Facultad de Ciencias Exactas, Ingeniería y Agrimensura

Universidad Nacional de Rosario

Lic. en Ciencias de la Computación

Tesina de Grado

"Selección de variables en problemas multiclase"

Andrés C. Burgos

Director: Dr. Pablo M. Granitto

Marzo 2009

Resumen

La selección de variables es una técnica de preprocesado comunmente usada en conjuntos de datos de alta dimensionalidad. Tiene como propósito reducir la dimensión del espacio de variables, eliminar variables irrelevantes o redundantes, mejorar la eficiencia de los algoritmos de aprendizaje e incrementar la interpretabilidad de los modelos construidos.

En este trabajo se introduce una nueva técnica de selección de variables para problemas multiclase. La técnica es una extensión del popular algoritmo RFE, consistente en resolver el problema de clasificación multiclase con una combinación One–Vs–All de clasificadores binarios, y seleccionar luego variables en cada uno de los subproblemas creados por el OVA usando RFE.

Usando datos reales de genómica y espectrometría de masa, y varios clasificadores para construir los rankings, se analiza en detalle la performance y estabilidad del nuevo método y se lo compara con el método RFE tradicional.

Índice

1. Introducción

Nuestro método científico está basado en gran parte en el proceso: (observación / recolección de datos) → (modelado) → (extracción de conocimiento). La introducción en las últimas décadas de métodos automáticos de medición y almacenamiento de datos produjo una revolución en las ciencias, donde la cantidad de nuevas observaciones disponibles supera ampliamente nuestra capacidad actual de modelado. El aprendizaje automatizado ("Machine Learning") intenta comprender los mecanismos por los cuales se adquiere el conocimiento a partir de los datos/experiencia y con ello lograr automatizar la etapa de modelado [1]. Uno de los objetivos principales es crear modelos que sean capaces de identificar patrones en los datos que permitan hacer predicciones útiles sobre observaciones nuevas. Otro objetivo importante es que los modelos creados sean interpretables y permitan extraer fácilmente conocimiento de ellos. Estos dos objetivos, lamentablemente, suelen estar contrapuestos. Las clases de modelos con mayor capacidad de generalización son típicamente "cajas negras/grises" (black/gray boxes) de difícil o nula interpretación.

La selección de variables ("feature selection") es una técnica de pre-procesado que se aplica normalmente en aprendizaje automatizado a conjuntos de datos de alta dimensionalidad. Esta técnica estudia cómo seleccionar eficientemente un subconjunto de atributos o variables que se usarán luego para construir los modelos. La selección de variables tiene diversos propósitos: reducir la dimensión del espacio de variables ("curse of dimensionality"), eliminar variables irrelevantes o redundantes y mejorar la eficiencia de los algoritmos de aprendizaje, entre otros [7]. Sin embargo, la mayor utilidad de la selección de variables en el contexto actual es que mejora la interpretabilidad de los modelos, al permitir concentrar los esfuerzos de extracción de conocimiento en un reducido número de variables, y contribuye por lo tanto a disminuir el efecto de caja negra de los modelos de mayor capacidad.

La aparición de las técnicas de "high-throughput" en biología, química, etc., ha in-

troducido un gran desafío en selección de variables: los datasets "anchos" formados por un número muy grande de variables medidas sobre unos pocos ejemplares. Ejemplos comunes son los datos de expresión genética (DNA microarrays) y proteómica (espectrometría de masa de suero sanguíneo).

En selección de variables se ha trabajado principalmente sobre problemas de dos clases o binarios. Los problemas de muchas clases o multiclase han recibido menos atención, por su mayor dificultad intrínseca y porque algunos clasificadores utilizados en la selección no son capaces de manejar problemas multiclase. La mayoría de los métodos y aplicaciones conocidos en esta área corresponden a lo que llamaremos el "método directo", es decir la utilización de un algoritmo de selección asociado a un clasificador que tiene la capacidad de manejar el problema multiclase directamente. El problema de realizar una clasificación en muchas clases tiene otra aproximación muy utilizada en los últimos años, la combinación de un conjunto de clasificadores binarios para formar un clasificador multiclase compuesto. En esta tesina se trata el problema de realizar la selección de variables en problemas multiclase a partir de una combinación de clasificadores binarios. Es lo que llamaremos el "método combinado".

Existe cierta literatura donde se proponen métodos combinados para ciertos clasificadores y métodos de combinación/ selección. Los resultados presentados parecen ser al menos tan buenos como los métodos directos [16]. En esta tesina se adaptó el método combinado a uno de los algoritmos de selección de variables más importante actualmente, el método RFE [9]. Se realizó una amplia evaluación, utilizando tres diferentes clasificadores y cuatro métodos distintos para producir rankings de variables. Los métodos se evaluaron no sólo en cuanto a la capacidad de encontrar conjuntos pequeños con gran capacidad de discriminación, sino también en cuanto a la estabilidad de las selecciones de los métodos.

Nuestros resultados, con distintos métodos y diversos tipos de problemas, muestran que el método combinado no presenta mejoras reales en comparación con el directo.

Creemos que esta diferencia con los resultados publicados previamente se debe principalmente a diferentes interpretaciones sobre los conjuntos de variables seleccionados. En particular, encontramos diferencias en las cantidades de variables que se dicen utilizar para realizar las clasificaciones, lo cual justificaría las aparentes mejoras publicadas.

Presentamos entonces ambas interpretaciones sobre la cantidad de variables y las comparamos con el método directo, analizando tanto los porcentajes de error como la estabilidad de cada una.

En la sección 2 se presentarán algunos conceptos básicos del aprendizaje automatizado para la adecuada comprensión de las secciones siguientes. En la sección 3 se describe en qué consiste la selección de variables y se presenta el algoritmo de selección de variables RFE, el cual es un método de clasificación directo. También se introduce el concepto de estabilidad. En la sección 4 se propone un clasificador combinado para luego, en la sección 5, analizar los porcentajes de error medio y estabilidad de cada método sobre los resultados obtenidos utilizando diferentes datasets, comparando el método de clasificación directo con el combinado. Finalmente en la sección 6 se presentan las conclusiones sobre los resultados analizados.

2. Conceptos básicos

2.1. Aprendizaje automatizado

El aprendizaje automatizado es una rama de la inteligencia artificial que contempla el desarrollo de algoritmos que son capaces de optimizar su performance usando datos de ejemplo o experiencia. El aprendizaje es necesario en casos donde no es posible escribir directamente un programa de computación para un problema dado. Estos casos se destacan por la falta de un conocimiento humano suficiente en el tema o cuando éste no es explicable por los humanos. Consideremos el reconocimiento del habla, es decir, la transformación de señales acústicas a texto. Esta tarea es llevada a cabo fácilmente por los humanos, aunque no somos capaces de explicar cómo. En aprendizaje automatizado, el enfoque dado es recolectar grandes cantidades de ejemplos de sonidos de diferentes personas y aprender a asociarlos con palabras.

Todos los algoritmos desarrollados asumen que existen procesos con cierto determinismo que explican los datos observados. Aunque no se conozcan los detalles de los procesos subyacentes en la generación de datos -por ejemplo, comportamiento de un consumidor- se asume que no es completamente azaroso, es decir, que existen ciertas correlaciones en los datos. Por ejemplo, las personas no van al supermercado a comprar cosas al azar. Cuando compran cerveza usualmente también compran maní, o compran helado en verano y chocolate en invierno.

Aunque no se puedan identificar enteramente estos procesos, se pueden construir aproximaciones buenas y útiles. Estas aproximaciones pueden servir para detectar ciertos patrones o regularidades, que a su vez se pueden utilizar para comprender mejor el proceso o para realizar predicciones [2]. Construir eficazmente estas aproximaciones útiles es el objetivo principal del aprendizaje automatizado.

2.2. Aprendizaje supervisado

Existen varios tipos de problemas que se estudian en aprendizaje automatizado. Entre las fundamentales se encuentran el aprendizaje supervisado y el no supervisado. El aprendizaje no supervisado intenta descubrir como están organizados los datos provistos, agrupándolos por similaridad. Los datos usados por este tipo de algoritmos no se encuentran clasificados. En cambio, en el aprendizaje supervisado el objetivo es crear, a partir de un conjunto de datos de entrenamiento para los cuales se conoce la respuesta adecuada, un modelo numérico capaz de realizar predicciones precisas para datos nuevos. Los problemas de clasificación son una parte de los problemas supervizados en los que el objetivo es asociar con los datos un número discreto de valores, clases o categorías.

2.2.1. Sesgo inductivo

Normalmente, para un problema de clasificación dado, existen muchas (potencialmente infinitas) funciones clasificadoras que son consistentes con los datos de entrenamiento. Una función clasificadora se dice que es consistente con un conjunto de datos si clasifica a todos éstos correctamente. Por lo tanto, para encontrar una solución a nuestro problema de clasificación, debemos realizar suposiciones extras sobre la función subyacente a los datos. A este conjunto de suposiciones que se hacen para hacer posible el aprendizaje se le llama *sesgo inductivo* del algoritmo de aprendizaje [2].

2.3. Clasificación

El problema de clasificación consiste formalmente en asignar una etiqueta o clase a un objeto. Cada objeto se describe por un conjunto de variables que representan medidas u observaciones sobre el mismo, y se le asocia un vector en un espacio n-dimensional, donde cada dimensión representa una variable. Así, el objetivo es clasificar a un objeto $\mathbf{x} = [x_1, x_2, ..., x_n] \in \mathbb{R}^n$, donde a $\mathbb{R}^n$ se lo denomina *feature space* (espacio de variables

o características). Cada objeto tiene asignada una clase o label $l(\mathbf{x})$. Para un problema con c clases $l(\mathbf{x})$ puede tomar c valores discretos distintos.

El objetivo entonces es encontrar una función clasificadora h tal que para cada objeto $\mathbf{x}$ sea $h(\mathbf{x}) = l(\mathbf{x})$.

2.3.1. Error de clasificación

Dado un conjunto de datos D, el error de clasificación del modelo h se define por

$$error(h) = \frac{N_{err}}{N_{tot}} \tag{1}$$

donde N_{err} es el número de clasificaciones erróneas de h en D y N_{tot} es el total de datos clasificados.

Formalmente, sea $h(\mathbf{x_i})$ la etiqueta o clase asignada por h a un objeto $\mathbf{x_i} \in \mathbf{D}$, y sea $l(\mathbf{x_i})$ la clase real correspondiente a ese objeto, entonces

$$error(h) = \frac{1}{N_{tot}} \sum_{i=1}^{N_{tot}} \{\phi(l(\mathbf{x_i}), \mathbf{h}(\mathbf{x_i}))\} \tag{2}$$

donde $\phi(a, b) = 1 \Leftrightarrow a \neq b$.

2.3.2. Estrategias de estimación del error

Para encontrar una función clasificadora, un algoritmo necesita utilizar datos de donde poder "aprender". Pero es conocido desde la estadística clásica que se necesita un nuevo conjunto de datos, independiente de los que se usaron para aprender, para poder estimar correctamente el error de un clasificador (estimación sin sesgo). Como en la práctica se dispone casi siempre de un conjunto limitado de datos, se utilizan distintas estrategias para realizar eficazmente las dos tareas (aprender y estimar el error).

Dado un conjunto de datos D con N ejemplos, algunos de los métodos utilizados usualmente incluyen:

- **Resubstitution:** Se entrena el clasificador con D y se lo evalúa con D. El error tiene sesgo optimista y no es confiable.
- **Hold-Out:** Se particiona D en dos subconjuntos. Se utiliza una parte para entrenar el clasificador y la otra para estimar el error. Usualmente se repite el procedimiento varias veces para tener una estimación más confiable.
- **Bootstraps:** Se generan L subconjuntos de tamaño N eligiendo aleatoriamente y con repetición elementos de D. Se estima el error utilizando los ejemplos no incluídos en cada repetición.
- **K-fold Cross Validation:** Se particiona D en k subconjuntos de igual tamaño, y se utilizan $k-1$ subconjuntos para entrenar y el restante para evaluar. Esto se repite k veces, dejando para evaluar en cada repetición un subconjunto distinto.
- **Leave-One-Out:** Corresponde a utilizar K-fold Cross Validation con $K = N$. Es decir, en N repeticiones el clasificador es entrenado utilizando todos los datos excepto uno, al cual se lo utiliza para testear.

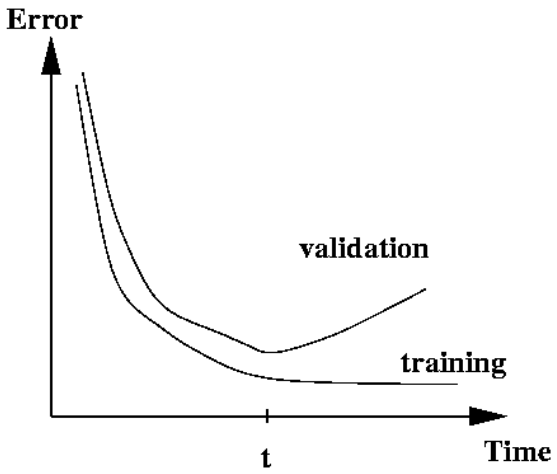
Muchas veces los modelos de aprendizaje cuentan con parámetros libres, como el número de neuronas en una red neuronal o la cantidad de árboles en un ensamble. En estos casos se emplean normalmente tres conjuntos en vez de dos, agregándose un conjunto de validación, el cual se utiliza para setear los parámetros libres del algoritmo. Luego se evalúa el error usando los parámetros a-priori más convenientes.

