Métodos de clasificación para datos funcionales basados en el Clasificador DDG

Johan Steward Rios Naranjo[†,*], Francisco Iván Zuluaga[◇]

[†] Estudiante Maestría en Ciencia de Datos y Analítica, Universidad EAFIT

[‡] Profesor Asesor del Departamento de Ciencias, Universidad EAFIT

1. Introducción

El clasificador DD^G es una extensión del clasificador DD, un claisifcador basado en profundidad funcional. Algunas comparaciones contra otros clasificadores han sido realizadas teniendo en cuenta algunas profundidades estadísticas. Este documento tiene como objetivo extender el clasificador DD^G proporcionando una estudio de simulación y aplicaciones a algunos conjuntos de datos existentes en el paquete roahd del software R a algunos métodos no explorados en Juan A. Cuesta-Albertos, 2016 con el fin de facilitar a las personas el uso de una herramienta efectiva para la clasificación de datos funcionales ya sean univariados o multivariados.

Un dato funcional se deriva de las observaciones discretas de una variable aleatoria, tomadas a lo largo de un intervalo continuo. Se convierte en un dato funcional cuando se describe una curva a lo largo de dicho intervalo, de forma que se es "estimado" el comportamiento real y continuo a lo largo de dicha variable independiente o intervalo continuo (por ejemplo: tiempo). Para decir que X es una variable aleatoria que es funcional, debe tomar valores en el espacio normado o seminormado completo, lo que implica que puede ser evaluado en cualquier punto del dominio del espacio ε J. O. RAMSAY, 2005, como el espacio L2 en el intervalo $I : [a, b]$, donde $I \subset R$. Esto mismo para el conjunto de datos funcionales de la variable aleatoria X, descrito por X_1,X_2 ... , X_n como las n mediciones de la variable X en el mismo intervalo continuo I.

Aunque los datos funcionales en sí mismos no son nuevos, es necesaria una nueva concepción de ellos como resultado de la creciente sofisticación de los datos. La tecnología ha evolucionado en las últimas dos décadas para permitir observaciones densamente muestreadas a lo largo del tiempo, espacio y otro tipo de dato continuo; estos datos generalmente reflejan la influencia de ciertas funciones suavizadas que asumimos subyacen y generan las observaciones **intro**.

Basándonos en lo anteriormente dicho, suponemos una ventaja sobre estos métodos por encima de los métodos estadísticos multivariados clásicos, que igual pueden ser aplicados a dichos datos, pero no pueden sacar provecho de la información adicional inducida por la suavidad de las funciones subyacentes.

El análisis de datos funcionales amplía las capacidades de las técnicas estadísticas tradicionales de varias maneras. Estudiar la variabilidad de un conjunto de datos cuando las observaciones son curvas en lugar de puntos requiere que las herramientas estándar se adapten para este marco funcional y que las nuevas herramientas sean creadas para aprovechar las características únicas de los datos funcionales. Por ejemplo, la disponibilidad de derivados permite el uso de ecuaciones diferenciales como modelos, y por lo tanto introduce descripciones de dinámica de procesos, así como sus características estáticas.**intro**

El análisis de datos funcionales es uno de esos enfoques hacia el modelado datos de series de tiempo que han comenzado a recibir atención en el literatura, particularmente en términos de su salud pública y aplicaciones biomédicas.

El aumento reciente del interés en la aplicación del modelado estadístico para medicina, biomedicina, salud pública, biología, la biomecánica y la ciencia ambiental ha sido en gran parte impulsado por la necesidad de buenos datos para informar al gobierno Políticas y procesos de planificación para la prestación de servicios de salud y prevención de enfermedades. Es importante destacar que tales modelos solo serán útiles a largo plazo si son precisos, basados en datos de buena calidad y generados mediante la aplicación de métodos estadísticos robustos apropiados.

Comúnmente, los datos de series temporales se tratan como datos multivariados datos porque se dan como una serie temporal discreta finita. Este enfoque multivariante habitual ignora por completo la información importante sobre el comportamiento funcional suave del proceso de generación que sustenta los datos Green PJ, 1994.

Como se explica en Shahid Ullah1, 2013, la idea básica tras el análisis de datos funcionales es expresar observaciones discretas que surjan de series de tiempo en forma de una función (para crear datos funcionales) que representen la función medida completa como una sola observación, y así crear modelos de predicción de una colección de datos funcionales aplicando conceptos estadísticos a partir del análisis de datos multivariados, al hacerlo, se tiene la ventaja de generar modelos que pueden describirse mediante dinámicas continuas y suaves, que luego permiten estimaciones precisas de los parámetros para su uso en la fase de análisis, reducción efectiva del ruido en los datos mediante suavizado de curvas y aplicabilidad a datos con tiempos de muestreo irregular.

Ramsay y Dalzell Ramsay JO, 1991 presentan varias razones prácticas para considerar los datos funcionales:

- Los procedimientos de suavizado e interpolación pueden producir representaciones funcionales de un conjunto finito de observaciones.
- Es más natural pensar en modelar problemas en forma funcional
- los objetivos de un análisis pueden ser funcionales en naturaleza, como sería el caso si se usaran datos finitos para estimar una función completa, sus derivadas o la valores de otros funcionales.

En Shahid Ullah1, 2013 encontramos que las caractéristicas más usadas para investigación correspondiente a datos funcionales son técnicas de suavizado, reducción de datos, pronósticos, modelos lineales funcionales, clustering y clasificación.

Es precisamente esta última característica la que más nos llama la atención y es que el propósito de este artículo recae en la extensión del clasificador DD^G y la comparación entre sus métodos de clasificación, más allá de facilitar el uso de la herramienta y explicar detalladamente el proceso del método. Sin embargo, algunos de estos métodos en el contexto funcional necesitan de una característica que también se encuentra en la estadística tradicional y es el concepto de profundidad estadística.

Tukey J, 1975 propuso la noción de profundidad de datos como una herramienta gráfica para visualizar datos bivariados tanto en casos univariados como multivariados Donoho D. L., 1992.

La profundidad de un punto en relación con un conjunto de datos determinado mide qué tan "profundo" se encuentra ese punto en la nube de datos. El concepto de profundidad de datos proporciona ordenación de puntos de centro a exterior en cualquier dimensión y conduce a un nuevo análisis estadístico multivariante no paramétrico en el que no se necesitan supuestos de distribución Neto, n.d.

En el caso de datos funcionales, se habla de profundidad funcional, que es la estimación de la centralidad (profundidad) que tiene una curva respecto a las demás. Lo que se busca es ordenar las curvas de forma tal que la curva más central corresponde a la mediana. En otras palabras dado un conjunto de observaciones $x_1, ..., x_n$ en R^p, una función de profundidad $D : R^p \to R^1$ proporciona un ordenamiento del centro hacia afuera de los datos. Las observaciones con gran profundidad están cerca del "centro", las observaciones de baja profundidad son valores atípicos. Esta medición depende de la métrica que se implemente, donde 3 de las métricas comúnmente implementadas en datos funcionales son la h-modal, Fraiman y Muniz y la de proyecciones aleatorias.

Como se dijo al principio, el objetivo nuestro trabajo es exponer, clarificar y expandir el método de clasificación DD^G el cual está basado en el gráfico derivado del concepto del QQ-plot, el DD-plot, un gráfico que funciona a través de profundidad estadística.

Los problemas de clasificación surgen en muchos campos de aplicación como la economía, la biología y la medicina. Naturalmente, el cerebro humano tiende a clasificar los objetos que nos rodean según su propiedades. Existen numerosos esquemas de clasificación que separan objetos en muy diferentes formas, dependiendo del campo de aplicación y la importancia que se le asigne a cada propiedad. La pertenencia a una clase de objetos nuevos observados puede ser determinada por sus propiedades usando conocimiento previamente obtenido. En las últimas décadas el aumento de la potencia informática permitió automatizar este proceso. Un nuevo campo surgió, el aprendizaje automático, que entre otras cosas, desarrolla algoritmos para el aprendizaje supervisado. La tarea del aprendizaje supervisado es definir una regla basada en datos mediante la cual los nuevos objetos son asignados a una de las clases. Para ello, se utiliza un conjunto de datos de entrenamiento que contiene objetos con pertenencia a una clase conocida.

Formalmente, consideramos los objetos como puntos en un espacio multivariado que está formado por sus propiedades. Es importante determinar las propiedades que son más relevantes para el problema de clasificación particular. Al analizar el conjunto de

entrenamiento, un clasificador genera un función de separación que determina la relación de clase de los objetos recién observados. Los procedimientos de clasificación tienen que determinar qué tan cerca está situado un objeto con respecto a una clase y qué tan típico es para esa clase. Esto se hace estudiando la ubicación, la escala y la forma de la distribución subyacente de las clases.

A continuación daremos una revisión a los conceptos necesarios para entender que es el clasificador DD^G y como se utiliza.

1.1. **QQ Plot.** Los gráficos estándar de cuantiles-cuantiles ($Q-Q$) (Wilk y Gnanadesikan, 1968) son un herramienta para evaluar visualmente un supuesto distributivo específico. Se construye un gráfico $Q-Q$ de una muestra, $x_1, ..., x_n$, trazando los cuantiles teóricos, $F^{-1}(F_n(x_i))$, contra el cuantiles de muestra, $x_(i)$. Si la distribución empírica, F_n, es consistente con la distribución teórica, F, los puntos en la gráfica $Q-Q$ caen en la línea de identidad (una línea de 45 grados). Para cualquier muestra probada contra una distribución dentro de una familia de distribuciones similares de escala de ubicación, como una normal, logarítmica normal o distribución exponencial, los cuantiles de la muestra todavía caen en una línea cuando se trazan contra los cuantiles teóricos de cualquiera de las distribuciones miembros de la familia.

1.2. **DD-Plot.** El artículo Jun Li, 2012 fue el primero en utilizar profundidades para clasificacióna través del clasificador de profundidad máxima (MD): dadas dos medidas de probabilidad (o clases, o grupos) P y Q, y una profundidad, D, nosotros clasificamos el punto x como producido por P si $D_P(x) > D_Q(x)$.

En Ghosh AK, 2005 se desarrolla completamente este procedimiento.

El clasificador MD parece bastante razonable, pero presenta algunos inconvenientes que pueden entenderse mejor con la ayuda del DD-Plot.

El DD-Plot fue introducido por primera vez por Liu RY, 1999 para comparaciones gráficas de dos distribuciones o muestras multivariadas basadas en la profundidad de los datos. Siempre es un gráfico bidimensional independientemente de las dimensiones de las muestras. Se mostró en Liu RY, 1999 que tanto diferencias de distribución, como diferencias de ubicación, escala, asimetría o curtosis, están asociados con diferentes patrones gráficos en el DD-Plot. Es así como los DD-Plot proporcionan herramientas de diagnóstico sencillas para realizar comparaciones visuales de dos muestras de cualquier dimensión.

Dadas dos distribuciones de probabilidad, P y Q en R^p, el DD-Plot es un gráfico bidimensional (independientemente de p) en el que, por cada $x \in R^p$, el par $(D_P(x), D_Q(x)) \in R^2$ está representado.

Si las dos distribuciones dadas son idénticas, entonces estos gráficos son segmentos de la linea diagonal de $(0,0)$ a $(1,1)$ en R^2. Gráficos que se desvían de esta linea recta, indican diferencia entre las dos distribuciones. Como se observará mas adelante, generalmente patrones de desviación diferentes corresponderán a diferentes tipos de variaciones entre las distribuciones,por ejemplo cambios en la localización con incrementos en la escala. Veamos

a continuación una definición más formal del DD-plot. Sean F y G dos distribuciones en R^d y sea $D()$ una medida de profundidad invariante afín, se define $DD(F,G)$ como:

$$DD(F,G) = \{(D(x,F), D(x,G)), \forall x \in R^d\} \tag{1.1}$$

donde F y G son fijas. Como DD es un subconjunto de R^2, este gráfico el cual llamaremos DD-plot puede ser fácilmente visualizado, es de resaltar que si la distribución base es invariante afín, el DD-plot también lo sera. Cuando la distribuciones son desconocidas se puede construir una versión empírica de un DD-plot, si la distribución base F es desconocida, con un conjunto de datos $\{X_1, ..., X_n\}$ se puede determinar si F es alguna distribución especificada, dígase G, examinando el siguiente DD-plot.

$$DD(F_n,G) = \{(D(x,F_n), D(x,G)), \forall x \in X_i, i = 1, ..., n\} \tag{1.2}$$

Si F y G son las distribuciones poblacionales para las muestras $\{X_1, \ldots, X_n\}(\equiv X)$ y $\{Y_1, \ldots, Y_m\}(\equiv Y)$, entonces el DD-plot mostrado a continuación puede ser usado para determinar cuando, o no, las dos distribuciones son idénticas.

$$DD(F_n,G_m) = \{(D(x,F_n), D(x,G_m)), x \in X \cup Y\} \tag{1.3}$$

para $d \geq 2$,y si F y G son absolutamente continuas, entonces el DD-plot es una región con una área no nula. Esta área puede servir como una medida invariante afín de la discrepancia entre F y G. Si las dos distribuciones son idénticas, el DD-plot debería estar concentrado a lo largo de la línea diagonal.

Ejemplos de diagramas DD aparecen en las figuras 1, 2 y 3.

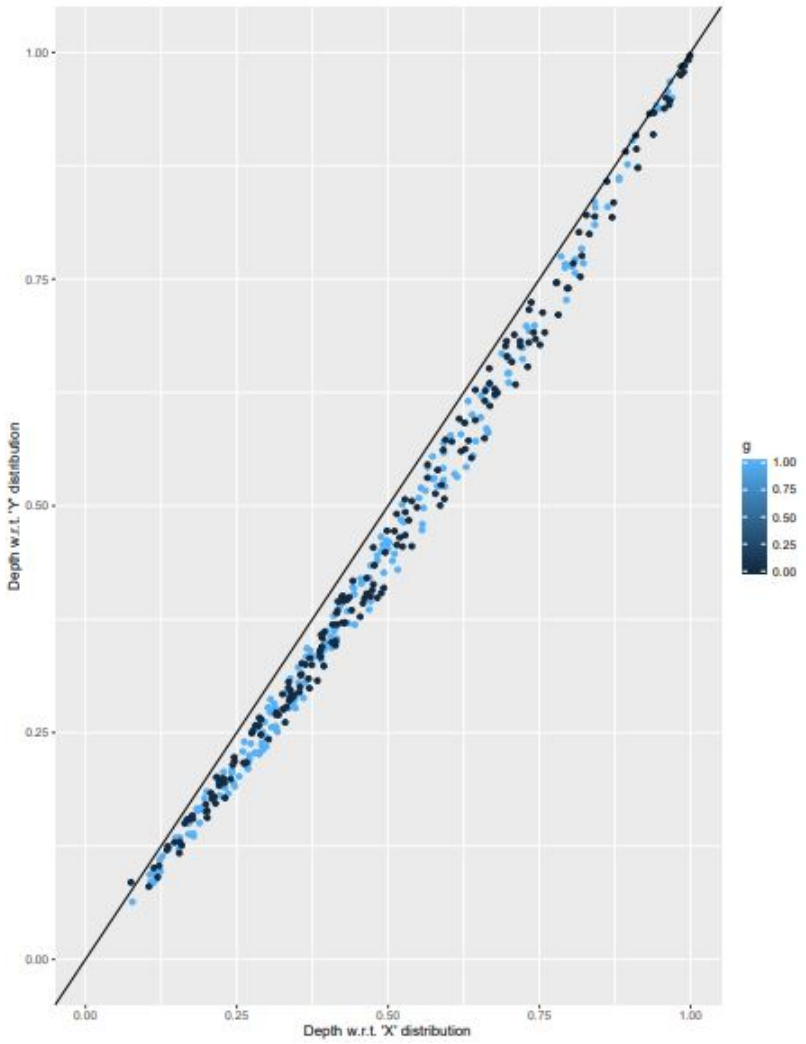

Figure 1. Ejemplo DD-plot 1

Como podemos observar los datos se agrupan sobre la linea de 45 grados. Diferentes patrones de desviaciones de la línea diagonal son indicaciones de diferencias en características específicas de F y G. Considere el caso donde la primera es generada a partir de una $N(0, I)$ y la segunda a partir de una $N(\mu, I)$ con $\mu = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$

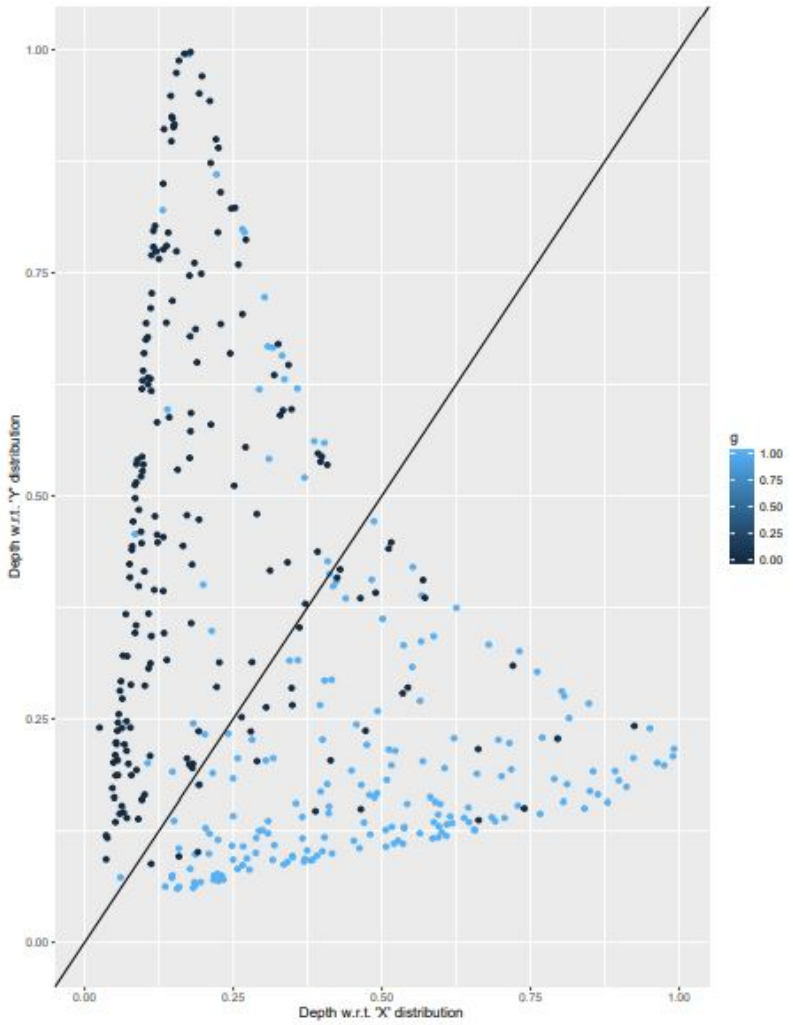

Figure 2. Ejemplo DD-plot 2

En este caso el DD-plot presenta una desviación notable de la línea diagonal. Desde un punto de vista teórico, si las dos muestras tienen el mismo centro (Punto más profundo), entonces el centro alcanza la profundidad mas alta dentro de cada muestra individual, en otras palabras un DD-plot con un punto máximo común para ambas coordenadas indica un centro común para las dos muestras base. Este no es el caso de la gráfica anterior y el gráfico en forma de ≪corazón≫ indica una diferencia en localización en las dos muestras.

