UCLV
Universidad Central
"Marta Abreu" de Las Villas

MFC
Facultad de Matemática
Física y Computación

Departamento de matemática

TRABAJO DE DIPLOMA

"Reducción de la dimensión no lineal en datos de microarreglos de ADN para el cáncer"

Autor: Lorenzo Antonio Pérez Carballo

Tutores: MSc. Yunier E. Tejeda Rodríguez & Dr. Carlos

Santa Clara, Mes y Año

Me gustaría dedicar este trabajo a mi querida familia y en especial a mi abuela por lo insustituible que ha sido y será siempre para mi. A esta larga listas de personas que quisiera dedicarle la investigación se le suman mis profesores de la carrera, resaltando a mis tutores. Y no por último menos importante, a todos mis amigos.

AGRADECIMIENTOS

Empezaré agradeciendo a mi tutor Yunier E. Tejeda Rodríguez por darme un tema de investigación que se convirtió en tesis y no solo eso, también en el trabajo que más me ha gustado. Luego, a mis profesores de la carrera que sin ellos seria imposible imaginarme ejerciendo otra profesión. Agradecerle a mis amigos, en especial a mis compañeros de beca, compañeros de aula y a los que viven conmigo en Sancti Spíritus. Dejé para el final a mi familia porque a ellos no les tengo que agradecer nada ya que siempre me darán su apoyo incondicional este en el lugar que esté. Gracias a todos y estaré agradecido de por vida.

SÍNTESIS

En la presente investigación, se desea reducir la dimensión de los datos de microarreglos de ADN para el cáncer mediante métodos de extracción de características no lineales, para obtener un procedimiento que permita distinguir entre muestras cancerígenas y no cancerígenas. Para lograr esto se estudian las potencialidades de los métodos de reducción de la dimensión siguientes: "Métodos de selección de características", "Análisis de componentes principales kernel" y "Descomposición matricial CUR". Luego, dichos métodos se utilizan en las metodologías propuestas "reducción de dimensión no lineal", "doble reducción de dimensión no lineal" y "doble reducción de dimensión no lineal en forma distribuida" de forma conjunta, para obtener un modelo de clasificación. En la etapa final del trabajo, se utilizan las metodologías propuestas sobre seis bases de datos para luego hacer una discusión de los resultados a partir de la complejidad de los datos y análisis horizontales. De una manera breve decir que esta investigación es parcial ya que debido al gran campo de estudio del tema tratado se realizarán próximamente otras búsquedas de solución para distinguir entre muestras cancerígenas y no cancerígenas.

ABSTRACT

In the present investigation, it is desired to reduce the size of DNA microarray data for cancer by nonlinear extraction methods, to obtain a procedure that allows to distinguish between carcinogenic and non-carcinogenic samples. To achieve this, the potentials of the following dimension reduction methods are studied: "Feature selection methods", "Principal component analysis kernel" and "Matrix decomposition CUR". Then, these methods are used in the proposed methodologies "reduction of non-linear dimension", "double reduction of non-linear dimension" and "double reduction of non-linear dimension in distributed form" jointly, to obtain a classification model. In the final stage of the work, the proposed methodologies are used on six databases to then make a discussion of the results based on the complexity of the data and horizontal analysis. In a brief way to say that this research is partial because due to the large field of study of the subject treated other solutions will soon be conducted to distinguish between carcinogenic and non-carcinogenic samples.

ÍNDICE

FIGURAS

TABLAS

INTRODUCCIÓN

Antecedentes

Los microarreglos son superficies sólidas donde se coleccionan los fragmentos de ADN en pozos, los cuales pueden ser de vidrio, plástico e incluso silicona, también son llamados chips de ADN o microarrays, en los anexos B.1 se muestran algunas imágenes. Estos se crearon como una solución rápida para analizar todos los genes de un organismo al mismo tiempo, muy diferente a lo se hacia antes de su llegada ya que se debían estudiar uno por uno todos los genes diferentes. Entre las aplicaciones de esta tecnología está analizar y monitorear la expresión genética, conocer el mecanismo molecular mediante el cual actúa un medicamento o fármaco, diagnosticar enfermedades congénitas en particular el cáncer, entre otras aplicaciones (1).

Existen microarreglos de diferentes tipos: microarreglos de Oligonucleótidos, Microarreglos de proteínas, Microarreglos de tejidos y por último los utilizados en la investigación, los microarreglos de ADNc o ADN complementario, siendo estos los más populares. Cabe resaltar que todos los tipos de microarreglos posee el mismo fundamento.

Los microarreglos de ADN se basan en la comparación de dos muestras biológicas diferentes, sondas y muestras a analizar. El ADN utilizado para la comparación es el se que obtiene de la retrotranscripción del ARNm (ARN mensajero). Dicha tecnología realiza los siguientes pasos: Extracción del ARN, marcado del ARN y síntesis de ARNc, impresión de Microarreglos, hibridación del ARNc y por último la lectura de Microarreglos (en los anexos B.2 se muestra el paso final).

Esta información es llevada a una matriz de orden $n \times p$ siendo n el número de muestras y p el número de genes medidos permitiendo así el estudio de diferentes problemas: (I) distinguir entre muestras cancerígenas y no cancerígenas, (II) clasificar los diferentes tipos de cáncer e (III) identificar subtipos de cáncer que pueden progresar agresivamente.

Debido al costo y las dificultades experimentales que suponen la obtención de esta información genética, resulta común encontrar estudios donde la cantidad de muestras es mucho menor

en relación con el número de genes. Estos problemas de alta dimensión en los datos se conocen como "large p small n" y representan un reto para los métodos estadísticos tradicionales resultando difícil o imposible su aplicación, de ahí que es necesaria la búsqueda de alternativas para lidiar con estos. Para resolver estos problemas se puede reducir la dimensión en los datos mediante la selección de características o la extracción de características que son de nuestro interés en la investigación, en particular, los métodos de extracción de características no lineales.

Es por eso que nuestro problema de investigación se centra en:

Problema científico

¿Cómo reducir la dimensión de los datos de microarreglos de ADN para el cáncer que permita distinguir entre muestras cancerígenas y no cancerígenas?

A partir del problema científico se trazan el objetivo general y los objetivos específicos.

Objetivo general

Reducir la dimensión de los datos de microarreglos de ADN para el cáncer mediante métodos de extracción de características no lineales, para obtener un procedimiento que permita distinguir entre muestras cancerígenas y no cancerígenas.

Objetivos específicos

1. Estudiar los métodos de selección de características en los datos de microarreglos de ADN para el cáncer.

2. Estudiar el método Análisis de Componentes Principales Kernel en datos de microarreglos de ADN para el cáncer.

3. Estudiar la descomposición matricial CUR en los datos de microarreglos de ADN para el cáncer.

4. Proponer una metodología que permita reducir la dimensión de los conjuntos de datos mediante el Análisis de Componentes Principales Kernel para obtener un modelo de clasificación que pronostique esta enfermedad.

5. Proponer una metodología que permita reducir la dimensión de los conjuntos de datos mediante la descomposición matricial CUR y luego aplicar Análisis de Componentes Principales Kernel para obtener un modelo de clasificación que pronostique esta enfermedad.

6. Proponer una metodología que permita distribuir los conjuntos de datos de microarreglos en diferentes subconjuntos y luego aplicar la metodología anterior a cada uno de ellos combinando sus resultados para obtener un modelo de clasificación que pronostique esta enfermedad.

7. Implementar las metodologías propuestas en el entorno de desarrollo integrado RStudio para que pueda ser empleada en el software R.

CAPÍTULO 1

Métodos de reducción de la dimensión

1.1. Métodos de selección de características

Los métodos de selección de características consisten en seleccionar un conjunto mínimo de características con una mejor precisión predictiva (2). Esto se puede considerar como un problema de búsqueda en un espacio de estado (3), donde cada estado corresponde a un subconjunto de características, y el espacio incluye todos los subconjuntos posibles que se pueden generar. En general, un método de selección de características se basa en dos pasos básicos: generación de características de subconjuntos y evaluación de subconjuntos. En la generación de nuevos subconjuntos, se define un punto de partida y luego una estrategia que se ejecuta en el espacio de búsqueda hasta que se cumple un criterio de detención.

La Tabla 1.1 muestra una comparación entre los diferentes métodos de selección de características en términos de sus características, ventajas y desventajas. Además, presenta cómo se clasifican los métodos de filtro y envolvente. Finalmente, se proporcionan los métodos más utilizados en el análisis de datos de microarreglos. En (4; 5; 6; 7) se hace una revisión de los principales resultados de los métodos de selección de características en los datos de microarreglos.

1.1.1. Selección de características distribuidas

La paralelización de los algoritmos es un enfoque adecuado para aumentar la eficiencia de los programas informáticos. Cuando un algoritmo se paraleliza, se puede ejecutar a la vez en muchos dispositivos de procesamiento diferentes (por ejemplo, un clúster con varios nodos). Los algoritmos distribuidos son una subclase de métodos paralelos que, por lo general, se ejecutan al mismo

Datos de Microarreglos	Desventajas	Ventajas	Características	Métodos	
t-test feature selection. Information gain (IG). Unconditional Mixture Modelling.	Ignora las dependencias de características. Ignora la interacción con el clasificador.	La complejidad del tiempo es $O(n)$, que es baja en comparación con otros métodos. Rápido, escalable y puede usarse con cualquier algoritmo de aprendizaje de manera efectiva.	Extrae las características de los datos sin ningún tipo de aprendizaje involucrado.	Univariado	**Filtro**
Correlation-based feature selection (CFS). ReliefF. minimum Redundancy Maximum Relevance (mRMR). Bayesian networks.	Más lento que las técnicas univariadas. Menos escalables que las técnicas univariadas. Ignora la interacción con el clasificador.	Los modelos tienen dependencias. Independiente del clasificador. Mejor complejidad computacional que los métodos envolventes.	Extrae las características de los datos sin ningún tipo de aprendizaje involucrado.	Multivariado	
Successive Feature Selection (SFS). Sequential forward selection (SFS). Sequential backward elimination (SBE).	Riesgo de sobreajuste. Más propensos que los algoritmos aleatorizados para obtener un óptimo local (búsqueda ambiciosa). Selección dependiente del clasificador.	Sencillo. Interactúa con el clasificador. Los modelos tienen dependencias. Menos intensivo computacionalmente que los métodos aleatorizados.	Usa técnicas de aprendizaje para evaluar qué características son útiles.	Determinista	**Envolvente**
Genetic algorithms (GA). Best Incremental Ranked Subset. Genetic Algorithm-Support Vector Machine.	Complejidad de tiempo exponencial. Selección dependiente del clasificador. Mayor riesgo de sobreajuste que los algoritmos deterministas.	Menos propenso al óptimo local Interactúa con el clasificador. Los modelos tienen dependencias.	Usa técnicas de aprendizaje para evaluar qué características son útiles.	Aleatorizado	
Decision trees. Support Vector Machine based on Recursive Feature Elimination (SVM-RFE). Random forests. Least absolute shrinkage and selection operator (LASSO).	Selección dependiente del clasificador.	Interactúa con el clasificador. Mejor complejidad computacional que los métodos envolventes. Los modelos tienen dependencias.	Combina el paso de selección de características y la construcción del clasificador.	**Embebido**	
SVM-RFE with MRMR. ReliefF, mRMR with GA (R-m-GA). CFS-TGA.	La complejidad del tiempo puede aumentar.	Combina las ventajas de varios enfoques.	Combina dos o más algoritmos de selección de características de diferentes orígenes conceptuales de forma secuencial.	**Híbrido**	

