Presentación Artículo para Análisis de Datos III

Prof.: Maikol Solís

Estudiante: Jimmy Calvo Monge

II-2021

Artículo de Referencia

- Título: *Invariant optimal feature selection: a distance discriminant and feature ranking based solution*.
- Autores: **J. Liang, S. Yang, A. Winstanley**.
- Pattern Recognition 41 (2008) 1429–1439.
- https://doi.org/10.1016/j.patcog.2007.10.018.

Introducción

- El objetivo de este artículo es presentar un nuevo método de selección de variables mediante un ranking de atributos y compararlo con otros métodos en experimentos con varios conjuntos de datos.
- El método propone una medida de ranqueo de variables denominada *FSDD*, basada en un discriminante de distancias para cada variable.
- **Selección de Variables** : tiene por seleccionar atributos importantes en un conjunto de datos para mejorar el desempeño de modelos predicitivos para así alcanzar: el alivio de la dimensionalidad y el reforzamiento de la interpretabilidad. Estudiaremos los métodos de ranking de variables.

FSDD

(Feature Selection based on Distance Discriminant)

- Este es el nuevo método que propone nuestro artículo de referencia.
- De acuerdo al artículo: *The basis of the proposed algorithm is to find out the features that promise good class separability among different classes as well as make the samples in the same classes as close as possible.*
- Para lograr este cometido, se introduce un **discriminante de distancias**, como un criterio para seleccionar buenas variables.

Este discriminante de distancias está definido incialmente para todo el conjunto de datos y viene dado por:

$$d_b - \beta d_w$$

Donde d_b (b: between) es un indicador que mide la distancia entre diferentes clases y d_w (w: within) mide la distancia entre las observaciones de una misma clase. El parámetro β se utiliza para controlar el impacto de d_w

¿Por qué considerar este discriminante de distancias?

Si lo vemos bien es muy natural querer maximizar este indicador: Queremos la mayor distancia entre las clases ($\max d_b$) y al mismo tiempo queremos minimizar la distancia de las observaciones dentro de una misma clase ($\min d_w$).

- Introducimos cierta terminología para explicar estas distancias. Contamos con N observaciones $Y_1, Y_2 \cdots Y_N$ de n variables, de manera que cada $Y_i \in \mathbb{R}^n$, y etiquetadas en c clases distintas.
- Definimos la distancia entre dos puntos $Y_i = (y_i^1, \cdots, y_i^n), Y_j = (y_j^1, \cdots, y_j^n) \in \mathbb{R}^n$ como

$$d(Y_i, Y_j) = \sum_{k=1}^{n} \frac{(y_i^k - y_j^k)^2}{\sigma_k^2},$$

  donde σ_k^2 es la varianza de la k-ésima columna.

- La distancia intra-clase, para la clase C viene dada por

$$d(C) = \frac{2}{N(N-1)} \sum_{i<j} d(Y_i, Y_j), \quad Y_i, Y_j \in C,$$

y la distancia intra-clase general entonces se define como

$$d_w = \sum_{i=1}^{c} \rho_i d(C_i),$$

donde ρ_i es la probabilidad anterior (*prior*) de la clase C_i.

- Por otro lado, la distancia entre clases se define como

$$d_b = \frac{1}{2} \sum_{i,j} \rho_i \rho_j d(m_i, m_j),$$

donde m_i es el centroide de la clase C_i, es decir

$$m_i = \frac{1}{|Y_\ell \in C_i|} \sum_{Y_\ell \in C_i} Y_\ell.$$

- En el artículo mencionado, se demuestra la fórmula

$$d_b - \beta d_w = \sum_{k=1}^{n} \frac{1}{\sigma_k^2} \left[\sigma_k''^2 - \beta \sum_{i=1}^{c} \rho_i \sigma_k^2(i) \right] \quad (1)$$

Donde:

- σ_k^2 es la varianza de la variable k-ésima, de todas las observaciones, esto es:

$$\sigma_k^2 = \frac{1}{N} \sum_{i=1}^{N} (y_i^k - \overline{y_k})^2, \quad \overline{y_k} = \frac{1}{N} \sum_{i=1}^{N} y_i^k.$$

- $\sigma_k^2(i)$ es la varianza de la variable k-ésima, de las observaciones en la clase $i-$ésima, esto es:

$$\sigma_k^2(i) = \frac{1}{|Y \in C_i| - 1} \sum_{Y \in C_i} (y^k - \overline{y_k(i)})^2, \quad \overline{y_k(i)} = \frac{1}{|Y \in C_i|} \sum_{Y \in C_i} y^k.$$

- $\sigma_k''^2$ es la varianza promediada del centroide de la clase i-ésima en la variable k-ésima. Esto es,

$$\sigma_k''^2 = \sum_{i=1}^{c} \rho_i (m_i^k - m_k)^2, \quad m_k = \sum_{i=1}^{c} \rho_i m_i^k.$$

La demostración es un poco extensa y bastante teórica, pero sólo involucra una serie de manipulaciones.

- El método de ranqueo de variables en este caso es utilizando la fórmula (1). Como los términos en la sumatoria de la fórmula dependen de cada variable y el objetivo principal es maximizar la métrica general $d_b - \beta d_w$, entonces los autores argumentan que, en cierto sentido, las mejores m variables son las m primeras variables con mayor valor en la métrica

$$\mathrm{FSDD}(k) = \frac{1}{\sigma_k^2}\left[\sigma_k'^2 - \beta \sum_{i=1}^{c} \rho_i \sigma_k^2(i)\right]$$

Y esta es la métrica para ranquear las variables que se propone en este artículo. Es un indicador por variable y depende de la naturaleza del conjunto de datos, y no de ningún modelo en particular (filter).

Metodología

Para comparar la eficacia del método de selección de variables FSDD el artículo propone comparar el algoritmo FSDD contra otros dos algoritmos establecidos de selección de variables que son

- Algoritmo ReliefF
- Uso del criterio de información mutua mediante el algoritmo mrmrMID.