Considere ahora el caso donde la primera es generada a partir de una $N(0, 9I)$ y la segunda con los mismos parámetros del caso anterior.

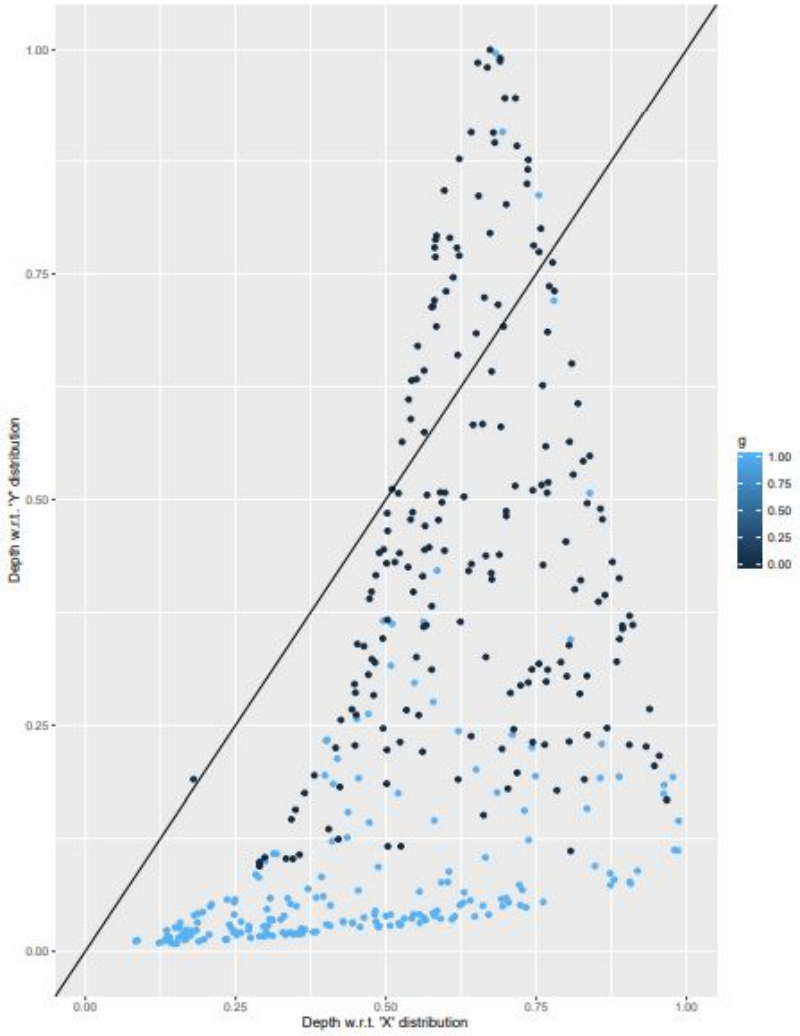

Figure 3. Ejemplo DD-plot 3

F y G tienen el mismo centro, pero con un ≪spread≫ mayor para una que para otra, entonces los puntos en el DD-plot tienden a arquearse sobre la linea diagonal en forma de una media luna.

La idea que se desarrolló en Jun Li, 2012 es que el DD-Plot contiene información que ayuda a obtener un buen clasificador.

1.3. **Clasificador DD.** En Jun Li, 2012, en un contexto de datos multivariados, usan el DD-plot para introducir un algoritmo de clasificación no paramétrico, el cual denominan el clasificador-DD. El algoritmo desarrollado es totalmente orientado por los datos y el resultado de la clasificación puede ser visualizado en un gráfico de dos dimensiones. La idea del clasificador-DD es buscar la curva que mejor separe las dos muestras en su DD-plot, en el sentido de que la separación dé como resultado el error de clasificación más pequeño en

su DD-plot. La curva que mejor separa en el DD-plot conduce a una regla de clasificación en el espacio muestral original de las dos muestras.

Otro aspecto a destacar es que el resultado de clasificación puede ser visualizado con el DD-plot bi-dimensional.

1.3.1. *Regla de clasificación.* Sean $\{X_, \ldots, X_m\}(\equiv X)$ y $\{Y_1, \ldots, Y_m\}(\equiv Y)$ dos muestras aleatorias de F y G respectivamente, que son distribuciones definidas en R^d, si $F = G$,el DD-plot debería estar concentrado a lo largo de la línea de 45 grados, ahora si las dos distribuciones difieren , el DD-plot exhibirá una desviación notable de la línea de 45 grados. Para facilitar el análisis replicamos las gráficas obtenidas anteriormente en 2, el gráfico es construido para dos normales bivariadas, cada una de tamaño 200, un estándar y la otra con una media $(2, 0)^{'}$, ambas construidas usando la medida de profundidad de Mahalanobis. Los colores de los puntos representan la membresía de los puntos muestrales, el color azul claro representa que esta observación corresponde a X , y el azul oscuro que pertenece Y. Como podemos observar las dos diferentes muestras se alejan de la línea de 45 grados mas o menos de manera simétrica. Si se usa una línea para separar las dos muestras en el DD-plot, la línea de 45 grados parece ser la mejor elección. Si se usa la línea de 45 grados como la línea separadora, su correspondiente regla de clasificación asignará x a F si $D(x, F_m) > D(x, G_n)$ y asignará x a G en cualquier otro caso.

En esta situación la linea que mejor separe las dos muestras en el DD-plot da como resultado una muy cercana mejor línea separadora en el espacio muestral original R^2. Como se puede observar en la siguiente gráfica , la línea se puede obtener mapeando la línea de 45 grados del DD-plot en el espacio R^2.

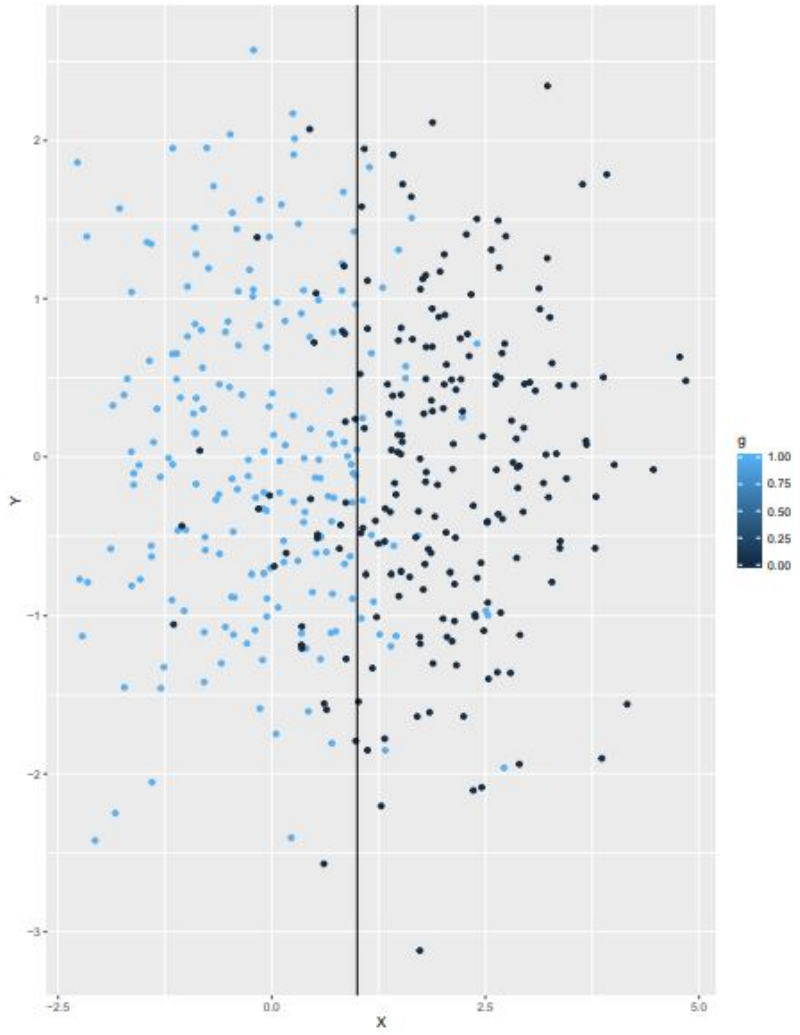

Figure 4. Ejemplo en R^2 del ejemplo DD-plot 2

Ahora si se consideramos en el caso en el que la primera muestra es saca de una normal bivariada con media $(0,0)^{'}$ y una matriz $9I_2$, como en la figura 3, en este caso las dos muestras no se alejan de manera simétrica de la línea de 45 grados, las observaciones de X parecen moverse hacia el eje x y las observaciones de Y hacia la línea vertical $x = 1$.

Por ejemplo si en este caso se usa el clasificador de máxima profundidad (MD), que es equivalente a dibujar la línea de 45 grados y asignar las observaciones por encima de la línea a G y a F ,los que caen por debajo,como podemos observar si se sigue esta regla de clasificación se asignarán la mayor parte de las observaciones de Y a F dando como resultado una tasa de mala clasificación muy alta, lo anterior significa que el clasificador de máxima profundidad no funciona muy bien cuando las distribuciones difieren en variabilidad.

Si observamos la siguiente gráfica se puede observar que por ejemplo un polinomio de grado podría separar mejor las dos muestras.

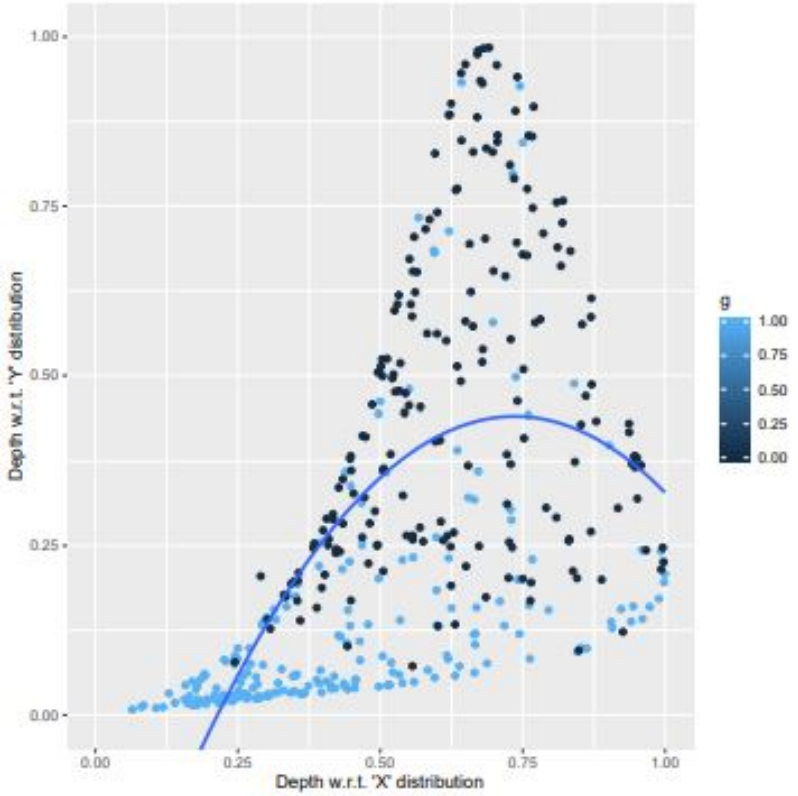

Figure 5. Ejemplo DD-plot 4

Si esta curva es mapeada en el espacio muestral original en R^2 se, puede observar que existe una curva, en este caso un circulo que separa muy bien las dos muestras.

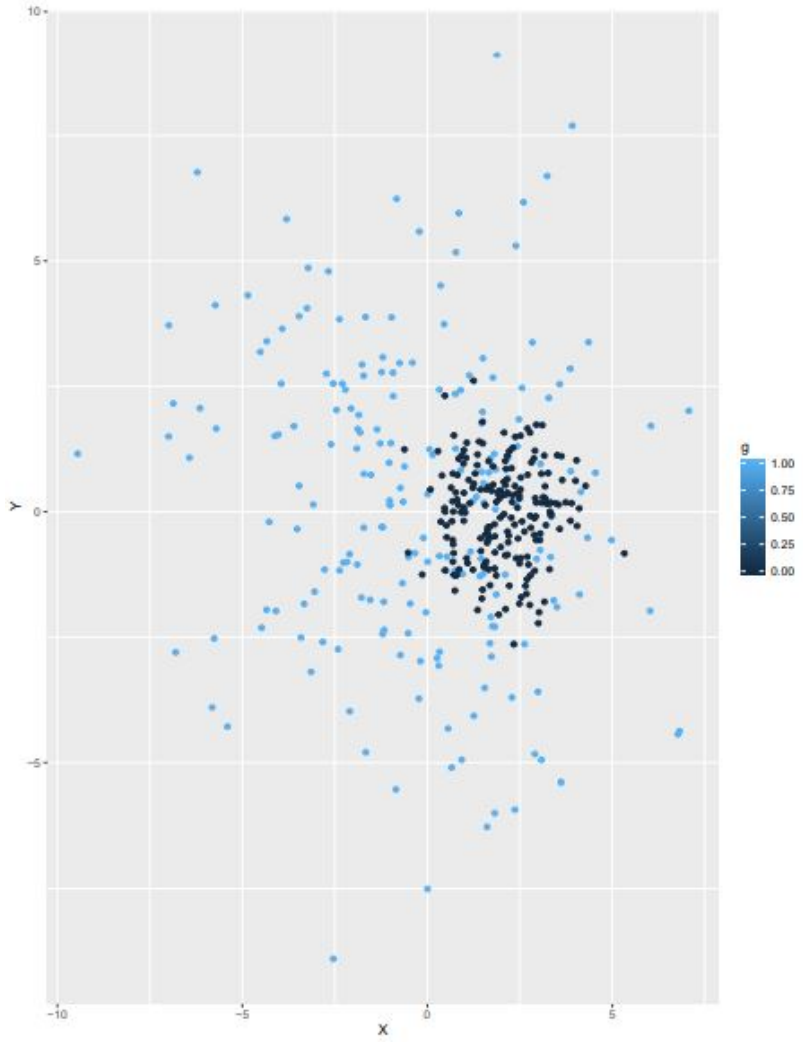

Figure 6. Ejemplo en R^2 del ejemplo DD-plot 4

Lo anterior indica que la curva o linea que mejor separe las dos muestras en el DD-Plot también será la mejor curva o superficie que separe las dos muestras en el espacio muestral original. Esto se formaliza en el siguiente resultado: Sean $f1()$ y $f2()$ las funciones de densidad de F y G respectivamente. La regla óptima de clasificación está dada por la siguiente regla de bayes:

$$\begin{aligned} \pi_2 f_2(x) > \pi_1 f_1(x) \quad & \textbf{asigne } x \textbf{ a } G \\ \pi_2 f_2(x) < \pi_1 f_1(x) \quad & \textbf{asigne } x \textbf{ a } F \end{aligned} \tag{1.4}$$

donde π_1 y π_2 son las probabilidades a priori de F y G, respectivamente, Asuma que $f1()$ y $f2()$ son ambas de la familia elíptica.

$$f_i(x) = c_i|\sum_i|^{-\frac{1}{2}} h_i((x\mu_i)' \sum_i^{-1}(x\mu_i)) \quad i = 1, 2 \tag{1.5}$$

Donde los $h_i()'s$,son funciones estrictamente decrecientes. Si $D_F(x) = l_1(f_1(x))$ y $D_G(x) = l_2(f_2(x))$ para algunas funciones estrictamente crecientes l_1 y l_2,entonces la regla de bayes es equivalente a:

$$\begin{array}{ll} D_G(x) > r(D_F(x)) & \textbf{asigne } x \textbf{ a } G \\ D_G(x) < r(D_F(x)) & \textbf{asigne } x \textbf{ a } F \end{array} \tag{1.6}$$

donde $r()$ es una función creciente real, el resultado anterior implica que para distribuciones unimodales y elípticas la mejor curva o superficie entre dos muestras en R^d es equivalente a la mejor curva que separa las dos muestras en en el DD-plot. Para estimar la función $r_0()$ que mejor separa en el DD-plot se parte en principio de una familia de funciones Γ y se busca $r_0()$ dentro de esta familia y se expande esta familia tanto como sea posible. Dado Γ y cualquier $r \in \Gamma$ se considera el siguiente algoritmo de clasificación:

$$\begin{array}{ll} D_{G_n}(x) > r(D_{F_m}(x)) & \textbf{asigne } x \textbf{ a } G \\ D_{G_n}(x) \leq r(D_{F_m}(x)) & \textbf{asigne } x \textbf{ a } F \end{array} \tag{1.7}$$