Tabla 1.1: “Comparación de varios enfoques de selección de características”

tiempo. En este caso, los datos se separan y se asignan en diferentes ubicaciones, ejecutándose simultáneamente en procesadores independientes que tienen información limitada sobre lo que están haciendo las otras partes del algoritmo. Los métodos de aprendizaje automático se pueden paralelizar distribuyendo los subconjuntos de datos de entrenamiento a múltiples procesadores, aprendiendo en paralelo y luego combinando los resultados.

Diversos enfoques para paralelizar y distribuir la selección de características se han propuesto recientemente en la literatura (8; 9; 10; 11). Los datos pueden dividirse y distribuirse de dos maneras: horizontalmente (es decir, por instancias) y verticalmente (es decir, por características). La partición horizontal es especialmente adecuada cuando se trata de conjuntos de datos con un gran número de instancias, mientras que la partición vertical es más apropiada para conjuntos de datos con un alto número de características. Transformar el problema de selección de características a gran escala en varios problemas a pequeña escala puede mejorar el tiempo de procesamiento y, a veces, la precisión del modelo (12).

1.2. Extracción de característica no lineal

1.2.1. Análisis de Componentes Principales Kernel

Para poder hablar de Análisis de Componete Principales Kernel, antes debemos describir el Análisis de Componentes Principales para un mejor comprensión. El Análisis de Componentes Principales (**ACP**) es una transformación de base para diagonalizar una estimación de la matriz de covarianza de los datos x_i, $i = 1,\ldots,n$, $x_i \in \mathbb{R}^n$, $\sum_{i=1}^{n} x_i = 0$ (matriz centralizada) definidos como

$$C = \frac{1}{n}\sum_{i=1}^{n} x_i x_i^T$$

Las nuevas coordenadas en el vector propio (Eigenvector), es decir, las proyecciones ortogonales en el Eigenvector, se llaman *componentes principales* (13).

El problema de valores propios $Cu = \lambda u$ implica que todas las soluciones de u deben estar en el espacio generado por el conjunto de vectores $x_1, x_2, \ldots x_n$; por lo cual (el creador del método fue Schölkopf en el 1996), escribió de forma análoga

$$\lambda\,(x_i, u) = (x_i, C)$$

dicho modelo está asociada a la matriz de rotación U, la cual permite calcular las componentes

principales, que no son más que combinaciones lineales de las variables iniciales $z = U^T x$ (14). Pero en el siguiente trabajo se realiza el Análisis de Componentes Principales Kernel por la Descomposición del Valor Singular (**DVS**). Esta descomposición ya fue realizada en el curso anterior (2016-2017) por el actual profesor Miguel A. Gutiérrez Arce (15).

Ya con esta idea se puede explicar brevemente Análisis de Componentes Principales Kernel (**ACPK**). De ahora en adelante, se le llamará por sus siglas. Según (14), es posible sustituir el espacio original de las observaciones X por un espacio provisto de producto punto H mapeado a través de $\phi : X \longmapsto H$ (creado por Schölkopf & Smola en el 2002). Partiendo de la misma suposición sobre la cual se construyó la matriz de covarianza C, la cual implica que todos los datos están centralizados en H, la matriz de covarianza quedaría de la siguiente forma

$$\bar{C} = \frac{1}{n} \sum_{i=1}^{n} \phi\left(x_i\right) \phi^T\left(x_i\right)$$

En este caso debemos encontrar los valores propios no nulos ($\lambda > 0$) y sus respectivos vectores propios v que satisfacen

$$\lambda v = \bar{C} v$$

Como en el método anterior, las soluciones de v deben estar dentro del espacio generado por $\{\phi(x_1), \phi(x_2), \ldots, \phi(x_n)\}$. Entonces

$$\lambda\left(\phi\left(x_k, v\right)\right) = \left(\phi\left(x_k\right), \overline{C} v\right) \tag{1.1}$$

Además, es posible definir los vectores propios en términos de los datos mapeados en H:

$$v = \sum_{i=1}^{n} a_i \phi\left(x_i\right) \tag{1.2}$$

Combinando 1.1 y 1.2, se obtiene

$$\lambda \sum_{i=1}^{n} a_i \left(\phi\left(x_k\right), \phi\left(x_i\right)\right) = \frac{1}{n} \sum_{i=1}^{n} a_i \left(\phi\left(x_k\right), \sum_{j=1}^{n} \phi\left(x_j\right)\right) \left(\phi\left(x_j\right), \phi\left(x_i\right)\right)$$

Definiendo la matriz K como $k_{ij} = \left(\phi\left(x_i\right), \phi\left(x_j\right)\right)$, obtiene

$$n\lambda K a = K^2 a \tag{1.3}$$

donde a denota el vector columna que sintetiza la representación de v dada en 1.2, a través del

conjunto de observaciones mapeadas por ϕ. Debido a la simetría de K, sus vectores propios generan el espacio completo; por tanto

$$n\lambda a = ka$$

genera las soluciones de la ecuación 1.3. De esta forma, los valores propios asociados a α corresponden a $n\lambda$; en consecuencia, cada uno de los v corresponden al mismo ordenamiento de los a. Es necesario trasladar la restricción de $\|v\| = 1$ a los correspondientes vectores propios de K:

$$1 = \sum_{i,j=1}^{n} a_i a_j \left(\phi\left(x_i\right), \phi\left(x_j\right)\right) = \lambda\left(a, a\right)$$

Para la extracción de componentes principales, deben proyectarse los datos mapeados a H sobre los respectivos vectores propios seleccionados. Podemos hacer uso de

$$\left(v, \phi\left(x\right)\right) = \sum_{i=1}^{n} a_i \left(\phi\left(x_i\right), \phi\left(x\right)\right)$$

La centralización de los datos es posible al reemplazar la matriz K por su correspondiente versión centralizada:

$$\widetilde{K} = K - 1_n K - K 1_n + 1_n K 1_n$$

donde 1_n es una matriz cuadrada de tamaño $n \times n$ cuyas entradas son $1/n$. Para los m datos de prueba en el vector t que deben ser mapeados se tiene:

$$k_{ij}^{test} = \left(\phi\left(t_i\right), \phi\left(x_j\right)\right)$$

y su versión centralizada:

$$\widetilde{K^{test}} = K^{test} - 1_n^{'} K - K^{test} 1_n + 1_n^{'} K 1_n$$

donde $1_n^{'}$ es una matriz de tamaño $m \times n$ cuya entrada son $1/m$.

Dentro de los procedimientos para escoger cuantas componentes serán consideradas se encuentra la de seleccionar aquellas cuyo valor propio excede al promedio, es decir,

$$\lambda_k > \frac{1}{r} \sum_{k}^{r} \lambda_k$$

donde λ_k es el k-ésimo valor propio correspondiente de la k-ésima componentes principal y r representa el rango de la matriz de datos en cuestión. En el caso de **ACPK** usando la matriz de correlación se optan por las que tienen sus valores propios mayores que 1, sin embargo, esta técnica puede conducir a ignorar información importante (16). Además, existe la posibilidad de elegir las componentes cuya varianza acumulativa explique un $Q100\,\%$ que garantice un alto comportamiento de la variabilidad total

$$\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{j=1}^{p} \lambda_j} \leq Q \tag{1.4}$$

Las funciones kernel más utilizadas son:

- Kernel Lineal: corresponde al producto punto en el espacio de entrada.

$$k(x,x') = \langle x,x' \rangle$$

- Kernel polinomial: representa la expansión a todos las combinaciones de monomios de orden d.

$$k(x,x') = \langle x,x' \rangle^d$$

- Kernel gaussiano: está contenido dentro de las funciones de base radial.

$$k(x,x') = \exp\left(-\frac{\|x-x'\|^2}{2\sigma^2}\right)$$

- Kernel hiperbólico: está asociado a las funciones de activación de las redes neuronales.

$$k(x,x') = \tanh\left(\xi \langle x,x' \rangle + b\right)$$

los parámetros $\xi > 0$ y $b < 0$ denotan escala y corrimiento, respectivamente.

1.3. Descomposición matricial CUR

La descomposición matricial CUR se emplea para nombrar aquellas descomposiciones matriciales de bajo rango que son explícitamente expresadas en términos de un número pequeño de columnas y/o filas de una matriz de datos, A. Estas descomposiciones permiten aproximar la

matriz A por medio del producto de tres matrices C, U y R donde C y R contienen algunas columnas y filas de A, respectivamente, mientras U es una matriz que se construye cuidadosamente de manera que garantice dicha aproximación.

Se conocen varias descomposiciones CUR que se diferencian en las cotas de error obtenidas y en el criterio para elegir las columnas y filas que forman las matrices C y R (17; 18; 19; 20; 21).

En (18) se propone la descomposición matricial CUR, la cual elige las columnas a incluir en C (similar en R) a partir de un factor de importancia para cada columna de la matriz A. Dicho factor se define a partir de la matriz A y un parámetro de entrada dado por el rango k, como se muestra a continuación:

$$\pi_j = \frac{1}{k}\sum_{p=1}^{k}\left(v_j^p\right)^2, \forall j = 1:n$$

donde v_j^p es la j-ésima componente del p-ésimo vector derecho de A.