2.4. Dificultades

2.4.1. Sobreajuste

Se dice que una función clasificadora sobreajusta un conjunto de datos si se ajusta muy bien a los datos de entrenamiento pero da predicciones muy pobres para datos nuevos. Esto ocurre cuando hay insuficientes datos para entrenar el clasificador y por

Figura 1: Gráfica de sobreajuste



lo tanto los datos no cubren enteramente el concepto que se quiere aprender o cuando los datos contienen ruido y éste es incorporado a la función clasificadora [2]. En general cuando hay *sobreajuste* existen otras funciones clasificadoras que se ajustan menos a los datos de entrenamiento pero que obtienen una mejor performance sobre todo el conjunto de datos (incluyendo datos más allá de los de entrenamiento).

La Figura 1 muestra, para una red neuronal genérica, el error de entrenamiento y validación (testing) en función del tiempo de entrenamiento. A partir del tiempo t se empieza a notar el sobreajuste. Las técnicas más utilizadas para evitar el sobreajuste consisten en estimar el error sobre datos nuevos, con las estrategias enunciadas previamente, y seleccionar el ajuste del modelo que da un resultado óptimo en esa estimación.

2.4.2. Generalización

Al entrenar una función clasificadora esperamos que ésta tenga una buena performance para datos nuevos, es decir, que pueda *generalizar* a partir de datos de entrenamiento. Para obtener una buena generalización, se debe igualar la complejidad de la función clasificadora con la complejidad de la función subyacente a los datos. Si el clasificador es menos complejo que la función, se tiene un subajuste ("underfitting"), y al

mejorar la complejidad del clasificador, los errores de entrenamiento y testing decrecen. Pero si hacemos al clasificador muy complejo, obtenemos sobreajuste.

En todos los algoritmos de aprendizaje que entrenan a partir de datos de ejemplo existe una compensación entre tres factores:

- la complejidad del clasificador,
- la cantidad de datos de entrenamiento,
- el error de generalización para datos nuevos.

Mientras la cantidad de datos de entrenamiento se incrementa, el error de generalización decrece. Mientras la complejidad del clasificador crece, el error de generalización decrece al principio y luego empieza a incrementarse [2].

2.4.3. Dimensionalidad

El problema de dimensionalidad surge cuando tenemos conjuntos de datos donde los objetos son representados por muchas variables y se dispone de un número pequeño de ejemplos. A este tipo de datos se los conoce como "problemas anchos". Este caso no es anómalo, sino que actualmente tiende a ser el caso general. Se pueden pensar en muchos problemas tecnológicos actuales en los cuales se tiene en potencia un número muy grande de medidas que cuantifiquen ese evento, y relativamente pocas instancias. Ejemplos de esto son:

- Actualmente se trata de relacionar directamente enfermedades con genes. Pero típicamente se cuenta con resultados para miles de genes y relativamente pocos pacientes con una enfermedad particular.
- En reconocimiento de voz, se utilizan muchas muestras del habla de una persona (muchas palabras), pero relativamente pocas personas en la muestra.

Además, las nuevas técnicas de laboratorio han cambiado la forma de obtener mediciones sobre distintos tipos de eventos. En vez de centrar el análisis sobre unas pocas variables elegidas, por ejemplo, usando conocimiento científico previo sobre la materia, se utiliza la recolección automática y masiva de datos o variables de entrada, sobre las cuales se desconoce la relevancia real de cada una.

Bajo estas circunstancias, si imaginamos que los datos medidos sobre algún problema están en un espacio *d–dimensional*, al aumentar las variables medidas aumenta el número de dimensiones, pero el número de instancias se mantiene relativamente pequeño. Esto produce una dispersión de los datos, una baja en la densidad de los mismos, conocida como "curse of dimensionality", que permite al clasificador obtener una buena performance para los datos de entrenamiento, pero da resultados pobres para datos nuevos (sobreajuste) [2].

Una de las formas de atacar este problema es mediante la aplicación de las técnicas de selección de variables que se describen más adelante.

2.5. Clasificadores

A continuación se presentan brevemente los modelos de clasificación que han sido utilizados en este trabajo.

2.5.1. Random Forest

Esta técnica está basada en la teoría de ensambles de clasificadores [6]. Random forest, introducido por L. Breiman en 1999, utiliza un conjunto (o bosque) formado por muchos árboles de clasificación (decision trees). Para clasificar un nuevo objeto, cada árbol en el ensamble lo toma como entrada y produce una salida, su clasificación. La decisión del ensamble se toma como la clase con mayoría de votos en el ensamble [6].

En Random Forest cada árbol individual se desarrolla de una manera particular:

1. Dado un conjunto de datos de entrenamiento de cardinalidad N, toma N ejemplos

aleatoriamente con repetición (un bootstrap). Este será el conjunto de entrenamiento para crear el árbol.

2. Para crear cada nodo del árbol se utiliza únicamente una pequeña cantidad de las variables del problema. Si cada objeto tiene M variables de entrada, se determina un número $m << M$ y para cada nodo del árbol se seleccionan m variables aleatoriamente. La variable más relevante de este subconjunto elegido al azar se usa en el nodo. El valor de m se mantiene constante durante la expansión del bosque.

3. Cada árbol es desarrollado hasta la mayor extensión posible. No se realiza poda (pruning).

Breiman [6] muestra que el error del ensamble depende de dos factores:

1. La *correlación* entre dos árboles cualesquiera en el bosque. El incremento en la correlación produce un incremento en el error del bosque. La utilización de un subconjunto de variables elegidas al azar y de un boostrap de datos tiende a reducir dicha correlación.

2. La *fuerza* de cada árbol individual en el bosque. Un árbol con un error bajo es un clasificador fuerte. El incremento de la fuerza de árboles individuales decrementa el error del bosque. La utilización de árboles sin pruning va en este sentido.

2.5.2. Penalized Discriminant Analysis

Linear Discriminant Analysis (LDA) es un método de clasificación tradicional, desarrollado por Fisher en 1936 [18]. LDA es conceptualmente simple. Primero se produce una tranformación lineal del espacio de entrada, hacia un sistema de coordenadas en el cual se maximice la distancia entre puntos de distinta clase y a la vez se minimice la distancia entre puntos de la misma clase. Una vez en ese espacio, se clasifican los puntos midiendo la distancia al centroide de cada clase.

LDA presenta un número favorable de propiedades, como una robustez razonable a anomalías y aún a covarianzas de clase ligeramente diferentes. Pero por otro lado, existen dos grandes deficiencias que se dan en circunstancias opuestas:

- LDA es muy flexible en situaciones con un número grande (i.e., cientos) de predictores o variables altamente correlacionadas.
- Es demasiado rígido en situaciones donde los límites de clase en el espacio del predictor son complejos y no simplemente lineales.

En el primer caso, LDA sobreajusta. En el segundo, LDA subajusta los datos.

En este trabajo estamos interesados en la primera situación. Recientemente [5] se introdujo la técnica del *Penalized Discriminant Analysis* (PDA). En esta técnica la tranformación lineal se realiza utilizando algún método de regresión fuertemente regularizada, como por ejemplo Ridge Regression [5]. El método resultante ha demostrado ser muy eficiente en espacios de alta dimensión.

2.5.3. Support Vector Machines

La máquina de vectores soporte [19], o Support Vector Machines (SVM), es el método más importante introducido en la última década. El método está basado en las teorías de V. Vapnik sobre minimización del riesgo estructural [20]. En su forma más básica, el modelo SVM resuelve un problema de dos clases construyendo un hiperplano en el espacio de las variables de manera que se maximice la distancia al punto más cercano de cada clase (margen). Está solución conceptualmente simple tiene importantes propiedades teóricas y excelentes resultados prácticos.

El algoritmo se extiende a problemas no–lineales mediante el uso de funciones de Kernel que realizan una transformación (implícita) a un espacio de variables de mayor dimensión, donde el problema no–lineal original se puede resolver con simples métodos lineales. Las SVM también pueden ser extendidas a problemas multiclase. Para esto, se

convierte un problema de múltiples clases a varios problemas de dos clases. Una técnica común, *one–versus–all*, es construir tantos clasificadores como clases existan. Esto es, para cada clase, construir un clasificador que la separa del resto de las clases. Luego se elige la clase que clasifica los datos de test con mayor margen. Otra estrategia es construir un clasificador que discrimine cada par de clases, conocido como *one–versus–one*, y elegir la clase que es seleccionada por la mayor cantidad de clasificadores. Esto puede parecer muy costoso en principio, ya que se construyen $c(c-1)/2$ clasificadores (donde c es la cantidad de clases del problema), pero la cantidad de datos utilizados por cada clasificador es considerablemente menor, reduciendo el tiempo de entrenamiento.

3. Selección de variables

La selección de variables es el proceso mediante el cual se selecciona un subconjunto de las variables originales de un problema. Es particularmente útil para dominios como la biología y la química, donde un objeto medido automáticamente puede ser representado por hasta 60.000 variables. Este proceso reduce considerablemente la cantidad de variables utilizadas produciendo varios beneficios: facilidad para visualizar y entender los datos, eliminación de variables irrelevantes o redundantes, reducción de los requerimientos de medición y almacenamiento, reducción de los tiempos de entrenamiento y mejora en la calidad de las predicciones al mitigar el problema de la dimensionalidad [7].

Un proceso típico de selección de variables en un loop que consiste en cuatro pasos básicos: selección de un subconjunto de variables, evaluación del mismo, aplicación de un criterio de terminación y validación del resultado final.

Los algoritmos de selección de variables se clasifican en principio en dos categorías: el modelo *filter* y el modelo *wrapper*. El modelo filter utiliza propiedades estadísticas generales de los datos para evaluar y seleccionar conjuntos de variables. Es el método más antiguo e históricamente más utilizado. El modelo wrapper requiere un algoritmo de aprendizaje predeterminado y utiliza su performance como criterio de evaluación. Busca directamente las variables más adecuadas para el algoritmo de minería con el objetivo de mejorar su performance. Aunque con mejores resultados en general, es computacionalmente más costoso que el modelo filter, en particular para espacios de muy alta dimensión [8].

En este trabajo hemos utilizado el algoritmo de selección de variables llamado Recursive Feature Elimination [9] ó Eliminación Recursiva de Variables (RFE), que se describe a continuación, el cual es un algoritmo perteneciente al modelo wrapper pero con una carga computacional muy baja.

3.1. Recursive Feature Elimination

El algoritmo RFE es un proceso simple y relativamente eficiente para seleccionar variables. La idea es construir un clasificador y utilizar información disponible en el mismo para ordenar las variables del problema de acuerdo a su importancia. Una vez ordenadas, se elimina el subconjunto de menor importancia (en algunos casos se elimina una variable a la vez, en otros un número mayor) y se itera el proceso. La iteración, que agrega una gran carga computacional, es necesaria para poder enfrentar el problema de la correlación entre variables. Cuando dos variables importantes tienen una alta correlación, los clasificadores tienden a repartir la "importancia" entre ambas. Recién cuando RFE remueve una de ellas, la otra toma toda la "importancia" y pasa a ocupar lugares más altos en el ranking. De aquí el significado de repetir el ranking después de cada eliminación [9]. El algoritmo se puede combinar con cualquier clasificador que permita extraer información sobre la influencia de cada variable en la asignación de clase.

3.2. Procedimiento experimental para la selección de variables

Recientemente, luego de la publicación de algunos trabajos con errores metodológicos [9], se redescubrió la importancia de utilizar un procedimiento experimental para la selección de variables que no tenga el "bias de selección" [10]. En este caso se implementó un procedimiento que consiste principalmente en dos procesos anidados. El bucle externo realiza n particiones aleatorias sobre un dataset, resultando en n subconjuntos de entrenamiento y n subconjuntos de test, y promedia los resultados de estos subconjuntos. El bucle interno realiza la selección de subconjuntos anidados de variables y el desarrollo de los clasificadores sobre estos subconjuntos (utilizando sólo el subconjunto de entrenamiento provisto por el bucle externo) [4].

Dados un conjunto de datos de entrenamiento y test (creados por el bucle externo), los pasos que sigue el bucle interno se pueden describir, a grandes rasgos, de la siguiente

manera:

1. Se entrena un clasificador con un conjunto de entrenamiento dado.
2. Se crea un ranking de variables.
3. Se calcula el error en conjuntos de test utilizando esas variables.
4. Se elimina un grupo de variables.

Estos pasos se repiten hasta que se cumpla el criterio de terminación, que en este caso es una condición sobre la cantidad de variables. A continuación se describen con más detalle los pasos enumerados.

3.2.1. Entrenamiento

El entrenamiento de una función de clasificación consiste en el ajuste de ciertos parámetros particulares del modelo utilizado de modo que éste represente una aproximación óptima de la función subyacente en los datos de entrenamiento. Para nuestro trabajo el loop externo realiza una partición con el 75 % de los datos para entrenar el clasificador y el 25 % para test. A su vez, el loop interno utiliza una división en k-folds del conjunto de entrenamiento para seleccionar el valor óptimo de los parámetros internos de los clasificadores (como la constante de regularización C de las SVM).

3.2.2. Ranking de variables

Crear un ranking de variables consiste en ordenar las mismas utilizando algún criterio de importancia. Consideremos un conjunto de m ejemplos $\{\mathbf{x}_k, y_k\}(k = 1, ..., m)$, donde cada ejemplo consiste en n variables de entrada $x_{k,i}(i = 1, ..., n)$ y una variable de salida y_k. Para armar un ranking de variables se hace uso de una función de puntajes $S(i)$ computada a partir de los valores $x_{k,i}$ y $y_k, k = 1, ..., m$. Por convención, se asume

que un puntaje alto corresponde a una variable valiosa y que se ordenan las variables en orden decreciente de $S(i)$ [7].