La idea es entonces para un Γ determinado encontrar el $r_0 \in \Gamma$ que minimiza la tasa de mala clasificación. Para cualquier $r \in \Gamma$ saque una curva correspondiente a $y = r(x)$ en el DD-plot, asigne las observaciones por encima de la curva a G y las que caen por debajo a F ,y entonces calcule la tasa de mala clasificación empírica.

$$\hat{\Delta}_N(r) = \frac{\pi_1}{m}\sum_{i=1}^{m} I(D_{G_n}(X_i) > r(D_{F_m}(X_i))) + \frac{\pi_2}{n}\sum_{i=1}^{n} I(D_{G_n}(Y_i) \leq r(D_{F_m}(Y_i))) \tag{1.8}$$

donde $N = (m, n)$. Se propone entonces estimar el optimo
r_0 por $\hat{r}_N = argmin_{r\in\Gamma}\Delta_N(r)$, el clasificador propuesto es entonces:

$$\begin{array}{ll} D_{G_n}(x) > \hat{r}(D_{F_m}(x)) & \textbf{asigne } x \textbf{ a } G \\ D_{G_n}(x) \leq \hat{r}(D_{F_m}(x)) & \textbf{asigne } x \textbf{ a } F \end{array} \tag{1.9}$$

1.3.2. *Propiedades.* Sea Γ una familia que mapea $[0, 1]$ en si mismo, Dado $r \in \Gamma$ y $d_1, d_2 \in [0, 1]$,definase

$$C_r(d_1, d_2) = \begin{Bmatrix} 1 & \textbf{si } d_2 > r(d_1) \\ 0 & \textbf{si } d_2 \leq r(d_1) \end{Bmatrix} \tag{1.10}$$

De esta manera el clasificador dado por la ecuación (9) puede ser expresado como:

$$\hat{C}_N = C_{\hat{r}_N}(D_{F_m}(z), D_{G_n}(z)) = \begin{Bmatrix} 1 & \textbf{asigne } z \textbf{ a } G \\ 0 & \textbf{asigne } z \textbf{ a } G \end{Bmatrix} \tag{1.11}$$

De igual manera ocurre con la contraparte poblacional $C_r(D_F(z), D_G(z))$,de acuerdo a lo anterior podemos definir las tasas de mala clasificación como:

(1.12)
$$\Delta_N(\hat{C}_N) = \Delta_N(\hat{r}_N) = \pi_1 P_F\{z : C_{\hat{r}_N}(D_{F_m}(z), D_{G_n}(z)) = 1 + \pi_2 P_G\{z : C_{\hat{r}_N}(D_{F_m}(z), D_{G_n}(z)) = 0\}$$

Donde esta ultima expresión corresponde a la probabilidad de mala clasificación condicional dada las muestras de entrenamiento cuando los valores $D_{G_n}(Z)$ y $\hat{r}_N(D_{F_m}(Z))$ son usados para clasificar la observación futura Z. La expectativa de esta expresión con respecto a la distribución de probabilidad de las muestras de entrenamiento es la probabilidad incondicional de mala clasificación del clasificador-DD. Entonces algunas propiedades del clasificador-DD son las siguientes:

(1) Asuma que $P_F\{z : r(D_F(z)) = D_G(z)\} = P_G\{z : r(D_F(z)) = D_G(z)\} = 0$ y que $D_F(.)$ y $D_G(.)$ son continuas y que $sup_z|D_{F_m}(z) - D_F(z)| \to^{c.s} 0$ y $sup_z|D_{G_n}(z) - D_G(z)|^{c.s.} \to 0$ cuando $min(m,n) \to \infty$. Entonces para cualquier $C \in \{C_r, r \in \Gamma\}, \hat{\Delta}_N(C) \to^{c.s} \Delta(C)$. cuando $min(m,n) \to \infty$

(2) Con Γ continua, entonces siempre existe algún $\hat{r}_N \in \Gamma$ tal que $\hat{r}_N = argmin_{r\in\Gamma}\hat{\Delta}_N(r)$. Ahora, si $r_0(\equiv argmin_{r\in\Gamma}\Delta(r))$ es única , entonces para casi todas las parejas de muestras, la sucesión $\{\hat{r}_N\}$ converge puntualmente a r_0 cuando $mßn(m,n) \to \infty$

(3) Asuma que F y G admiten funciones de densidad f_1 y f_2 respectivamente. Si existe una función $r_B \in \Gamma$, que da como resultado la regla de Bayes,entonces se tiene que: $\Delta_N(\hat{C}_N) \to^{c.s} \Delta(C_{r_B})$ y $E(\Delta_N(\hat{C}_N)) \to \Delta(C_{r_B})$

(4) Asuma que para todo $\delta \in R, P_F\{z : D_F(z) = \delta\} = P_G\{z : D_F(z)\} = 0$, y que F y G pertenecen a la familia de funciones elípticas. Si Γ pertenece a la familia de funciones crecientes y las profundidades usadas en el clasificador-DD, son las profundidades de Mahalanobis,Half-Space,Simplicial, o profundidad de proyección,entonces $E(\Delta_N(\hat{C}_N))$ converge a la tasa de error de bayes cuando $min(m,n) \to \infty$

La propiedad (2) enuncia que si la función separadora optima r_0 es única y pertenece a Γ, entonces la sucesión de curvas separadoras $\hat{r}_N$ en el clasificador-DD converge a r_0, y la propiedad (3) enuncia que si existe una función separadora en Γ que da como resultado la tasa de error de Bayes ,entonces la sucesión $\Delta_N(\hat{C}_N)$ del clasificador-DD converge a la tasa de error de Bayes.

1.3.3. *Más de 2 clases.* En el caso de $g > 2$, Jun Li, 2012 proponen el siguiente procedimiento. Sea $\{X_{i1}, \ldots, X_{in_i}\}(\equiv X_i)$ ser la muestras del grupo $i(i = 1, \ldots, g)$. Para cualquier para de grupos, dígase los grupos i y j. Ahora aplique el clasificador-DD a $X_{iy}X_j$. Para cualquier futura observación Z, denote los resultados de clasificación de las clases i y j por $M(Z,i,j)$, y sea $M(Z,i,j) = ioj$ dependiendo de cuando Z es asignada al grupo i o al grupo j. Repita este procedimiento para cada posible par de grupos y asocie Z con un vector de membresía $\{M(Z;i,j)\}_{i=1,j=i+1}^{k,k}$. La membresía final de Z es el grupo en el cual aparezca con mayor frecuencia, en el anterior vector, si lazos ocurren , se pondera $M(Z;i,j)$ por $|D_{F_i,n_i}(Z)r_{\hat{a}}(D_{F_j,n_j}(Z))|$, donde $D_{F_l,n_l}()$ es la profundidad muestral con respecto a X_l.

Aquí $\hat{a}$ es el vector de coeficientes del polinomio obtenido por el procedimiento anterior que minimiza la tasa de mala clasificación empírica.

El clasificador DD es una gran mejora sobre el clasificador MD y, en el problema citado anteriormente según Jun Li, 2012, la clasificación del clasificador DD es muy cercana de la clasificación óptima.

1.4. **Clasificador DD^G.** Juan A. Cuesta-Albertos, 2016 extienden el análisis anterior al caso funcional con la posibilidad de aplicar al DD-plot otros métodos de clasificación tradicionales. De igual manera se trabaja de forma más eficiente la clasificación para más de 2 grupos. Este clasificador además puede trabajar en datos funcionales multivariados que permite utilizar información adicional para clasificar, tal como las derivadas de las curvas en conjunto con las trayectorias originales. La G en DD^G hace referencia el incremento en la dimensión que va de 2 en el DD-plot al número de grupos multiplicado por el número de diferentes fuentes de información manejadas.

En Pokotylo, 2016 definen el DD-Plot en el caso que involucra solo dos grupos, como una gráfica bidimensional que traza los pares $(D_1(x), D_2(x))$. Aquí, $D_i(x)$ es la profundidad del punto x con respecto a los datos del i-ésimo grupo. Con esta notación, el DD plot es, en pocas palabras, un mapa entre el espacio funcional X que define los datos y ^{2}:

$$x \to (D_1(x), D_2(x)) \in^2 \tag{1.13}$$

El clasificador DD intenta identificar los dos grupos utilizando la información proporcionada por el DD-plot. Dado que hemos transformado nuestros datos para incluirlos en ^{2}, el marco para separar clases es mucho más simple, dado que las profundidades contienen información sobre cómo separar los grupos. Por lo tanto, la elección de una profundidad es ahora un paso crucial. En Jun Li, 2012, la regla de clasificación es una función polinomial (hasta un orden seleccionado k), asegurando que el punto $(0,0)^t$ le pertenece. Esta regla tiene tres principales inconvenientes. Primero, el número de polinomios diferentes de orden k que pueden servir como regla de clasificación es $\binom{N}{k}$, donde N es el tamaño de la muestra. Este es el número de formas posibles de seleccionar k puntos de N, y cada una de las selecciones tiene un orden k polinomio que se interpola entre estos k puntos y $(0,0)^t$. Claramente, como N aumenta, la complejidad del proceso de estimación crece a la tasa N^k. Segundo, el problema de clasificar más de dos grupos se resolvió en Jun Li, 2012 utilizando votación por mayoría que requiere repetir el procedimiento para cada combinación de grupos. Esto significa que la optimización debe resolverse en $\binom{g}{2}$ veces, donde g es el número de grupos.

El clasificador de DD^G propuesto aquí intenta ofrecer una solución unificada a los inconvenientes previamente establecidos. Supongamos que tenemos un proceso en el espacio del producto $X = X_1$ x ... x X_p, datos multivariados (funcionales), en los que tenemos g grupos (clases o distribuciones). Comencemos asumiendo que $p = 1$. El clasificador de DD^G comienza seleccionando una profundidad D y calculando el siguiente mapa:

$$x \to d = (D_1(x), ..., D_g(x)) \in^g \tag{1.14}$$

$$d = \begin{pmatrix} D_1(x) \\ D_2(x) \\ . \\ . \\ . \\ D_g(x) \end{pmatrix}_{(gx1)} \tag{1.15}$$

Entonces se aplica ahora cualquier clasificador disponible que trabaje en un espacio gdimensional para separar los g grupos. La extensión del procedimiento al caso $p > 1$, solo se necesita una profundidad apropiada D^j para cada subespacio q_j y considere la aplicación

$$\chi = \chi_1 x \dots x \chi_p \to R^G \tag{1.16}$$

$$x = (x_1, \dots, x_p) \to d = (D^1(x_1), \dots, D^p(x_p)) \tag{1.17}$$

$$d = \begin{pmatrix} D_1(x_1) & D_1(x_2) & \dots & D_1(x_p) \\ D_2(x) & D_2(x_2) & \dots & D_2(x_p) \\ . & . & \dots & . \\ . & . & \dots & . \\ . & . & \dots & . \\ D_g(x_1) & D_g(x_2) & \dots & D_g(x_p) \end{pmatrix}_{(gxp)} \tag{1.18}$$

Donde $D^i(x_i)$ es el vector gdimensional de las profundidades del punto $x_i \in \chi_i$ con respecto a los grupos $1, \dots, g$ y $G = gp$. El clasificadorDD es interesante en el contexto funcional porque nos permite disminuir la dimensión del problema de clasificación de infinito a G.

Un interesante escenario surge como vimos anteriormente cuando nos enfrentamos con datos funcionales multivariados esto es, cuando los elementos pertenecen a un espacio producto de espacios funcionales: $\chi = \chi_1 \dots \chi_p$.

1.5. **Otras formas de clasificación en Datos Funcionales.**

1.5.1. *Clasificación Basada en Profundidad.* El procedimiento $DD\alpha$ se desarrolla para resolver el problema de clasificar objetos $d-$dimensionales en clases $q \geq 2$. El procedimiento es completamente no paramétrico; utiliza gráficos de profundidad $q - dimensional$ y un algoritmo muy eficiente para el análisis de discriminación en el espacio de profundidad $[0, 1]^q$. Específicamente, la profundidad es la profundidad del zonoide y el algoritmo es el $alpha - procedure$. En el caso de más de dos clases se realizan varias clasificaciones binarias

y se aplica una regla de mayoría. Se discuten tratamientos especiales para los "extraños", es decir, los datos que tienen un vector de profundidad cero.

1.5.2. *Clasificación en Datos Funcionales.* En Tatjana Lange, 2012 se introduce un procedimiento no paramétrico rápido para clasificar datos funcionales. Consiste en una transformación de dos pasos de los datos originales más un clasificador que opera en una baja dimensión hipercubo. Los datos funcionales se mapean primero en un espacio de pendiente de ubicación de dimensión finita y luego transformado por una función de profundidad multivariante en el gráfico DD, que es un subconjunto del unidad hipercubo. Esta transformación produce también una nueva noción de profundidad para los datos funcionales. Tres Para ello se emplean funciones de profundidad alternativas, así como dos reglas para la clasificación final en $[0,1]^q$. Toda la metodología no implica técnicas de suavizado y es completamente no paramétrica.

En este artículo, exploramos las posibilidades de profundidades en problemas de clasificación en espacios multidimensionales o funcionales. Las profundidades son herramientas relativamente simples que ordenan puntos en un espacio dependiendo de cuán profundos sean con respecto a una distribución de probabilidad P. En el caso unidimensional, es fácil ordenar puntos con respecto a P. La mediana es el punto más interno, y los percentiles extremos son los puntos más externos. Además, si F_P denota la función de distribución de P, entonces

$$DP(x) = min\{F_P(x), 1 - F_P(x)\} \tag{1.19}$$

es un índice que mide qué tan profundo es xR con respecto a P. Este índice también puedese aplicará a las muestras reemplazando F_P con la función de distribución empírica. Otras posibilidades disponibles para definir $D_P(x)$ incluyen aquellas en las que $D_P(x)$ disminuye con la distancia entre x y la media de P, en este caso, el punto más profundo. La mayoría de ellos son positivos y acotados; cuanto más grande es el índice, es más profundo el punto.

1.6. **Profundidad estadística para datos funcionales.** Como se mencionó anteriormente, el clasificador DD es especialmente interesante en el contexto funcional, porque nos permite disminuir la dimensión de un problema de clasificación muy grande. Esta sección revisa varias profundidades de datos funcionales que se utilizan más tarde con el clasificador DD^G y proporciona algunas extensiones para cubrir funciones multivariadas datos.

En Juan A. Cuesta-Albertos, 2016 se realiza una descripción de las profundidades y una forma de utilizar las mismas para datos funcionales multivariados así:

1.6.1. *Fraiman y Muniz (FM).* También se conoce como profundidad integrada por su definición. Dada una muestra $x_1, ..., x_N$ de funciones definidas en el intervalo $[0,T]$, sea $S_t = x_1(t), ..., x_N(t)$ los valores de esas funciones en un intervalo $t \in [0,T]$ dado. Denote por $F_{N,t}$, la distribución empírica de la muestra S_t y por $D_i(t)$ una profundidad univariante de $x_i(t)$ en esta muestra. Entonces, la profundidad de FM para el ith dato es:

$$FM_i = \int_0^t D_i(t)dt \tag{1.20}$$

Una generalización obvia de la profundidad FM es considerar la integración de diferentes profundidades univariadas, como la profundidad de medio espacio (HS, que se define en (1)), la Profundidad simple (SD) o profundidad Mahalanobis (MhD):

$$D_p(x) = min[F_p(x), 1 - F_p(x)] \tag{1.21}$$

$$D_i^{SD}(t) = 2F_{N,t}(x_i(t))(1 - F_{N,t}(x_i(t)^{-1}) \tag{1.22}$$

$$D_i^{MhD}(t) = [1 + (x_i(t) - \hat{\mu}(t))^2/\hat{\sigma}^2(t)]^{-1} \tag{1.23}$$

donde $\hat{\mu}2(t)$, $\hat{\sigma}2(t)$ son estimaciones de la media y la varianza en el punto t.

Por ejemplo, la curva más profunda puede variar según esta selección. Un escenario interesante surge cuando nos enfrentamos a datos funcionales multivariados; es decir, cuando los elementos pertenecen a un espacio producto de espacios funcionales: $\chi = \chi^1\chi^p$. Una combinación de profundidades de la información de todos los componentes es una idea atractiva, porque puede mantener baja la dimensión de nuestro problema de clasificación. Sin embargo, esto correría el riesgo de perder información. Podemos hacer esto de dos formas:

Profundidad ponderada: dado $x_i = (x_i^1, ..., x_i^p) \in \chi$, calcula la profundidad de cada componente, obteniendo los valores FM(x_i^j), $j = 1, ..., p$, y luego defina una versión ponderación de la profundidad FM(FM^w) como:

$$FM_i^w = \sum_{j=1}^{p} w_j FM(x_i^j) \tag{1.24}$$

donde $w = w_1, ..., w_p$ es un vector adecuado de pesos. En la elección de w, debemos tener en cuenta las diferencias en las escalas de las profundidades (por ejemplo, la profundidad FM usando SD como la profundidad univariante toma valores en $[0, 1]$, mientras que la profundidad HS siempre pertenece a el intervalo $[0, 1/2]$).