A continuación, se muestrea aleatoriamente un pequeño número de columnas de A usando ese factor de importancia como una distribución de probabilidad.

El algoritmo básico para seleccionar las columnas de una matriz denominado ColumnSelect (18) toma como entrada cualquier matriz de orden $m \times n$, un parámetro de rango k y un parámetro de error ε.

El resultado teórico más importante que avala dicho algoritmo establece que con probabilidad al menos 99 %, esta elección de columnas satisface que

$$\|A - P_C A\|_F \leq \left(1 + \frac{\varepsilon}{2}\right)\|A - A_k\|_F$$

donde $P_C A$ denota la matriz de proyection sobre el espacio columna generado por C y A_k es la matriz de rango k más próxima a A en norma de Frobenius (ver en (22) la demostración). De esta forma el resultado garantiza que si A es una matriz cercana a una matriz de rango k entonces con alta probabilidad, el subespacio generado por las columnas de A está próximo al subespacio generado por las columnas de C. Esto justifica el hecho de poder utilizar un método en forma paralela distribuyendo la matriz de datos A en varias matrices C (23).

En (24) se mencionan los métodos experimentales “random”, “exact.num.random”, “top.scores”, “ortho.top.scores” y “highest.ranks” los cuales se encuentran implementados en el paquete rCUR (25). En dicho trabajo comentamos que tales métodos proporcionan la misma precisión que el algoritmo ColumSelect.

1.4. Conclusiones del capítulo

En este capítulo se estudiaron tres métodos de reducción de la dimensión: selección de características, extracción de características no lineales y algoritmos aleatorios. En la selección de características se mostró una tabla donde se evidencia una clasificación muy detallada de estos en datos de microarreglos de ADN. Por otro lado, se enseñó el **ACPK** como método de extracción de características no lineal, mostrando su fundamentación teórica. Por último, se presentó un algoritmo de muestreo aleatorio que resuelve el problema de aproximación de matrices de bajo rango, siendo este la descomposición matricial CUR.

Una vez presentados los métodos procedemos en el siguiente capítulo a proponer tres metodologías en las cuales se fusiona dicha teoría.

CAPÍTULO 2

Metodologías Propuestas

En el siguiente capitulo se proponen tres metodologías para reducir la dimensión de los datos. La primera cuenta con la reducción de la dimensión por el método de **ACPK**. Luego, en la segunda y tercera metodología se utilizan la descomposición matricial CUR y el método de extracción de características no lineales anterior.

2.1. Reducción de la dimensión no lineal - RDNL

La metodología que se propone a continuación consiste en dos etapas:

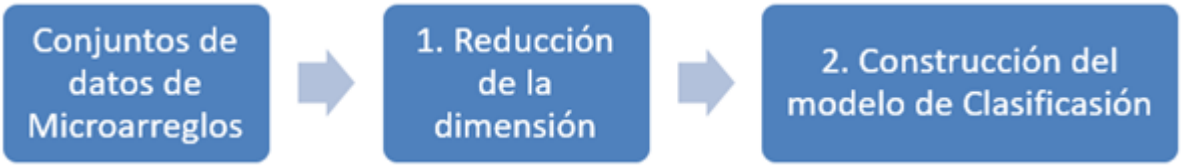


Figura 2.1: "Diagrama de la Metodología Propuesta - RDNL"

1. Reducir la dimensión de los datos de microarreglos utilizando **ACPK**.
2. Calcular un modelo de clasificación por Análisis Discriminante Lineal (**ADL**) a partir de las k-ésimas componentes obtenidas mediante **ACPK**.

Primeramente, pasamos por la obtención de los datos que se van a analizar, pero como el proceso de obtenerlo no forma parte de la investigación, no lo consideramos un paso para la metodología propuesta.

En la etapa uno, se reduce la dimensión de los datos empleado el método de extracción de características no lineal **ACPK**.

Luego, ya en la fase dos, después de haber obtenido las k-ésimas componentes principales (c.p.) mediante **ACPK**, se construye un modelo de clasificación por **ADL** utilizando, en dependencia del conjunto de datos, los siguientes criterios:

- Para los conjuntos de microarreglos que no presentan conjuntos de prueba se utiliza validación cruzada k-campos.
- Para los conjuntos de microarreglos que tienen conjunto de prueba se construye el modelo de clasificación con el conjunto de entrenamiento y se emplea validación holout (26) para el conjunto de prueba.

Por último, se obtiene la matriz de confusión para cada método de extracción y se emplean las medidas de sensibilidad (Se), especificidad (Es) y exactitud (Ex) para determinar cuan bueno es el modelo de clasificación.

Estas, se describen en términos de positivos verdaderos (PV), negativos verdaderos (NV), negativos falsos (NF) y positivos falsos (PF):

$$Se = \frac{PV}{PV + NF};\ 0 \leq Se \leq 1$$

$$Es = \frac{NV}{NV + PF};\ 0 \leq Es \leq 1$$

$$Ex = \frac{PV + NV}{PV + NV + NF + PF};\ 0 \leq Ex \leq 1$$

La sensibilidad y especificidad son medidas que permiten indicar que el modelo de clasificación soluciona el problema del desbalance de las clases, mientras que la exactitud muestra que dicho modelo enmienda el problema de la complejidad de los datos.

2.2. Doble reducción de la dimensión no lineal - DRDNL

A continuación, se presenta la metodología que consiste de tres etapas para la obtención de un modelo de clasificación:

1. Reducir la dimensión de los datos de microarreglos utilizando la descomposición de la matriz CUR para obtener la matriz C.

2. Reducir la dimensión de la matriz C usando el método **ACPK**.

3. Calcular un modelo de clasificación por **ADL** a partir de las k-ésimas componentes obtenidas mediante **ACPK**.

En el Figura 2.2 se muestra en un diagrama de progreso, las siguientes fases de la metodología propuesta.

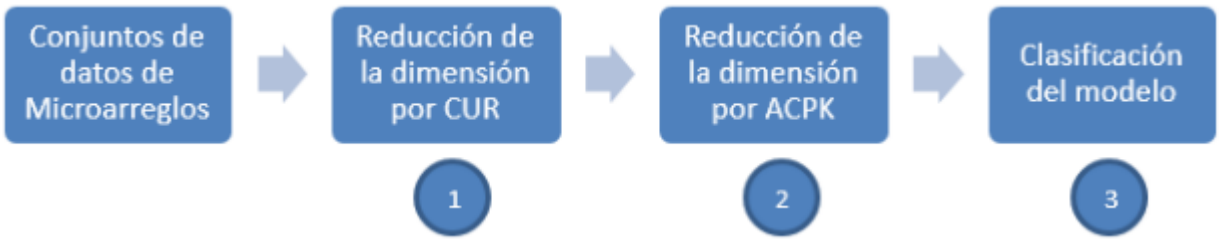


Figura 2.2: "Diagrama de la Metodología Propuesta - DRDNL"

A partir de la matriz C se reduce la dimensión empleando el método de extracción de características no lineal **ACPK**.

Luego de obtener las k-ésimas componentes mediante el método anterior, se construye un modelo de clasificación por **ADL**, utilizándose el criterio de validación cruzada k-campos.

Posteriormente se obtiene la matriz de confusión para cada método de extracción y se emplean las medidas de sensibilidad (Se), especificidad (Es) y exactitud (Ex) para determinar cuan bueno es el modelo de clasificación.

2.3. Doble reducción de la dimensión no lineal en forma distribuida - DRDNLD

En este estudio, seguimos la metodología propuesta en (23) con el propósito de realizar una distribución de datos de microarreglos vertical para la reducción de dimensión. El procedimiento es especialmente adecuado para la aplicación a datos de microarreglos porque facilita una relación más equilibrada entre las características y las instancias, lo que también contribuye a evitar el sobreajuste y la reducción del tiempo de clasificación de ejecución.

El algoritmo consiste en dividir el conjunto de datos en pequeños subconjuntos disjuntos y luego aplicar la metodología de reducción de dimensión dos veces a cada uno de ellos. Este algoritmo cuenta con cinco pasos:

1. Particionar el conjunto de datos de microarreglos en varios subconjuntos pequeños disjuntos $(D_1, D_2, \ldots, D_n)$.

2. Reducir la dimensión de los subconjuntos$D_i \forall i = 1 : n$ usando la descomposición de la matriz CUR para obtener la matriz i-th C para cada partición D_i.

3. Reducir la dimensión de la matriz i-th C $\forall i \in 1, \ldots, N$ por el método **ACPK**.

4. Calcular el modelo de clasificación i-ésimo $\forall i = 1 : n$ mediante algoritmos de aprendizaje automático a partir de los componentes k-ésimo obtenidos mediante los métodos anteriores.

5. Seleccionar el modelo que tenga el error de clasificación erróneo más bajo.

La Figura 2.3 muestra estos pasos. A continuación se muestra una explicación sintetizada de lo que ocurre en cada paso de la metodología

2.4. Conclusiones del capítulo

En este capítulo se propusieron tres metodologías que resuelven el problema de la reducción de la dimensión en datos de microarreglos de ADN para obtener un modelo de clasificación que pronostique si un paciente padece o no el cáncer. La primera metodología consiste en dos pasos: reducir la dimensión mediante el **ACPK** y obtener un modelo de clasificación por **ADL**. En cambio, la segunda metodología (**DRDNL**) parte de la descomposición matricial CUR para reducir la dimensión y luego se aplica la metodología **RDNL**. Por último, la metodología **DRDNLD** consiste en distribuir el conjunto de datos en forma vertical y luego aplicar la metodología **DRDNL** obteniendo el mejor modelo de todas las particiones.

En el siguiente capítulo se aplicaran las tres metodologías propuestas, obteniendo resultados para cada una de ellas.