Por ejemplo, para un problema con dos clases, (+) y (-), se puede utilizar para formar un ranking el coeficiente de correlación [13], definido como

$$w_i = \frac{\mu_i(+) - \mu_i(-)}{\sigma_i(+) + \sigma_i(-)} \tag{3}$$

donde μ_i y σ_i son la media y desviación estándar de los valores de la variable i para todos los ejemplos de clase (+) o de clase (-), con $i = 1, ..., n$. Valores grandes positivos de w_i indican una correlación fuerte con la clase (+), mientras que valores grandes negativos indican una correlación fuerte con la clase (-). En este procedimiento básico los coeficientes w_i son computados con información sobre una sola variable, no se tiene en cuenta información mutua entre variables. Como se explicó anteriormente, este problema se soluciona creando el ranking de variables a partir de los modelos de clasificación, que incluyen la información de todas las variables a la vez. En este trabajo se utilizan los siguientes clasificadores para crear rankings de variables:

Random Forest Random Forest crea una estimación interna de la importancia de las variables con dos métodos distintos.

Estimación con los conjuntos out-of-bag Cuando se crea un bootstrap para entrenar un árbol, aproximadamente un tercio de los casos son dejados fuera de la muestra. Estos datos, llamados out-of-bag (*oob*), son utilizados para obtener una estimación no sesgada del error de clasificación mientras los árboles se agregan al bosque. También son utilizados para obtener una estimación de la importancia de las variables [6].

En primer término se estima el error de clasificación del ensamble sobre estos conjuntos *oob*. Luego se toma una variable m y se permutan sus valores dentro de los datos de entrenamiento, decorrelacionando esa variable de lo aprendido por el modelo. Se re-

pite ahora la estimación del error *oob*. Intuitivamente, si la variable no era importante para el ensamble el error no debería cambiar mucho al desordenar la variable. Por el contrario, un gran cambio en el error implica una gran importancia de la variable. Se repite este procedimiento y se ordenan las variables de acuerdo al cambio relativo en la estimación del error *oob* que produce cada una.

Importancia GINI Otro método más directo de estimar la importancia de las variables es el criterio GINI. En este caso, cada vez que un nodo de un árbol utiliza una variable dada m se computa el cambio en el criterio de pureza GINI Index entre el nodo original y las dos ramas resultantes. Este cambio se promedia para todas las veces que se usa esa variable, dentro de cada árbol y en todo el ensamble. El valor resultante da una medida relativa de la importancia, al evaluar en cierto modo la bondad de las decisiones provocadas por la variable.

Penalized Discriminant Analysis Como se describió anteriormente, PDA realiza una transformación lineal a un nuevo sistema de coordenadas, en donde se utiliza la distancia de Mahalanovis a los centroides de las clases para realizar la clasificación. El nuevo sistema crea ejes en los cuales la discriminación entre clases es máxima. Para crear un ranking de variables se analiza en este caso la participación de cada variable original en estos nuevos ejes, ordenándolas de acuerdo al peso relativo promedio de cada una en la transformación lineal. Es simple entender que una variable con gran discriminación tendrá un gran peso en la combinación lineal que crea los nuevos ejes, mientras que una variable independiente de las clases tendría que tener un peso nulo.

Support Vector Machine SVM crea una superficie de decisión usando un hiperplano. Como indico Guyon et al. [9], la componente de cada variable del problema en el vector normal al hiperplano da una medida natural de la importancia de una variable. Como en el caso del PDA, una variable importante define la dirección del hiperplano, y

tiene por lo tanto una componente grande. Una variable irrelevante tiende a tener una componente nula. En este trabajo se utiliza un clasificador SVM multiclase construido a partir de clasificadores *One–Vs–One*. En este caso, como se había mencionado, un problema con c clases se descompone en $p = c(c-1)/2$ problemas binarios. Para resolver cada uno, entrenamos un clasificador SVM lineal, obteniendo p funciones clasificadoras

$$D_i(x) = \mathbf{x}\mathbf{w}_i \qquad i = 1, ..., p \tag{4}$$

Los vectores peso $\mathbf{w}_i$ correspondientes a los problemas binarios luego son promediados

$$\mathbf{W} = \frac{1}{p}\sum_{i=1}^{p}\mathbf{w}_i \tag{5}$$

y las componentes de $\mathbf{W}$ son utilizadas para rankear las variables [4].

3.2.3. Error de clasificación

En cada ejecución del bucle interno se obtiene, para los ejemplos pertenecientes al conjunto de test, la probabilidad de que ese ejemplo pertenezca a cada clase. Estas probabilidades se calculan para todos los subconjuntos de variables utilizados. Para predecir la clase de cada ejemplo para un subconjunto de variables determinado, se toma la probabilidad más alta.

A partir de estas predicciones se calcula el error de clasificación para un conjunto de test. El error de clasificación se define como la cantidad de ejemplos mal clasificados sobre la cantidad total de ejemplos. Esta medida no afecta el comportamiento del algoritmo RFE, sólo muestra la performance sobre el conjunto de test utilizado.

Finalmente, sobre todas las corridas del bucle externo, y por cada cantidad de variables consideradas en el bucle interno, se promedian los errores de clasificación y se obtiene una estimación del error para dicho caso.

3.2.4. Eliminación de variables

La cantidad de variables a eliminar en cada paso se determina por dos parámetros de entrada del algoritmo RFE. En primera instancia, en cada paso de elimina un porcentaje de las variables, típicamente el 10 %. El segundo identifica un número de variables a partir del cual se realiza un procedimiento menos agresivo y se eliminan las variables de una a la vez. Las variables eliminadas siempre corresponden a las últimas en el ranking, y no son vueltas a considerar. De esta manera, el algoritmo RFE produce subconjuntos anidados $S_1 \subset S_2 \subset \cdots \subset S_n$, donde la posición en el ranking de una variable cualquiera en el subconjunto S_i puede diferir de la posición en el subconjunto S_{i+1}, con $i = 1, ..., n-1$. Por cada iteración del bucle externo se obtiene un ranking total de variables, el cual se compone de la suma de ranking parciales de variables presentes en el subconjunto S_{i+1} y eliminadas en el subconjunto S_i con $i = 1, ..., n-1$.

3.3. Estabilidad

Se puede pensar a la *estabilidad* de un algoritmo de selección de variables como una medida de la sensibilidad de la solución (variables seleccionadas) a perturbaciones en los datos de entrada. La motivación para explorar la estabilidad es dar evidencias de que el proceso de selección es relativamente robusto frente a cambios leves en los datos, en otras palabras, que la información que se extrae del método es confiable. Los métodos de selección de variables que producen soluciones consistentes son siempre preferibles a aquellos con resultados muy volátiles [15].

Vale la pena aclarar que la estabilidad es una medida independiente de la performance de las variables seleccionadas. Aún con buenos resultados en selección, grandes variaciones en los subconjuntos de variables seleccionadas indican que algo anda mal. Por ejemplo, el algoritmo de selección de variables no es apropiado para los datos analizados, no hay suficientes muestras, etc. Por lo tanto, se debe confiar menos en conjuntos de variables que cambian radicalmente con variaciones leves en los datos de entrada.

3.3.1. Cuantificación

En cada iteración del bucle externo en el algoritmo RFE, se produce un ranking de las variables del problema. Para medir qué estables son estos rankings, se asigna a cada variable una posición relativa en el ranking. Este es un valor entre 0 y 1, en una escala lineal, desde la última posición del ranking a la primera. Para evaluar la estabilidad del algoritmo se puede medir la estabilidad de las posiciones relativas que toma cada variable en el ranking. Un método estable tiende a dar posiciones bastante cercanas en cada corrida, mientras que uno inestable producirá una gran variación en las posiciones asignadas. Evaluando la dispersión de estos rankings relativos, por ejemplo con la mediana o la distancia inter–cuartiles (IQR), se puede tener una evaluación cuantitativa de la estabilidad.

4. RFE sobre clasificadores combinados

En este capítulo se presentará una técnica de selección de variables para problemas multiclase basada en el algoritmo RFE. Esta técnica combina los resultados individuales de las ejecuciones *one-vs-all* formando un nuevo clasificador, sobre el cual se seleccionan las variables. Se analizarán los errores de clasificación obtenidos y la estabilidad para los modelos utilizados.

4.1. One Versus All

El esquema de clasificación "one-versus-all" (OVA) es conceptualmente simple. Consiste en entrenar C clasificadores binarios (donde C es la cantidad de clases del problema), cada uno creado para distinguir los ejemplos de una clase de los ejemplos de las clases restantes. Para elegir variables en un clasificador OVA, básicamente se ejecuta el algoritmo RFE descripto anteriormente tantas veces como clases exista. Las particiones de datos a utilizar para entrenamiento y para test en cada ejecución se mantienen constantes. En la i-ésima ejecución, se etiqueta la clase i con el número 1 y todas las restantes clases con 2. Al aplicar el método RFE sobre estos problemas binarios se seleccionarán las variables más significativas para cada clase, es decir, aquellas que distingan la clase del resto. Discutiremos a continuación las ventajas y desventajas de este enfoque.

4.1.1. Fundamentos

El clasificador combinado intenta explotar la idea de que cada clase puede ser distinguida de las demás por los valores que asume un conjunto de variables particular. Así, dos clases pueden separarse utilizando subconjuntos de variables que tengan unas pocas o ninguna variable en común. En cambio, al utilizar un clasificador único para todo el problema es necesario seleccionar un subconjunto de variables capaz de distinguir cada clase de las restantes a la vez. Es probable que los conjuntos de variables seleccionados en uno y otro caso sean diferentes en cantidad o composición.

4.1.2. Clasificación

Nuestro clasificador combinará los resultados obtenidos en cada ejecución del RFE–OVA. En la ejecución i-ésima del algoritmo RFE con las clases renombradas (es decir, cuando se evalúa a la clase i contra todas las demás), en la j- ésima iteración del bucle externo y en la k-ésima iteración del bucle interno se obtiene una probabilidad sobre el ejemplo e de un conjunto de test. Llamemos a este valor p^e_{ijk}. El clasificador combinado asignará al ejemplo e la clase $c = argmax_i(p^e_{ijk})$.

4.1.3. Cantidad de variables seleccionadas

Dado que uno de los objetivos primordiales del algoritmo es conseguir buenas clasificaciones con subconjuntos relativamente pequeños de variables, un valor importante de nuestro clasificador es la cantidad real de variables que utiliza. Algunos trabajos previos consideran al número de variables usada en cada clasificador OVA como la cantidad de variables del clasificador combinado. En este trabajo llamaremos a esta cantidad el número *aparente* de variables. Este número no es una indicación fiel de la cantidad de variables utilizadas, en particular en problemas con muchas clases. Al tener cada clasificador OVA la libertad de elegir sus propias variables, la cantidad real de variables involucradas en el clasificador combiando puede ser mucho mayor.

Sea V_{ikj} el subconjunto de variables obtenido luego de la ejecución i-ésima del algoritmo RFE, en la j-ésima iteración del bucle externo y en la k-ésima iteración del bucle interno de dicha ejecución. Si llamamos con V^c_{kj} al subconjunto real de variables utilizados por el clasificador combinado en la j-ésima iteración del bucle externo y en la k-ésima iteración del bucle interno, entonces:

$$V^c_{kj} = \bigcup_{i=1}^{C} V_{ijk} \tag{6}$$

donde se tiene que $|V^c_{kj}| \geq |V_{ijk}|$ para todo $i = 1, ..., C$, donde la norma del subcon-

junto es la cantidad de elementos que la componen.

4.1.4. Estabilidad del RFE combinado

Como discutimos en el capítulo anterior, para analizar la estabilidad de las selecciones se necesita asignar un ranking relativo a cada variable en el problema. En el caso del RFE combinado esto no es tan directo como en el caso de un único clasificador multiclase que se definió previamente.

Para armar el ranking relativo en este caso se adoptó la estrategia más simple posible para combinar las selecciones de los distintos clasificadores OVA. Se considera en principio la primer posición del ranking. Se toman todas las variables que ocupan esa posición para cada clase y se eliminan las repetidas. Supongamos que tenemos entonces un subconjunto de v variables, que deberían ocupar las primeras v posiciones del ranking relativo. Les asignaremos a las v variables la misma posición, la media entre 1 y v. Pasamos entonces a mirar la segunda posición del ranking original de cada clasificador. Tomamos todas las variables que ocupan esa posición para cada clase y se eliminan ahora no sólo las repetidas, sino también las que ya fueron seleccionadas en la posición anterior. A este nuevo subconjunto, supongamos con t variables, se le asigna (a todas las variables por igual) la media entre $v + 1$ y $v + t$. Este proceso se reitera hasta que todas las variables tienen un ranking asignado, y recién entonces el ranking se normaliza dividiendo por la cantidad total de variables, para conseguir el ranking relativo final.

A partir de este ranking relativo se puede analizar la estabilidad de las selecciones considerando medidas de dispersión del ranking, como la MIQ que se describió previamente.

5. Resultados

En este capítulo se describen los experimentos realizados para comparar los métodos directo y combinados del RFE, incluyendo los datasets utilizados. Luego se muestran los resultados obtenidos y se los analiza.