Éstos dos algoritmos filter han sido ampliamente utilizados para la selección de variables. En el reporte del artículo se puede encontrar lo que hacen estos algoritmos específicamente, por falta de tiempo no entraremos en estos detalles.

😉

El experimento hace lo siguiente:

- Se tienen varios conjuntos de datos de muestra. Para cada uno se selecciona un subconjunto de datos con las mejores m variables de acuerdo a los tres algoritmos (FSDD, ReliefF, mrmrMID).
- Sobre el subconjunto de datos se aplica un clasificador (KNN, NB, DT y SVM) y se mide su precisión.
- Esto se hace para cada $m = 1$ hasta $m =$ #columnas.
- En un gráfico se comparan las precisiones obtenidas con cada método de selección de variables.

Los conjuntos de datos utilizados en el artículo de referencia sobre el valor FSDD son los siguientes:

1. MFeat (Multiple Features Dataset)
2. Satimage
3. Spambase
4. Spectrometer
5. Wine
6. Analcatdata
7. Iris
8. Vowel

Todos, excepto Analcatdata que está en Statlib, se encuentran en el repositorio UCI, de libre acceso y descarga. Para este trabajo decidí utilizar sólo los que siguen, por cuestiones de tiempo y duración de algunos de los algoritmos.

1. MFeat
2. Satimage
3. Spambase
4. Wine

En la siguiente tabla se encuentra la información sobre el tamaño de cada uno de estos conjuntos de datos y el método de CV a utilizar de acuerdo al artículo.

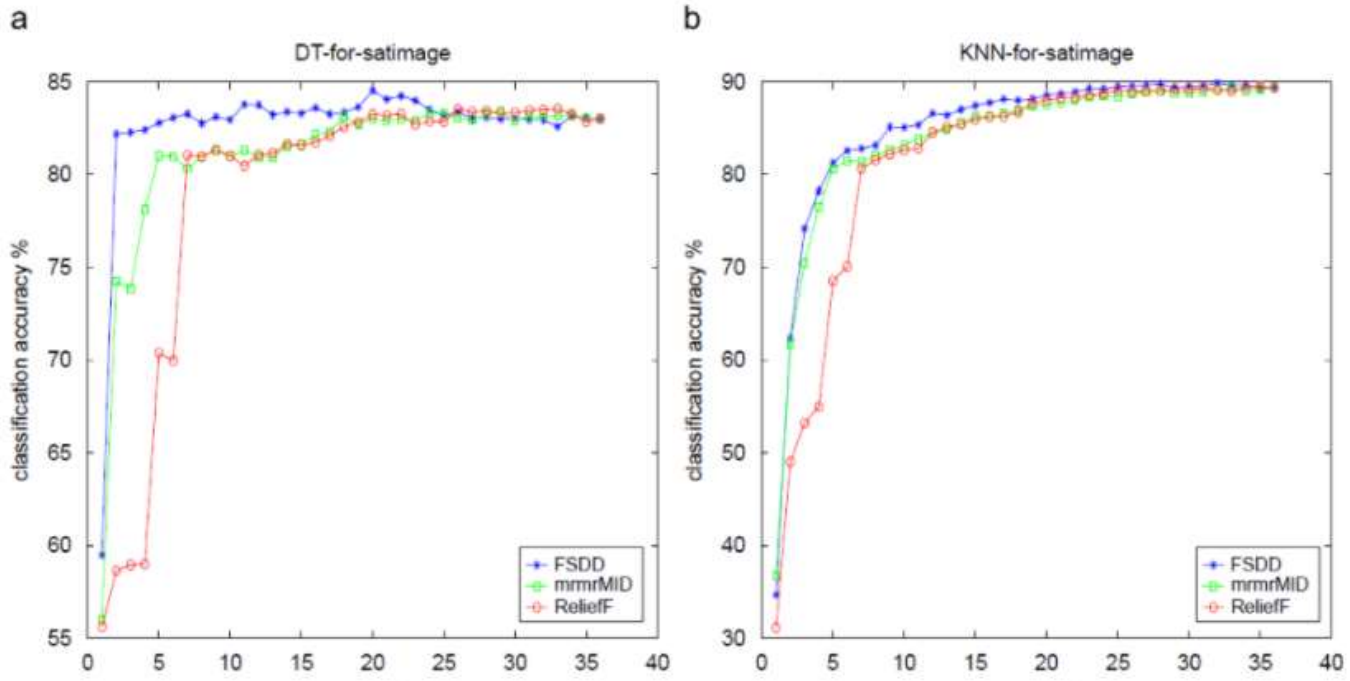
Conjunto de Datos	#Variables	#Instancias	CV Fold
Mfeat	649	2000	2-Fold CV
Satimage	36	6435	2-Fold CV
Spambase	57	4601	2-Fold CV
Wine	13	178	10-Fold CV

Descargué los conjuntos de datos como csv y en las siguientes líneas de código los leo para el análisis. El conjunto Mfeat se puede descargar desde **`mklearn`**, sin embargo requiere de cierta preparación porque viene un poco fragmentado.

- En los artículos no viene referencia a alguna biblioteca o repositorio en donde se encuentren estos métodos o experimentos, o no lo pude encontrar. Los autores mencionan que implementaron el análisis en Matlab.
  😕
- Por eso, para replicar los experimentos, decidí emular el análisis en python, por comodidad.
- Todo el código se puede encontrar aquí:

Resultados de los experimentos del artículo

Siguiendo la metodología propuesta, los autores presentan los resultados de sus experimentos en gráfocos como los que siguen. Aquí se puede apreciar la ventaja en precisión que ofrece el método propuesto en comparación a los otros dos. En este caso se muestran los resultados para el conjunto de datos Satimage.