Soporte común: suponga que todos los χ^i tienen el mismo soporte $[0, T]$ (esto sucede, por ejemplo, cuando se utilizan las curvas y sus derivadas). En este caso, podemos definir una versión resumida p de la profundidad FM(FM^p) como:

$$FMi^p = \int_0^T D_i^p(t)dt \tag{1.25}$$

donde $D_i^p(t)$ es una profundidad p-variada del vector $(x_i^1(t), ..., x_i^p(t))$ respecto a S_t.

1.6.2. *h-modal (hM).* Cuevas A, 2007 propuso la profundidad de hM como una generalización funcional de la Profundidad de probabilidad para medir qué tan rodeada está una curva con respecto a las demás. La profundidad de población hM de un dato x_0 viene dada por

$$f_h(x_0) = E[K(m(x_0, X)/h)] \tag{1.26}$$

donde X es un elemento aleatorio que describe la población, m es una métrica o semimétrica adecuada, $K()$ es un núcleo y h es el parámetro de ancho de banda. Dada una muestra aleatoria $x_1, ..., x_N$ de X, la profundidad h-modal empírica es definida como:

$$\hat{f_h}(x_0) = N^{-1} \sum_{i=1}^{N} K(m(x_0, X)/h) \tag{1.27}$$

Esta ecuación es similar al estimador de densidad kernel no paramétrico habitual, con un diferencia: nuestro interés se centra en lo que sucede en un barrio de cada punto, por lo que no se pretende que el ancho de banda converja a cero cuando $N \to \infty$, y la única restricción es que el ancho de banda debe ser lo suficientemente grande para evitar situaciones patológicas.

Podemos aplicar una profundidad ponderada de los componentes y usarla con datos funcionales multivariantes. Otra alternativa es construir una nueva métrica que combine la los definidos en los componentes del espacio del producto utilizando una métrica p-dimensional, como, por ejemplo, la euclideana; es decir, tomar:

$$m((x_0^1, ..., x_0^p), (x_i^1, ..., x_i^p)) := \sqrt{m_1(x_0^1, x_i^1)^2 + ... + m_p(x_0^p, x_i^p)^2} \tag{1.28}$$

donde m_i denota la métrica en el componente i del espacio del producto. Es importante aquí para asegurar que las diferentes métricas de los espacios tengan escalas similares para evitar un solo componente de dominar la distancia total.

1.6.3. *Proyecciones aleatorias.* Dado una muestra $x_1, ..., x_N$ de funciones en un espacio de Hilbert con producto escalar $\langle , \rangle$, seleccionamos aleatoriamente un vector unitario a en este espacio (independientemente de x_i) y proyectamos los datos en el subespacio unidimensional generado por a. La profundidad de la muestra de un dato x es la profundidad univariante de la proyección $\langle a, x \rangle$ con respecto a la muestra proyectada $\langle a, xi \rangle_{i=1}^{N}$.

La RP mencionada en Juan A. Cuesta-Albertos, 2016, utiliza la profundidad Half Space univariante y resume las profundidades de las proyecciones a través de la media (usando $R = 50$ como opción predeterminada). Por lo tanto, si $D_{a_r}(x)$ es la profundidad asociada con el rth proyección, entonces:

$$RP(x) = RP^{-1} \sum_{r=1}^{R} D_{a_r}(x) \tag{1.29}$$

Las extensiones de datos funcionales multivariados son similares a las propuestas para el Profundidad FM, excepto por el hecho de que el uso de una profundidad variable p con las proyecciones aquí no requiere un soporte común para todos los componentes. La profundidad de RPD propuesta en Cuevas A, 2007 es un ejemplo de esta extensión utilizando las curvas originales y sus derivados como componentes de datos funcionales multivariados, que son de dos dimensiones en este caso.

2. Metodología

Así como para datos funcionales tenemos distintas profundidades estadísticas, basados en la estructura de los datos y la posibilidad o no de aplicar conceptos tradicionales de la estadística sobre datos funcionales como la curva mediana, se hace necesario que los clasificadores sean llevados a este contexto funcional. En el paquete fda.usc del software R existe el método classif.DD, que es el visto en Juan A. Cuesta-Albertos, 2016, el cual hace uso de unas profundidades y unos clasificadores ya establecidos (se pueden encontrar en la documentación del paquete). La intención de este trabajo es poder abordar clasificadores ya conocidos, adicionales a los preestablecidos en el método fda.usc::classif.DD, y que tienen una versión para datos funcionales de su forma tradicional.

2.1. **KNN.** La lógica básica detrás de KNN es explorar su vecindario, asumir que el punto de datos de prueba es similar a ellos y derivar el resultado. En KNN, buscamos k vecinos y hacemos la predicción. En el caso de la clasificación KNN, se aplica una votación mayoritaria sobre los k puntos de datos más cercanos, mientras que, en la regresión KNN, la media de los k puntos de datos más cercanos se calcula como resultado. Como regla general, seleccionamos números impares como k. KNN es un modelo de aprendizaje perezoso donde los cálculos ocurren solo en tiempo de ejecución.

KNN entra en los algoritmos de aprendizaje supervisado. Esto significa que tenemos un conjunto de datos con etiquetas de medidas de entrenamiento (x, y) y querríamos encontrar el vínculo entre xey. Nuestro objetivo es descubrir una función $h : X \to Y$ de modo que teniendo una observación desconocida x, $h(x)$ pueda predecir positivamente la salida idéntica y.

Primero, hablaremos sobre el funcionamiento del algoritmo de clasificación KNN. En el problema de clasificación, el algoritmo de K-vecino más cercano esencialmente dice que para un valor dado de K, el algoritmo encontrará el K vecino más cercano del punto de datos no visto y luego asignará la clase al punto de datos no visto teniendo la clase que tiene el mayor número de puntos de todas las clases de K vecinos. Para las métricas de distancia, usaremos la métrica euclidiana.

$$d(x, x') = \sqrt{(x_1 - x_1')^2 + ... + (x_n - x_n')^2} \tag{2.1}$$

Finalmente, la entrada x se asigna a la clase con la mayor probabilidad.

$$P(y = j|X = x) = \frac{1}{K} \sum_{i \in A} I(y^{(i)} = j) \tag{2.2}$$

Para regresión, la técnica será la misma, pero, en lugar de las clases de los vecinos, tomaremos el valor del objetivo y encontraremos el valor del objetivo para el punto de datos invisible tomando un promedio, media o cualquier función adecuada que se desee.

Ahora surge el problema de definir el valor de la variable K y cómo afectará al clasificador. Bueno, como la mayoría de los algoritmos de aprendizaje automático, la K en KNN es un hiperparámetro que como usuario, debemos decidir para obtener el ajuste más adecuado para el conjunto de datos. Cuando K es pequeño, mantenemos la región de una predicción dada y presionamos a nuestro clasificador para que sea "más ciego" a la distribución general. Un valor pequeño para K proporciona un más preciso ajuste, que tendrá un sesgo bajo pero una gran variación. Gráficamente, nuestro límite de decisión será más irregular. Por otro lado, un K más alto promedia más votantes en cada predicción y, por lo tanto, es más flexible a los valores atípicos. Los valores más grandes de K tendrán límites de decisión más suaves, lo que significa una menor varianza pero un mayor sesgo.

2.1.1. *Propiedades del KNN.* La versión Bayes del algoritmo es fácil de implementar calculando las distancias desde el ejemplo de prueba a todos los ejemplos almacenados, pero es computacionalmente intensivo para grandes conjuntos de entrenamiento. El uso de un algoritmo de búsqueda de vecino más cercano aproximado hace que k-NN sea computacionalmente manejable incluso para grandes conjuntos de datos. Se han propuesto muchos algoritmos de búsqueda de vecinos más cercanos a lo largo de los años; estos generalmente buscan reducir el número de evaluaciones a distancia realmente realizadas.

k-NN tiene algunos resultados de gran consistencia. A medida que la cantidad de datos se acerca al infinito, se garantiza que el algoritmo k-NN de dos clases producirá una tasa de error no peor que el doble de la tasa de error de Bayes (la tasa de error mínima alcanzable dada la distribución de los datos).

2.1.2. *Ventajas KNN.*

- Sencillo y fácil de interpretar
- No hace ningún supuesto por lo que se puede implementar en tareas no lineales.
- Funciona bien en la clasificación con varias clases.
- Funciona tanto en tareas de clasificación como de regresión.

2.1.3. *Desventajas KNN.*

- Se vuelve muy lento a medida que aumenta el número de puntos de datos porque el modelo necesita almacenar todos los puntos de datos.
- No es eficiente en memoria.
- Sensible a valores atípicos.

2.2. **SVM.** Antes de entrar en el algoritmo SVM, hablemos primero de algunas definiciones que debemos utilizar más adelante.

Longitud de un vector: La longitud de un vector x se llama su norma, que se escribe como ||x||. La fórmula de la norma euclidiana para calcular la norma de un vector $x = (x_1, x_2, \ldots, x_n)$ es:

$$||x|| = \sqrt{x_1^2 + x_2^2 + ... + x_n^2} \tag{2.3}$$

Dirección de un vector: La dirección de un vector $x = (x1, x2)$ se escribe como y se define como:

$$w = (\frac{x_1}{||x||}, \frac{x_2}{||x||}) \tag{2.4}$$

Producto escalar: El producto escalar de dos vectores devuelve un escalar. Nos da algunas ideas sobre cómo se relacionan los dos vectores.

$$x \cdot y = ||x||||y||cos(\theta) \tag{2.5}$$

La fórmula algebraica del producto escalar: En general, el producto escalar se puede calcular como sigue para dos vectores n-dimensionales:

$$x \cdot y = \sum_{i=1}^{n} x_i y_i \tag{2.6}$$

Separabilidad lineal: La separabilidad lineal es un concepto importante en SVM. Aunque en casos prácticos los datos pueden no ser linealmente separables, comenzaremos con los casos linealmente separables (ya que son fáciles de entender y tratar) y luego derivaremos los casos no linealmente separables.

Cuando datos bidimensionales están separados por una línea decimos que los datos son linealmente separables. Existen datos separables de forma no lineal, lo que significa que no podemos encontrar una línea para separar los datos bidimensionales. De manera similar, para los datos tridimensionales, decimos que los datos son linealmente separables si podemos encontrar un plano para separarlos.

Hiperplano: Veamos primero el caso bidimensional. Los datos bidimensionales linealmente separables se pueden separar con una línea. La función de la recta es $y = ax + b$. Cambiamos el nombre de x con x_1 y y con x_2 y obtenemos:

$$ax_1 - x_2 + b = 0 \tag{2.7}$$

Si definimos $x = (x1, x2)$ y $w = (a, 1)$, obtenemos:

$$w \cdot x + b = 0 \tag{2.8}$$

Esta ecuación se deriva de vectores bidimensionales. Pero, de hecho, también funciona para cualquier número de dimensiones. Esta es la ecuación del hiperplano.

2.2.1. *Clasificador.* Una vez que tenemos el hiperplano, podemos usar el hiperplano para hacer predicciones. Definimos la función de hipótesis h como:

$$\mathrm{h}(\mathrm{x}_i) = \begin{cases} +1 & if w \cdot x + b \geq 0 \\ -1 & if w \cdot x + b < 0 \end{cases}$$

El punto arriba o sobre el hiperplano se clasificará como clase +1, y el punto debajo del hiperplano se clasificará como clase -1.

Básicamente, el objetivo del algoritmo de aprendizaje SVM es encontrar un hiperplano que pueda separar los datos con precisión. Puede haber muchos de esos hiperplanos. Y necesitamos encontrar el mejor, que a menudo se denomina hiperplano óptimo.

2.2.2. *Problema de optimización de SVM.* Si está familiarizado con el perceptrón, este encuentra el hiperplano actualizando iterativamente sus pesos y tratando de minimizar la función de costo. Sin embargo, si ejecuta el algoritmo varias veces, probablemente no obtendrá el mismo hiperplano cada vez. SVM no sufre este problema. SVM funciona al encontrar el hiperplano óptimo que podría separar mejor los datos.

Entonces surge la pregunta de cómo elegimos el hiperplano óptimo y cómo comparamos los hiperplanos.

2.2.3. *Métricas para comparar hiperplanos.* Primera versión: Consideremos primero la ecuación del hiperplano $w \cdot x + b = 0$. Sabemos que si el punto (x, y) está en el hiperplano, $w \cdot x + b = 0$. Si el punto (x, y) no está en el hiperplano, el valor de $w \cdot x + b$ podría ser positivo o negativo. Para todos los puntos de ejemplo de entrenamiento, queremos saber el punto más cercano al hiperplano. Podríamos calcular $\beta = |w \cdot x + b|$. Para definir formalmente el problema:

Dado un conjunto de datos $D = \{(x_i, y_i)|x_i \in R^n, y_i \in \{-1, 1\}\}_{i=1}^m$, calculamos β para cada ejemplo de entrenamiento, y B es el β más pequeño que obtenemos.

$$B = \min_{i=1...m} |w \cdot x + b| \tag{2.9}$$

Si tenemos s hiperplanos, cada uno de ellos tendrá un valor B_i, y seleccionaremos el hiperplano con el mayor valor B_i.

$$H = \max_{i=1...s} \{h_i|B_i\} \tag{2.10}$$

El problema con esta métrica es que podría no distinguir entre un hiperplano bueno y uno malo. Debido a que tomamos el valor absoluto de $w \cdot x + b$, podríamos obtener el mismo valor para un hiperplano correcto y uno incorrecto. Necesitamos ajustar esta métrica.

Segunda versión: Podríamos utilizar la información de la etiqueta y. Definamos $f = y(w \cdot x + b)$, y el signo de f siempre será positivo si el punto se clasifica correctamente y será negativo si se clasifica incorrectamente.

Para hacerlo formal, dado un conjunto de datos D, calculamos f para cada ejemplo de entrenamiento, y F es la f más pequeña que obtenemos. En la literatura, F se denomina margen funcional del conjunto de datos.

$$F = \min_{i=1...m} y_i(w \cdot x + b) \tag{2.11}$$

Al comparar hiperplanos, se seleccionará favorablemente el hiperplano con el F más grande.

Parece que encontramos la métrica correcta. Sin embargo, esta métrica adolece de un problema llamado variante de escala. Por ejemplo, tenemos dos vectores $w_1 = (3, 4)$ y $w_2 = (30, 40)$. Como tienen el mismo vector unitario $u = (0.6, 0.8)$, los dos vectores w_1 y w_2 representan el mismo hiperplano. Sin embargo, cuando calculamos F, el que tiene w_2 devolverá un número mayor que el que tiene w_1. Necesitamos encontrar una métrica que sea invariante en escala.

Tercera versión: Dividimos f por la longitud del vector w. Definimos $\gamma = y(\frac{w}{\|w\|} \cdot x + \frac{b}{\|w\|})$.

Para hacerlo formal, dado un conjunto de datos D, calculamos γ para cada ejemplo de entrenamiento, y M es el γ más pequeño que obtenemos. En la literatura, M se denomina margen geométrico del conjunto de datos.

$$M = \min_{i=1...m} y_i(\frac{w}{||w||} \cdot x + \frac{b}{||w||}) \tag{2.12}$$

Al comparar hiperplanos, se seleccionará favorablemente el hiperplano con el M más grande.

Ahora tenemos una métrica perfecta para comparar diferentes hiperplanos. Nuestro objetivo es encontrar un hiperplano óptimo, lo que significa que necesitamos encontrar los valores de w y b del hiperplano óptimo.

El problema de encontrar los valores de w y b se llama problema de optimización.

2.2.4. *Derivación del problema de optimización de SVM.* Para encontrar los valores de w y b del hiperplano óptimo, necesitamos resolver el siguiente problema de optimización, con la restricción de que el margen geométrico de cada ejemplo debe ser mayor o igual que M:

$$\begin{gathered} \max_{w,b} M \\ \textbf{subject to } \gamma_i \geq M, i = 1...m \end{gathered} \tag{2.13}$$

También sabemos que $M = \frac{F}{\|w\|}$, el problema anterior se puede reescribir como:

$$\begin{gathered} \max_{w,b} M \\ \textbf{subject to } f_i \geq F, i = 1...m \end{gathered} \tag{2.14}$$

Si cambiamos la escala de w y b, seguimos maximizando M y el resultado de la optimización no cambiará. Cambiemos la escala de w y b y hagamos $F = 1$, el problema anterior se puede reescribir como:

$$\begin{gathered} \max_{w,b} \frac{1}{||w||} \\ \textbf{subject to } f_i \geq 1, i = 1...m \end{gathered} \tag{2.15}$$

Este problema de maximización es equivalente al siguiente problema de minimización:

$$\begin{aligned} &\min_{w,b} ||w|| \\ &\textbf{subject to } f_i \geq 1, i = 1...m \end{aligned} \tag{2.16}$$

Este problema de minimización es equivalente al siguiente problema de minimización:

$$\begin{aligned} &\min_{w,b} \tfrac{1}{2}||w||^2 \\ &\textbf{subject to } y_i(w \cdot x + b) - 1 \geq 0, i = 1...m \end{aligned} \tag{2.17}$$

La declaración anterior es el problema de optimización de SVM. Se llama problema de optimización cuadrática convexa. Y hablaremos sobre cómo resolver este problema en la siguiente sección.

2.2.5. *Resolviendo el problema de optimización de SVM - Hard Margin SVM.* Podemos reformular el problema de optimización de SVM usando el método del multiplicador de Lagrange.

2.2.6. *Problema de SVM Lagrange.* Lagrange afirmó que si queremos encontrar el mínimo de f bajo la restricción de igualdad g, solo necesitamos resolver:

$$\nabla f(x) - \alpha \nabla g(x) = 0 \tag{2.18}$$

α se llama multiplicador de Lagrange.