5.1. Codificación

El código del RFE combinado fue escrito en R, un lenguaje open–source para computación estadística y gráficas [17]. Se construyó sobre el código del paquete RFE, anteriormente desarrollado en el grupo de trabajo, haciendo previamente una reingeniería del mismo para obtener mayor granularidad en el código, así como también facilitar la expansión futura del mismo. Los detalles de la implementación se describen en el Apéndice C.

5.2. Datasets

Se analizaron en todos los casos datos reales de interés tecnológico actual. Los primeros dos datasets corresponden a problemas de expresión de genes medida con DNA microchips, donde se busca predecir el tipo de enfermedad a partir de la expresión genética. Los cuatro problemas restantes son mediciones directas del contenido de volátiles (del olor) emitidos por distintos productos agroalimentarios, utilizando un espectrómetro de masa del tipo PTR-MS. El objetivo es discriminar entre variedades de especies o clases de productos típicos, para control de calidad o tipicalidad. A continuación se describen brevemente las características de los mismos. Durante el desarrollo del trabajo se utilizaron otros 6 datasets reales, 5 de genómica y uno de actividad química (QSAR). No se incluyen en este informe por un problema de espacio, y por ser los resultados cualitativamente iguales a los presentados.

Brain Tumor 1 En este problema se utilizaron 90 ejemplos, cada uno descripto por 5921 variables. Las 5 clases corresponden a distintos clases de tumores cerebrales (entre paréntesis se muestra la cantidad de ejemplos de la clase): *0*:Medulloblastoma (60); *1*: Malignant glioma (10); *2*: AT/RT (10); *3*: Normal cerebellum (4); *4*: PNET (6).

Brain Tumor 2 Nuevamente, las clases corresponden a tejidos de tumores cerebrales. Se utilizaron 50 ejemplos, cada uno descripto por 10367 variables. Las 4 clases corresponden a *0*: Classic Glioblastomas (14); *1*: Classic Anaplastic Oligodendrogliomas (7); *2*: Non-classic Glioblastomas (14); *3*: Non-classic Anaplastic Oligodendrogliomas (15).

Fragola Se utilizaron 233 ejemplos, cada uno descripto por 232 variables. Las 9 clases corresponden a distintas variedades de frutillas, evaluadas durante 3 años en distintos campos experimentales en el norte de Italia. Las clases están bastante balanceadas, la más numerosa tiene 30 muestras y la menor 21.

Grana En este problema se midieron con PTR-MS 15 de ejemplos de cuatro quesos tipo Grana de diferenes orígenes y grado de madurez. En total hay 60 muestras, cada una descripta por 235 variables. Las 4 clases corresponden a *0*:Grana Padano (15); *1*:Parmegiano Reggiano (15); *2*:Grana Trentino (15); *3*:Grana Trentino joven (15).

Lampone Se analizaron 92 ejemplares, cada uno descripto por 232 variables. Las 5 clases corresponden a distintas variedades de Arándanos cultivadas en la provincia de Trento, norte de Italia. Las clases están bastante balanceadas, hay 3 con 19 ejemplos, una con 18 y una con 17.

Mirop Este dataset es similar al Grana, pero las muestras corresponden a 6 quesos semi-duros típicos del Trentino, llamados Nostrani. Se utilizaron 48 ejemplos, cada uno descripto por 240 variables.

5.3. Parámetros utilizados

Para cada dataset, cada modelo de clasificación y cada ejecución OVA se realizaron 100 iteraciones. En cada iteración se tomó una partición al azar con el 75 % de los datos para entrenamiento y el 25 % restante para test. En cada iteración del bucle interno, en el algoritmo RFE, se eliminó el 10 % de las variables, hasta alcanzar las 20 variables. A partir de este valor el algoritmo eliminó una variable por iteración. Cada vez que se entrenó un clasificador sus parámetros internos fueron optimizados a partir de valores en grilla usando K-folds sobre el conjunto de entrenamiento solamente.

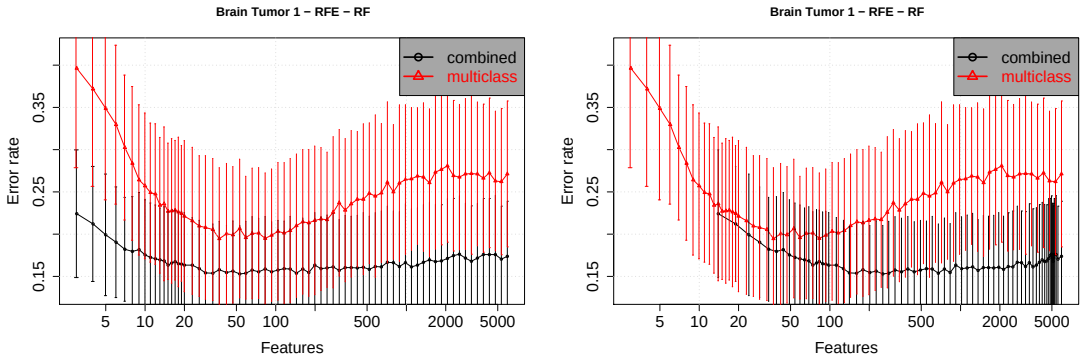
5.4. Evaluación: Calidad de los subconjuntos

Para evaluar la calidad de los conjuntos de variables seleccionadas con cada método, para cada dataset se generaron curvas de error de clasificación en función del número de variables utilizadas. Este porcentaje de error es una media de los errores obtenidos en las 100 iteraciones del bucle externo, tanto para el método directo como para el combinado. Cabe destacar que los subconjuntos de variables utilizados para un número particular de variables generalmente no son iguales, debido a que en cada iteración del bucle externo se utilizan distintos conjuntos de datos de entrenamiento y test. Por lo tanto la gráfica no muestra el porcentaje de error de clasificación para un subconjunto de variables particular, sino que presenta una media para una cierta cantidad de variables.

Para cada dataset y para cada modelo de clasificación se generaron dos gráficas. En la primera mostramos el resultado de ejecutar el RFE combinado tomando la cantidad de variables aparente, discutida en el capítulo anterior. En la segunda gráfica mostramos los mismos valores de error, pero en función del número real de variables usadas en cada caso. En ambas figuras se incluye siempre el resultado del método directo correspondiente, para tener una comparación de los resultados.

Como primer ejemplo, en la Figura 2 se muestran los resultados sobre el dataset Brain Tumor 1, utilizando para el ranking el modelo de clasificación Random Forest

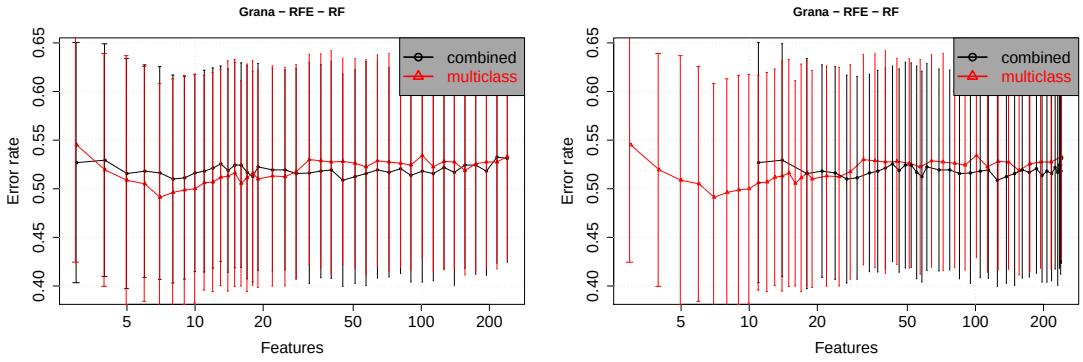
Figura 2: Gráfica aparente y real para el dataset Brain Tumor 1



con GINI.

En este caso el resultado aparente combinado resulta significativamente mejor que el directo. Primero, mejora el promedio de error para todas las cantidades de variables utilizadas. En particular, utilizando 3 variables se mejora el promedio de error en más de un 15 %, y con 15 variables en un poco más que 5 %. Pero esta misma comparación para el resultado real, en el panel de la derecha de la Figura, da muy diferente. Podemos notar, como se ha dicho, que la línea del combinado se ha desplazado hacia la derecha, porque el número real de variables usadas en cada caso es mayor al aparente. Por ejemplo, el mínimo de variables utilizadas en el resultado aparente es 3, cuando en realidad se utilizaron 15 variables para obtener tal porcentaje de error. En este caso se utilizaron 3 variables para cada una de las 5 ejecuciones *one-versus-all* (una por cada clase), donde casualmente cada una de estas seleccionó un conjunto de variables con intersección vacía con los conjuntos restantes, dando un total de $5 \times 3 = 15$ variables. Es importante destacar que el encogimiento no es lineal, es decir, si para 3 variables en el resultado aparente corresponden 15 variables en el real, para n variables en el aparente no necesariamente corresponden $n \times 5$ ó $n + 12$ variables en el resultado real. Esta diferencia en las variables utilizadas hace que, para 15 variables, en vez de haber algo más que un 5 % de mejora en el porcentaje de error, haya una mejora mucho menor.

Figura 3: Gráfica aparente y real para el dataset Grana



Sin embargo, el combinado real mejora al directo para todas las cantidades de variables utilizadas en este ejemplo.

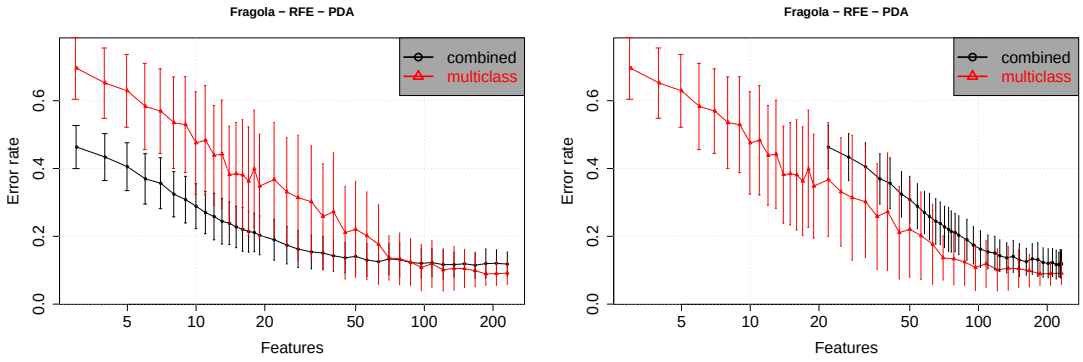
Veamos ahora otro ejemplo donde se obtuvieron resultados similares a los del clasificador directo. En la Figura 3 se muestran los resultados obtenidos con el dataset Grana, utilizando para el ranking el modelo de clasificación Random Forest.

En las figuras se puede observar resultados similares para ambos métodos combinados. En este caso, al no haber grandes cambios en el error al reducir la cantidad de variables, no hay una influencia del número real o aparente de variables. Para cualquier cantidad de variables el porcentaje de error no varía más que un 3 %.

Veamos, por último, un caso donde el método RFE combinado mostró resultados peores a la selección sobre el clasificador multiclase directo. En la Figura 4 se muestran los resultados para el dataset Fragola, utilizando el modelo de clasificación PDA.

Comparándo ambas figuras se puede ver un cambio radical entre la medición aparente y la real de la performance del método combinado, en relación al directo. En la gráfica del combinado aparente, a partir de las 70 u 80 variables éste supera la performance del directo, con porcentajes de error de hasta un 20 % menor en los casos con pocas variables. Sin embargo, el combinado real muestra resultados totalmente desfavorables. Cuando se aparentaba usar 3 variables, promediando un error de alrededor de un 45 %,

Figura 4: Gráfica aparente y real para el dataset Fragola

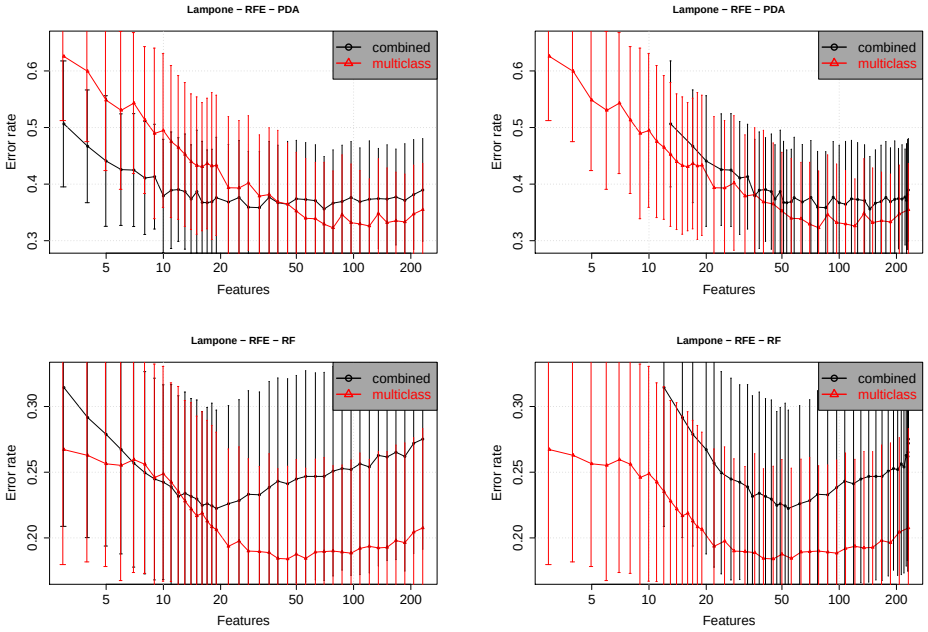


en realidad se estaban utilizando algo más de 20 variables (recordemos que este es un dataset de 9 clases). Con esa cantidad de variables, el método directo obtuvo menos del 40 % de error, considerablemente menor al combinado. Además, es facil ver que el directo obtiene porcentajes de errores menores para todas las cantidades de variables evaluadas.