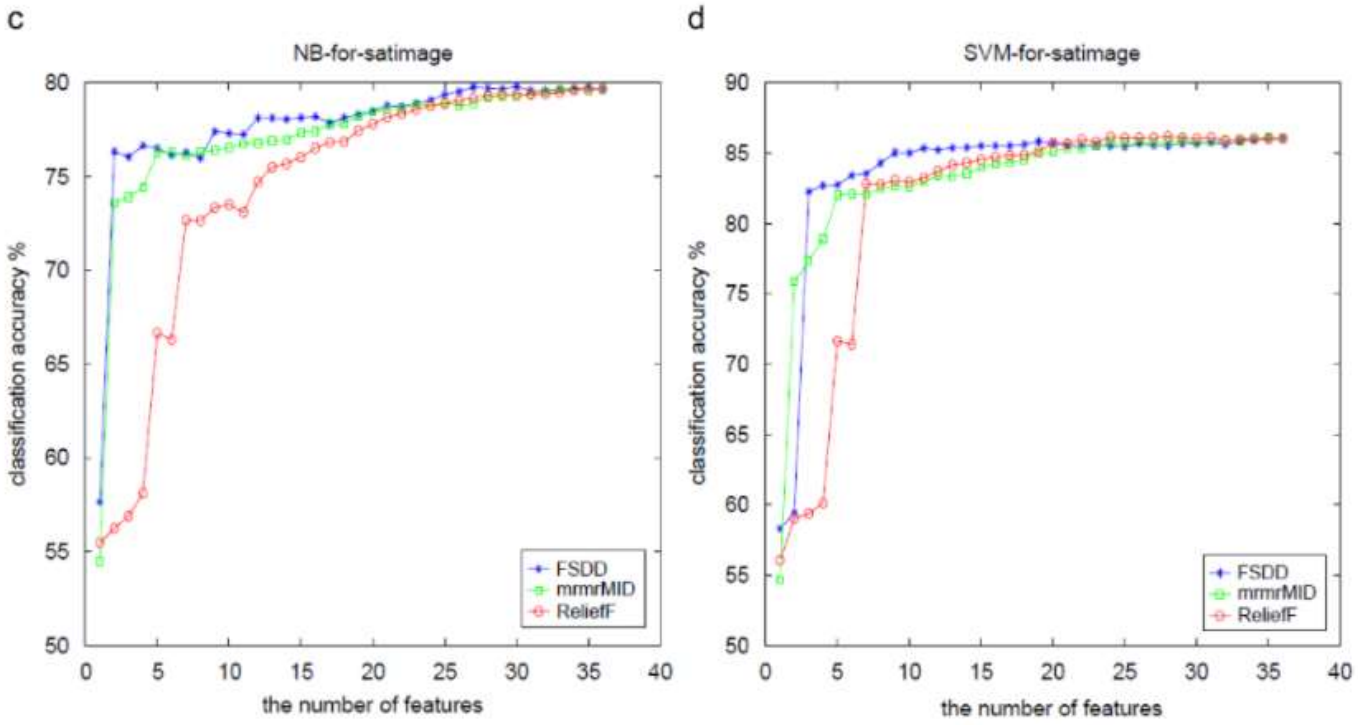
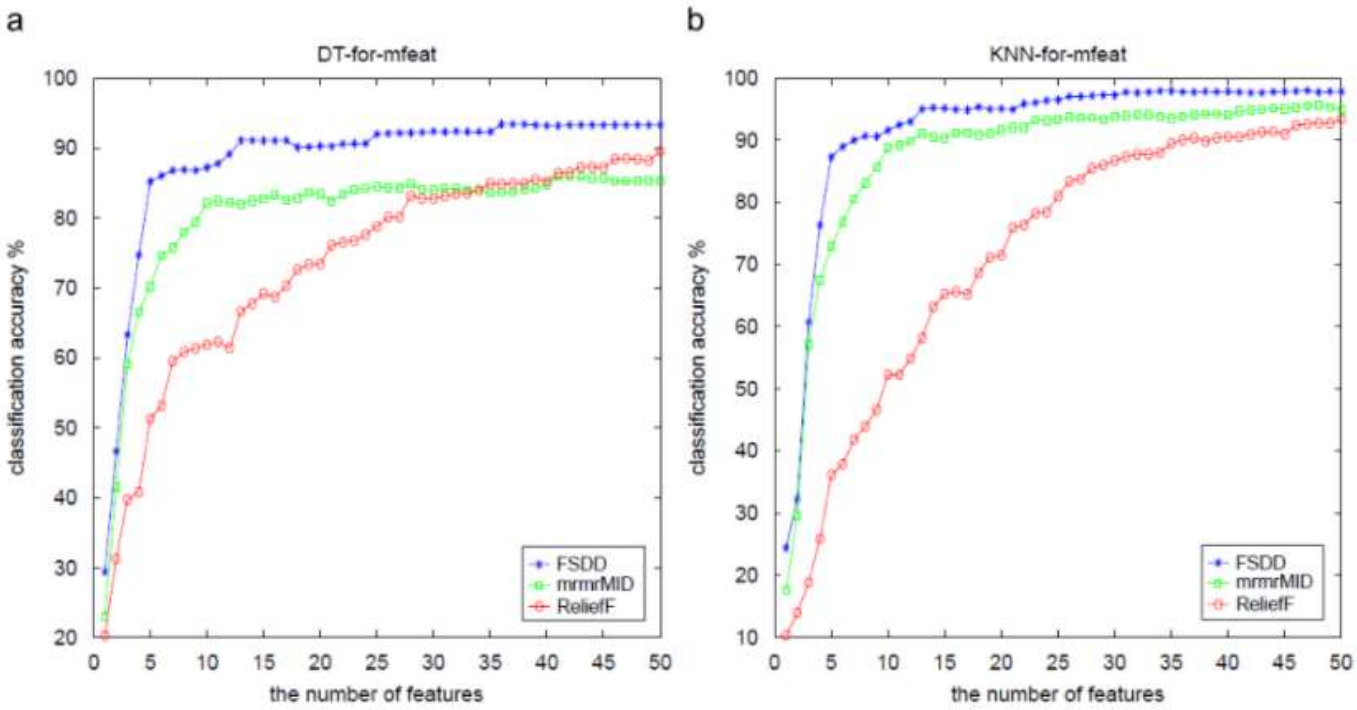


Fig. 2. Two-fold CV classification accuracy for Satimage with four classifiers.