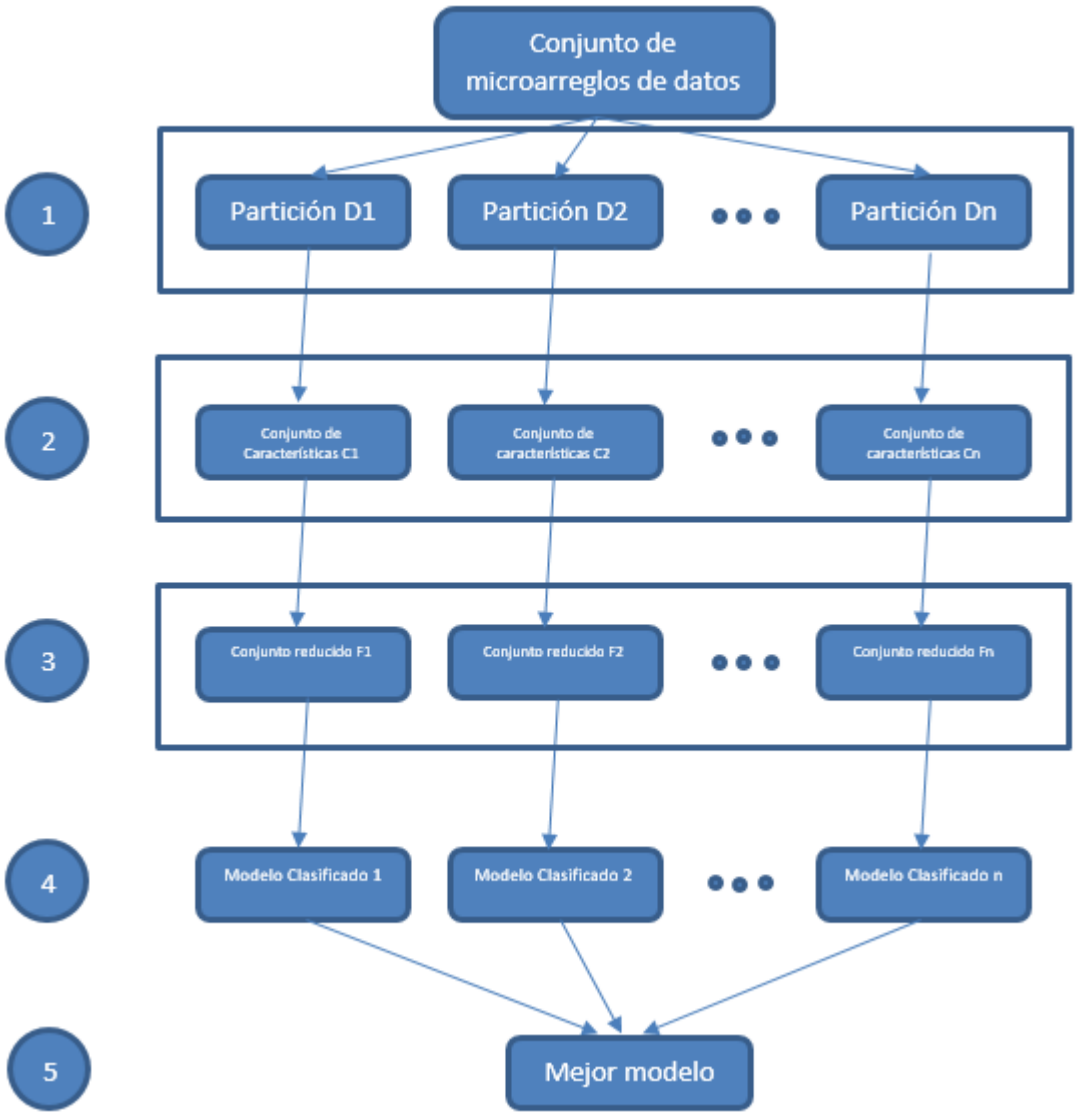


Figura 2.3: "Diagrama de la Metodología Propuesta - DRDNLD"

CAPÍTULO 3

Resultados y discusiones

3.1. Conjuntos de datos de microarreglos de ADN para el cáncer

En esta investigación se estudian 6 conjuntos de datos de microarreglos binarios estudiados por (5). En la Tabla 3.1 se muestra una breve descripción de estos conjuntos de datos. Siendo n y p el número de muestras y genes respectivamente, en tanto IR representa la tasa de desbalance definida como la cantidad de muestras de clases negativas dividido por la cantidad de muestras de clases positivas. Consecuentemente, $F1$ simboliza la máxima de las tasas discriminantes de Fisher (26) que puede ser calculada a través de la siguiente relación:

$$F1 = \max_{i,j=1:n} \left\{ \frac{\left(\mu_i - \mu_j\right)^2}{\sigma_i^2 + \sigma_j^2} \right\}, \; donde \; n = p$$

Los conjuntos de datos Colon, DLBCL y Ovarian fueron descargados del repositorio Kent

Conjunto de datos	n	p	IR	$F1$
CNS/Embrional-T	60	7 129	1.86	0.45
Colon	62	2 000	1.82	1.08
DLBCL	47	4 026	1.04	2.91
GLI-85	85	22 283	2.27	2.35
Ovarian	253	15 154	1.75	6.94
SMK-CAN-187	187	19 993	1.08	0.41

Tabla 3.1: “Descripción de los conjuntos de datos”

Ridge Bio-Medical Repository, desde el Agency for Sciency, Technology and Research (27), el conjunto de datos CNS/Embrional-T fue descargado del repositorio de Dataset Repository, desde el Bioinformatics Research Group of Universida Pablo de Olavide (28) mientras que los conjuntos de datos GLI-85 y SMK-CAN-187 fueron descargados del repositorio de Feature Selction Dataset, desde Arizona State University (29).

3.2. Resultados de la clasificación

Los resultados de la investigación son obtenidos en el software R (en su versión 3.4.3) utilizando los siguientes paquetes: caret (6.0-76), kernlab (0.9-25), rCUR (1.3), foreach (1.4.3) y doSNOW (1.0.12).

En el ambiente de entorno integrado RStudio (versión 1.1.442) se implementan las metodologías mencionadas en el **capítulo 2** a partir de las funciones "nldr", "nldr2" y "nldr2d". Los pseudocódigos de estas funciones se pueden encontrar en los **Anexos** B.3, B.4 y B.5, en ese orden. Para el cálculo de las componentes principales por medio de **ACPK** se utilizan diferentes funciones kernel del paquete kernlab y para reducir la dimensión de los datos mediante la descomposición de la matriz CUR, se usa el paquete rCUR. Por otro lado, para realizar ejecuciones distrivuidas, el paquete foreach se usado con un clúster de 10 procesadores virtuales. Finalmente, el paquete caret se utiliza para validar el modelo de clasificación calculado.

En la **Tabla** 3.2 se muestran todos los resultados obtenidos en la investigación por los indicadores sensibilidad, especificidad y exactitud. Para dar una explicación más detallada de la tabla partimos de la primera fila, la cual contiene las metodologías o métodos especificando la función kernel que es utilizada en cada caso. Luego la tabla continúa con seis filas las cuales representan los conjuntos de datos que fueron entrenados. Además, cada fila se divide en tres para así poder colocar cada indicador por conjunto de datos, por tanto la tabla muestra los tres indicadores en cada conjunto de datos contra los diferentes métodos utilizados y sus respectivas funciones kernel.

3.3. Discusión

Para tener una primera imagen de los datos de la investigación los cuales presentan problemas de dimensionalidad, desbalance (IR) y solapamiento entre las clases (F1) se decidió ordenar estos datos de mayor a menor complejidad. Dicha lista fue numerada según los problemas anteriores y luego se dio un orden total de complejidad, como muestra la tabla 3.3.

Métodos		**RDNL**				**DRDNL**				**DRDNLD**			
		Lineal	Polin.	Gauss.	Tanh.	Lineal	Polin.	Gauss.	Tanh.	Lineal	Polin.	Gauss.	Tanh.
Colon	Se	0.82	0.82	0.55	0.46	0.82	0.77	0.36	0.64	0.77	0.77	0.68	0.68
	Es	0.88	0.88	0.65	0.90	0.88	0.88	0.78	0.88	0.93	0.90	0.73	0.90
	Ex	0.86	0.86	0.61	0.74	0.86	0.84	0.63	0.79	0.87	0.86	0.71	0.82
CNS	Se	0.24	0.33	0.48	0.33	0.04	0.09	0.24	0.24	0.29	0.24	0.57	0.19
	Es	0.62	0.64	0.44	0.78	0.72	0.67	0.54	0.74	0.67	0.72	0.54	0.74
	Ex	0.48	0.53	0.45	0.62	0.48	0.47	0.43	0.57	0.53	0.55	0.55	0.55
DLBCL	Se	1.00	1.00	0.78	0.52	0.87	0.87	0.70	0.91	0.91	0.91	0.74	1.00
	Es	0.75	0.75	0.83	0.46	0.67	0.67	0.54	0.83	0.75	0.75	0.58	0.83
	Ex	0.87	0.87	0.82	0.49	0.77	0.77	0.62	0.87	0.83	0.83	0.66	0.91
GLI-85	Se	0.77	0.81	0.42	0.73	0.73	0.73	0.58	0.81	0.81	0.81	0.39	0.77
	Es	0.88	0.88	0.53	0.92	0.92	0.92	0.64	0.92	0.90	0.90	0.59	0.92
	Ex	0.85	0.86	0.49	0.86	0.86	0.86	0.62	0.88	0.87	0.87	0.53	0.87
Ovarian	Se	0.79	0.79	0.52	0.86	0.79	0.79	0.52	0.85	0.79	0.79	0.65	0.86
	Es	0.93	0.92	0.53	0.98	0.91	0.91	0.55	0.94	0.94	0.94	0.74	0.96
	Ex	0.88	0.88	0.53	0.93	0.87	0.87	0.54	0.92	0.89	0.89	0.71	0.93
SMK	Se	0.67	0.62	0.52	0.54	0.66	0.66	0.49	0.63	0.59	0.59	0.50	0.52
	Es	0.77	0.74	0.58	0.76	0.77	0.77	0.54	0.75	0.79	0.79	0.56	0.71
	Ex	0.72	0.69	0.55	0.66	0.72	0.72	0.51	0.70	0.70	0.70	0.53	0.62

Tabla 3.2: “Resultado para el LDA en los conjuntos de datos”

Conjunto de datos	Dimensionalidad de los datos	*IR*	$F1$	Complejidad Total
CNS/Embrional-T	3	5	5	5
Colon	1	4	4	2
DLBCL	2	1	2	1
GLI-85	6	6	1	6
Ovarian	4	3	3	3
SMK	5	2	6	4

Tabla 3.3: "Orden según la complejidad total"

De esta manera es fácil detectar en un orden jerárquico, cuales de los conjuntos de datos presenta mayor o menor complejidad según los indicadores que se midieron anteriormente. Por lo tanto, los conjuntos que presentan mayor complejidad son GLI-85 y CNS, mientras que los de menor complejidad son DLBCL y Colon. De esta forma se pudieran considerar tres grupos para agrupar o caracterizar los conjuntos de datos: *Nivel alto de complejidad* (GLI-85 y CNS), *Nivel medio de complejidad* (Ovarian y SMK) y *Nivel bajo de complejidad* (DLBCL y Colon). El grupo de mayor complejidad presenta altos niveles de desbalance, mientras que el de menor complejidad presentan bajos niveles de dimensionalidad. Sin embargo, el Nivel medio de complejidad es un conjunto que por estar en la media presenta casos de los dos tipos siendo destacable el bajo desvalanse y la alta dimensionalidad.