En trabajos previos, otros autores [16, 21] mostraron resultados con métodos combinados que parecen ser claramente superiores a los métodos directos, pero utilizando el método aparente para medir la cantidad de variables. En oposición, la mayoría de nuestros resultados, que se muestran en el Apéndice A, son del tipo de los mostrados en las últimas dos figuras, donde las mejoras son sólo en la gráfica aparente, y en el caso real el método directo funciona igual o mejor al combinado. En nuestros experimentos se observa que en la gráfica real se produce un "encogimiento" del resultado combinado aparente. Esto se debe a que los porcentajes de error no cambian, sólo cambia la cantidad de variables correspondiente a esos valores y, en general, esa cantidad aumenta considerablemente, convirtiendo resultados que aparentaban ser buenos en resultados similares para ambos métodos.

Otro punto interesante de analizar es cómo cambian los resultados en función del método de clasificación que se utiliza. En las Figuras 5 y 6, usando el dataset Lampone,

Figura 5: Gráfica aparente y real para el dataset Lampone, para los rankings basados en PDA y RF

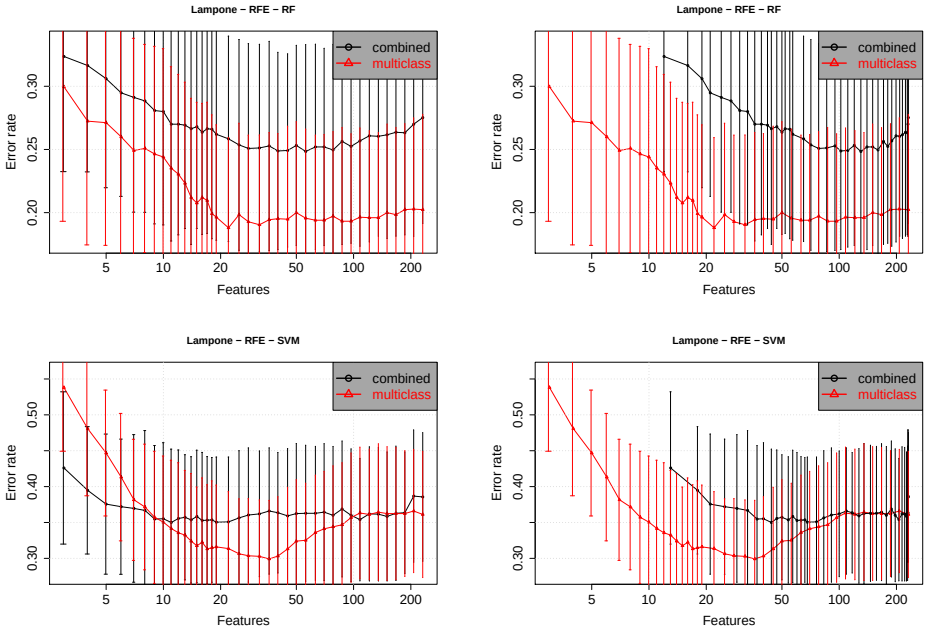


mostramos una comparación de los métodos directo y combinado sobre cuatro diferentes modelos de clasificación, usados para producir los rankings y las discriminaciones. Del análisis de esta figura y las correspondientes a los otros datasets analizados, que se muestran en el apendice A, se desprende que en general la relación entre los métodos directo y combinado es casi independiente del tipo de clasificador usado para construir el ranking. En las Figuras 5 y 6 se observa que, cualitativamente, tanto para las gráficas reales como las aparentes, las diferencias entre los métodos se mantienen. El efecto de usar distintos clasificadores se observa como un cambio global en los niveles de error de la figura, que afecta por igual a ambos métodos para seleccionar variables.

5.5. Evaluación: Estabilidad

Partiendo del concepto y cálculo de estabilidad presentados, podemos decir que un modelo de clasificación es *"más estable"* cuando, a pesar de una variación en los datos de

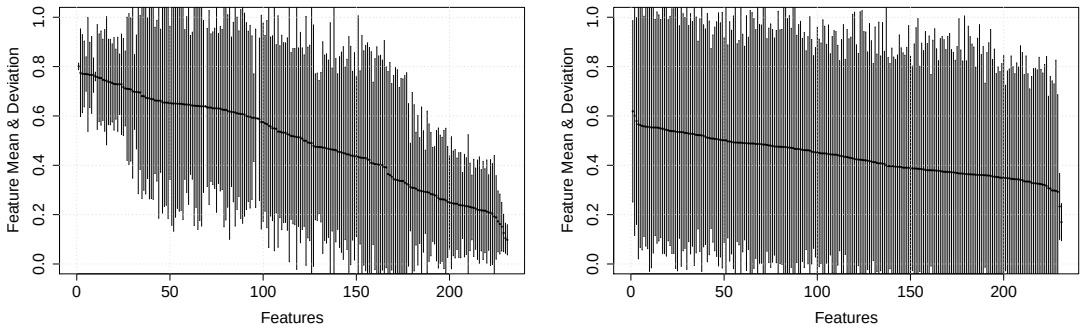
Figura 6: Gráfica aparente y real para el dataset Lampone, para los rankings basados en RF GINI y SVM



entrenamiento, el ranking de variables que genera no varía mucho. Podemos pensar en el modelo más estable posible; éste generaría el mismo ranking de variables sin importar el conjunto de datos provisto para su entrenamiento. La variación de las posiciones de las variables en el ranking sería nula, y la media de cada variable pertenecería a una línea recta que comienza en 1 para la variable primera en el ranking y que termina en 0 para la última. Por otro lado, el caso opuesto corresponde a aquel donde las variables varían de posición en los rankings prácticamente al azar. En este caso para todas las variables la posición media sería 0.5, y su variación, medida por el IQR, toma valores grandes. Cabe destacar que una variable puede tener media 0.5 debido que siempre obtiene el puesto de la mitad del ranking en todos los generados, lo que sería un comportamiento estable, pero en este caso son todas las variables al mismo tiempo y asociadas a un gran IQR.

Para comparar la estabilidad de los métodos directo y combinados, vamos a recurrir

Figura 7: Gráfica de estabilidad del método directo y combinado para el dataset Fragola utilizando clasificadores SVM



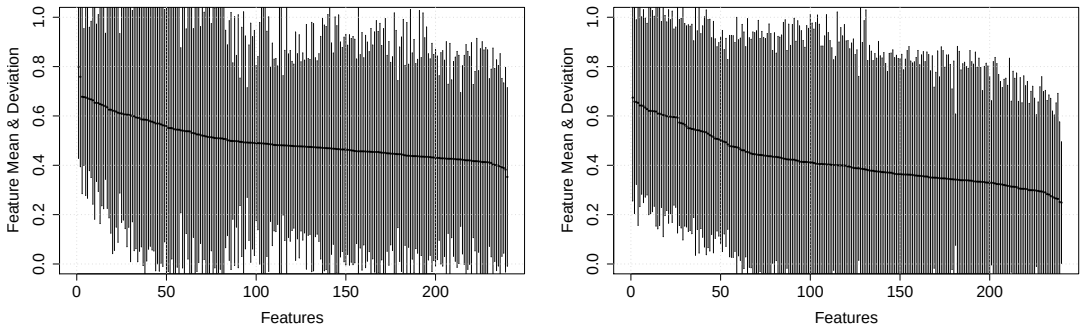
al análisis de la distribución de las posiciones relativas de las variables en las distintas ejecuciones de los métodos. La forma de medir las posiciones relativas se discutió en el capítulo anterior. En un primer análisis se han confeccionado gráficas con los valores de las medias e IQR para cada método, el RFE directo y el combinado.

En la Figura 7 se muestran los resultados obtenidos para el dataset Fragola utilizando clasificadores SVM. Podemos decir que el método directo, panel de la izquierda, es más estable que el combinado. Las medias obtenidas por el directo se acercan más al ideal de estabilidad descripto anteriormente, mientras que las variaciones en los rankings son menores para este método. El método combinado muestra su mayor inestabilidad al estar las medias de todas las variables cercanas al valor 0.5, con grandes variaciones.

Para el dataset Grana, utilizando el modelo RF con GINI, Figura 8 no se observan diferencias significativas en las gráficas obtenidas entre los dos modelos, directo y combinado.

En el apéndice B se muestran las gráficas correspondientes a los demás datasets. En todos los casos se puede observar que cualitativamente los comportamientos son independientes del clasificador utilizado para generar los rankings. En general, el método combinado es equivalente o más inestable que el método directo, pero nunca más estable.

Figura 8: Gráfica de estabilidad del método directo y combinado para el dataset Grana utilizando clasificadores RF con GINI



6. Conclusiones

En este trabajo se introdujo un nuevo método de selección de variables para problemas de clasificación multiclase, el método RFE combinado. El método está basado en aplicar un clasificador *one-versus-all* al problema de C clases y realizar una selección de variables con el algoritmo RFE sobre cada unos de los C subproblemas binarios creados. La unión de las variables seleccionadas por cada subproblema se toma como la selección producida por el método combinado. El nuevo método se comparó con el método directo (utilizar RFE sobre un clasificador capaz de resolver directamente un problema multiclase). Utilizando varios datasets reales de genómica y espectrometría de masa y diversos clasificadores, se evaluó la calidad de las soluciones obtenidas en niveles de error y en estabilidad de las mismas.

Se consideraron dos interpretaciones distintas sobre la cantidad de variables utilizadas para clasificar. A la forma más simple, considerar la cantidad de variables seleccionadas por cada subproblema, la llamamos aparente, debido a que entendemos que no contabiliza adecuadamente la cantidad de variables que se utilizan para clasificar. Al segundo método lo llamamos real, ya que creemos que es la manera correcta de calcular estas cantidades.

En general, nuestros resultados con la medición aparente del error muestran que el nuevo método combinado no obtiene mejoras significativas en relación al método directo, con excepción de los resultados correspondientes al dataset Brain Tumor 1 y un caso para los datos de Fragola y Lampone. El método presenta algunas mejoras cuando la cantidad de variables utilizadas para clasificar es pequeña, lo cual sería importante por ser la zona de mayor interés en selección de variables. Sin embargo, al analizar el número real de variables usados por los modelos se verifica que, con excepción del dataset Brain Tumor 1, se obtienen siempre resultados iguales o peores que los del método directo.

El resultado es contradictorio, porque trabajos publicados anteriormente utilizando otros métodos combinados mostraban ganancias detectables sobre los métodos directos.

Creemos que es posible que los trabajos previos hayan utilizado la forma incorrecta de medir la cantidad de variables, lo que nosotros llamamos cantidad aparente. Lamentablemente, en ningún trabajo se detalla el proceso utilizado para medirlas.

El análisis de la estabilidad para el método directo en comparación con el método combinado arrojó resultados contundentes. Para ningún dataset el método combinado se mostró más estable que el método directo. Esto es probablemente una consecuencia de la mayor libertad para elegir variables que tiene el método combinado.

Los trabajos que se plantean a partir de esta tesis incluyen el uso de otros métodos de combinación de clasificadores binarios, como por ejemplo el one–vs–one, o el uso de clasificadores ECOC.

Referencias

[1] Tom M. Mitchell, "Machine Learning", Mcgraw–Hill, 1997.

[2] Ethem Alpaydin. Introduction to Machine Learning (Adaptive Computation and Machine Learning). The MIT Press, October 2004.

[3] John Loughrey, Pádraig Cunningham. Overfitting in Wrapper-Based Feature Subset Selection: The Harder You Try the Worse it Gets.

[4] Pablo M. Granitto, Franco Biasioli, Cesare Furlanello, Flavia Gasperi. Efficient Feature Selection for PTR-MS Fingerprinting of Agroindustrial Products. Chemometrics and Intelligent Laboratory Systems, 83:2, 42-51, 2006.

[5] Trevor Hastie, Andreas Buja, Robert Tibshirani. Penalized Discriminant Analysis. Tech Report AT&T Bell Laboratories, May 1994.

[6] Leo Breiman. Random Forests. Machine Learning, 45, 5–32, 2001.

[7] I. Guyon, A. Elisseeff, "An Introduction to Variable and Feature Selection", Journal of Machine Learning Research, 3(Mar), 1157-1182, 2003.

[8] Huan Liu, Lei Yu, "Toward Integrating Feature Selection Algorithms for Classification and Clustering,"IEEE Transactions on Knowledge and Data Engineering, vol. 17, no. 4, pp. 491-502, Apr. 2005.

[9] Isabelle Guyon , Jason Weston , Stephen Barnhill , Vladimir Vapnik, Gene Selection for Cancer Classification using Support Vector Machines, Machine Learning, v.46 n.1-3, p.389-422, 2002.

[10] Ambroise, C., & McLachlan, G. J. Selection Bias in Gene Extraction on the Basis of Microarray Gene–Expression Data. PNAS, 99(10), 6562-6566, 2001.

[11] Huan Liu , Edward R. Dougherty , Jennifer G. Dy , Kari Torkkola , Eugene Tuv , Hanchuan Peng , Chris Ding , Fuhui Long , Michael Berens , Lance Parsons , Zheng Zhao , Lei Yu , George Forman, Evolving Feature Selection, IEEE Intelligent Systems, v.20 n.6, p.64-76, November 2005.