En términos del problema de optimización de SVM, $f(w) = \frac{1}{2}\|w\|^2$, $g(w, b) = y_i(w \cdot x + b) - 1, i = 1...m$. La función lagrangiana es entonces $\mathcal{L}(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{m} \alpha_i[y_i(w \cdot x + b) - 1]$.

Para resolver analíticamente $\nabla\mathcal{L}(w, b, \alpha) = 0$, el número de ejemplos tiene que ser pequeño. Por tanto, reescribiremos el problema utilizando el principio de dualidad.

De manera equivalente, necesitamos resolver el siguiente problema primario lagrangiano:

$$\begin{aligned} &\min_{w,b} \max \mathcal{L}(w, b, \alpha) \\ &\textbf{subject to } \alpha_i \geq 0, i = 1...m \end{aligned} \tag{2.19}$$

Más exactamente, α debería ser multiplicadores KKT (Karush-Kuhn-Tucker) porque aquí estamos tratando con restricciones de desigualdad. Pero usaremos el término multiplicadores lagrangianos para la continuidad.

Hay un α para cada ejemplo, necesitamos maximizar $\mathcal{L}(w, b, \alpha)$ para todos los ejemplos. Y hay un (w, b) para cada hiperplano, necesitamos minimizar el $max\mathcal{L}(w, b, \alpha)$ mientras tanto.

2.2.7. *Problema dual de Wolfe.* Veamos cuál es el problema dual del problema primario anterior. La función lagrangiana es:

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2} w \cdot w - \sum_{i=1}^{m} \alpha_i[y_i(w \cdot x + b) - 1] \tag{2.20}$$

Para el problema dual, tenemos que:

$$\begin{array}{c} \nabla_w \mathcal{L}(w,b,\alpha) = w - \sum_{i=1}^{m} \alpha_i y_i x_i = 0 \\ \nabla_b \mathcal{L}(w,b,\alpha) = -\sum_{i=1}^{m} \alpha_i y_i = 0 \end{array} \tag{2.21}$$

De las dos ecuaciones anteriores, obtenemos $w = \sum_{i=1}^{m} \alpha_i y_i x_i$ y $\sum_{i=1}^{m} \alpha_i y_i = 0$. Los sustituimos en la función lagrangiana $\mathcal{L}$ y obtenemos:

$$W(\alpha, b) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j x_i \cdot x_j \tag{2.22}$$

Por tanto, el problema dual se plantea como:

$$\begin{array}{c} \max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j x_i \cdot x_j \\ \textbf{subject to } \alpha_i \geq 0, i = 1...m, \sum_{i=1}^{m} a_i y_i = 0 \end{array} \tag{2.23}$$

La ventaja del problema dual de Wolfe sobre el problema primario de Lagrange es que la función objetivo ahora solo depende de los multiplicadores de Lagrange, que es más fácil de resolver analíticamente.

Tenga en cuenta que debido a que las restricciones son desigualdades, en realidad extendemos el método de los multiplicadores de Lagrange a las condiciones KKT (Karush-Kuhn-Tucker). La condición de holgura complementaria de las condiciones KKT establece que:

$$\alpha_i [y_i (w \cdot x^{\star} + b) - 1] = 0 \tag{2.24}$$

x^* son el punto / puntos donde alcanzamos el óptimo. El valor α es positivo para estos puntos. Y el valor α de otros puntos está cerca de cero. Entonces $y_i(w \cdot x^{\star} + b) - 1$ debe ser cero. Estos ejemplos se denominan vectores de soporte, que son los puntos más cercanos al hiperplano.

Calcular w y b: Después de resolver el problema dual de Wolfe, obtenemos un vector de α que contiene el valor del multiplicador de Lagrange para cada ejemplo. Luego podemos proceder a calcular w y b, lo que determina el hiperplano óptimo.

Según la ecuación anterior:

$$w - \sum_{i=1}^{m} \alpha_i y_i x_i = 0 \tag{2.25}$$

Obtenemos:

$$w = \sum_{i=1}^{m} \alpha_i y_i x_i \tag{2.26}$$

Para calcular el valor de b, obtuvimos:

$$y_i(w \cdot x^\star + b) - 1 = 0 \tag{2.27}$$

Multiplicamos ambos lados por y_i y sabemos que $y_i^2 = 1$. Obtenemos:

$$b = y_i - w \cdot x^\star \tag{2.28}$$

Por lo tanto, podríamos calcular b como:

$$b = \frac{1}{S} \sum_{i=1}^{S} (y_i - w \cdot x) \tag{2.29}$$

S es el número de vectores de soporte.

2.2.8. *Clasificador.* Una vez que tenemos el hiperplano, podemos usar el hiperplano para hacer predicciones. La función de hipótesis h es:

$$\mathrm{h}(\mathrm{x}_i) = \begin{cases} +1 & if w \cdot x + b \geq 0 \\ -1 & if w \cdot x + b < 0 \end{cases}$$

2.2.9. *Resolviendo el problema dual de Wolfe.* Podemos resolver el problema dual de Wolfe usando algún paquete o biblioteca analíticamente. Por ejemplo, podemos usar un paquete de Python llamado CVXOPT, que es para optimización convexa. Este paquete proporciona un solucionador QP para problemas de programación cuadrática. Podemos reorganizar nuestro problema dual, establecer los parámetros requeridos, enviarlos al solucionador de QP y obtener la solución esperada. La solución contendrá los valores de los multiplicadores de Lagrange para cada ejemplo. Entonces podemos calcular w y b y obtener el hiperplano óptimo. No enumeraré el código aquí porque está fuera del alcance de este artículo, pero puede implementarlo por su cuenta, lo que no debería ser muy difícil.

En la práctica, la mayoría de las bibliotecas de aprendizaje automático utilizan un algoritmo creado específicamente para resolver este problema rápidamente: el algoritmo SMO (optimización secuencial mínima). Comparado con el solucionador CVXOPT QP, SMO intenta resolver un problema más simple y trabaja bastante más rápido. No declararé los detalles de SMO, pero puede encontrar más materiales en línea y obtener más información al respecto.

A diferencia de los Perceptrons, ejecutar SVM varias veces siempre devolverá el mismo resultado.

2.2.10. *SVM de margen duro.* La formulación de SVM anterior se denomina SVM de margen duro. El problema con Hard Margin SVM es que no tolera valores atípicos. No funciona con datos separables de forma no lineal debido a valores atípicos. La razón es que si recuerda nuestro problema de optimización inicial, las restricciones son $y_i(w \cdot x_i + b) \geq 1$ para cada ejemplo. Para que el problema de optimización se pueda resolver, deben cumplirse todas las restricciones. Si hay un ejemplo atípico que hace que la restricción no se satisfaga,

la optimización no se podrá resolver. En la siguiente sección, hablaremos sobre cómo lidiar con esta limitación utilizando una variante llamada SVM de margen suave.

2.2.11. *Resolviendo el problema de optimización de SVM - Soft Margin SVM.* El problema con Hard Margin SVM es que solo funciona para datos separables linealmente. Sin embargo, este no sería el caso en el mundo real. Es muy probable que, en casos prácticos, los datos contengan algo de ruido y es posible que no se puedan separar linealmente. Veremos cómo Soft Margin SVM maneja este problema.

Básicamente, el truco que utiliza Soft Margin SVM es muy simple, agrega variables de holgura ζ_i a las restricciones del problema de optimización. Las restricciones ahora se convierten en:

$$y_i(w \cdot x_i + b) \geq 1 - \zeta_i, i = 1...m \tag{2.30}$$

Al agregar las variables de holgura, al minimizar la función objetivo, es posible satisfacer la restricción incluso si el ejemplo no cumple con la restricción original. El problema es que siempre podemos elegir un valor de ζ suficientemente grande para que todos los ejemplos satisfagan las restricciones.

Una técnica para manejar esto es usar la regularización. Por ejemplo, podríamos usar la regularización L1 para penalizar valores grandes de ζ. El problema de optimización regularizada se convierte en:

$$\begin{gathered} \min_{w,b,\zeta} \frac{1}{2}||w||^2 + \sum_{i=1}^{m} \zeta_i \\ \textbf{subject to } y_i(w \cdot x_i + b) \geq 1 - \zeta_i, i = 1...m \end{gathered} \tag{2.31}$$

Además, queremos asegurarnos de que no minimizamos la función objetivo eligiendo valores negativos de ζ. Agregamos las restricciones $\zeta_i \geq 0$. También agregamos un parámetro de regularización C para determinar qué tan importante debería ser ζ, lo que significa cuánto queremos evitar clasificar erróneamente cada ejemplo de entrenamiento. El problema de optimización regularizada se convierte en:

$$\begin{gathered} \min_{w,b,\zeta} \frac{1}{2}||w||^2 + C\sum_{i=1}^{m} \zeta_i \\ \textbf{subject to } y_i(w \cdot x_i + b) \geq 1 - \zeta_i, \zeta_i \geq 0, i = 1...m \end{gathered} \tag{2.32}$$

Nuevamente, si usamos el método de multiplicadores de Lagrange como el anterior y hacemos todos los cálculos matemáticos, el problema de optimización podría transformarse en un problema dual:

$$\begin{gathered} \max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i\alpha_j y_i y_j x_i \cdot x_j \\ \textbf{subject to } 0 \leq \alpha_i \leq C, i = 1...m, \sum_{i=1}^{m} a_i y_i = 0 \end{gathered} \tag{2.33}$$

Aquí la restricción $\alpha_i \geq 0$ se ha cambiado a $0 \leq \alpha_i \leq C$.

2.2.12. *Parámetro de regularización C.* Entonces, ¿qué hace el parámetro de regularización C? Como dijimos, determina la importancia que debe tener ζ. Una C menor enfatiza la importancia de ζ y una C mayor disminuye la importancia de ζ.

Otra forma de pensar en C es que le da control de cómo la SVM manejará los errores. Si establecemos C en infinito positivo, obtendremos el mismo resultado que el Hard Margin SVM. Por el contrario, si establecemos C en 0, ya no habrá restricción y terminaremos con un hiperplano que no clasifica nada. Las reglas generales son: valores pequeños de C resultarán en un margen más amplio, a costa de algunas clasificaciones erróneas; valores grandes de C le darán el clasificador de margen rígido y tolerarán la violación de restricción cero. Necesitamos encontrar un valor de C que no haga que la solución se vea afectada por datos ruidosos.

2.2.13. *Truco de kernel.* Ahora, Soft Margin SVM puede manejar los datos separables no linealmente causados por datos ruidosos. ¿Qué pasa si la separabilidad no lineal no es causada por el ruido? ¿Qué pasa si los datos son característicamente no linealmente separables? ¿Todavía podemos separar los datos usando SVM? La respuesta es, por supuesto, sí. Y hablaremos de una técnica llamada truco del kernel para lidiar con esto.

Imagen que tiene un conjunto de datos bidimensional no linealmente separable, le gustaría clasificarlo usando SVM. Parece que no es posible porque los datos no se pueden separar linealmente. Sin embargo, si transformamos los datos bidimensionales a una dimensión superior, digamos, tridimensional o incluso de diez dimensiones, podríamos encontrar un hiperplano para separar los datos.

El problema es que, si tenemos un gran conjunto de datos que contiene, digamos, millones de ejemplos, la transformación tardará mucho en ejecutarse, y mucho menos los cálculos en el problema de optimización posterior. Repasemos el problema dual de Wolfe:

$$\begin{gathered} \max_\alpha \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j x_i \cdot x_j \\ \textbf{subject to } \alpha_i \geq 0, i = 1...m, \sum_{i=1}^m a_i y_i = 0 \end{gathered} \tag{2.34}$$

Para resolver este problema, en realidad solo nos preocupamos por el resultado del producto escalar $x_i \cdot x_i$. Si hay una función que pudiera calcular el producto escalar y el resultado es el mismo que cuando transformamos los datos en una dimensión superior, sería fantástico. Esta función se llama función del núcleo.

Entonces, el truco del kernel es: si define una función del kernel como $K(x_i, x_j) = x_i \cdot x_j$, reescribimos el problema dual de Wolfe:

$$\begin{gathered} \max_\alpha \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j K(x_i \cdot x_j) \\ \textbf{subject to } \alpha_i \geq 0, i = 1...m, \sum_{i=1}^m a_i y_i = 0 \end{gathered} \tag{2.35}$$

Este es un pequeño cambio, pero en realidad es un truco poderoso. Lo que hace es calcular el resultado de un producto escalar realizado en otro espacio. Ahora tenemos la capacidad de cambiar la función del kernel para clasificar datos separables de forma no lineal.

Hay varios tipos de kernel que podemos usar para clasificar los datos. Algunos de los más populares son kernel lineal, kernel polinomial y kernel RBF.

El kernel lineal se define como:

$$K(x_i, x_j) = x_i \cdot x_j \tag{2.36}$$

Este es el mismo que usamos en la discusión anterior. En la práctica, debe saber que un kernel lineal funciona bien para la clasificación de texto.

El núcleo polinomial se define como:

$$K(x_i, x_j) = (x_i \cdot x_j + c)^d \tag{2.37}$$

Este núcleo contiene dos parámetros: una constante cy un grado de libertad d. Un valor d con 1 es solo el núcleo lineal. Un valor mayor de d hará que el límite de decisión sea más complejo y podría resultar en un sobreajuste.

El kernel RBF se define como:

$$K(x_i, x_j) = exp(-\gamma||x_i - x_j||^2) \tag{2.38}$$

El kernel RBF (Radial Basis Function) también se denomina kernel gaussiano. Resultará en un límite de decisión más complejo. El kernel RBF contiene un parámetro γ. Un pequeño valor de γ hará que el modelo se comporte como una SVM lineal. Un gran valor de γ hará que el modelo se vea muy afectado por los ejemplos de vectores de soporte.

En la práctica, se recomienda probar primero el kernel RBF porque normalmente funciona bien.

2.3. **Propiedades del SVM.**

- Flexibilidad para elegir una función de similitud.
- Escasa solución cuando se trata de grandes conjuntos de datos.
- Solo se utilizan vectores de soporte para especificar el hiperplano de separación.
- Capacidad para manejar grandes espacios de funciones.
- La complejidad no depende de la dimensionalidad del espacio de características.
- El sobreajuste puede controlarse mediante un enfoque de margen suave.
- Buena propiedad matemática: un simple problema de optimización convexa que es garantizado para converger en una única solución global.
- Selección de variables.

2.4. **SVM en el contexto funcional.**

En Fabrice Rossi, 2006 hablan sobre algunas consideraciones que deben ser tenidas en cuenta a la hora de pensar en un support vector machine para datos funcionales. Algunas de estas son:

En espacios de Hilbert de dimensión infinita, el problema del margen duro (P_0) siempre tiene una solución cuando los datos de entrada están en posiciones generales, es decir,

cuando N observaciones abarcan un subespacio N-dimensional de X. Por tanto, una solución muy ingenua consistiría en evitar márgenes blandos y kernels no lineales. Esto no daría resultados muy interesantes en la práctica debido a la falta de regularización.

El SVM lineal con margen suave también puede conducir a malos resultados. De hecho, es bien sabido que el problema (P_C) es equivalente al siguiente problema de optimización sin restricciones:

$$(R_\lambda) min_{w,b} \frac{1}{N} \sum_{i=1}^{N} max(0, 1 - y_i(\langle w, x_i \rangle + b)) + \lambda \langle w, w \rangle \tag{2.39}$$

con $\lambda = \frac{1}{CN}$. Esta forma de ver (P_C) enfatiza el aspecto de regularización y vincula el modelo SVM a la regresión Ridge.

La penalización utilizada en la regresión Ridge se comporta mal con datos funcionales. Por supuesto, la función de pérdida utilizada por SVM (la pérdida de bisagra, es decir, $h(u, v) = max(0, 1 - uv)$) es diferente de la pérdida cuadrática utilizada en la regresión Ridge.

En cualquier caso estos puntos muestran que se pueden esperar malos resultados con el SVM lineal aplicado directamente a los datos funcionales.

Por tanto, es interesante considerar el SVM no lineal para datos funcionales, mediante la introducción de kernels adaptados. (P_C, κ) es equivalente a:

$$(R_{\lambda,\kappa}) min_{f \in \kappa} \frac{1}{N} \sum_{i=1}^{N} max(0, 1 - y_i(f(x_i))) + \lambda \langle f, f \rangle_\kappa \tag{2.40}$$

El uso de un kernel corresponde, por lo tanto, a reemplazar un clasificador lineal por un no lineal, y también para reemplazar la penalización de Ridge por una penalización inducida por el kernel que podría estar más adaptado al problema.

2.5. **Random Forest.** Bosque aleatorio, el modelo de Bagging más popular utilizado hoy en día para conjuntos de datos de bajo sesgo y alta varianza.

En bagging todo el muestreo se realiza como muestreo con reemplazo. Cada modelo se basa en un subconjunto de datos diferente. Se dice que un modelo tiene una alta variación si el modelo varía mucho con los cambios en los datos de entrenamiento. Así que el bagging es un concepto para reducir la variación en el modelo sin afectar el sesgo. $Bagging = DT + muestreodehileras$ Bagging es la aplicación del procedimiento Bootstrap a un algoritmo de aprendizaje automático de alta varianza, generalmente árboles de decisión.

Al tomar un pequeño conjunto de datos de muestra para cada modelo, el modelo agregado no cambia mucho porque solo afecta a un pequeño subconjunto de un conjunto de datos.

La operación de agregación típica es media, mediana o moda. Media o mediana en caso de regresión y Majority voting en caso de problema de clasificación. El paso de agregación final es el modelo final (h).

Toma un montón de modelos de bajo sesgo y alta varianza ($h1, h2, h3, h4\dots$) y se combinan con el bagging. Se obtendrá un sesgo bajo y un modelo de varianza reducida (h).

