varios casos es de 0.5, lo que indica que la calidad de esos dos modelos es la misma que un modelo aleatorio. Por la varianza que hay en los resultados de la métrica AUC, posiblemente haya bastante desbalanceo de las clases y los resultados que se han obtenido no sean fiables, por ello, se probarán diferentes tipos de preprocesamientos de los datos para mejorar los resultados y que sean más fiables. Nos interesa tener valores para las métricas lo más parecidos posibles, ya que si existe mucha diferencia entre la precisión y el AUC es posible que solamente se esté aprendiendo una clase. Mirando la cantidad de instancias de cada clase, se puede ver que para la clase positiva existen 900000 ejemplos y para la clase negativa 100000; por lo que habrá que intentar balancear ambas clases.

Pruebas oversampling

Lo primero que se va a hacer es probar a realizar oversampling solamente sobre los datos y ver que resultados obtienen los clasificadores. El método utilizado para oversampling es Random Oversampling, que genera instancias con valores aleatorios de la clase minoritaria hasta alcanzar un cierto porcentaje. Los resultados obtenidos son los siguientes:

Model	Class Balance	Noise Reduction	Instance Selection	Precision	AUC
DecisionTree	ROS 25	None	None	0.878566	0.668053
RandomForest	ROS 25	None	None	0.886161	0.679921
PCARD	ROS 25	None	None	0.893088	0.626520
GBT	ROS 25	None	None	0.885300	0.68484
SVM	ROS 25	None	None	0.9	0.5
DecisionTree	ROS 50	None	None	0.827279	0.640367
RandomForest	ROS 50	None	None	0.827299	0.642837
PCARD	ROS 50	None	None	0.828762	0.74723
GBT	ROS 50	None	None	0.831362	0.648739
SVM	ROS 50	None	None	0.877203	0.647348
DecisionTree	ROS 75	None	None	0.780685	0.625716
RandomForest	ROS 75	None	None	0.780922	0.627723
PCARD	ROS 75	None	None	0.775904	0.77359
GBT	ROS 75	None	None	0.787382	0.632340
SVM	ROS 75	None	None	0.760637	0.760767
DecisionTree	ROS 100	None	None	0.735856	0.614128
RandomForest	ROS 100	None	None	0.73969	0.617169
PCARD	ROS 100	None	None	0.733355	0.778623
GBT	ROS 100	None	None	0.751577	0.62179
SVM	ROS 100	None	None	0.666628	0.763411

Según los resultados de las pruebas, se puede ver que los resultados más interesante se encuentran cuando se realiza ROS 50 y ROS 75, donde el que mejores resultados obtiene es el modelo PCARD; para otros valores de oversampling hay demasiada diferencia entre AUC y Precision, por lo cual es posible que los algoritmos sigan estando demasiado desbalanceado y el modelo no esté aprendiendo ambas clases; o también que la Precision baje demasiado porque el número de instancias generadas son demasiadas y ruidosas (ya que ROS genera las instancias de forma aleatoria y pueden coincidir con instancias de la clase mayoritaria)..

Pruebas oversampling más otro preprocesamiento

Lo siguiente que se va a hacer es probar diferentes métodos de preprocesamiento sobre los datos con ROS; las pruebas se realizarán sobre conjuntos de datos con ROS 50 y ROS 75. Los preprocesamientos que se van a utilizar son selección de instancias con FCNN y limpieza de ruido con HME-BD por separado, tras esto se harán pruebas con ambos métodos de preprocesamiento y se analizarán los resultados obtenidos.

Model	Class Balance	Noise Reduction	Instance Selection	Precision	AUC
DecisionTree	ROS 50	None	FCNN	0.9	1.0
RandomForest	ROS 50	None	FCNN	0.9	1.0
PCARD	ROS 50	None	FCNN	0.9	0.5
GBT	ROS 50	None	FCNN	0.9	1.0
SVM	ROS 50	None	FCNN	0.9	0.5

Model	Class Balance	Noise Reduction	Instance Selection	Precision	AUC
DecisionTree	ROS 75	None	FCNN	0.9	1.0
RandomForest	ROS 75	None	FCNN	0.9	1.0
PCARD	ROS 75	None	FCNN	0.9	0.5
GBT	ROS 75	None	FCNN	0.9	1.0
SVM	ROS 75	None	FCNN	0.9	0.5

Model	Class Balance	Noise Reduction	Instance Selection	Precision	AUC
DecisionTree	ROS 50	HMEBD	None	0.817689	0.632300
RandomForest	ROS 50	HMEBD	None	0.827823	0.642736
PCARD	ROS 50	HMEBD	None	0.813235	0.751677
GBT	ROS 50	HMEBD	None	0.816193	0.63332
SVM	ROS 50	HMEBD	None	0.855529	0.68352

Model	Class Balance	Noise Reduction	Instance Selection	Precision	AUC
DecisionTree	ROS 75	HMEBD	None	0.768858	0.61862
RandomForest	ROS 75	HMEBD	None	0.779708	0.626713
PCARD	ROS 75	HMEBD	None	0.770629	0.76984