Para las tres metodologías creadas se hace variar la función kernel logrando así una mayor variedad de resultados. Lo que nos lleva a querer detectar la mejor función kernel, para esto se confeccionó la gráfica 3.1 en la cual la función tangente hiperbólica es la que obtiene mejores resultados, siendo destacable el 91 % y 93 % para DLBCL y Ovarian, respectivamente. Mientras que la función Gaussiana con un argumento de $sigma = 0{,}1$ obtiene los peores resultados de la investigación. No obstante, esta opinión es de manera general porque cuando se analiza de un punto de vista más puntual, la función lineal y polinómica de grado uno obtiene los mejores resultados para Colon (87 %) y SMK (70 %). El solapamiento entre las clases influye sustancialmente en la exactitud como es en los casos de CNS y SMK los cuales están en los Niveles altos y medios de complejidad, respectivamente, mientras que los conjuntos de Nivel bajo de complejidad presentan altos niveles de Exactitud. Cabe destacar como Ovarian siendo un conjunto que está dentro de los Niveles medio de complejidad tiene altos indices de exactitud, lo cual da un grado de potencia del método utilizado.

De igual manera se analizan los indicadores de sensibilidad y especificidad, los cuales se pueden encontrar en las **gráficas** 3.2 y 3.3, en ese orden. Para la sensibilidad la función kernel

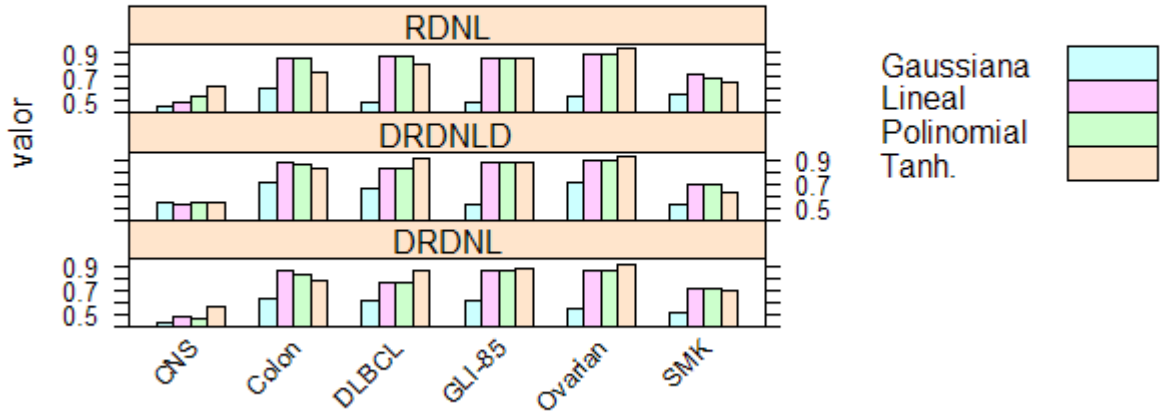


Figura 3.1: "Medidas del indicador Exactitud según las metodologías propuestas"

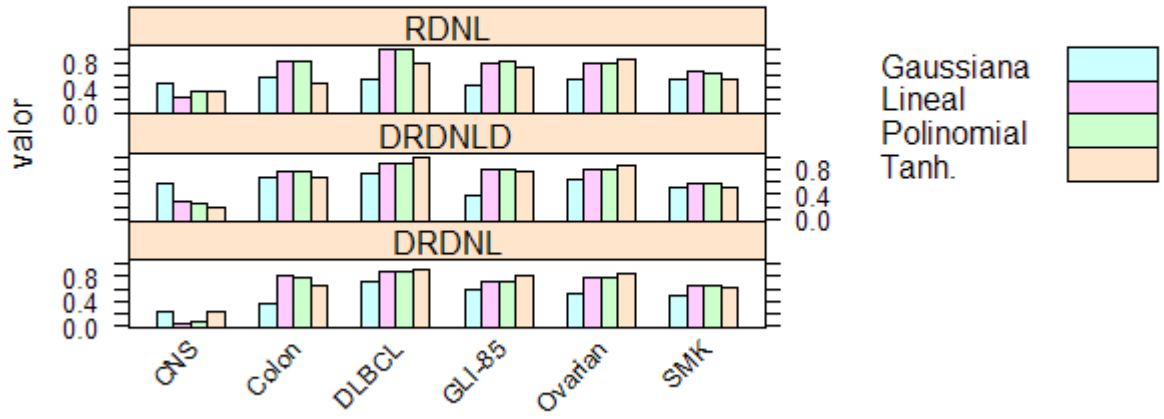


Figura 3.2: "Medidas del indicador Sensibilidad según las metodologías propuestas"

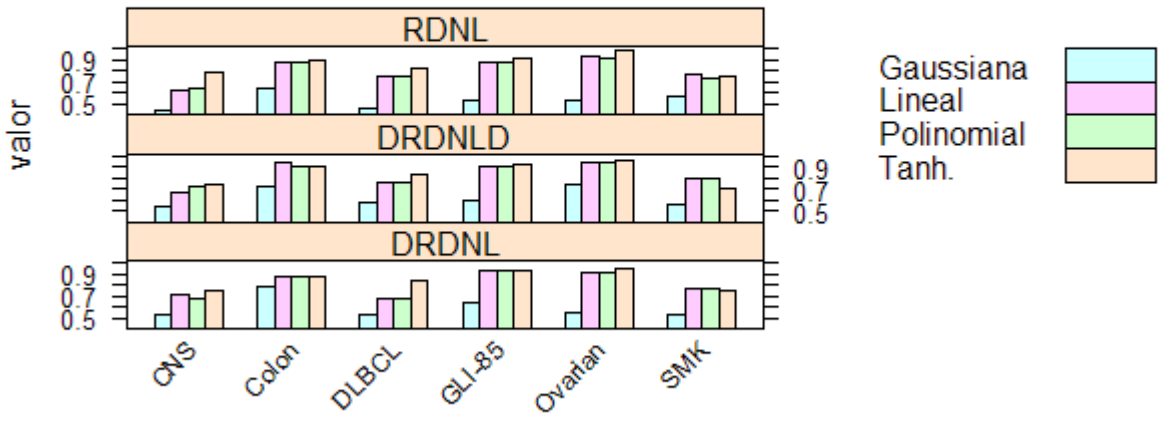


Figura 3.3: "Medidas del indicador Especificidad según las metodologías propuestas"

dominante es la polinomial con argumento degree (grado, traducción al español) igual uno ya que esta obtiene los resultados más altos, como puede ser en el caso de DLBCL (100 %) y Colon (87 %) aplicando DRDNLD en ambos casos. A diferencia del caso de la exactitud, la tangente hiperbólica obtiene los resultados más bajos con respecto a la sensibilidad en la mayoría de los conjuntos. Mientra que la especificidad se comporta parecido a la exactitud al momento de seleccionar la función kernel que mejor resultados obtiene ya que la tangente hiperbólica es superior en todos los conjuntos (con todos los métodos) exceptuando el caso de SMK y la gaussiana devuelve los peores resultados aplíquese el conjunto y método que se quiera. Analizando ambos indicadores según el desbalance de las clases el resultado que se obtiene es positivo ya que el conjunto de datos con más desbalance de clases es GLI-85 y los resultados de la sensibilidad es de un 88 % utilizando el método DRDNL con la función tanh y la especificidad es de un 92 % con el mismo método y función kernel anterior. Dado que GLI-85 está en el conjunto Nivel alto de complejidad y se obtienen resultados buenos implica directamente la idea de que el método con la función kernel correspondiente es efectivos.

Por último, se realiza un análisis horizontal de los datos, ¿como sería dicho análisis? Primeramente para cada función se buscan los resultados del indicador Exactitud para RDNL, DRDNL y DRDNLD. En el gráfico 3.4 se detecta que uno de los resultados esperados en la investigación

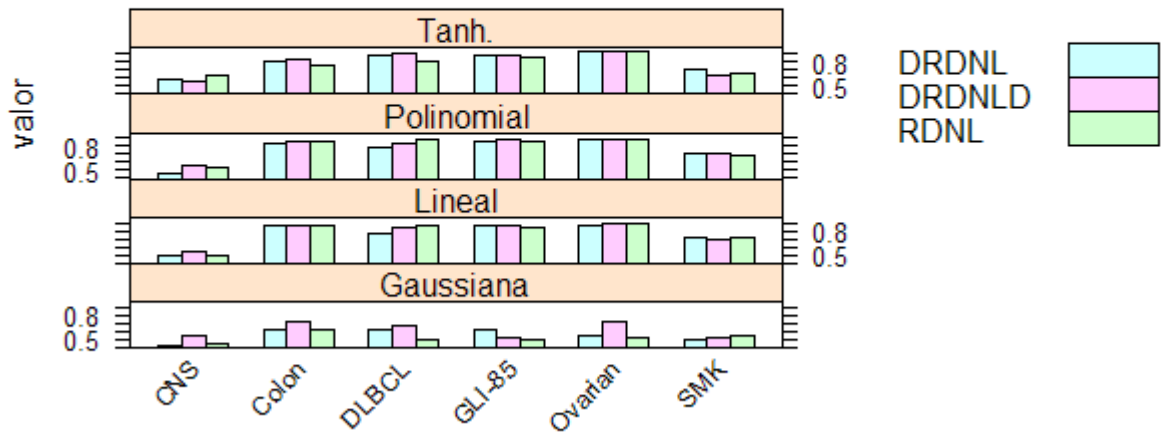


Figura 3.4: "Análisis Horizontal de las metodologías"

se cumplió porque la metodología DRDNLD supera a las dos restantes con solo mirar la gráfica. Solo cuesta darse cuenta en el caso de la función kernel "tanh." ya que la diferencia no es fácil de detectar en el gráfico, como las demás. En cambio las metodologías restantes se comportan de formas disparejas, lo que hace que no se pueda encontrar un patrón de forma sencilla.