Similarmente, los siguientes son los resultados obtenidos por los autores para el conjunto Mfeat:



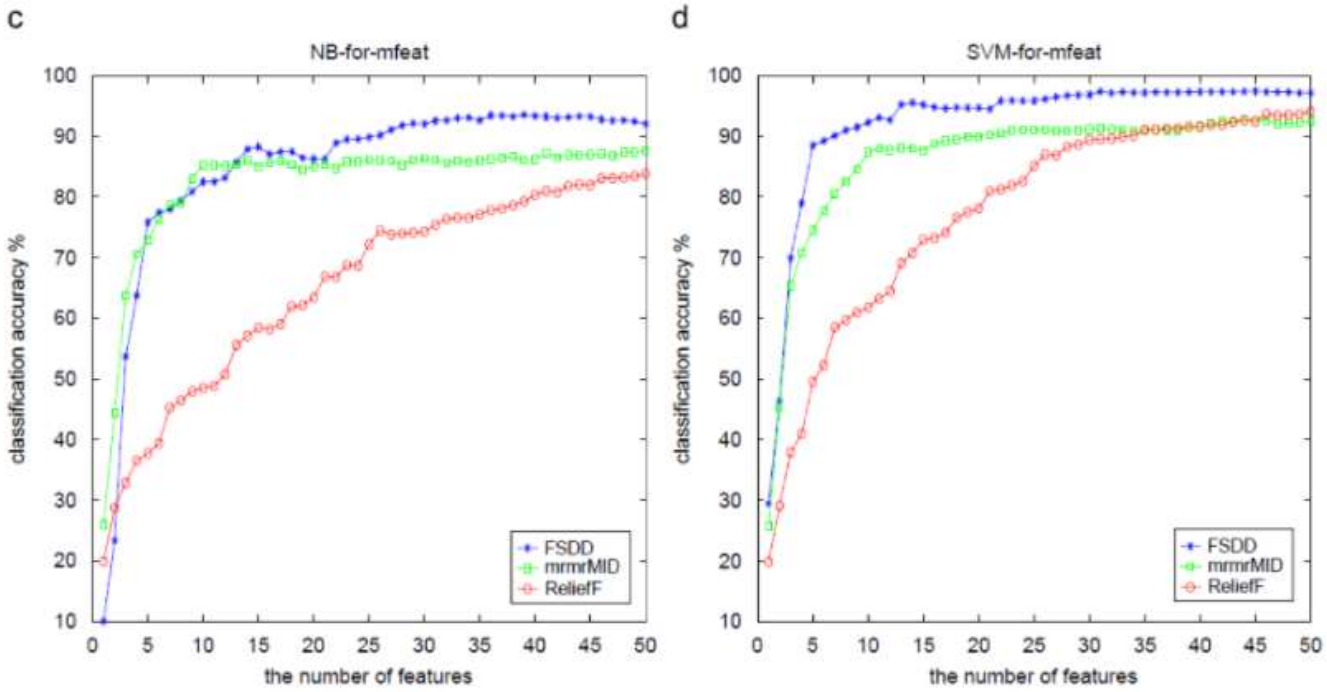
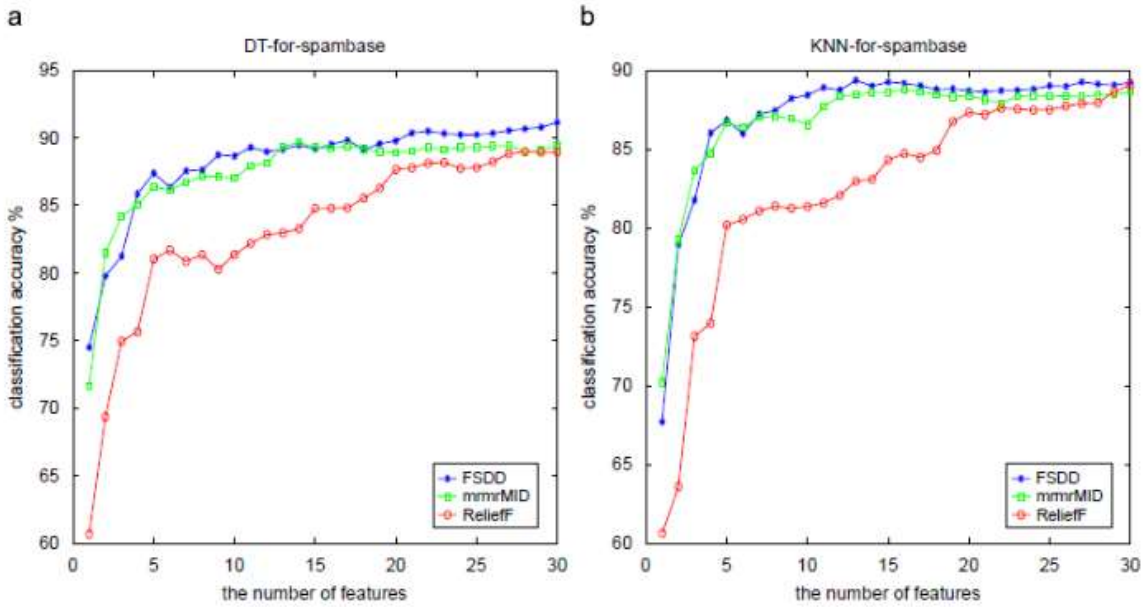
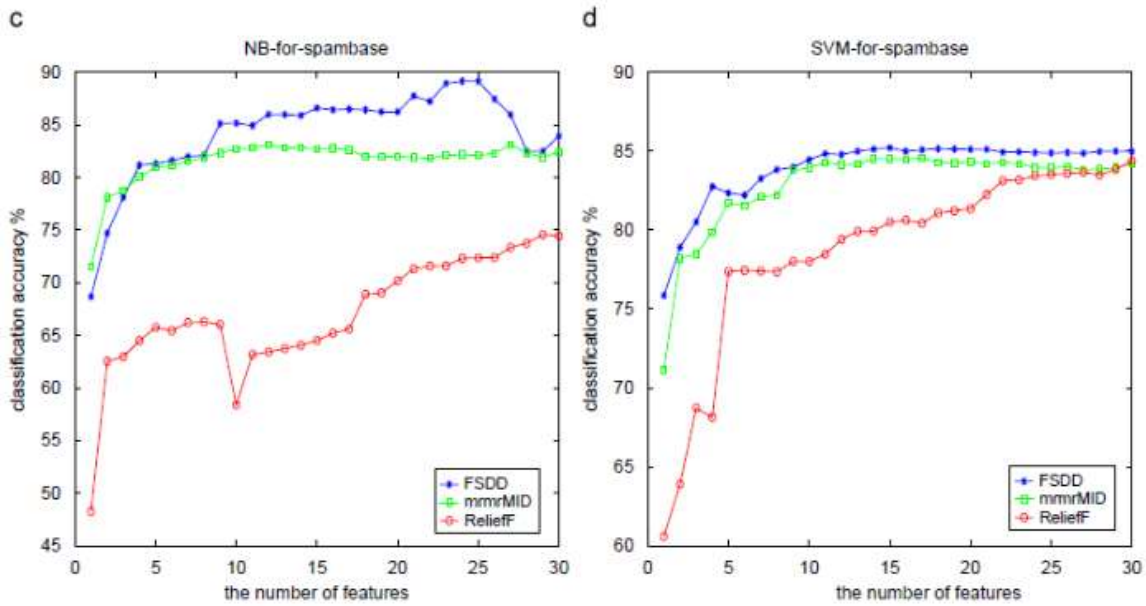