Al hacer bagging con árboles de decisión, nos preocupa menos que los árboles individuales sobreajusten los datos de entrenamiento. Por esta razón y por eficiencia, los árboles de decisión individuales se cultivan en profundidad (por ejemplo, pocas muestras de entrenamiento en cada nodo de la hoja del árbol) y los árboles no se podan. Estos árboles tendrán una alta varianza y un sesgo bajo. Estas son características importantes de los submodelos cuando se combinan predicciones mediante bagging.

Los bosques aleatorios realizan tanto el muestreo de filas como el muestreo de columnas con el árbol de decisiones como base. Los modelos h1, h2, h3, h4 son más diferentes que el de hacer solo bagging debido al muestreo en columna.

Los bosques aleatorios construyen muchos árboles de decisiones individuales durante el entrenamiento. Las predicciones de todos los árboles se combinan para hacer la predicción final; la moda de las clases para la clasificación o la predicción media para la regresión. Dado que utilizan una colección de resultados para tomar una decisión final, se denominan técnicas de conjunto.

La importancia de la característica se calcula como la disminución de la impureza del nodo ponderada por la probabilidad de alcanzar ese nodo. La probabilidad del nodo se puede calcular por el número de muestras que llegan al nodo, dividido por el número total de muestras. Cuanto mayor sea el valor, más importante será la característica.

Las fórmulas utilizadas para el criterio de impurezas son las siguientes, para la clasificación, se usa la impureza de Gini de forma predeterminada, pero existe la entropía como alternativa. Para la regresión, se calcula la reducción de la varianza utilizando el error cuadrático medio. Además, la reducción de la varianza se puede calcular con el error absoluto medio.

Impurity	**Tarea**	**Fórmula**
Gini	Clasificación	$\sum_{i=1}^{C} f_i(1-f_i)$
Entropía	Clasificación	$\sum_{i=1}^{C} -log(f_i)$
Varianza (MSE)	Regresión	$\frac{1}{N}\sum_{i=1}^{N}(y_i-\mu)^2$
Varianza (MAE)	Regresión	$\frac{1}{N}\sum_{i=1}^{N}\lvert y_i-\mu\rvert$

Table 1. Tabla de fórmulas de Impuridad

Para cada árbol de decisión, se calcula la importancia de un nodo utilizando Importancia de Gini, asumiendo solo dos nodos secundarios (árbol binario):

$$ni_j = w_j C_j - w_{left(j)} C_{left(j)} - w_{right(j)} C_{right(j)} \tag{2.41}$$

donde ni_j es la importancia del nodo j, w_j es número ponderado de muestras que llegan al nodo j, C_j es el valor de impureza del nodo j, $left(j)$ es nodo hijo de la división izquierda en el nodo j, $right(j)$ es nodo hijo de la división derecha en el nodo j.

La importancia de cada característica en un árbol de decisiones se calcula como:

$$fi_i = \frac{\sum_{j:elnodojsedivideenlacaractersticai} ni_j}{\sum_{k \in todoslosnodos} ni_k} \tag{2.42}$$

donde fi_i es la importancia de la característica i, ni_j es la importancia del nodo j.

Luego, estos se pueden normalizar a un valor entre 0 y 1 dividiendo por la suma de todos los valores de importancia de la característica:

$$normfi_i = \frac{fi_i}{\sum_{j \in todoslascaractersticas} fi_j} \tag{2.43}$$

La importancia de la característica final, a nivel de bosque aleatorio, es el promedio de todos los árboles. La suma del valor de importancia de la característica en cada árbol se calcula y se divide por el número total de árboles:

$$RFfi_i = \frac{\sum_{j \in todoslosrboles} fi_{ij}}{T} \tag{2.44}$$

donde $RFfi_i$ es la importancia de la característica i calculada a partir de todos los árboles en el modelo Random Forest, $normfi_{ij}$ es la importancia de la característica normalizada para i en el árbol j, T es número total de árboles.

A medida que aumenta el número de learners base (k), la variación disminuirá. Cuando disminuye k, aumenta la varianza. Pero el sesgo permanece constante durante todo el proceso. k se puede encontrar mediante validación cruzada.

$Bosquealeatorio = DT(alumnobase) + Bagging(muestreodefilasconreemplazo) + Baggingdecaracterstica$
$agregacin(media/mediana, votomayoritario)$

Aquí se quiere que el learner base tenga un sesgo bajo y una varianza alta. así que se entrena un árbol a la longitud completa. no importa la profundidad, se deja crecer porque al final la varianza disminuye en la agregación. Para el modelo h1, el conjunto de datos $(D - D^{'})$ que no se utiliza en el modelado está fuera del conjunto de datos y se utilizaran para la validación cruzada del modelo h1.

2.5.1. *Pasos de implementación.* Suponga que hay N observaciones y M características en el conjunto de datos de entrenamiento. Primero, una muestra del conjunto de datos de entrenamiento se toma al azar con reemplazo. Se selecciona aleatoriamente un subconjunto de características M y la característica que ofrezca la mejor división se utiliza para dividir el nodo de forma iterativa. El árbol crece hasta el más grande. Los pasos anteriores se repiten y la predicción se da en función de la agregación de predicciones de n números de árboles.

2.5.2. *Complejidad de entrenamiento y tiempo de ejecución.* $Tiempodeentrenamiento = O(log(nd) * k)$.

$Tiempodeejecucin = O(profundidad * k)$.

$Espacio = O(almacenarcadaDT * K)$

A medida que aumenta el número de modelos base, aumenta el tiempo de ejecución del entrenamiento, por lo que siempre use la validación cruzada para encontrar el hiperparámetro óptimo.

2.5.3. *Random Forest en el caso funcional.* Supongamos que tenemos observaciones ruidosas para m curvas:

$$y_{ij} = x_i(t_{ij}) + \epsilon_{ij} \tag{2.45}$$

donde $x_i(t), i = 1, , m$, son curvas suaves desconocidas, $t_{ij}, j = 1, , n_i$, son algunos puntos discretos con observaciones en el intervalo $[0, T]$, y ϵij son los errores de medición con media 0 y varianza σ_{ij}^2. Las curvas deben estimarse a partir de datos ruidosos utilizando algunos métodos de estimación no paramétricos.

2.5.4. *Propiedades Random Forest.* Los bosques aleatorios se pueden utilizar para clasificar la importancia de las variables en un problema de regresión o clasificación de forma natural.

2.5.5. *Ventajas de Random Forest.*

- Supera el problema del sobreajuste promediando o combinando los resultados de diferentes árboles de decisión.
- Los bosques aleatorios funcionan bien para una amplia gama de elementos de datos que un solo árbol de decisiones.
- Los bosques aleatorios son muy flexibles y poseen una precisión muy alta.

2.5.6. *Desventajas de Random Forest.*

- La complejidad es la principal desventaja de los algoritmos de bosque aleatorio.
- Se requieren más recursos computacionales para implementar el algoritmo Random Forest y requiere mucho tiempo en comparación con otros algoritmos.
- Es menos intuitivo en el caso de que tengamos una gran colección de árboles de decisión.

2.6. **Naive-Bayes.** Dado un vector de características $X = (x_1, x_2, \ldots, x_n)$ y una variable de clase C_k, el Teorema de Bayes establece que:

$$P(C_k|X) = \frac{P(X|C_k)P(C_k)}{P(X)} \tag{2.46}$$

Para $k = 1, 2, \ldots, K$.

Llamamos $P(C_kX)$ la probabilidad posterior, P (XCk) la probabilidad ,$P(C_kX)$ la probabilidad previa de una clase y $P(X)$ la probabilidad previa del predictor. Estamos interesados en calcular la probabilidad posterior a partir de la probabilidad y las probabilidades previas. Usando la regla de la cadena, la probabilidad P (XCk) se puede descomponer como:

(2.47)

$$P(X|C_k) = P(x_1, ..., x_n|C_k) = P(x_1|x_2, .., x_n, C_k)P(x_2|x_3, .., x_n, C_k)...P(x_{n-1}|x_n, C_k)P(x_n|C_k)$$

Es bastante tedioso calcular el conjunto de probabilidades para todos los valores una y otra vez. Afortunadamente, con el supuesto de Naive de independencia condicional, las probabilidades condicionales son independientes entre sí. En términos de aprendizaje automático, queremos decir que las funciones que se nos proporcionan son independientes y no se afectan entre sí, y esto no sucede en la vida real. Las características dependen de la ocurrencia o el valor de otro, que simplemente es ignorado por el clasificador Naive Bayes y, por lo tanto, recibe el término "NAIVE".

Así, por independencia condicional, tenemos:

$$P(y|x_1,...,x_n) = \frac{P(x_1|y)P(x_2|y)...P(x_n|y)(P(y))}{P(x_1)P(x_2)...P(x_n)} \tag{2.48}$$

Y la probabilidad posterior se puede escribir como:

$$P(y|x_1,...,x_n) \propto P(y)\prod_{i=1}^{n} P(x_i|y) \tag{2.49}$$

ya que el denominador permanece constante para todos los valores.

La discusión hasta ahora ha derivado el modelo de características independientes, es decir, el modelo de probabilidad de Naive Bayes. El clasificador Naive Bayes combina este modelo con una regla de decisión. Una regla común es elegir la hipótesis más probable; esto se conoce como la regla de decisión máxima a posteriori o MAP. El clasificador correspondiente, un clasificador de Bayes, es la función que asigna una etiqueta de clase para algunos k de la siguiente manera:

$$y = argmax_y P(y)\prod_{i=1}^{n} P(x_i|y) \tag{2.50}$$

2.6.1. *Tipos de clasificadores de Naive Bayes.* $P(x_i|y)$ varia según la distribución que se asume.

Gaussiana: se utiliza en la clasificación y supone que las características siguen una distribución normal.

Multinomial: se utiliza para recuentos discretos. Por ejemplo, digamos que tenemos un problema de clasificación de texto. Aquí podemos considerar los ensayos de Bernoulli, que son un paso más allá, y en lugar de contar las instancias de una palabra que aparece en el documento, tenemos un recuento de la frecuencia con la que aparece una palabra en el documento. En términos generales, puede considerarlo como el número de veces que se observa el número de resultado x_i durante los n ensayos.

Bernoulli: El modelo binomial es útil si sus vectores de características son binarios (es decir, ceros y unos). Una aplicación sería la clasificación de texto con un modelo de bolsa de palabras, donde los 1 y 0 son recuentos de la palabra aparece en el documento y que la palabra no aparece en el documento, respectivamente.

2.6.2. *Propiedades de Naive Bayes.* El clasificador de Naive Bayes tiene varias propiedades que lo hacen sorprendentemente útil en la práctica. En particular, el desacoplamiento de las distribuciones de características condicionales de clase significa que cada distribución puede estimarse independientemente como una distribución unidimensional. Esto ayuda a aliviar los problemas derivados de la maldición de la dimensionalidad, como la necesidad de conjuntos de datos que escalen exponencialmente con el número de características. Aunque Naive Bayes a menudo no produce una buena estimación de las probabilidades de clase correctas, esto puede no ser un requisito para muchas aplicaciones. Por ejemplo, el clasificador de Naive Bayes hará la clasificación correcta de la regla de decisión siempre que la clase correcta sea más probable que cualquier otra clase. Esto es cierto independientemente de si la estimación de probabilidad es leve o incluso extremadamente inexacta. De esta manera, el clasificador general puede ser lo suficientemente robusto como para ignorar deficiencias graves en su modelo de probabilidad de Naive subyacente.

2.6.3. *Cómo preparar sus datos para Naive Bayes.* Entradas categóricas: Naive Bayes asume atributos de etiqueta como binarios, categóricos o nominales.

Probabilidades logarítmicas: el cálculo de la probabilidad de diferentes valores de clase implica multiplicar muchas probabilidades pequeñas juntas, lo que puede resultar en un desbordamiento de valores o valores computacionalmente muy bajos. Como tal, es una buena práctica utilizar una transformación logarítmica de las probabilidades para evitar este desbordamiento. Funciones del núcleo: en lugar de suponer una distribución gaussiana para los valores de entrada numéricos, que normalmente se ajusta a todo tipo de valores numéricos, se pueden utilizar distribuciones complejas.

Entradas gaussianas: si las variables de entrada tienen un valor real, se supone una distribución gaussiana. En este caso, el algoritmo funcionará mejor si las distribuciones univariadas de sus datos son gaussianas o casi gaussianas. Esto puede requerir la eliminación de valores atípicos.

Actualizar probabilidades: cuando haya nuevos datos disponibles, simplemente puede actualizar las probabilidades de su modelo.

2.6.4. *Ventajas de Naive Bayes.*

- Cuando la suposición de predictores independientes es cierta, un clasificador Naive Bayes funciona mejor en comparación con otros modelos. Si se cumple el supuesto de independencia condicional de Naive Bayes, convergerá más rápido que los modelos discriminativos como la regresión logística.
- Naive Bayes requiere una pequeña cantidad de datos de entrenamiento para estimar los datos de prueba. Entonces el período de entrenamiento toma menos tiempo.
- Muy simple, fácil de implementar y rápido. Puede hacer predicciones probabilísticas.
- Es altamente escalable. Escala linealmente con el número de características de predicción y puntos de datos.
- Se puede utilizar para problemas de clasificación binarios y de clases múltiples.

- Maneja datos continuos y discretos y no es sensible a características irrelevantes.

2.6.5. *Desventajas de Naive Bayes.*

- La principal limitación de Naive Bayes es la suposición de características predictoras independientes. Naive Bayes asume implícitamente que todos los atributos son mutuamente independientes. En la vida real, es casi imposible que obtengamos un conjunto de predictores que sean completamente independientes entre sí.
- Si una variable categórica tiene una categoría en el conjunto de datos de prueba, que no se observó en el conjunto de datos de entrenamiento, el modelo asignará una probabilidad de 0 (cero) y no podrá realizar una predicción. Esto a menudo se conoce como frecuencia cero. Para solucionar esto, podemos utilizar una técnica de suavizado.

2.6.6. *Naive Bayes en el plano funcional.* Los clasificadores de Bayes para datos funcionales plantean un desafío. Esto se debe a que las funciones de densidad de probabilidad no existen para los datos funcionales. Como consecuencia, el clasificador clásico de Bayes usando es necesario modificar los cocientes de densidad. En Xiongtao Dai, 2016 proponen utilizar relaciones de densidad de proyecciones en una secuencia de funciones propias que son comunes a los grupos a clasificar. Las relaciones de densidad luego se puede factorizar en las relaciones de densidad de los componentes principales funcionales individuales en donde el problema de clasificación se reduce a una secuencia de densidad unidimensional no paramétrica estimada. Esta es una extensión a datos funcionales de algunos de los primeros clasificadores de Bayes no paramétricos que se basaron en relaciones de densidad simples en el caso unidimensional. Mediante la factorización de los cocientes de densidad, se puede evitar la maldición de la dimensionalidad que de otro modo afectaría gravemente a los clasificadores de Bayes para datos funcionales.

Consideramos la situación en la que los datos observados provienen de una distribución común (X, Y), donde X es una función aleatoria integrable cuadrada completamente observada en $L^2(\tau)$, τ es un intervalo compacto y $Y \in 0, 1$ es una etiqueta de grupo. Suponga que X se distribuye como $X^{[}k]$ si X es de población Π_k, $k = 0, 1$, esto implica que, $X^{[}k]$ es la distribución condicional de X dado $Y = k$. también sea $\pi_k = P(Y = k)$ la probabilidad prior de que una observación caiga en Π_k. Nuestro objetivo es inferir la etiqueta de grupo Y de una nueva observación X.

La regla de clasificación de Bayes óptima, que minimiza el error de clasificación errónea, clasifica un observación $X = x$ a Π_1 si:

$$Q(x) = \frac{P(Y = 1|X = x)}{P(Y = 0|X = x)} > 1 \tag{2.51}$$

donde denotamos observaciones funcionales realizadas por x y funciones predictoras aleatorias por X. Si existen las densidades condicionales de las observaciones funcionales X, donde el condicionamiento está en la etiqueta de grupo respectiva, denotamos como g_0 y g_1 cuando se condiciona en el grupo 0 o 1. Entonces el teorema de Bayes implica:

$$Q(x) = \frac{\Pi_1 g_1(x)}{\Pi_0 g_0(x)} \tag{2.52}$$

Sin embargo, el problema está en que las densidades de datos funcionales no suelen existir. Para superar esta dificultad, en Xiongtao Dai, 2016 consideran una secuencia de aproximaciones a las observaciones funcionales, donde el número de componentes está aumentando, y luego usan las razones de densidad previamente descritas. Adicionalmente hacen una seria de simplificaciones y supuestos como asumir que las auto-covarianzas de los procesos estocásticos que generan los datos observados tienen las mismas funciones propias ordenadas para ambas poblaciones.

En Delaigle and Hall, 2010 amplian el enfoque de Xiongtao Dai, 2016 a varios casos de poblaciones y se comparan varios otros enfoques sobre datos simulados y reales. Los resultados muestran que el clasificador ingenuo de Bayes para datos funcionales es aplicable a problemas de clasificación de múltiples categorías y tiene un rendimiento de muestra finita preferible a otros métodos.

2.7. **Majority Voting.** Una colección de varios modelos que trabajan juntos en un solo conjunto se llama conjunto. El método se llama Ensemble Learning. Es mucho más útil utilizar todos los modelos diferentes en lugar de uno solo. Esto permite un eError más bajo y menos sobreajuste. La votación es una de las formas más sencillas de combinar las predicciones de varios algoritmos de aprendizaje automático. El clasificador de votación no es un clasificador real, sino un contenedor para un conjunto de diferentes claisificadores que se entrenan y evalúan en paralelo para explotar las diferentes peculiaridades de cada algoritmo.