[12] Gregory Piatetsky-Shapiro , Pablo Tamayo, Microarray data mining: facing the challenges, ACM SIGKDD Explorations Newsletter, v.5 n.2, December 2003.

[13] Todd Golub, Donna Slonim, Pablo Tamayo, Christine Huard, Michelle Gaasenbeek, Jill Mesirov, Hilary Coller, Mignon Loh, James Downing, Mark Caligiuri, Clara Bloomfield, and Eric S. Lander. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. Science, 286(5439), 531-537, October 1999.

[14] J.R. Quinlan. C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, 1993.

[15] Pavel Krízek, Josef Kittler, Václav Hlavác. Improving stability of feature selection methods. Proceedings of CAIP 2007, LNCS 4673, pp. 929–936, 2007.

[16] Lipo Wang, Nina Zhou, Feng Chu. A General Wrapper Approach to Selection of Class-Dependent Features. IEEE TRANSACTIONS ON NEURAL NETWORKS, 19:7, 1267–1278, 2008.

[17] The r-project for statistical computing. http://www.r-project.org/.

[18] Fisher, R.A. The Use of Multiple Measurements in Taxonomic Problems. Annals of Eugenics, 7: 179-188, 1936.

[19] Vladimir Vapnik. Statistical Learning Theory. Wiley Interscience, 1998.

[20] Nello Cristianini y John Shawe-Taylor, An introduction to Support Vector Machines, Cambridge University Press, 2000.

[21] Alexander Statnikov, Constantin F. Aliferis, Ioannis Tsamardinos, Douglas Hardin and Shawn Levy. A comprehensive evaluation of multicategory classification methods for microarray gene expression cancer diagnosis. Bioinformatics, 21:5, 631–643, 2005.

Apéndice A: Código fuente

El código del RFE combinado fue escrito en R, un lenguaje para computación estadística y gráficas [17]. Se construyó sobre el código del paquete RFE, haciendo previamente una reingeniería de éste para obtener mayor granularidad en el código, así como también facilitar la expansión del mismo.

El código se reescribió de forma que adopte una arquitectura estratificada, en la cual se pueden distinguir las siguientes capas:

Capa 5: Gráficas
Capa 4: RFE Combinado
Capa 3: Partición de datos
Capa 2: Selección de variables
Capa 1: Ranking de variables
Capa 0: Modelos de selección

Veamos una breve descripción de cada capa:

- Capa 0: Realiza el entrenamiento de los clasificadores. En este trabajo se utilizaron los paquetes de R para Random Forest, PDA y SVM.
- Capa 1: Previo al entrenamiento realiza un tunning de los parámetros del clasificador a utilizar. Entrena el clasificador mediante una llamada a funciones de la capa 0 y luego arma un ranking de variables y calcula el error en test para el clasificador entrenado.
- Capa 2: Para cada conjunto de entrenamiento y test, selecciona diferentes subconjuntos de variables de acuerdo al criterio utilizado. Con esta selección como parámetros se llama a las funciones de capa 1.
- Capa 3: Realiza una partición de los datos en entrenamiento y test por cada corrida. Luego llama a alguna función de la capa 2.

- Capa 4: Modifica las clases de los datos y luego realiza tantas llamadas a la capa 3 como clases haya. Luego de obtener los resultados los combina para obtener el nuevo clasificador. También arma el ranking de variables combinado y realiza el cálculo de la matriz de estabilidad por cada variable.

- Capa 5: Toma los resultados devueltos por la capa 4 ó la capa 3 y los grafica. Para los resultados obtenidos en la capa 4, aquí se realiza el cálculo de la cantidad de variables utilizadas en el combinado. También grafica la estabilidad de los diferentes métodos.

Veremos más en detalle la capa 4, la cual fue desarrollada completamente en este trabajo, y luego parte de la capa 5, para la cual se agregaron algunas funciones. Se describirá para cada función los parámetros que toma y el resultado que arroja.

Capa 4: RFE Combinado

```
one.vs.all <- function(x, y, debug.flags = ‘‘’’, depth = 1, ...)
```

Esta función toma como parámetros un conjunto de datos x y sus respectivas clases y. Los parámetros debug.flags y depth son usados con propósitos de debug. Los "..." corresponden a los parámetros que serán utilizados por las llamadas que realize esta función. Los parámetros pueden recibir valores por defecto, como es el caso de debug.flags y depth. Si otros valores son pasados a esta función, esos son los utilizados y los valores por defecto son descartados.

La función realiza una llamada a feat.sel.ranking (capa 3) con los datos x e y tal cual son tomados. El resultado de esta llamada corresponde al método directo. A continuación se realiza, para alguna clase i, el renombramiento del vector y, donde aquellos valores que no sean iguales a i serán reemplazados por el valor 2, mientras que los restantes serán reemplazados por 1. Se llama la función de la capa 3 feat.sel.eval.same.sets, pasándole el vector y modificado, junto con el resultado del método directo, debido a

que las particiones entre datos de entrenamiento y test utilizadas por esta función son las determinadas por la función feat.sel.ranking en la primer llamada. La función retorna un arreglo de longitud 2. Por un lado se devuelve una lista con las clases del vector i, y por el otro los $C + 1$ resultados de las ejecuciones del RFE, donde C es la cantidad de clases de y.

```
feat.pred.combined <- function(results, debug.flags = ‘‘’’, depth = 1,
dataset = ‘‘’’, ...)
```

Esta función es la encargada de combinar los diferentes resultados obtenidos en la llamada one.vs.all. El parámetro de entrada *results* corresponde a un arreglo que debe ser de la forma del devuelto por la función one.vs.all.

```
get.stability.matrix <- function(result, debug.flags = ‘‘’’,
depth = 1, ...)
```

A partir de un resultado obtenido por la ejecución de una función de la capa 3, la función calcula la matriz de estabilidad, cuyas filas corresponden a cada corrida del RFE y sus columnas a cada variable utilizada por el RFE. Los valores asignados pertenecen al rango [0, 1).

```
ranking.stab <- function(stab.matrix, debug.flags = "",
depth = 1, ...)
```

Esta función toma como parámetro una matriz de estabilidad y calcula, para cada variable, su media entre todas las corridas y su *inter-quartile range*. La función devuelve un lista con los valores calculados, ordenados por la media.

```
feat.miq <- function(stab.list, feat.per = NULL, debug.flags = "",
depth = 1)
```

El parámetro stab.list corresponde al devuelto por la función ranking.stab. Esta función calcula la media de los *inter-quartile ranges* para diferentes cantidades de variables, para cada ejecución *one-versus-all* y para el método directo. Para indicar las cantidades que se quieren calcular, se debe pasar como argumento un arreglo *feat.per* con estas cantidades. Por default, se calculan las medias para el 1 %, el 5 %, el 10 % y el 25 % de las variables. Las medias se calculan sobre las primeras n variables indicadas ordenadas por la media obtenida a partir de las posiciones en los rankings (la media que devulve ranking.stab).

```
combined.ranking <- function(result.list, debug.flags = "", depth = 1)
```

Esta función calcula el ranking combinado. Toma como parámetro a una lista de resultados, *result.list*, en la cual *result.list[[i]]* debe contener el resultado de la ejecución *i vs. all*, con $i = 1, ..., C$. La función devuelve una matriz con tantas filas como corridas tuvo el RFE y con tantas columnas como variables se hayan utilizado. Esta matriz corresponde al ranking de variables del combinado.

```
combined.miq <- function(csm, feat.per = NULL, debug.flags = "",
depth = 1)
```

El parámetro csm corresponde a una lista de medias y MIQs obtenido a partir del ranking combinado. Esta función calcula las medias los *inter-quartile ranges* para diferentes porcentajes de variables, los cuales son especificados en el parámetro feat.per, y por default es el 1 %, el 5 %, el 10 % y el 25 % de las variables.

Capa 5: Gráficas

```
plot.combined <- function(result.list, plot.all = FALSE,
postscript = FALSE, pdf = FALSE, plot.title = "", file.name = "",
internals = FALSE, feats = NULL, drift.l = 0.02, fact = 1,
horizontal.l = 0.1, ylim = c(0, 0.5), ...)
```

Esta función grafica los resultados pasados en result.list. Esta deberá ser una lista cuyo primer elemento sea el resultado combinado, es decir, aquel obtenido con la función feat.pred.combined. Su segundo elemento debe corresponder al resultado del método directo, con el cual se comparará. El resto de la lista deben ser los resultados de las ejecuciones *i vs. all*. A partir de esto la función calcula la cantidad de variables utilizadas en cada corrida, para cada cantidad de variables utilizadas. Si el parámetro plot.all es TRUE, se grafican todos los resultados, incluyendo los resultados de las ejecuciones *i vs. all*. Si es FALSE, grafica sólo el combinado real y el directo.

```
plot.stab <- function(stab, postscript = FALSE, pdf = FALSE,
plot.title = "", file.name = "", filter = 1, ...)
```

Toma como parámetro a stab, una lista de medias y IQRs (un valor devuelto por la función ranking.stab) y las grafica.

Apéndice B: Gráficas de porcentajes de errores de clasificación y estabilidad

Figura 9: Gráfica del error en función del número aparente de variables (panel izquierdo) y del número real de variables (panel derecho) para el dataset Brain Tumor 1. Cada fila corresponde a rankings de variables construidos con un clasificador distinto.

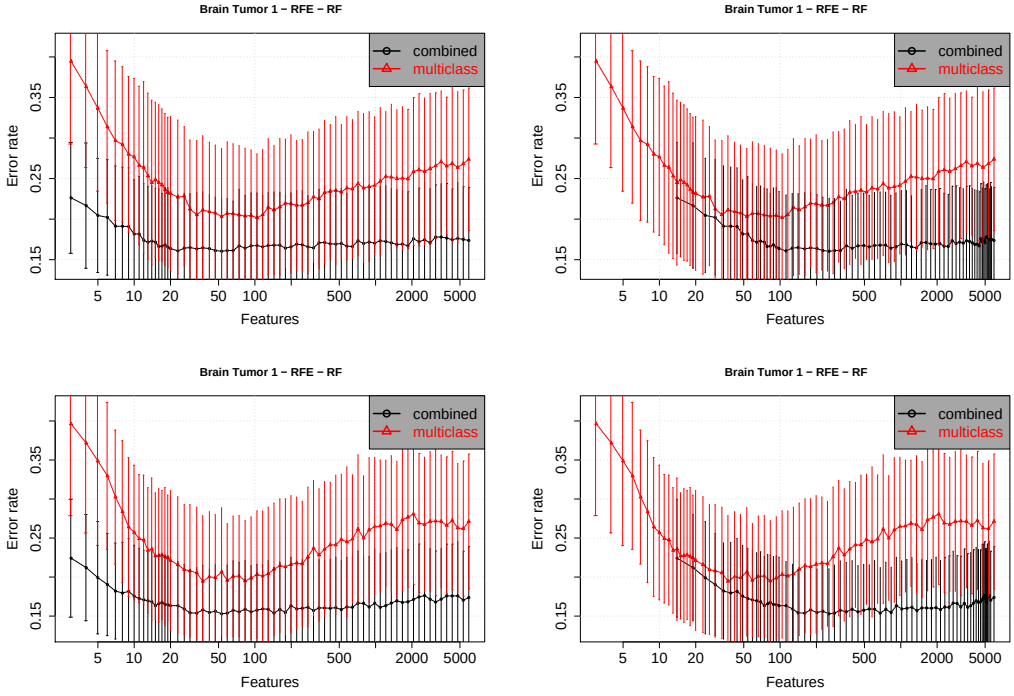


Figura 10: Gráfica del error en función del número aparente de variables (panel izquierdo) y del número real de variables (panel derecho) para el dataset Brain Tumor 2. Cada fila corresponde a rankings de variables construidos con un clasificador distinto.

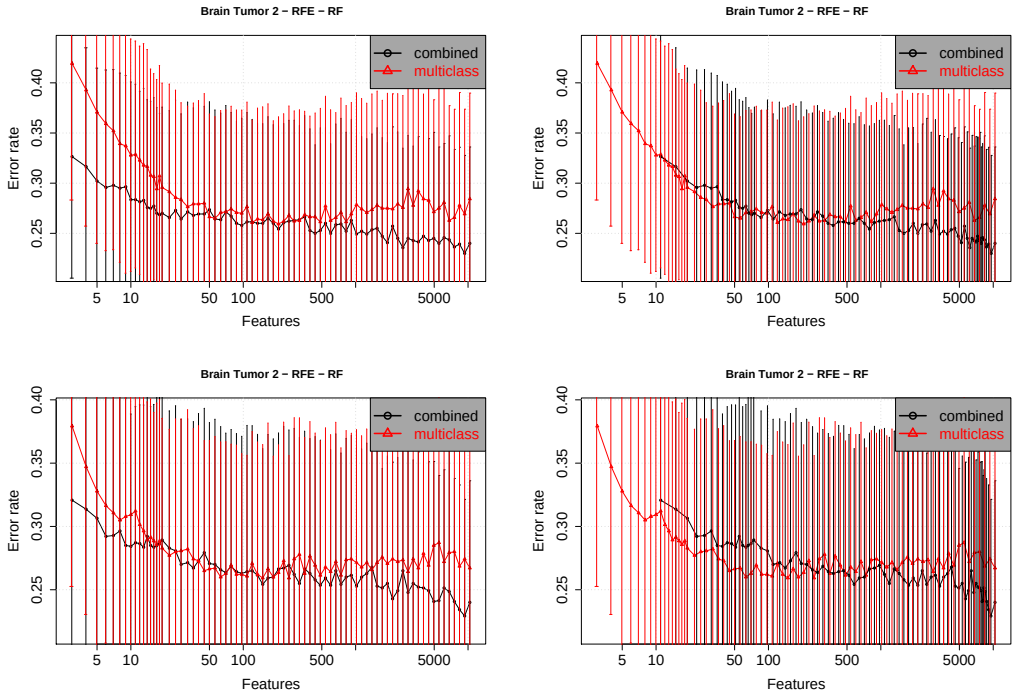


Figura 11: Gráfica del error en función del número aparente de variables (panel izquierdo) y del número real de variables (panel derecho) para el dataset Fragola. Cada fila corresponde a rankings de variables construidos con un clasificador distinto.