Fig. 1. Two-fold CV classification accuracy for Mfeat with four classifiers.

Los siguientes son los resultados obtenidos por los autores para el conjunto Spambase:




c
NB-for-spambase
classification accuracy %
the number of features
FSDD
mmrMID
ReliefF
d
SVM-for-spambase
classification accuracy %
the number of features
FSDD
mmrMID
ReliefF

También se presentan algunas tablas de precisión, en lugar de gráficos, para algunos otros conjuntos de datos. Cada tabla muestra en esencia los resultados que antes, para cada m se muestra la precisión obtenida con cada clasificador al filtrar el conjunto de datos usando cada algoritmo estudiado. Aquí la tabla para el conjunto de datos **`Wine`**.

Table 3
Ten-fold classification accuracy for Wine

Classifier	mtds	m												
		1	2	3	4	5	6	7	8	9	10	11	12	13
KNN	FSDD	70.23	85.39	90.45	92.14	94.94	96.07	96.07	96.07	95.51	96.07	96.07	97.19	95.51
	mrmrMID	38.76	65.17	75.84	75.84	82.58	80.9	85.39	91.01	91.57	94.38	93.82	93.26	95.51
	ReliefF	69.1	82.58	91.57	92.7	93.26	94.94	95.51	95.51	95.51	95.51	94.38	94.94	95.51
NB	FSDD	79.21	88.76	91.57	94.94	94.38	97.19	97.19	96.07	96.63	96.07	96.07	96.07	97.75
	mrmrMID	57.3	73.6	78.09	79.21	84.27	86.52	88.2	94.94	94.38	94.94	95.51	95.51	97.75
	ReliefF	76.97	88.2	91.01	91.57	93.26	94.38	94.38	94.38	94.94	96.63	96.07	97.19	97.75
DT	FSDD	71.91	87.64	92.7	91.57	95.51	95.51	95.51	95.51	95.51	95.51	95.51	94.94	94.94
	mrmrMID	49.44	66.29	73.6	78.65	76.97	78.65	86.52	92.14	92.14	93.82	94.38	93.82	94.94
	ReliefF	70.79	81.46	92.14	92.7	95.51	95.51	95.51	95.51	95.51	95.51	95.51	94.94	94.94
SVM	FSDD	79.21	88.2	92.7	95.51	96.07	97.75	97.19	97.75	97.75	98.32	98.32	97.75	98.32
	mrmrMID	56.18	75.28	82.02	80.9	83.71	85.39	91.57	94.94	95.51	97.19	96.07	97.75	98.32
	ReliefF	75.84	87.64	92.7	92.7	95.51	96.63	97.19	97.19	98.32	98.32	98.32	97.75	98.32

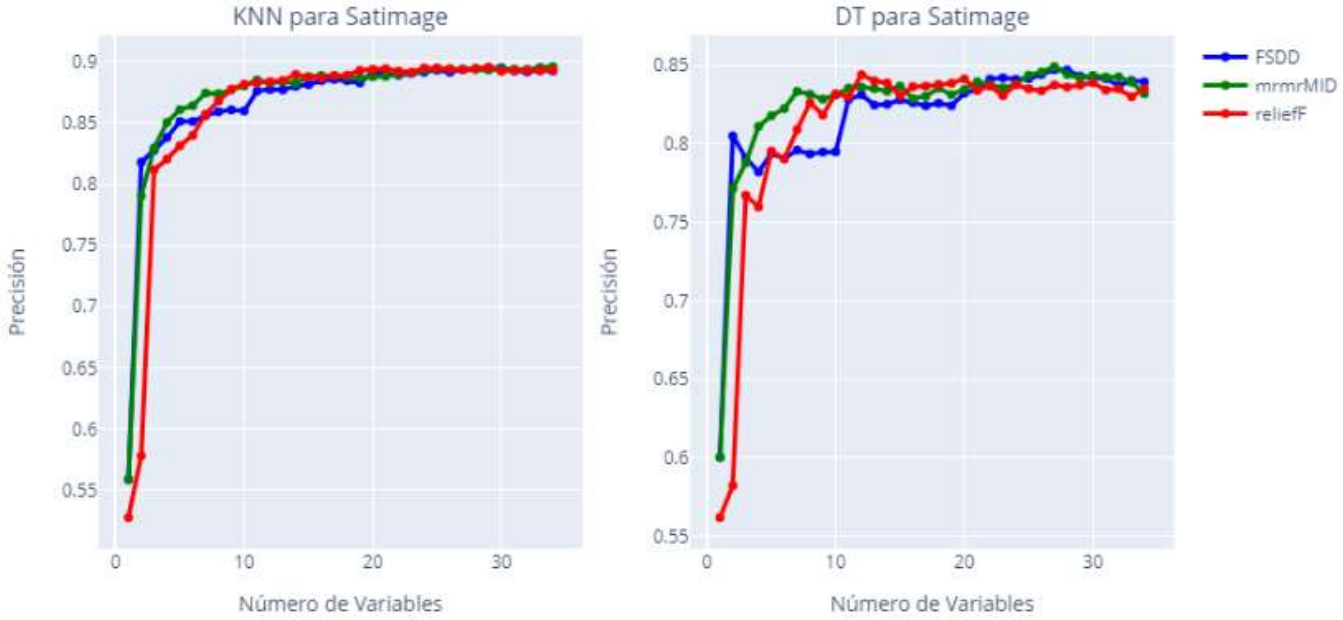
m is the number of features selected. mtds is the abbreviation of the word 'methods'.