Podemos entrenar conjuntos de datos usando diferentes algoritmos y conjuntos para predecir el resultado final. El resultado final de una predicción se toma por mayoría de votos de acuerdo con dos estrategias diferentes: Hard Voting / Majority Voting: El voto duro es el caso más simple de voto mayoritario. En este caso, se elegirá la clase que haya obtenido el mayor número de votos $N_c(y_t)$. Aquí predecimos la etiqueta de clase $\hat{y}$ a través del voto mayoritario de cada clasificador.

$$\hat{y} = argmax(N_c(y_t^1), N_c(y_t^2), ..., N_c(y_t^n)) \tag{2.53}$$

Soft Voting: en este caso, el vector de probabilidad para cada clase predicha (para todos los clasificadores) se suma y se promedia. La clase ganadora es la que corresponde al valor más alto (solo se recomienda si los clasificadores están bien calibrados).

$$\hat{y} = argmax \frac{1}{\#Classifiers} \sum_{Classifiers} (P_1, P_2, ..., P_n) \tag{2.54}$$

Un clasificador de votación puede ser una buena opción siempre que una sola estrategia no pueda alcanzar el umbral de precisión deseado. En cambio, el clasificador de votación abreviada permite la mezcla de diferentes clasificadores adoptando un voto mayoritario para decidir qué clase debe ser considerada ganadora durante una predicción.

Una aproximación similar es el de tomar clasificadores binarios para distintos pares de clases y luego evaluar los nuevos datos en los distintos clasificadores para elegir el que mejores métricas arroje.

3. Datos

Para todos los conjuntos de datos se realizó una partición aleatoria en datos de entrenamiento y de prueba (70, 30), y se corren para las mismas particiones todos los clasificadores usando, por practicidad, como profundidad una ponderación de varias h-modal en caso de ser multivariado, o la h-modal en caso de usar univariado. Adicional se corren varias particiones del conjunto de datos para evaluar la consistencia de los clasificadores, con el fin de medir la generalización del modelo. Todo esto se encuentra en el código en R.

3.1. **Tecator.** Los datos, tratados en este documento como funcionales multivariados, se extrajeron para un análisis espectrométrico para predecir el contenido de grasa de las rebanadas de carne utilizando 215 curvas de absorbancia proporcionadas por el dispositivo analizador de alimentos por infrarrojos tecator Las clases se crearon con un criterio de porcentaje de grasa. Aquellas mayor a 15 son las clase 1 y las menores la clase 0.

3.2. **Fonemas.** El conjunto de datos tiene 250 log-periodogramas de registro de 32 ms de duración correspondientes a cinco diferentes fonemas (sh, dcl, iy, aa, ao). El reto en este dataset está en que para cada iteración de las particiones se debe garantizar un correcto balance de las clases.

3.3. **Clima Canadiense.** Contiene la temperatura y la precipitación diarias en 35 lugares diferentes en Canadá promediaron entre 1960 y 1994. El dataset es multivariado, la primera variable es la temperatura promedio diaria, la segunda es la precipitación y la tercera el logaritmo de la precipitación para 35 estaciones meteorológicas diferentes.

4. Comparision Examples

4.1. **Tecator.** En la data de tecator se tomaron los datos originales y la segunda derivada como si fuera un caso multivariado. En las figuras 1 a 4, generadas con los datos de entrenamiento parece sugerir que las profundidades de los datos en ambas clases son muy similares, lo que lo hace un problema difícil de separar, es por esta razón que el KNN y el Random Forest son los modelos que mejor se desempeñan, teniendo en cuenta que se utilizó un kernel lineal en SVM y una distribución normal para Naive Bayes. Siendo KNN el mejor dado a su consistencia entre el entrenamiento y el testeo.

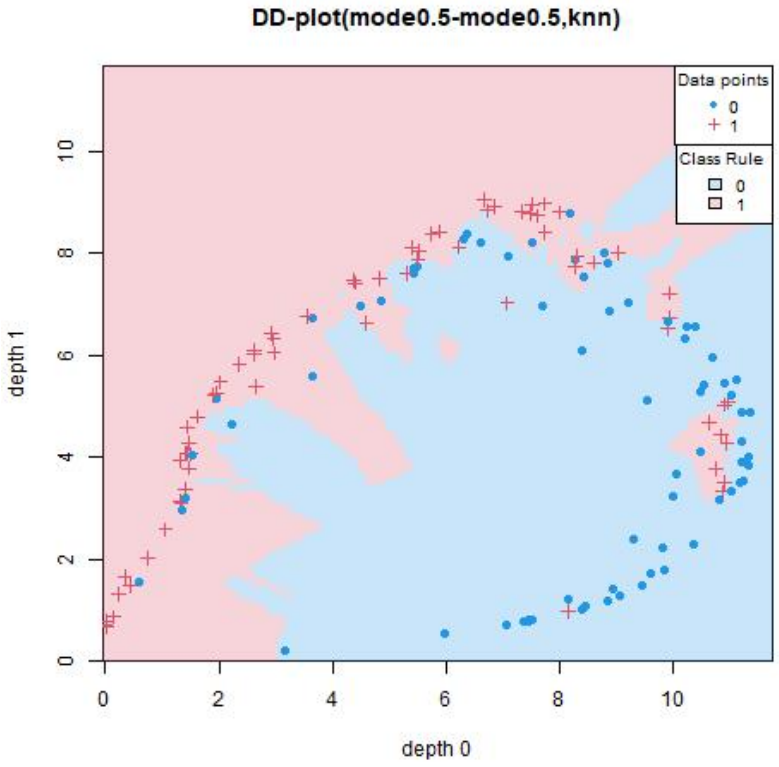

Figure 7. Knn Entrenamiento

		Train			**Test**	
Prob.		0.7	0.75		0.7	0.74
		0	1		0	1
Conf M.	0	55	24	0	23	10
	1	18	54	1	8	23

Table 2. Confusion Matrix and classification probabilities KNN

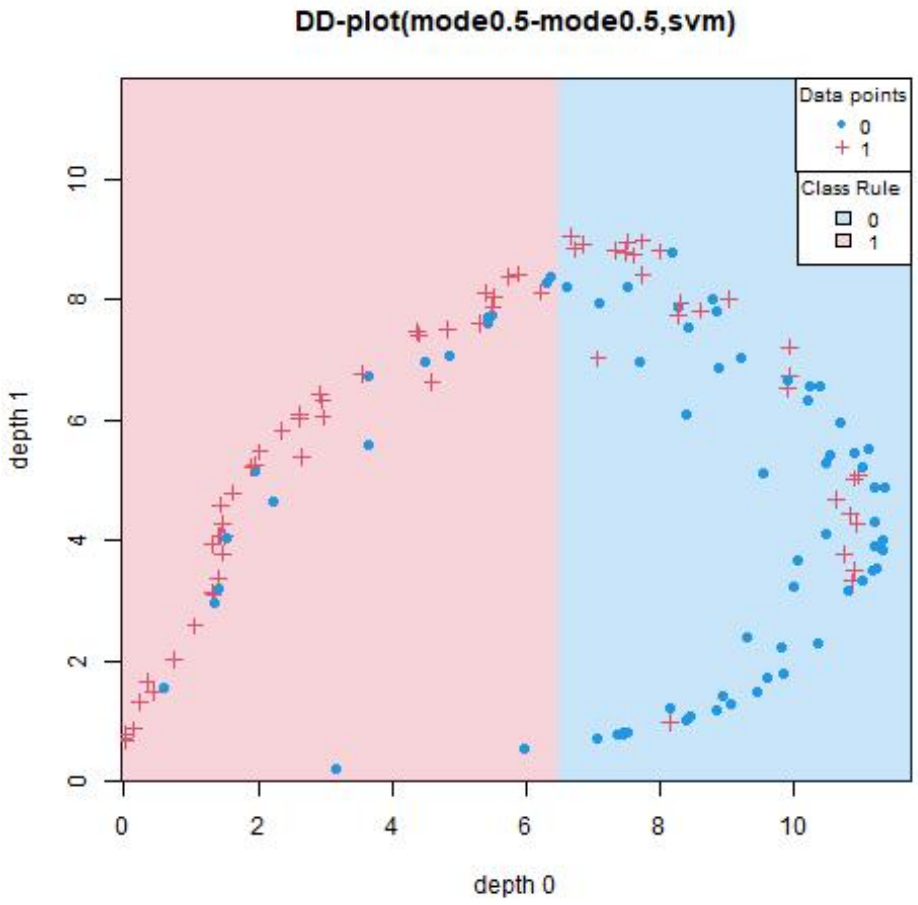

Figure 8. Svm Entrenamiento

		Train			**Test**	
Prob.		0.76	0.61		0.63	0.61
		0	1		0	1
Conf M.	0	60	19	0	21	12
	1	28	44	1	12	19

Table 3. Confusion Matrix and classification probabilities SVM

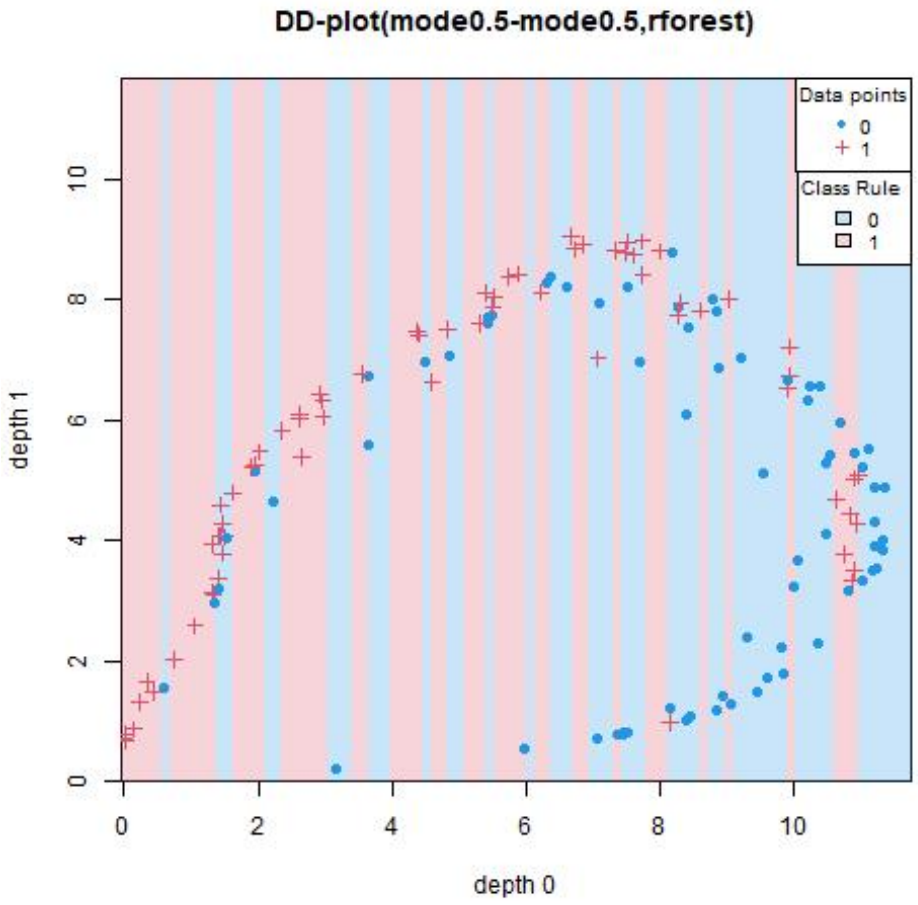

Figure 9. Random Forest Entrenamiento

		Train			**Test**	
Prob.		1	1		0.76	0.64
		0	1		0	1
Conf M.	0	79	0	0	25	8
	1	0	72	1	11	20

Table 4. Confusion Matrix and classification probabilities Random Forest

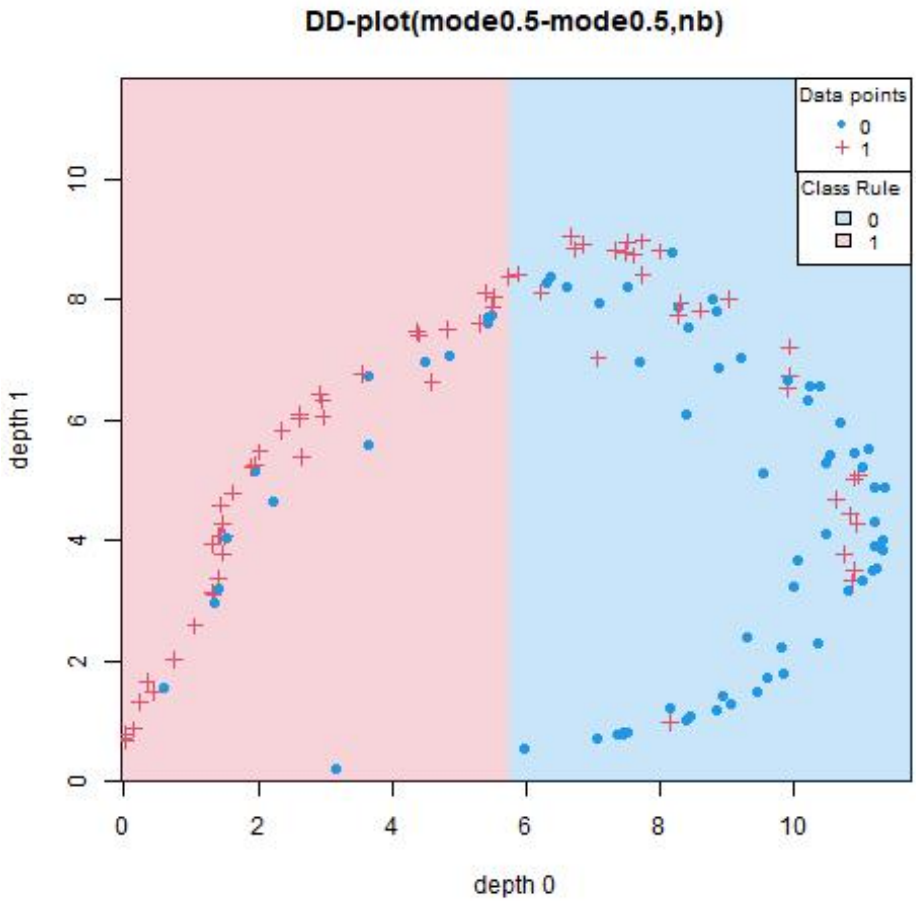

Figure 10. Naive Bayes Entrenamiento

		Train			Test	
Prob.		0.8	0.6		0.72	0.55
		0	1		0	1
Conf M.	0	63	16	0	24	9
	1	31	41	1	14	17

Table 5. Confusion Matrix and classification probabilities Naive Bayes

4.2. **Phoneme.** En las figuras 5 a 8 tenemos gráficas diferentes a las demás, esto debido a que el DDG-plot para acomodarse a más de 2 clases requiere aumentar su dimensión, haciendo comparaciones 1 a 1 de las clases y asignando finalmente una clase bajo el modelo de votos mayoritarios. Este problema se ve fuertemente afectado por la cantidad de datos que quedenen cada una de las clases. En este escenario el SVM y el Naive Bayes tuvieron un buen desempeño pero fallaron tanto en entrenamiento como en prueba al clasificar las clases 3 y 4, por otro lado el Random Forest tuvo el mejor desempeño en entrenamiento pero uno pésimo en prueba por lo que se entiende que no generalizó bien. El ganador fue nuevamente el KNN.