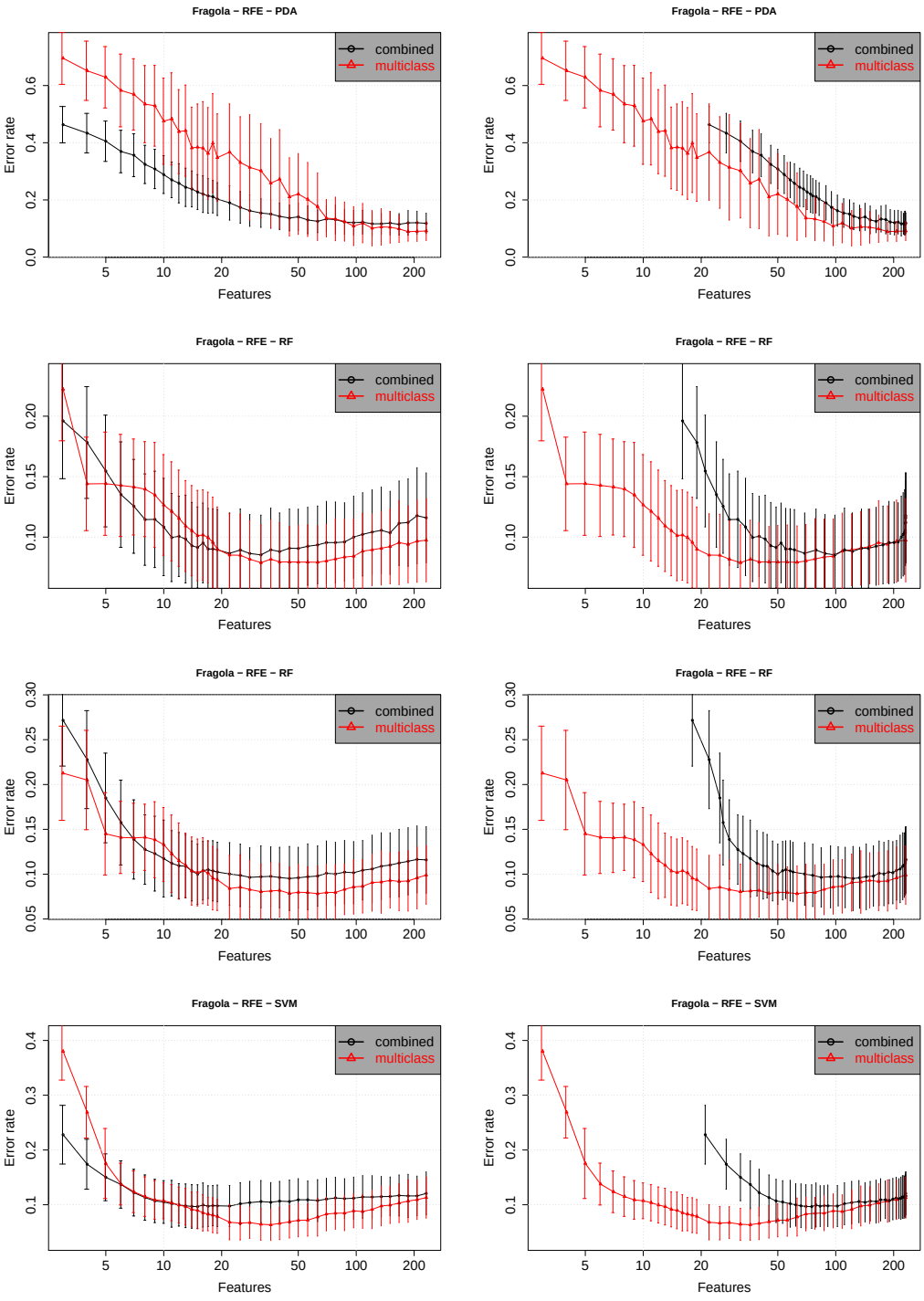


Figura 12: Gráfica del error en función del número aparente de variables (panel izquierdo) y del número real de variables (panel derecho) para el dataset Grana. Cada fila corresponde a rankings de variables construidos con un clasificador distinto.

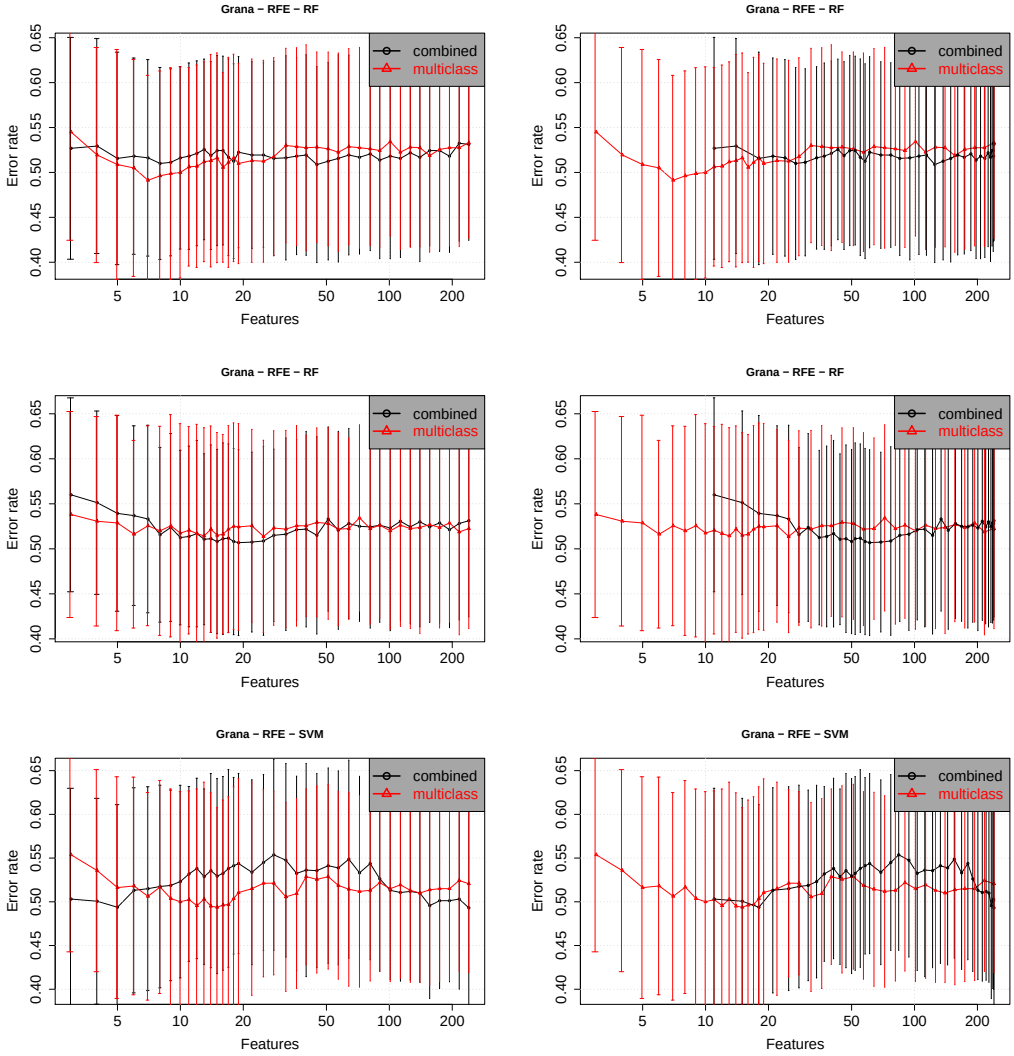


Figura 13: Gráfica del error en función del número aparente de variables (panel izquierdo) y del número real de variables (panel derecho) para el dataset Lampone. Cada fila corresponde a rankings de variables construidos con un clasificador distinto.

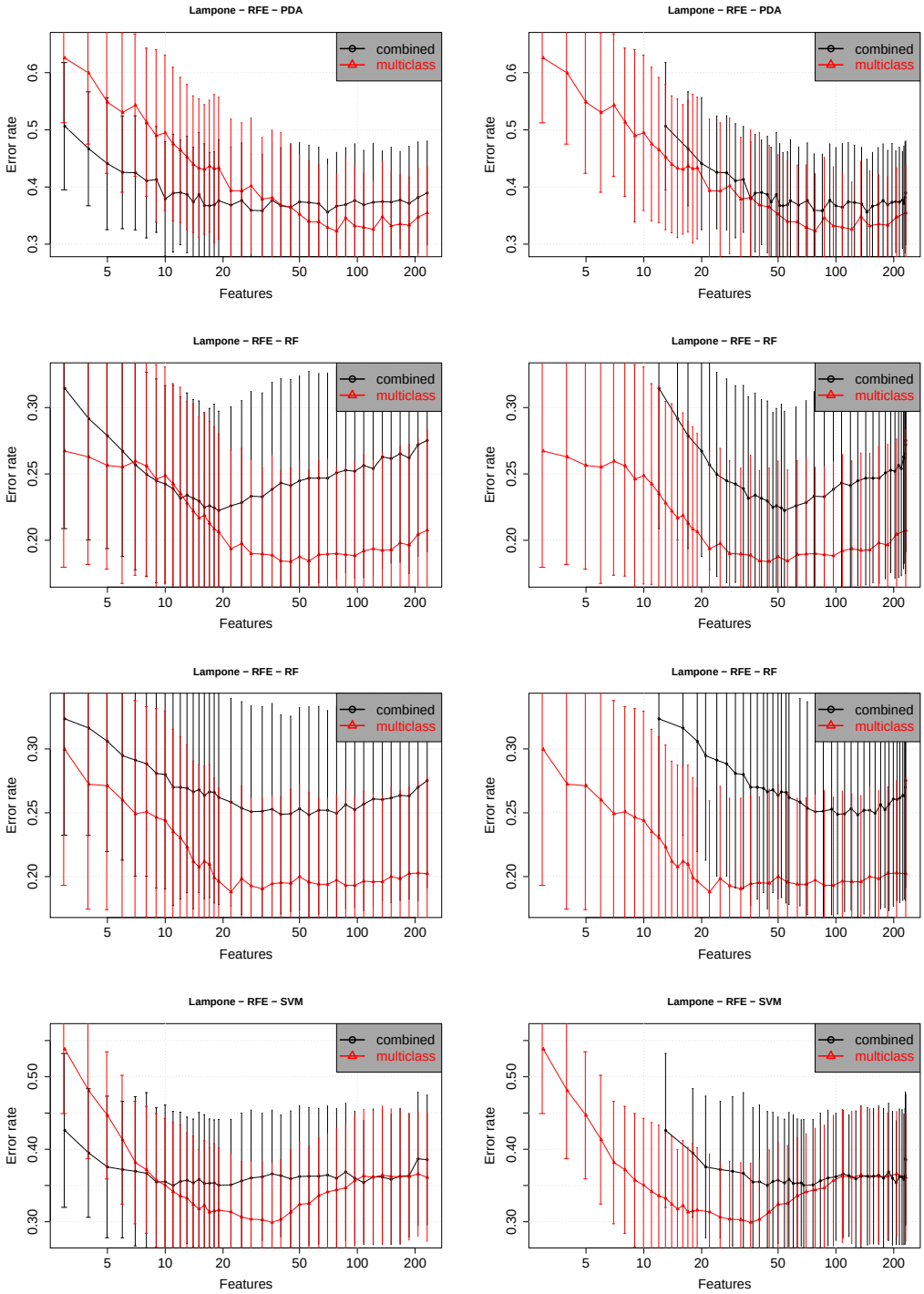


Figura 14: Gráfica del error en función del número aparente de variables (panel izquierdo) y del número real de variables (panel derecho) para el dataset Mirop. Cada fila corresponde a rankings de variables construidos con un clasificador distinto.