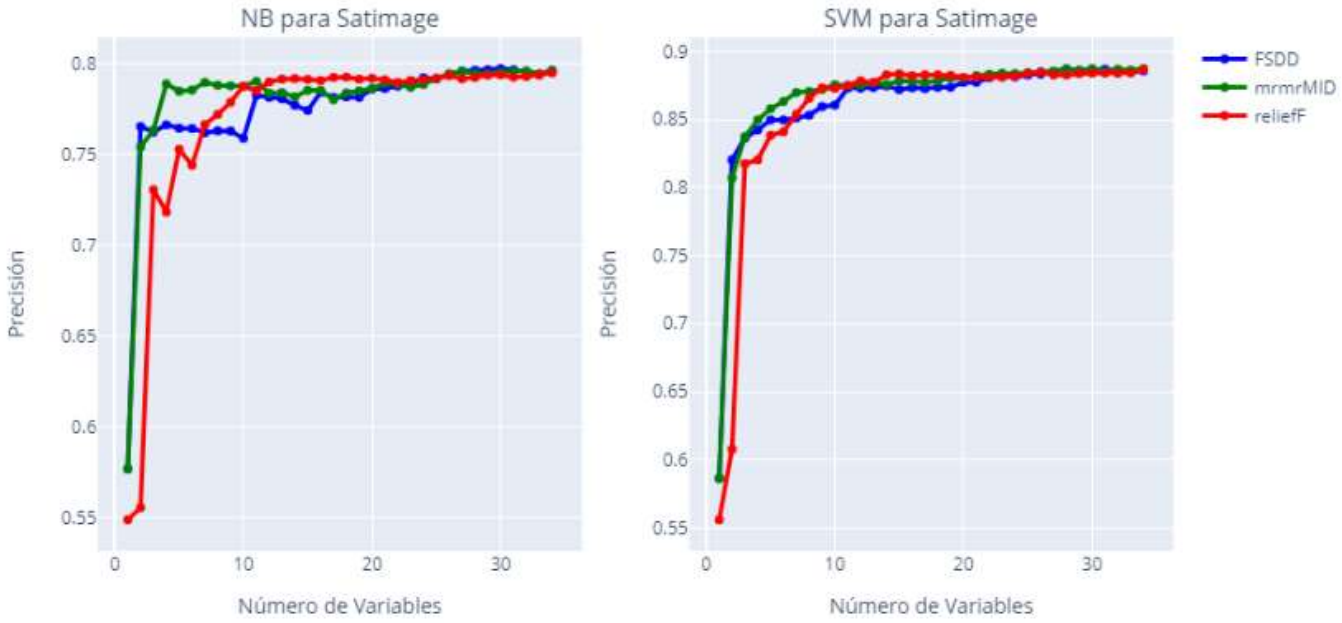
Resultados de mis experimentos

Se intentó programar los pseudo códigos tal y como se han especificado en los artículos (Excepto el de ReliefF, para el cual se usó una biblioteca). De mi parte creo que la implementación que he hecho en python sigue al pie de la letra lo escrito en los artículos. Comparemos los resultados obtenidos con mi implementación con los originales. 😅

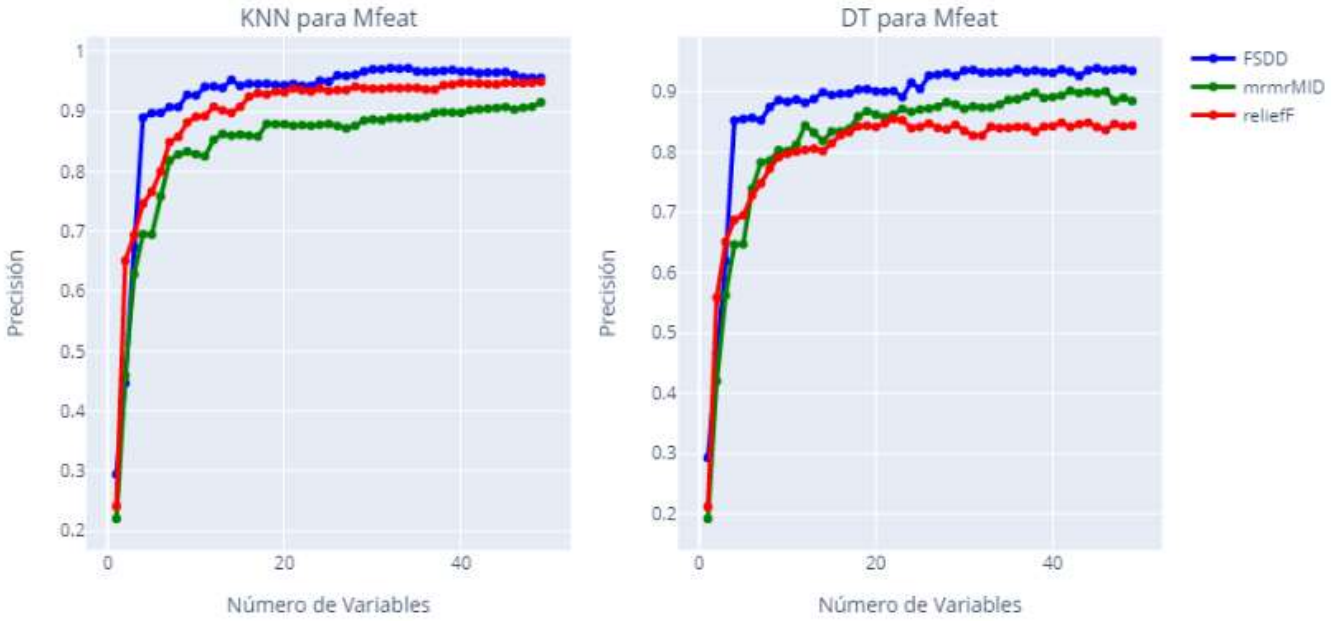
Cualquiera puede acceder al github y darme su opinión en qué se puede reforzar el código 😊