3.4. Conclusiones del capítulo

En este capítulo se muestran los conjuntos de datos de microarreglo de ADN y se comentan sus principales problemas agrupándolos en tres grupos: *Nivel bajo de complejidad*, *Nivel medio de complejidad* y *Nivel alto de complejidad*. Los resultados de la investigación se obtuvieron en el software R implementando las metodologías propuestas en el capítulo anterior. Por último, la función kernel "tanh." es la que mejores métricas obtiene entre la "lineal", "polinomial" y "Gaussiana", por otro lado gracias al análisis horizontal se concluye que la metodología en forma distribuida (**DRDNLD**) es superior a **RDNL** y **DRDNL** debido a que obtuvo los índice de sensibilidad, especificidad y exactitud más altos en cuanto a los conjuntos de datos.

CONCLUSIONES

- Se estudiaron los métodos de selección de características, **ACPK** y descomposición matricial CUR como métodos de reducción de la dimensión en los datos de microarreglo de ADN para el cáncer.
- Se propusieron las metodologías: **RDNL**, **DRDNL**, **DRDNLD** que resuelven el problema de la reducción de la dimensión en datos de microarreglos de ADN para obtener un modelo de clasificación que pronostique si un paciente padece o no el cáncer.
- La función kernel "tanh." es la que mejores métricas obtiene entre la "lineal", "polinomial" y "Gaussiana".
- La metodología en forma distribuida (**DRDNLD**) supera a las metodologías **RDNL** y **DRDNL** de acuerdo al análisis horizontal que se efectuó.
- En el ambiente de desarrollo integrado RStudio se implementaron las metodologías **RDNL**, **DRDNL**, **DRDNLD** a partir de las funciones "nldr", "nldr2" y "nldr2d", en ese orden.

REFERENCIAS

[1] D. Galvis, A. Orjuela, Microarreglos.

[2] S. W. Siedlecki, On automatic feature selection, 1.

[3] E. A. Guyon I., An introduction to variable and Feature Selection, 23.

[4] P. V. Ammu P. K., Review on Feature on Feature Selection Techiques Techniques of DNA Microarray Data., 61.

[5] V. Bolon Canedo, A review of microarray datasets and applied feature selection methods., 282.

[6] Z. M. a. D. Hira, A reviw of feature selection and feature extraction methods applied in microarray data.

[7] Y. S. Pedro Larranaga, A review of feature selection techniques in bioinformatics., 23.

[8] P. Colectivo de autores, Distributed feature selection using vertical partitioning for high dimensional data.

[9] Z. Zheng, Massively parallel feature selection: an approach based on variance preservation. (2013).

[10] S. Zhanquan, Paralel feature selection based on MapReduce.

[11] H. Qing, Parallel feature selection using positive approximation based on mapreduce.

[12] M. F. Colectivo de autores, Centralized vs. Distributed feature selection methods base don data on data complexity measures., 117.

[13] B. S. Klaus-Robert Muller, Kernel Prncipal Component Analysis.

[14] L. G. S. Julio Fernando Suarez, Introduccion a kernel ACP y otros metodos espectrales aplicados al aprendizaje no supervisado., 31.

[15] M. A. G. A. Yunier E. Tejeda Rodriguez, Metodos de extraccion de caracteristicas en datos de microarrays de ADN para enfermedades oncologicas (Parte I)., Informe de Tesis, Santa Clara, Cuba (2017).

[16] H. Williams, Principal component analysis.

[17] R. A. Frieze, Kast Monte-Carlos Algorithms for finding low-rank approximations, 51.

[18] M. Drineas, CUR matrix decompositions for improved data analysis., pNAS.

[19] R. P. Drineas, Fast Monte-Carlo algorithms for matrices III: Computing a compressed approximate matrix decomposition, 36.

[20] G. Stewart, Four algorithms for the efficient computation of truncated QR approximations to a sparse matrix, numer. Math (1999).

[21] S. Tyrtyshnikov, The maximum-volume concept in approximation by low-rank matrices.

[22] M. P. Drineas, Relative-error CUR matrix decompositions., 30.

[23] N. V. Bolon Canedo, Distributed feature selection: An application to microarray data classification., 30.

[24] I. A. Bodor, Mahoney and N. Solymosi, Rcur: an R package for CUR matrix decomposition., 13.

[25] A. Solymosi, rCUR: CUR decomposition package (2012).
URL http://CRAN.R-project.org/package=rCUR

[26] B. C. V., Novel feature selection methods for high dimensional data., deparament of Computer Science, University of a Coruna (2014).

[27] K.R.B.M., Dataset (2014).

[28] F.S.D.a.A.S., Dataset Repository. (2014).

[29] F.S.D.a.A.S., University (2014).

ANEXO A

Anexos Principales

A.1. “Matriz de confusión para RDNL”

C. Datos vs F. Kernel	Polinomial	Lineal	Tanh.	Gaussiana
Colon	$\begin{bmatrix} 18 & 5 \\ 4 & 35 \end{bmatrix}$	$\begin{bmatrix} 18 & 5 \\ 4 & 35 \end{bmatrix}$	$\begin{bmatrix} 10 & 4 \\ 12 & 36 \end{bmatrix}$	$\begin{bmatrix} 12 & 14 \\ 10 & 26 \end{bmatrix}$
CNS	$\begin{bmatrix} 7 & 14 \\ 14 & 25 \end{bmatrix}$	$\begin{bmatrix} 5 & 15 \\ 16 & 24 \end{bmatrix}$	$\begin{bmatrix} 7 & 9 \\ 14 & 30 \end{bmatrix}$	$\begin{bmatrix} 10 & 22 \\ 11 & 17 \end{bmatrix}$
DLBCL	$\begin{bmatrix} 23 & 6 \\ 0 & 18 \end{bmatrix}$	$\begin{bmatrix} 23 & 6 \\ 0 & 18 \end{bmatrix}$	$\begin{bmatrix} 18 & 4 \\ 5 & 20 \end{bmatrix}$	$\begin{bmatrix} 12 & 13 \\ 11 & 11 \end{bmatrix}$
GLI-85	$\begin{bmatrix} 21 & 7 \\ 5 & 52 \end{bmatrix}$	$\begin{bmatrix} 20 & 7 \\ 6 & 52 \end{bmatrix}$	$\begin{bmatrix} 19 & 5 \\ 7 & 54 \end{bmatrix}$	$\begin{bmatrix} 11 & 28 \\ 15 & 31 \end{bmatrix}$
Ovarian	$\begin{bmatrix} 72 & 12 \\ 19 & 150 \end{bmatrix}$	$\begin{bmatrix} 72 & 11 \\ 19 & 151 \end{bmatrix}$	$\begin{bmatrix} 78 & 4 \\ 13 & 158 \end{bmatrix}$	$\begin{bmatrix} 47 & 76 \\ 44 & 86 \end{bmatrix}$
SMK	$\begin{bmatrix} 56 & 25 \\ 34 & 72 \end{bmatrix}$	$\begin{bmatrix} 60 & 22 \\ 30 & 75 \end{bmatrix}$	$\begin{bmatrix} 49 & 23 \\ 41 & 74 \end{bmatrix}$	$\begin{bmatrix} 47 & 41 \\ 43 & 56 \end{bmatrix}$

A.2. "Matriz de confusión para DRDNL"

C. Datos vs F. Kernel	Polinomial	Lineal	Tanh.	Gaussiana
Colon	$\begin{bmatrix} 17 & 5 \\ 5 & 35 \end{bmatrix}$	$\begin{bmatrix} 18 & 5 \\ 5 & 35 \end{bmatrix}$	$\begin{bmatrix} 14 & 5 \\ 8 & 35 \end{bmatrix}$	$\begin{bmatrix} 8 & 9 \\ 14 & 31 \end{bmatrix}$
CNS	$\begin{bmatrix} 2 & 13 \\ 19 & 26 \end{bmatrix}$	$\begin{bmatrix} 1 & 11 \\ 10 & 28 \end{bmatrix}$	$\begin{bmatrix} 5 & 10 \\ 16 & 29 \end{bmatrix}$	$\begin{bmatrix} 5 & 18 \\ 16 & 21 \end{bmatrix}$
DLBCL	$\begin{bmatrix} 20 & 8 \\ 3 & 16 \end{bmatrix}$	$\begin{bmatrix} 20 & 8 \\ 3 & 16 \end{bmatrix}$	$\begin{bmatrix} 21 & 4 \\ 2 & 20 \end{bmatrix}$	$\begin{bmatrix} 16 & 11 \\ 7 & 13 \end{bmatrix}$
GLI-85	$\begin{bmatrix} 19 & 5 \\ 7 & 54 \end{bmatrix}$	$\begin{bmatrix} 19 & 5 \\ 7 & 54 \end{bmatrix}$	$\begin{bmatrix} 21 & 5 \\ 5 & 54 \end{bmatrix}$	$\begin{bmatrix} 15 & 21 \\ 11 & 38 \end{bmatrix}$
Ovarian	$\begin{bmatrix} 72 & 15 \\ 19 & 147 \end{bmatrix}$	$\begin{bmatrix} 72 & 15 \\ 19 & 147 \end{bmatrix}$	$\begin{bmatrix} 77 & 10 \\ 14 & 152 \end{bmatrix}$	$\begin{bmatrix} 47 & 73 \\ 44 & 89 \end{bmatrix}$
SMK	$\begin{bmatrix} 59 & 22 \\ 31 & 75 \end{bmatrix}$	$\begin{bmatrix} 59 & 22 \\ 31 & 75 \end{bmatrix}$	$\begin{bmatrix} 57 & 24 \\ 33 & 73 \end{bmatrix}$	$\begin{bmatrix} 44 & 45 \\ 46 & 52 \end{bmatrix}$