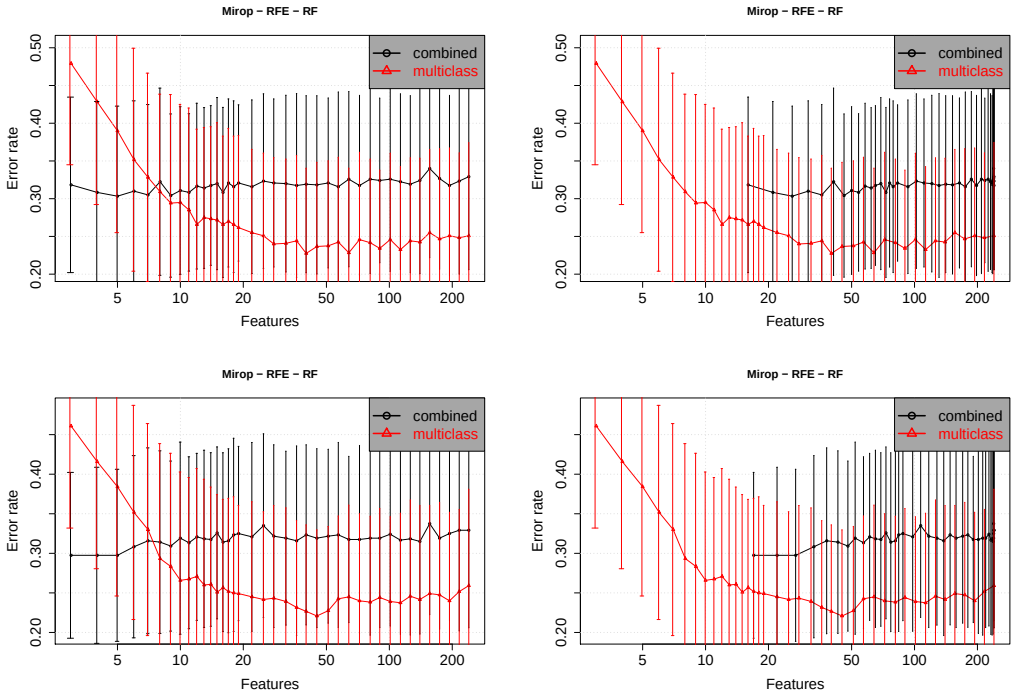


Figura 15: Gráfica de la estabilidad del método directo (panel izquierdo) y del método combinado (panel derecho) para el dataset Fragola. Cada fila corresponde, de arriba hacia abajo, a rankings de variables construidos con PDA, RF, RF GINI y SVMs.

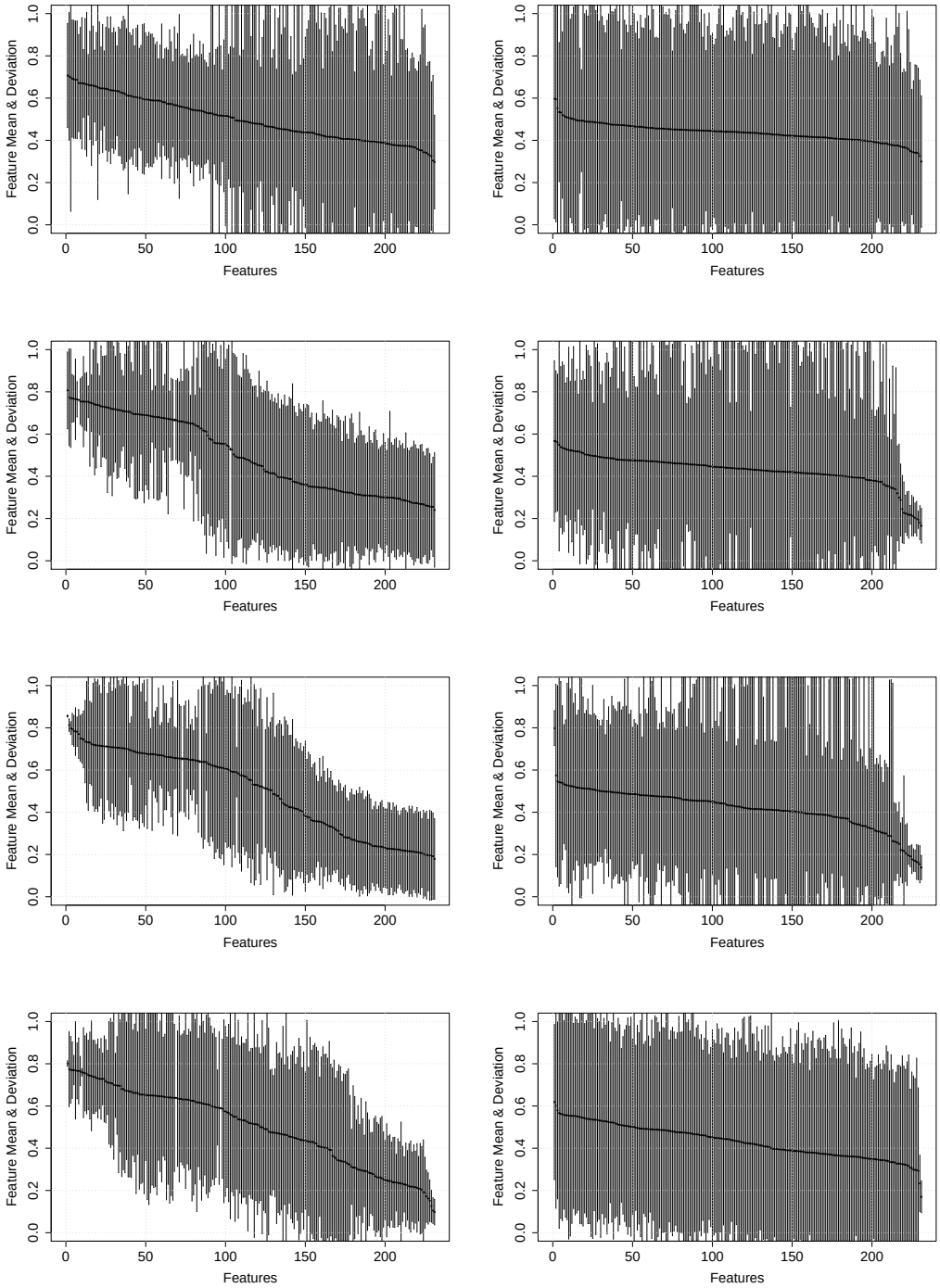


Figura 16: Gráfica de la estabilidad del método directo (panel izquierdo) y del método combinado (panel derecho) para el dataset Grana. Cada fila corresponde, de arriba hacia abajo, a rankings de variables construidos con RF, RF GINI y SVMs.

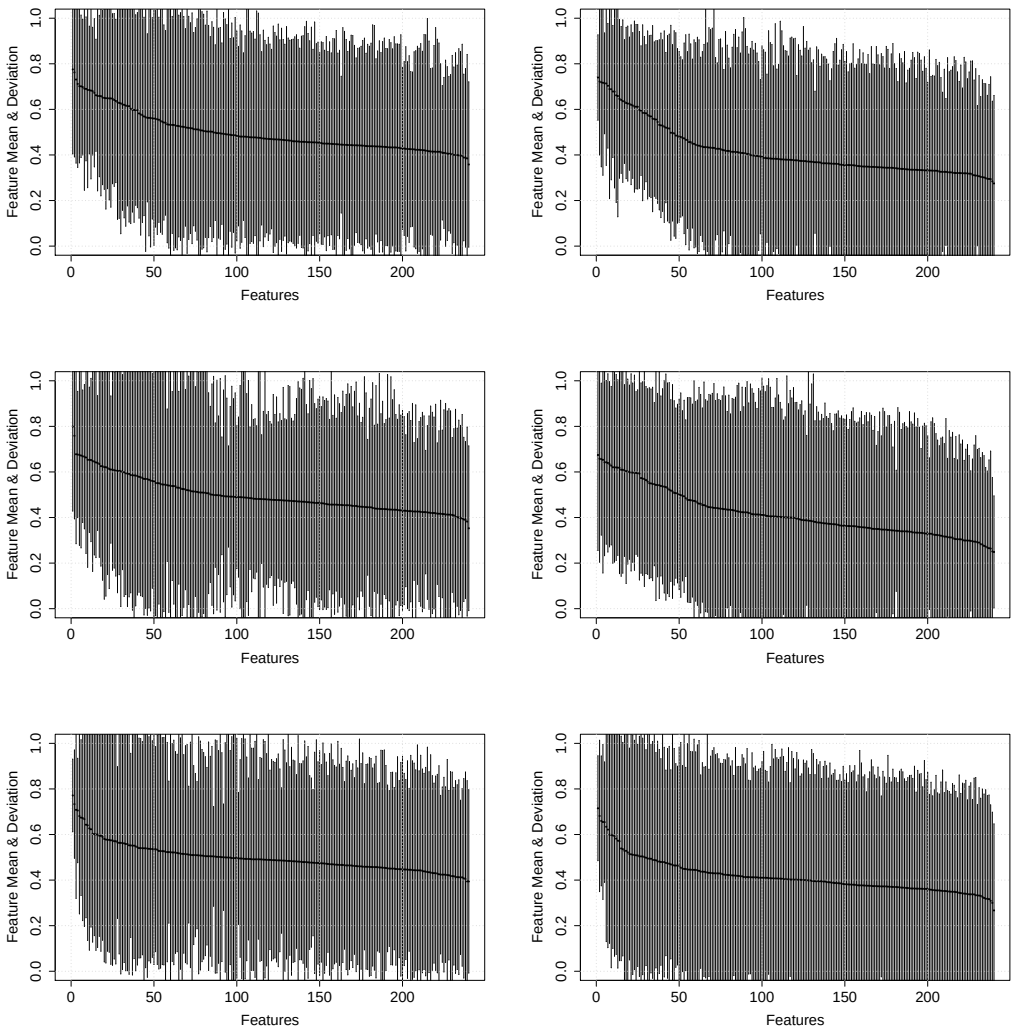


Figura 17: Gráfica de la estabilidad del método directo (panel izquierdo) y del método combinado (panel derecho) para el dataset Lampone. Cada fila corresponde, de arriba hacia abajo, a rankings de variables construidos con PDA, RF, RF GINI y SVMs.

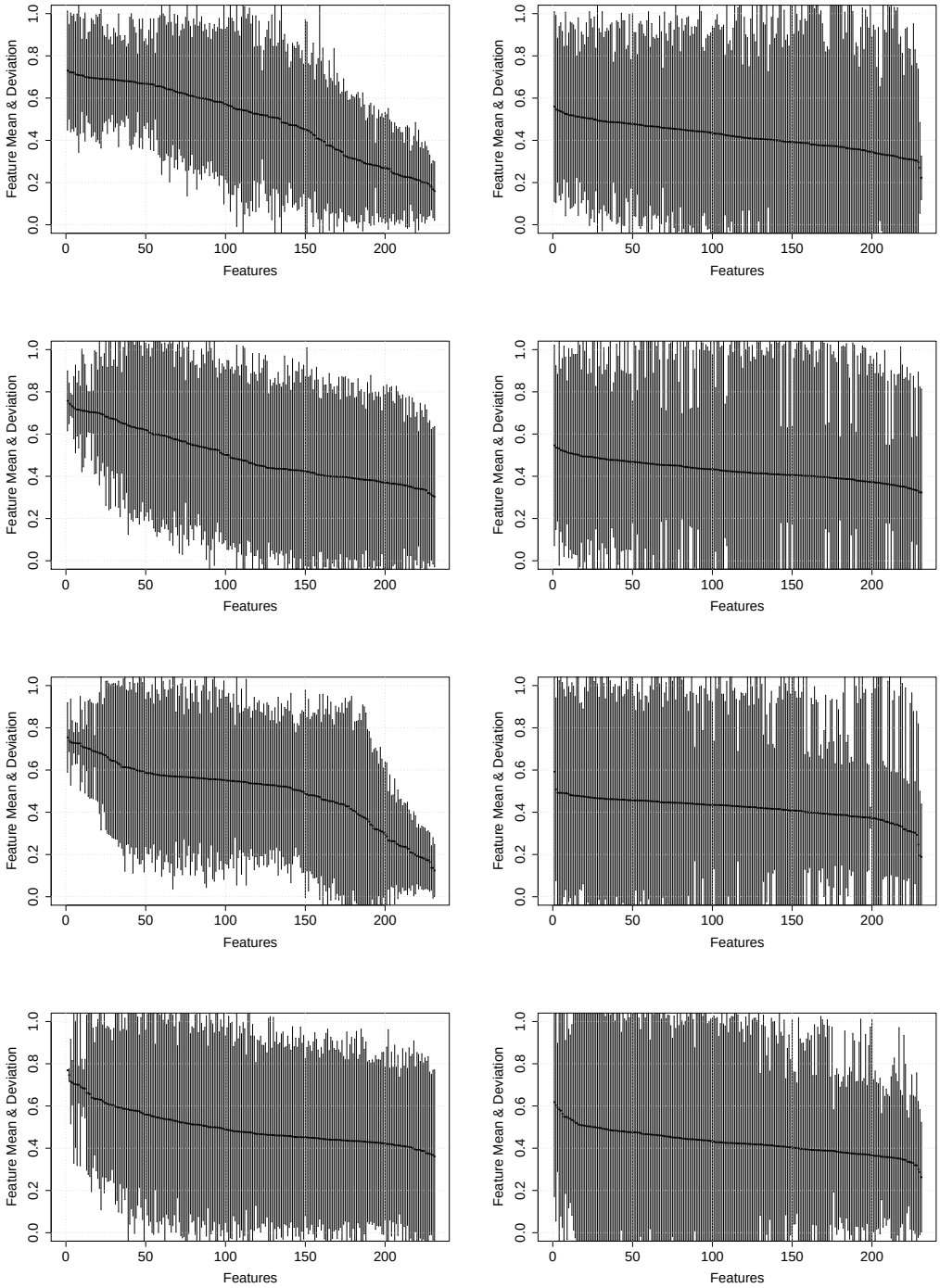


Figura 18: Gráfica de la estabilidad del método directo (panel izquierdo) y del método combinado (panel derecho) para el dataset Mirop. Cada fila corresponde, de arriba hacia abajo, a rankings de variables construidos con RF y RF GINI.