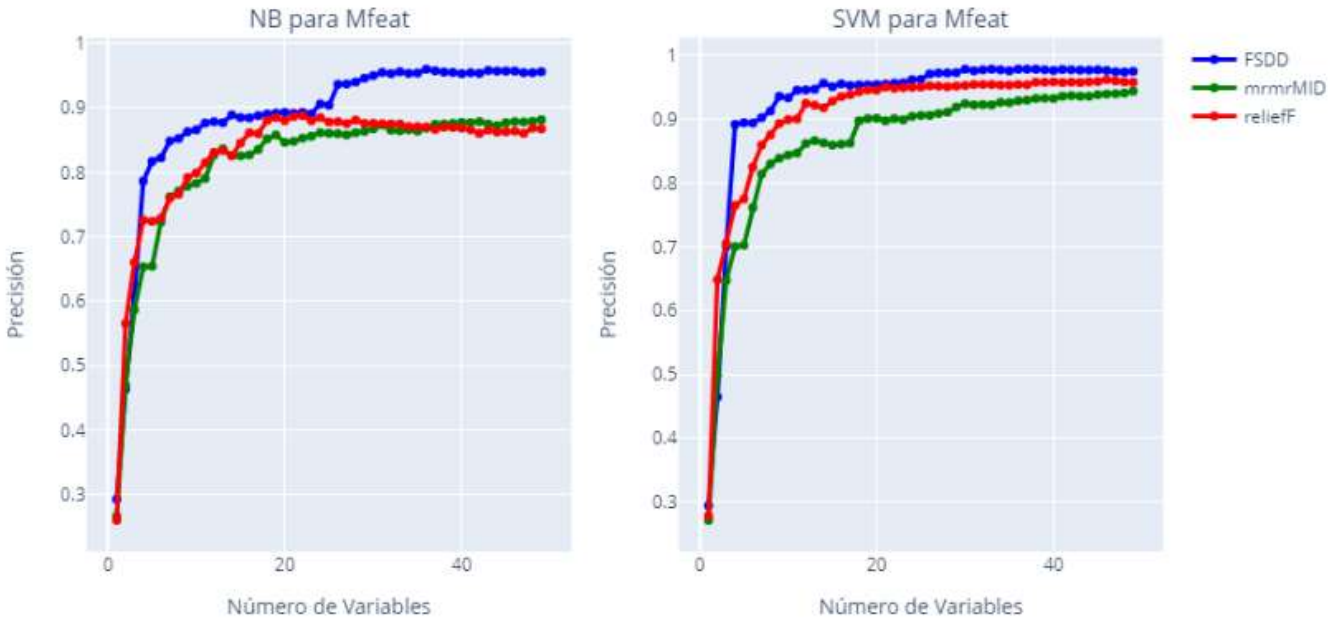
Estos son los gráficos que obtuve al aplicar el mismo proceso al conjunto de datos **`Satimage`**, usando la implementación en python que hice. 🤓