GBT	ROS 75	HMEBD	None	0.766911	0.618450
SVM	ROS 75	HMEBD	None	0.747247	0.759830

Model	Class Balance	Noise Reduction	Instance Selection	Precision	AUC
DecisionTree	ROS 75	HMEBD	FCNN	0.755587	0.61222
RandomForest	ROS 75	HMEBD	FCNN	0.784781	0.628502
PCARD	ROS 75	HMEBD	FCNN	0.778581	0.774287
GBT	ROS 75	HMEBD	FCNN	0.771964	0.619617
SVM	ROS 75	HMEBD	FCNN	0.763047	0.758830

Model	Class Balance	Noise Reduction	Instance Selection	Precision	AUC
DecisionTree	ROS 50	HMEBD	FCNN	0.817297	0.6310248
RandomForest	ROS 50	HMEBD	FCNN	0.82535	0.64188
PCARD	ROS 50	HMEBD	FCNN	0.82536	0.75097
GBT	ROS 50	HMEBD	FCNN	0.817036	0.63116
SVM	ROS 50	HMEBD	FCNN	0.831983	0.717866

Como se puede ver en las tablas, se puede ver que el algoritmo FCNN por sí solo no ofrece buenos resultados, de hecho son igual de malos que si no se realizara ningún preprocesamiento en los datos. Para el algoritmo HMEBD se puede ver que tampoco ha habido una mejora considerable para ninguno de los modelos a excepción de PCARD y SVM con ROS 75. Para las pruebas con HMEBD y FCNN juntos los resultados son muy parecidos a los obtenidos a los anteriores; por lo cual se puede concluir que realmente estos dos preprocesamientos no son demasiado efectivos para el conjunto de datos.

Otro preprocesamiento

En este apartado se muestran más pruebas con diferentes valores para ROS y mezclandolos con algún otro método de preprocesamiento como RNG o HMEBD. Además de estas pruebas también se han realizado pruebas con undersampling para algunos modelos para ver su rendimiento en comparación con el rendimiento obtenido con ROS.

Model	Class Balance	Noise Reduction	Instance Selection	Precision	AUC
DecisionTree	ROS 150	None	None	0.690729	0.60157750

PCARD	ROS 150	None	None	0.658488	0.767813
PCARD 15 trees	ROS 150	None	None	0.658381	0.7667005
PCARD 20 trees	ROS 150	None	None	0.65994	0.76874444
Random Forest	ROS 150	None	None	0.675794	0.6041420
DecisionTree	ROS 160	None	None	0.62149	0.591881
PCARD	ROS 160	None	None	0.635929	0.761920
Random Forest	ROS 160	None	None	0.667461	0.602560
PCARD	ROS 170	None	None	0.631286	0.759212
Random Forest	ROS 170	None	None	0.655581	0.600518
PCARD	ROS 180	None	None	0.631245	0.760256
Random Forest	ROS 180	None	None	0.646953	0.599186
PCARD	ROS 150	None	FCNN	0.9	0.5
PCARD	ROS 150	HMEBD	None	0.638526	0.760509
PCARD	ROS 150	HMEBD	FCNN	0.678874	0.77265888
PCARD	ROS 150	RNG	FCNN	0.901997	0.5213627
Random Forest	ROS 150	HMEBD	None	0.682042	0.6046111
Random Forest	ROS 160	HMEBD	None	0.673708	0.6031979
DecisionTree	ROS 150	HMEBD	None	0.650236	0.595623
DecisionTree	ROS 160	HMEBD	FCNN	0.670183	0.5992478

Model	**Class Balance**	**Noise Reduction**	**Instance Selection**	**Precision**	**AUC**
PCARD	RUS	None	None	0.72915	0.77727
Random Forest	RUS	None	None	0.736881	0.616514
GBT	RUS	None	None	0.744938	0.613431
SVM	RUS	None	None	0.663992	0.7605066
PCARD	RUS	HMEBD	None	0.714941	0.772345
GBT	RUS	HMEBD	None	0.737792	0.612049
SVM	RUS	HMEBD	None	0.662635	0.759841
PCARD	RUS	None	FCNN	0.9	0.5

PCARD	RUS	NCNedit	None	0.739019	0.777046

Por lo que se puede ver, los resultados del resto de pruebas con ROS son bastante malas en comparación con las primeras pruebas realizadas, ya que la Precision baja demasiado, posiblemente porque el valor para ROS es demasiado grande. Los resultados obtenidos para RUS son bastante parecidos a los obtenidos con ROS 50 o ROS 75.

Conclusiones

Según todas las pruebas realizadas, se puede concluir que los mejores modelos se obtienen con el algoritmo PCARD, el cual obtiene los resultados más altos para AUC y Precision. Para balanceo de clases, los métodos de RUS y ROS obtienen resultados bastante parejos, pero al añadir más métodos de preprocesamiento ROS muestra mejores resultados que RUS, posiblemente porque el procesamiento con datos con undersampling puede hacer que se pierda información valiosa sobre los datos. Sobre los métodos de preprocesamiento adicionales, realmente no aportan demasiada mejora o empeoran bastante los resultados (como en el caso de FCNN); por lo cual no los utilizaría para generar un modelo final.

Además de las pruebas que se muestran en esta documentación, también se han probado otros modelos, como KNN, y otros métodos de selección de instancias, como SSMASFLSDE; pero estos no han conseguido terminar. Además de esto, se podría haber probado con otras medidas como el TPR y TNR para ver el rendimiento de los modelos en clasificación para ambas clases.