Centro de Investigación en Matemáticas (CIMAT). Unidad Monterrey

Tarea 1: Manejo de ruido

Victor Manuel Gómez Espinosa

1 de octubre de 2020

1. Métodos de manejo de ruido

Se realizó una clase en Python llamada NoiseHandling, para manejo de ruido en clases con alguno de los siguientes 5 métodos:

1. Métodos de filtro de ruido: Edited Nearest Neighbor (ENN)
2. Métodos de filtro de ruido: Ensemble Filter (EF)
3. Métodos de filtro de ruido: Complementary Neural Network (CMTNN)
4. Métodos de pulido de ruido: Iterative Edition
5. Métodos de pulido de ruido: Hybrid Repairing Filter

Se seleccionaron 12 (1 por cada nivel de ruido) conjuntos de datos presentan diferentes características, la cantidad de ejemplos que esta entre 8000 y 4000, todos son de 2 clases, con 5, 10, 15 y 20% de ruido en las clases. Se utilizaron conjuntos de prueba del 20% del original, estos no contienen ruido y se utilizaron para obtener el accuracy. Algunos de los conjuntos de datos muestran un desbalance en las clases. (Figura 1.1).

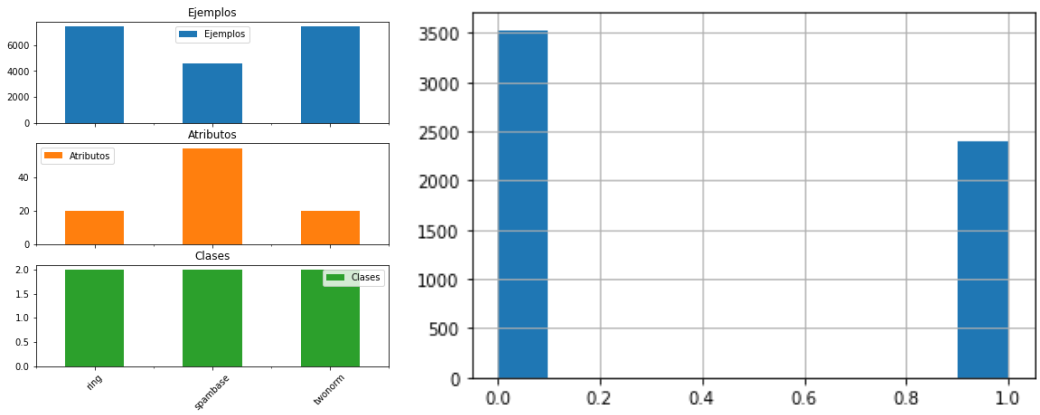

Figura 1.1: a) 12 Conjuntos de datos con ruido (5,10,15,20 %). b) clases desbalanceadas en el conjunto de datos.

Para generar datos para comparar cada método de manejo de ruido, para cada conjunto de datos se aplicó cada uno de los métodos y para cada uno se utilizaron 3 modelos de Machine Learning para clasificación ajustados mediante búsqueda aleatoria y validación cruzada (5x2) y empleando la métrica de accuracy con el conjunto de prueba, que es el que no contiene ruido. Los modelos son los siguientes:

1. KNeighbors: k: [2-40]
2. XGBoost: learning rate: [0.1,0.3,0.6], n_estimators: [50,150,300,500], subsample: [0.5,0.75,0.9], colsample_bytree: [0.5,0.75,0.9], regularización: gamma: [0,10,100], lambda: [1,10,100]
3. SVM: parameters = C: [0.1, 10], kernel: ['linear'], C: [1,5, 10,20], gamma: [0.001, 0.01, 0.1], kernel: ['rbf']

2. Análisis estadístico de los resultados

A partir de las muestras generadas como se explicó anteriormente, mediante un análisis descriptivo se puede observar que la mayoría de los métodos tienen una variación en un rango de 0.9 a 0.99, con media y mediana muy similar, pero aparentemente el método IterativeEdition se sale de este rango, es mucho mayor y hacia la izquierda, lo cual da una idea de un pobre desempeño (Figura 2.1 a). Otro método de manejo de ruido son los Robust Learners, por ejemplo basados en árboles, por ese motivo se utilizó el modelo XGBoost como uno de los clasificadores, pero se puede observar que el desempeño de este antes y después de ruido no parece ser muy significativo, y se tiene un comportamiento muy similar a las máquinas de soporte vectorial, que aunque su rango es mayor, su mediana está más cargada a la derecha, lo cual parece indicar que ambos son muy buenos para el manejo de ruido, principalmente en este experimento las SVM (Figura 2.1 b).

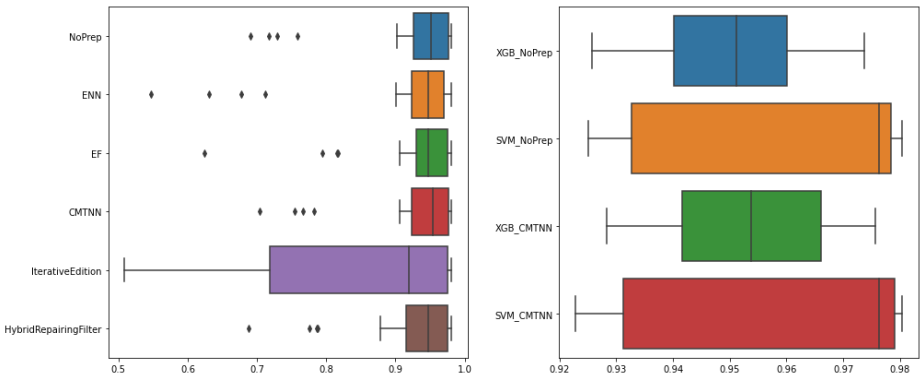

Figura 2.1: a) Histograma de los métodos de manejo de ruido y sin preprocesar (NoPrep). b) Box-plot de Robust Learners (XGBoost) y SVM antes y después de preprocesar (CMTNN).

Posteriormente se aplicó la prueba de Friedman al 5% la cual se rechaza lo que indica que si hay diferencias significativas entre nuestras mediciones por lo que después se aplicó la prueba de Nemenyi Friedman y se observó que la mayoría de los test pareados son similares excepto el método IterativeEdition como se había visto anteriormente (Figura 2.2).

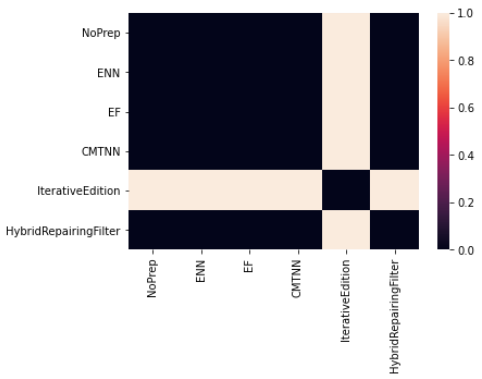

Figura 2.2: Heatmap de la prueba de Nemenyi-Friedman. Valores de 1 significan los que rechazan la prueba, es decir que son significativamente diferentes.

Lo anterior tiene sentido considerando que el método Iterative Edition se sabe es sensible a las clases desbalanceadas, lo cual ocurre en este experimento para algunos conjuntos de datos, de hecho, durante las pruebas hubo errores por lo cual se incluyó un criterio de paro si se rebasa una tolerancia para la clase minoritaria, lo cual explica que en las pruebas haya resultado significativamente diferente a todos los demás.