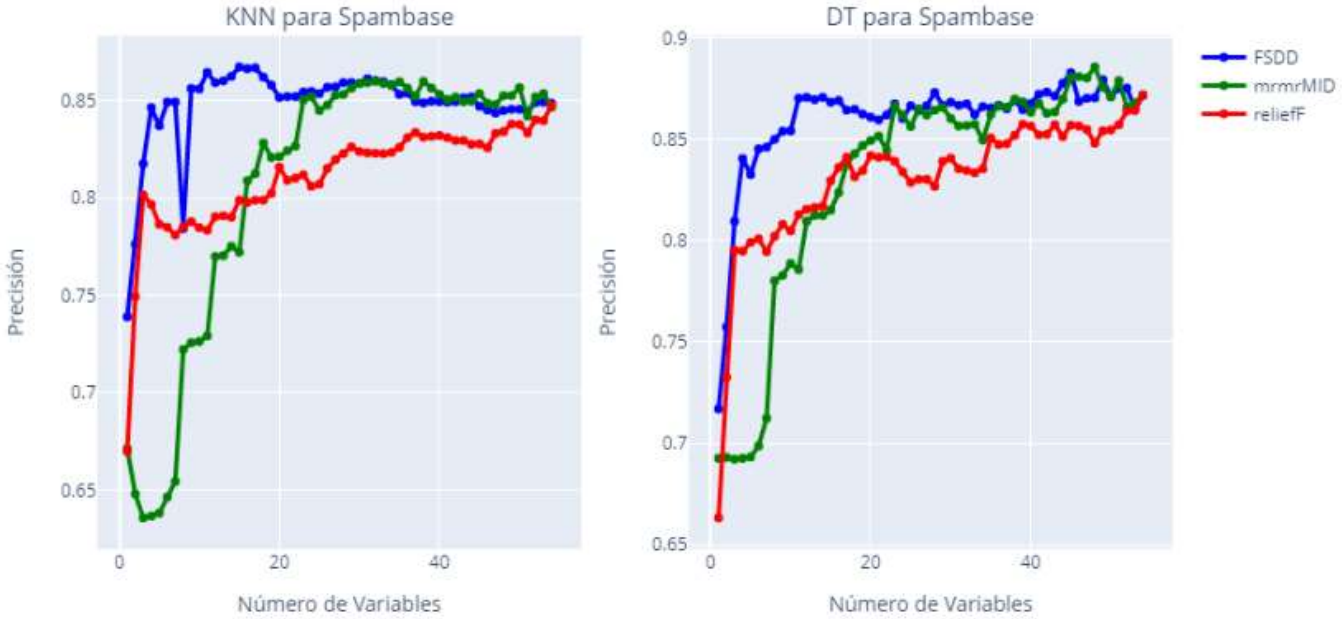

NB para Satimage
SVM para Satimage
Precisión
Número de Variables
FSDD
mrmrMID
reliefF
0.8
0.75
0.7
0.65
0.6
0.55
0.9
0.85
0
10
20
30

Los siguientes son los gráficos que obtuve para el conjunto Mfeat.




NB para Mfeat
SVM para Mfeat
Precisión
Número de Variables
FSDD
mrmrMID
reliefF
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
20
40

Seguidamente los resultados de la réplica para Spambase




NB para Spambase
Precisión
Número de Variables
0.9
0.85
0.8
0.75
0.7
0.65
0.6
0.55
0.5
0.45
0
20
40
SVM para Spambase
Precisión
Número de Variables
0.9
0.85
0.8
0.75
0.7
0.65
0
20
40
FSDD
mrmrMID
reliefF

Finalmente, ésta fue la tabla de precisiones obtenida en la réplica para el conjunto **`Wine`**.

Clasif.	Método	m												
		1	2	3	4	5	6	7	8	9	10	11	12	13
KNN	FSDD	0.798	0.899	0.933	0.944	0.955	0.961	0.955	0.944	0.944	0.949	0.961	0.944	0.938
	mrmrMID	0.798	0.719	0.736	0.803	0.809	0.803	0.787	0.803	0.809	0.904	0.927	0.944	0.938
	reliefF	0.528	0.781	0.803	0.949	0.966	0.944	0.921	0.916	0.933	0.91	0.927	0.91	0.938
NB	FSDD	0.753	0.848	0.815	0.876	0.949	0.933	0.933	0.938	0.938	0.916	0.933	0.927	0.921
	mrmrMID	0.753	0.719	0.736	0.86	0.854	0.82	0.843	0.831	0.809	0.888	0.899	0.91	0.921
	reliefF	0.511	0.719	0.764	0.938	0.949	0.944	0.944	0.949	0.938	0.916	0.921	0.916	0.921
DT	FSDD	0.787	0.899	0.91	0.938	0.944	0.966	0.961	0.966	0.972	0.972	0.966	0.972	0.978
	mrmrMID	0.787	0.775	0.831	0.854	0.893	0.876	0.871	0.893	0.916	0.938	0.972	0.983	0.978
	reliefF	0.567	0.775	0.826	0.944	0.961	0.944	0.933	0.927	0.938	0.938	0.949	0.949	0.978
SVM	FSDD	0.803	0.893	0.916	0.949	0.966	0.972	0.972	0.972	0.978	0.972	0.966	0.978	0.978
	mrmrMID	0.803	0.736	0.809	0.86	0.904	0.893	0.871	0.916	0.933	0.955	0.972	0.989	0.978
	reliefF	0.556	0.781	0.82	0.938	0.955	0.949	0.949	0.955	0.944	0.955	0.961	0.944	0.978

Problemas y Limitaciones

- Los artículos no brindaban referencias para accesar al código original que fue usado por los autores 😢

- Se menciona que la implementación fue hecha en matlab. Por esta razón se recurrió a una implementación en python, que puede tener errores o aspectos que no he considerado todavía. Es muy probable que esto haya afectado los resultados. De hecho existen algunos hiperparámetros en el método ReliefF de python que no se pueden controlar, como el número de vecinos a seleccionar en cada iteración. Por falta de tiempo no se implementó este método y en su lugar se utilizó una biblioteca de python hecha para éste.

Conclusiones

- La motivación teórica detrás de la definición de la métrica FSDD es muy intuitiva y la fórmula desarrollada por los autores permite obtener una nueva métrica para calificar variables dentro del problema de clasificación planteado.

- A pesar de no contar con la implementación explícita realizada por los autores, se logró emular el código apropiado siguiendo las descripciones de los algoritmos hechas en la bibliografía. Por detalles del software utilizado en algunos casos los experimentos efectuados originalmente no se han podido emular con exactitud, aunque esto también es razonable debido a la diferencia desde el punto de vista del software empleado.

- A pesar de esta limitación, los resultados obtenidos en este proyecto muestran un desempeño relativamente superior de la métrica FSDD en muchas instancias, y en el peor de los casos las tres métricas terminan siendo equivalentes con respecto al objetivo de precisión.

Bibliografía

- Artículo de Referencia:
    - J.Liang, S.Yang, A.Winstanley, *Invariant optimal feature selection: a distance discriminant and feature ranking based solution*, Pattern Recognition 41 (2008) 1429–1439.
- Otros artículos utilizados en este trabajo:
    - Jimin Lee, Nomin Batnyam, Sejong Ohn, *RFS: Efficient feature selection method based on R-value*. Comp. in Bio. and Med. 43, Issue 2, (2013) 91–99.
    - C.Ding, H.Peng, *Minimum redundancy feature selection from microarray gene expression data*. Proceedings of the IEEE Computer Society, Conference on Bioinformatics, IEEE Computer Society, 2003, p.523.
    - J.Liang, S.Yang, A.Winstanley, *Invariant optimal feature selection: a distance discriminant and feature ranking based solution*, Pattern Recognition 41 (2008) 1429–1439.
    - M.Robnik-Sikonja, I.Kononenko, *Theoretical and empirical analysis of ReliefF and RReliefF*, Mach.Learn. 53 (2003) 23–69.