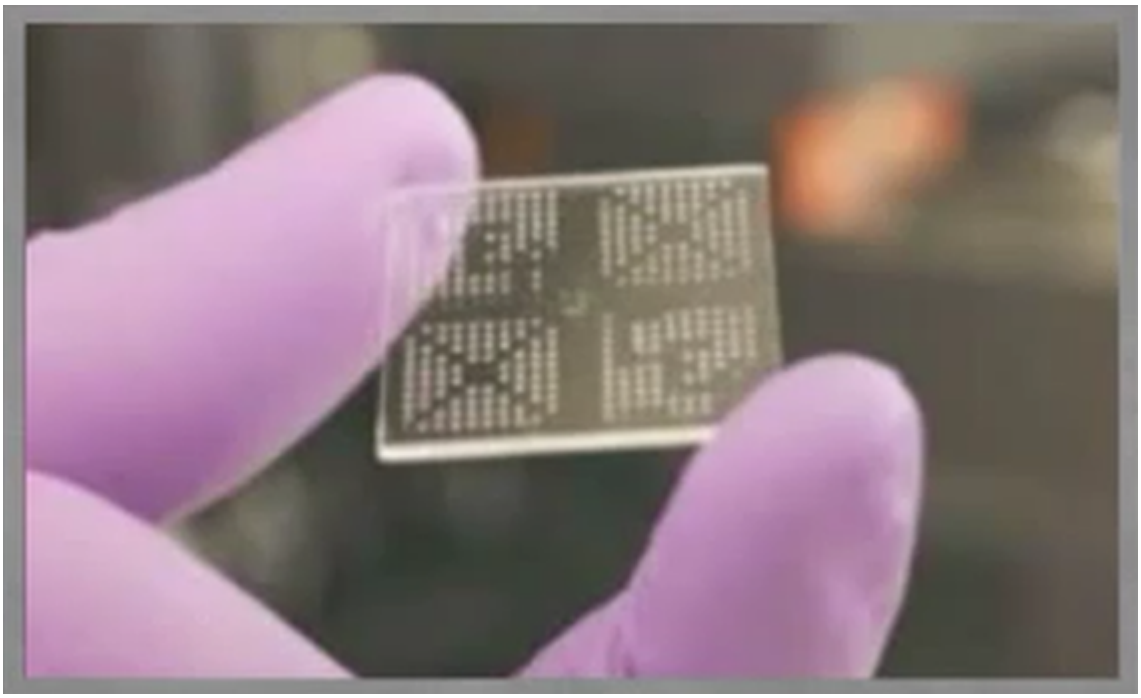
A.3. "Matriz de confusión para DRDNLD"

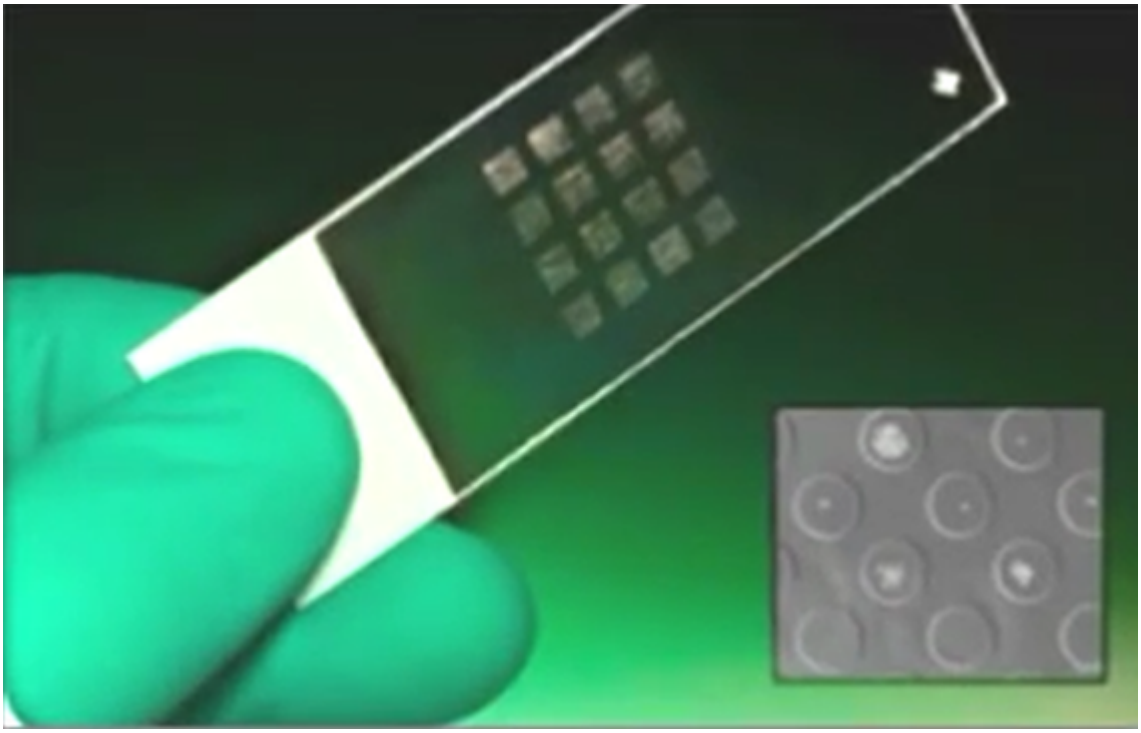
C. Datos vs F. Kernel	Polinomial	Lineal	Tanh.	Gaussiana
Colon	$\begin{bmatrix} 17 & 4 \\ 5 & 36 \end{bmatrix}$	$\begin{bmatrix} 17 & 3 \\ 5 & 37 \end{bmatrix}$	$\begin{bmatrix} 15 & 4 \\ 7 & 36 \end{bmatrix}$	$\begin{bmatrix} 15 & 11 \\ 7 & 29 \end{bmatrix}$
CNS	$\begin{bmatrix} 5 & 11 \\ 16 & 28 \end{bmatrix}$	$\begin{bmatrix} 6 & 13 \\ 15 & 26 \end{bmatrix}$	$\begin{bmatrix} 4 & 10 \\ 17 & 29 \end{bmatrix}$	$\begin{bmatrix} 12 & 18 \\ 9 & 21 \end{bmatrix}$
DLBCL	$\begin{bmatrix} 21 & 6 \\ 2 & 18 \end{bmatrix}$	$\begin{bmatrix} 21 & 6 \\ 2 & 18 \end{bmatrix}$	$\begin{bmatrix} 23 & 4 \\ 0 & 20 \end{bmatrix}$	$\begin{bmatrix} 17 & 10 \\ 6 & 14 \end{bmatrix}$
GLI-85	$\begin{bmatrix} 21 & 6 \\ 5 & 53 \end{bmatrix}$	$\begin{bmatrix} 21 & 6 \\ 5 & 53 \end{bmatrix}$	$\begin{bmatrix} 20 & 5 \\ 6 & 54 \end{bmatrix}$	$\begin{bmatrix} 10 & 24 \\ 16 & 35 \end{bmatrix}$
Ovarian	$\begin{bmatrix} 72 & 9 \\ 19 & 153 \end{bmatrix}$	$\begin{bmatrix} 72 & 9 \\ 19 & 153 \end{bmatrix}$	$\begin{bmatrix} 78 & 6 \\ 13 & 156 \end{bmatrix}$	$\begin{bmatrix} 59 & 42 \\ 32 & 120 \end{bmatrix}$
SMK	$\begin{bmatrix} 53 & 20 \\ 37 & 77 \end{bmatrix}$	$\begin{bmatrix} 53 & 20 \\ 37 & 77 \end{bmatrix}$	$\begin{bmatrix} 47 & 28 \\ 43 & 69 \end{bmatrix}$	$\begin{bmatrix} 45 & 43 \\ 45 & 54 \end{bmatrix}$

ANEXO B

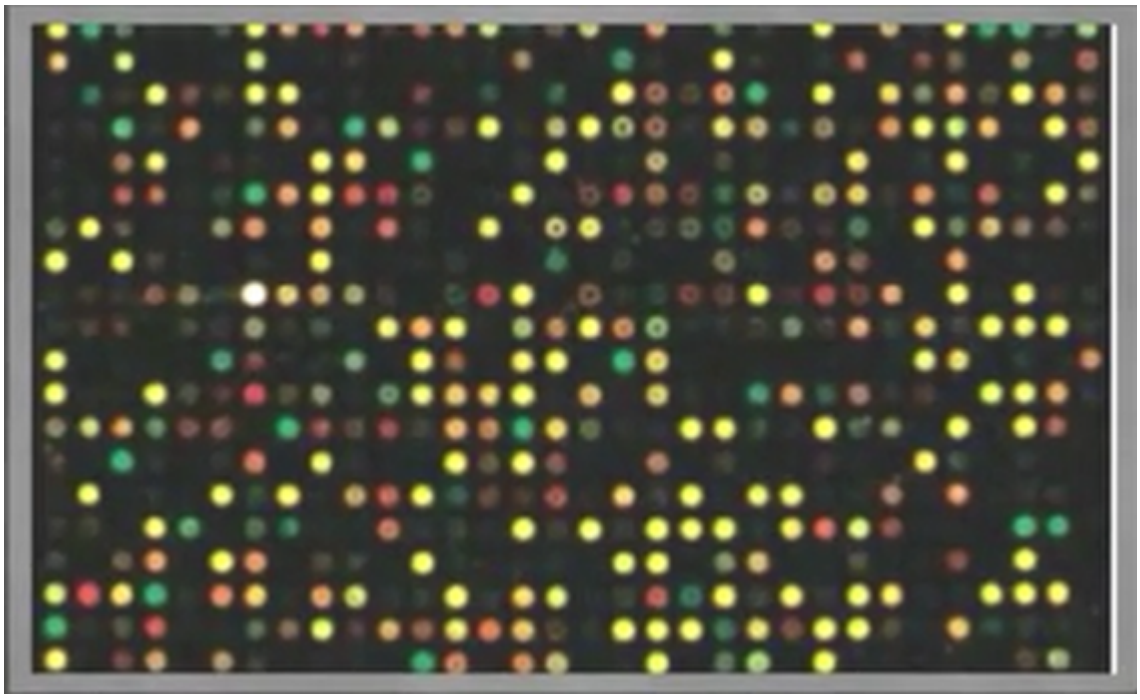
Anexos Secundarios

B.1. “Microarreglos de ADN”





B.2. Matriz final de microarreglos de ADN



B.3. “Pseudocódigo de RDNL”

```
nldr <- function(X, Y, prop = 0.75, scale = TRUE, functionK =
  "vanilladot"){
 ptm <- proc.time()
 if(!X %>% is.matrix) X <- X %>% as.matrix
 if(X %>% is.na %>% any){
  X.imputed <- X %>% impute.knn #Impute missing value
  X <- X.imputed$data
 }
 if(!Y %>% is.factor) Y <- Y %>% as.factor
 if(prop < 0 && prop > 1) stop('prop is a real number between 0 and 1')
 if(!scale %>% is.logical)
 stop("scale must be a logical value")
 functionK <- match.arg(functionK, c("rbfdot", "vanilladot",
  "polydot","tanhdot"))
 reduction.method <- list() # a list with the reduction methods to use

 #Step 1: Dimension reduction by kpca
 Xscale <- X %>% scale(., scale) # Estandarisa y centraliza la matriz
 reduction.method$Xkpca <- Xscale %>% kpca(., kernel = functionK,
  kpar = list())
 D2 <- reduction.method$Xkpca@eig # the corresponding eigenvalues
 T <- reduction.method$Xkpca@pcv
 # scores matrix: the principal component vectors
 propVar <- D2/sum(D2) #proportion of the variance
 propAcum <- cumsum(propVar) #proportion of the variance acumulate
 #N <- length(propAcum[propAcum <= prop]) #n-th principals components
 N <- (propAcum >= prop) %>% which %>% min #n-th principals components
 Ttrain <- T[,1:N] #n-th principals components for the train set
 colnames(Ttrain) <- paste(1:ncol(Ttrain))

 #Step 2: Classification model <- c("lda", "lda2", "LogitBoost", "knn",
  "nb", "svmLinear")
```

```
 #"C5.0", "rf", "rpart", "avNNet"
 trControl <- trainControl(method = "cv") #sampling
 lista <- foreach(i = 1:length(model)) %do% {
  matriz <- Ttrain %>% {
   train(., Y, method = model[i], trControl = trControl)
 } %>% confusionMatrix.train(norm = "none")
 matriz$table
 }
 confusionMatrix <- c(lapply(lista, confusionMatrix), list(model))
 time <- proc.time()-ptm
 return(list(reduction.method = reduction.method,
  confusion.Matrix = confusionMatrix,
  pA = propAcum, n = N, time = time))
}
```

B.4. "Pseudocódigo de DRNL"

```
nldr2 <- function(X, Y, prop = 0.75, scale = TRUE, c = dim(X)[2],
  r = dim(X)[1], method = "random", alpha = 1, weighted = FALSE,
  beta = 4, matrix.return = TRUE, error.return = FALSE,
  functionK = "rbfdot"){
 ptm <- proc.time()
 if(!X %>% is.matrix) X <- X %>% as.matrix
 if(X %>% is.na %>% any){
  X.imputed <- X %>% impute.knn #Impute missing value
  X <- X.imputed$data
 }
 if(!Y %>% is.factor) Y <- Y %>% as.factor
 if(prop < 0 && prop > 1) stop('prop is a real number between 0 and 1')
 if(!scale %>% is.logical) stop("scale must be a logical value")
 method <- match.arg(method,c("random","exact.num.random","top.scores",
  "ortho.top.scores", "highest.ranks"))
 functionK <- match.arg(functionK, c("rbfdot", "vanilladot", "polydot",
  "tanhdot"))
```

```
dimension.reduction <- list()

#Step 1: Dimension reduction by CUR
npc <- X %>% pca.SVD(., prop, scale)
#n-th principals components for the train set
dimension.reduction$cur <- X %>% CUR(., c, r, k = npc$n, sv = npc,
 method, alpha, weighted, beta, matrix.return, error.return)
C.index <- dimension.reduction$cur@C.index
C <- dimension.reduction$cur@C # X[,C.index]

#Step 2: Dimension redcution by kpca
Cscale <- C %>% scale(., scale) # Estandarisa y centraliza la matriz
dimension.reduction$Ckpca <- Cscale %>% kpca(., kernel = functionK,
 kpar = list(sigma=0.1))
D2 <- dimension.reduction$Ckpca@eig # the corresponding eigenvalues
T <- dimension.reduction$Ckpca@pcv
# scores matrix: the principal component vectors
propVar <- D2/sum(D2) #proportion of the variance
propAcum <- cumsum(propVar) #proportion of the variance acumulate
#N <- length(propAcum[propAcum <= prop]) #n-th principals components
N <- (propAcum >= prop) %>% which %>% min #n-th principals components
Ttrain <- T[,1:N] #n-th principals components for the train set
colnames(Ttrain) <- paste(1:ncol(Ttrain))

#Step 3: Classification 1
model <- c("lda", "lda2", "LogitBoost", "knn", "nb", "svmLinear")
#"C5.0", "rf", "rpart", "avNNet"
trControl <- trainControl(method = "cv") #sampling
lista <- foreach(i = 1:length(model)) %do% {
 matriz <- Ttrain %>% {
 train(., Y, method = model[i], trControl = trControl)
 } %>% confusionMatrix.train(norm = "none")
 matriz$table
}
```

```
 confusionMatrix <- c(lapply(lista, confusionMatrix),
 list(model)) time <- proc.time()-ptm
 return(list(dimension.reduction = dimension.reduction,
  confusionMatrix = confusionMatrix, time = time))
}
```

B.5. "Pseudocódigo de DRNLD"

```
nldr2d <- function(X, Y, order.col = "none", prop = 0.75,
  scale = TRUE, c = dim(X)[2], r = dim(X)[1],
  method = "random", alpha = 1, weighted = FALSE, beta = 4,
  matrix.return = TRUE, error.return = FALSE, percent = 0.10,
  functionK = "rbfdot") {
 ptm <- proc.time()
 if(!X %>% is.matrix)
 X <- X %>% as.matrix
 if(X %>% is.na %>% any){
  X.imputed <- X %>% impute.knn #Impute missing value
  X <- X.imputed$data
 }
 if(!Y %>% is.factor) Y <- Y %>% as.factor
 order.col <- match.arg(order.col, c("none", "random"))
 if(prop < 0 && prop > 1) stop('prop is a real number between 0 and 1')
 if(!scale %>% is.logical) stop("scale must be a logical value")
 method <- match.arg(method, c("random", "exact.num.random",
  "top.scores", "ortho.top.scores", "highest.ranks"))
 functionK <- match.arg(functionK, c("rbfdot", "vanilladot", "polydot",
  "tanhdot"))

 #Validate the field percent
 i <- j <- k <- l <- NULL
 model <- c("lda", "lda2", "LogitBoost", "knn", "nb", "svmLinear")
 #"C5.0", "rf", "rpart", "avNNet"
 if(order.col == "none"){
```

```
 C_0 <- seq(1, ncol(X), round(ncol(X)*percent))
 if(C_0 %>% length != percent*100){
  C_0 <- C_0[1:(percent*100)]
 }
 C_i <- c(foreach(i = 1:(length(C_0)-1)) %do% C_0[i]:(C_0[i+1]-1),
  list(C_0[length(C_0)]:ncol(X)))
}
drdnld <- foreach(i = 1:length(C_0), .packages = c("foreach", "rCUR",
 "magrittr", "caret", "caTools", "class", "e1071", "kernlab")) %dopar%{
  pca.SVD <- function(X, prop = 0.75, scale = TRUE){
   ptm <- proc.time()
   if(prop < 0 && prop > 1) stop('prop is a real number between 0 and 1')
   if(!scale %>% is.logical) stop("scale must be a logical value")
   Xsvd <- X %>% scale(., scale) %>% svd # Singular Value Decomposition
   D <- Xsvd$d #diagonal matrix with the singulars values
   U <- Xsvd$u #matrix with the singulars values left
   V <- Xsvd$v #matrix with the singulars values rithg
   T <- U%*%diag(D) #scores matrix
   P <- V #loading matrix
   propVar <- D^2/sum(D^2) #proportion of the variance
   propAcum <- cumsum(propVar) #proportion of the variance acumulate
   #N <- length(propAcum[propAcum <= prop]) #n-th principals components
   N <- which(propAcum >= prop) %>% min #n-th principals components
   time <- proc.time()-ptm
   return(list(d = D, u = U, v = V, t = T, p = P, pA = propAcum, n = N,
    time = time))
   }

   #Step 1: Dimension reduction by CUR
   X_i <- X[,C_i[[i]]]
   npc.SVD <- X_i %>% pca.SVD(., prop, scale) #n-th principals components
    for the train set
   dimension.reduction.cur <- X_i %>% CUR(., c, r, k = npc.SVD$n,
    sv = npc.SVD, method, alpha, weighted, beta, matrix.return,
```

```
 error.return)
C.index <- dimension.reduction.cur@C.index
C <- dimension.reduction.cur@C # X[,C.index]

#Step 2: Dimension reduction by kpca
Cscale <- C %>% scale(., scale) # Estandarisa y centraliza la matriz
dimension.reduction.train <- Cscale %>% kpca(., kernel = functionK,
 kpar = list(sigma=0.1))
D2 <- dimension.reduction.train@eig #the corresponding eigenvalues
T <- dimension.reduction.train@pcv
#scores matrix: the principal component vectors
propVar <- D2/sum(D2) #proportion of the variance
propAcum <- cumsum(propVar) #proportion of the variance acumulate
#N <- length(propAcum[propAcum <= prop]) #n-th principals components
N<-(propAcum >= prop) %>% which %>% min #n-th principals components
Ttrain <- T[,1:N] #n-th principals components for the train set
colnames(Ttrain) <- paste(1:ncol(Ttrain))

#Step 3: Classification
trControl <- trainControl(method = "cv") #sampling
lista <- foreach(i = 1:length(model)) %do% {
 matriz <- Ttrain %>%{
  train(., Y, method = model[i], trControl = trControl)
 } %>% confusionMatrix.train(norm = "none")
 matriz$table
}
confusionMatrix <- c(lapply(lista, confusionMatrix), list(model))
list(dimension.reduction.cur, dimension.reduction.train,
 confusionMatrix)
}
Ac.nmod <- foreach(k = 1:length(C_0), .combine = rbind,
.packages = c("foreach")) %dopar% {
foreach(l = 1:length(model), .combine = c) %dopar% {
 drdnld[[k]][[3]][[l]][[3]][1]
```

```
    }
   }
   index <- Ac.nmod %>% apply(., 1, max) %>% which.max
   dimension.reduction.cur.best <- drdnld[[index]][[1]]
   dimension.reduction.train.best <- drdnld[[index]][[2]]
   confusionMatrix.best <- drdnld[[index]][[3]]
   time <- proc.time()-ptm
   return(list(dimension.reduction.cur.best = dimension.reduction.cur.best,
    dimension.reduction.train.best = dimension.reduction.train.best,
    confusionMatrix.best = confusionMatrix.best, time = time))
}
```