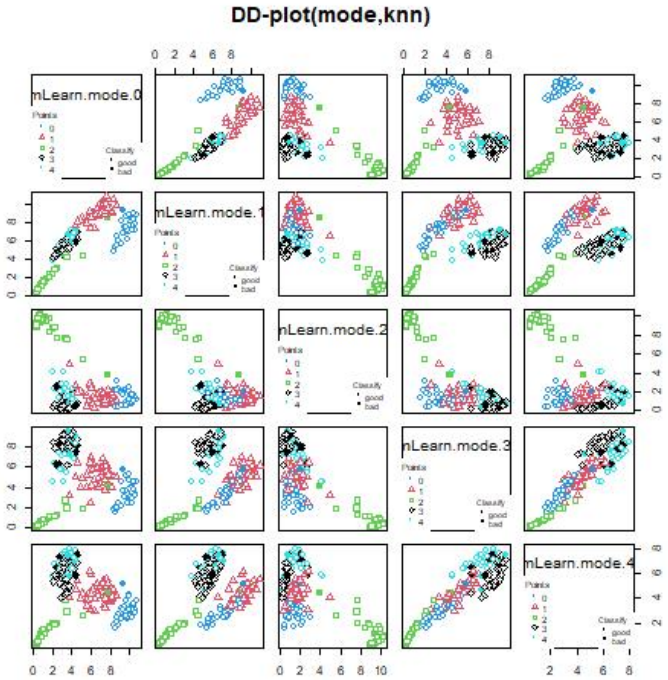

Figure 11. Knn Entrenamiento

		Train						Test				
Prob.		0.97	1	0.97	0.8	0.79		1	1	0.85	0.73	0.76
		0	1	2	3	4		0	1	2	3	4
Conf M.	0	35	1	0	0	0	0	14	0	0	0	0
	1	0	39	0	0	0	1	0	11	0	0	0
	2	0	1	35	0	0	2	0	2	12	0	0
	3	0	0	0	28	7	3	0	0	0	11	4
	4	0	0	0	6	23	4	0	0	1	4	16

Table 6. Confusion Matrix and classification probabilities KNN

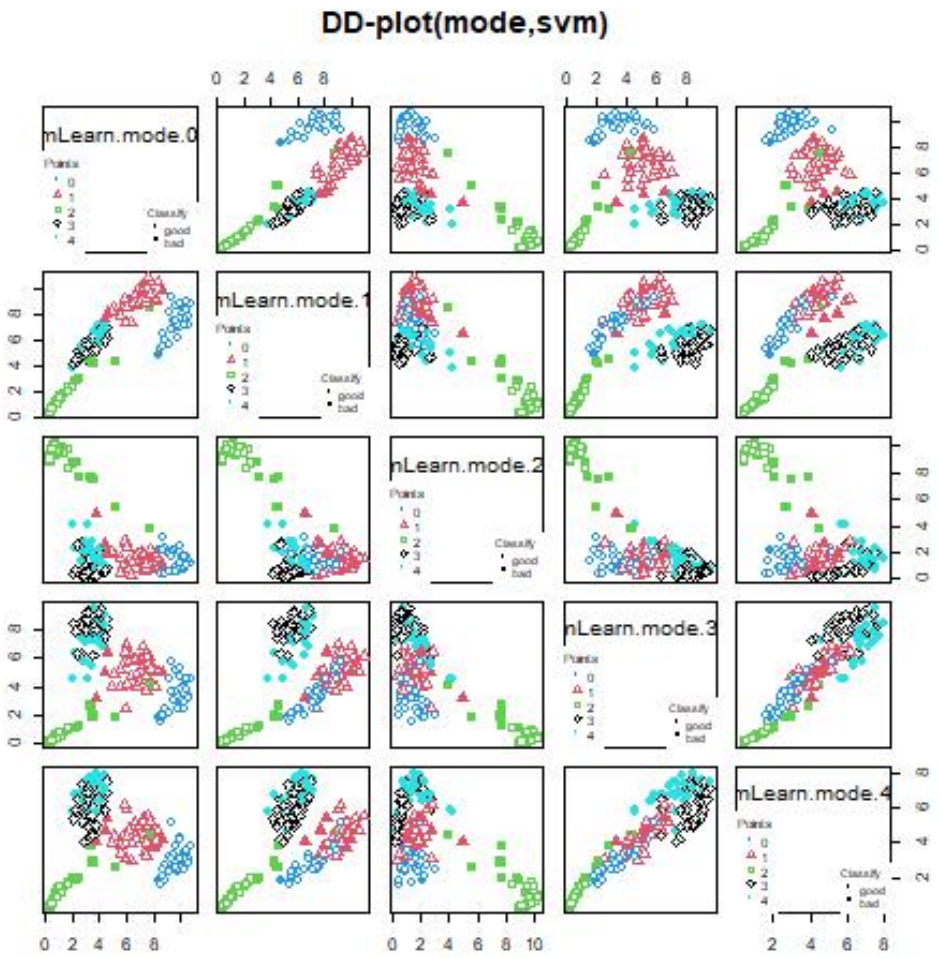

Figure 12. Svm Entrenamiento

		Train						Test				
Prob.		0.97	0.9	0.83	1	0		1	0.72	0.85	0.93	0
		0	1	2	3	4		0	1	2	3	4
	0	35	1	0	0	0	0	14	0	0	0	0
Conf M.	1	1	35	0	3	0	1	1	8	0	2	0
	2	0	2	30	4	0	2	0	2	12	0	0
	3	0	0	0	35	0	3	0	0	1	14	0
	4	0	0	0	29	0	4	0	0	2	19	0

Table 7. Confusion Matrix and classification probabilities SVM

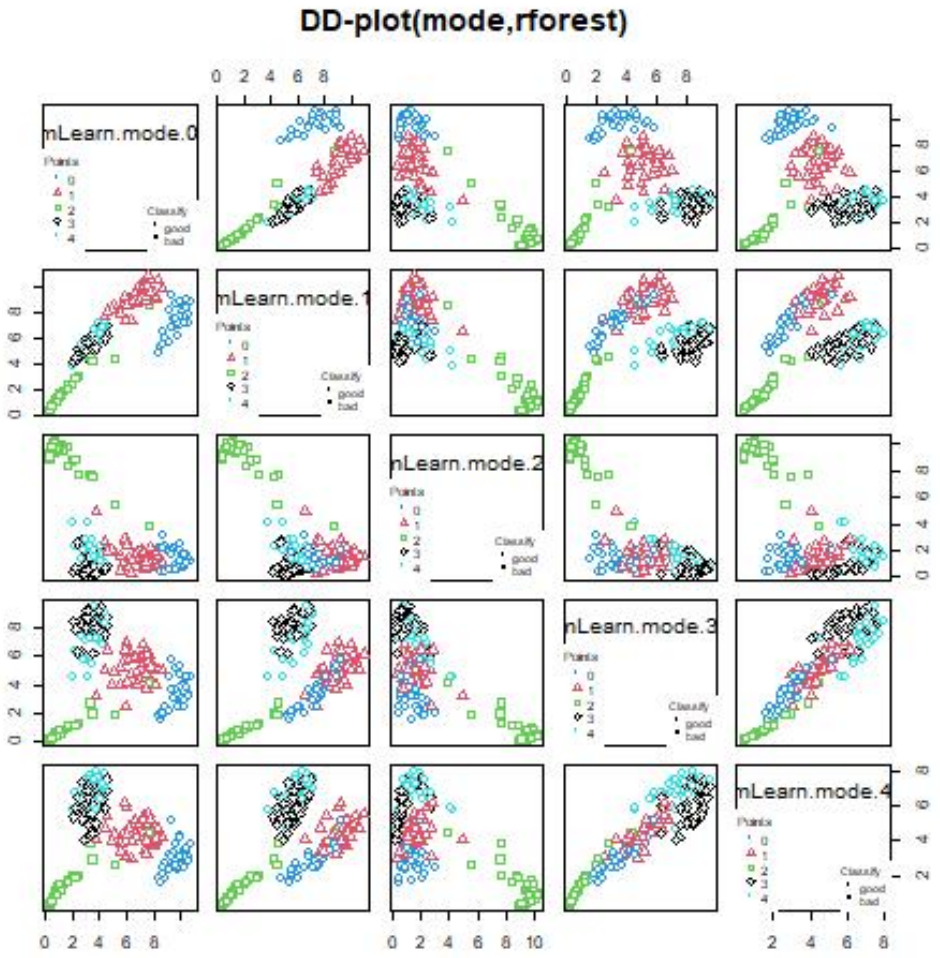

Figure 13. Random Forest Entrenamiento

		Train						Test				
Prob.		1	1	1	1	1		1	0.72	0.85	0.33	0.47
		0	1	2	3	4		0	1	2	3	4
Conf M.	0	36	0	0	0	0	0	14	0	0	0	0
	1	0	39	0	0	0	1	1	8	1	0	1
	2	0	0	36	0	0	2	0	2	12	0	0
	3	0	0	0	35	0	3	0	2	2	5	6
	4	0	0	0	0	29	4	0	0	3	8	10

Table 8. Confusion Matrix and classification probabilities Random Forest

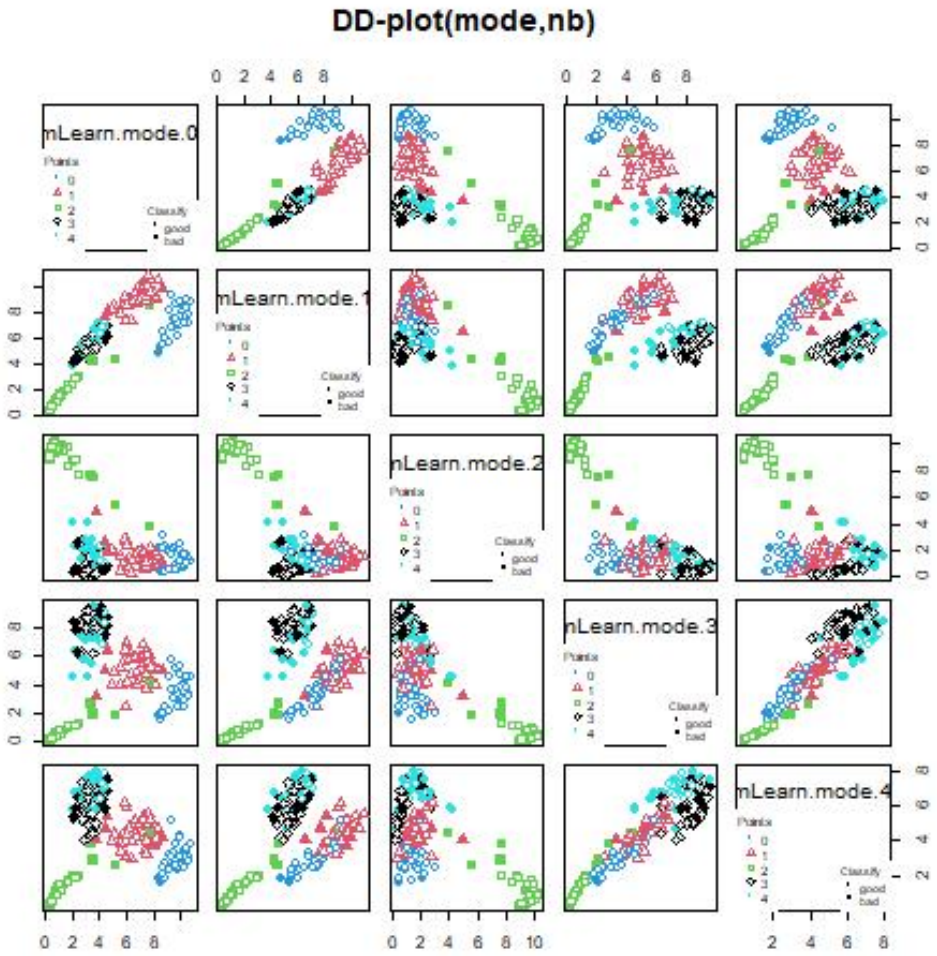

Figure 14. Naive Bayes Entrenamiento

Train

Prob.	0.94	0.9	0.88	0.63	0.38

Conf M.		0	1	2	3	4
	0	34	2	0	0	0
	1	1	35	0	1	2
	2	0	2	32	2	0
	3	0	0	3	22	10
	4	0	0	2	16	11

Test

	1	0.81	0.85	0.8	0.33

		0	1	2	3	4
	0	14	0	0	0	0
	1	1	9	0	0	1
	2	0	2	12	0	0
	3	0	0	1	12	2
	4	0	0	4	10	7

Table 9. Confusion Matrix and classification probabilities Naive Bayes

4.3. **Canadian Weather Univariate.** En las figuras 9 a 12 se observa que en general todos los clasificadores tuvieron un buen desempeño en este dataset en el cual solo tomamos la variable temperatura para las pruebas. El único que falló es el SVM al momento de ajustar con el conjunto de prueba.

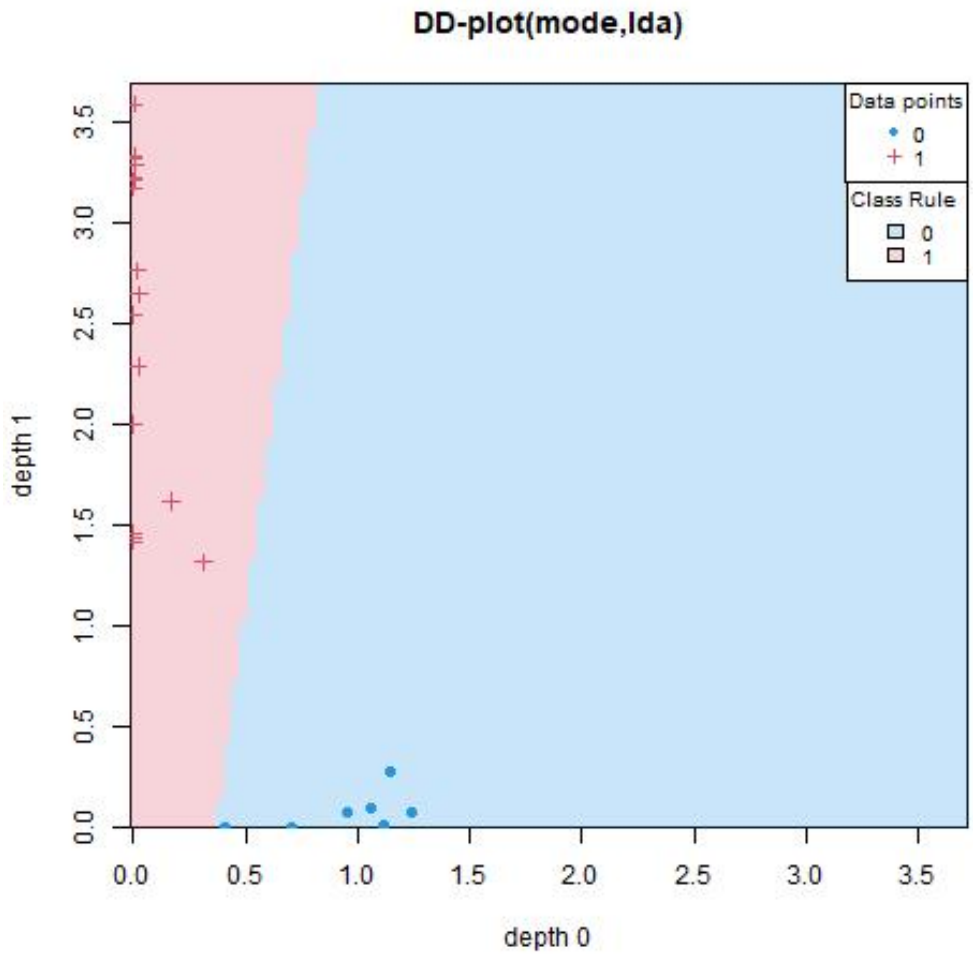

Figure 15. KNN Entrenamiento

	Train			**Test**		
Prob.	1	1		1	1	
		0	1		0	1
Conf M.	0	7	0	0	1	0
	1	0	18	1	0	9

Table 10. Confusion Matrix and classification probabilities KNN

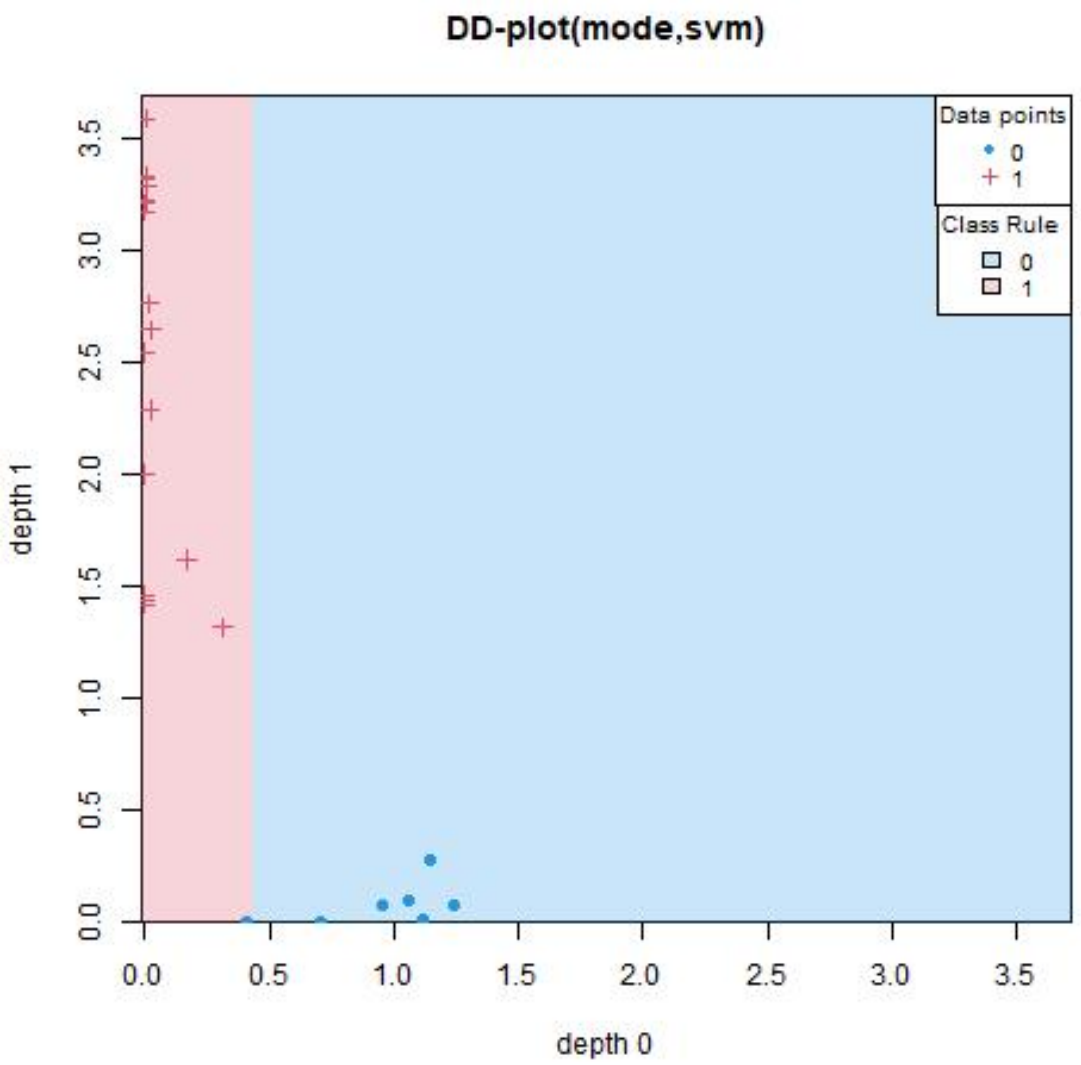

Figure 16. Svm Entrenamiento

		Train			**Test**	
Prob.		0.85	1		0	1
		0	1		0	1
Conf M.	0	6	1	0	0	1
	1	0	18	1	0	9

Table 11. Confusion Matrix and classification probabilities SVM

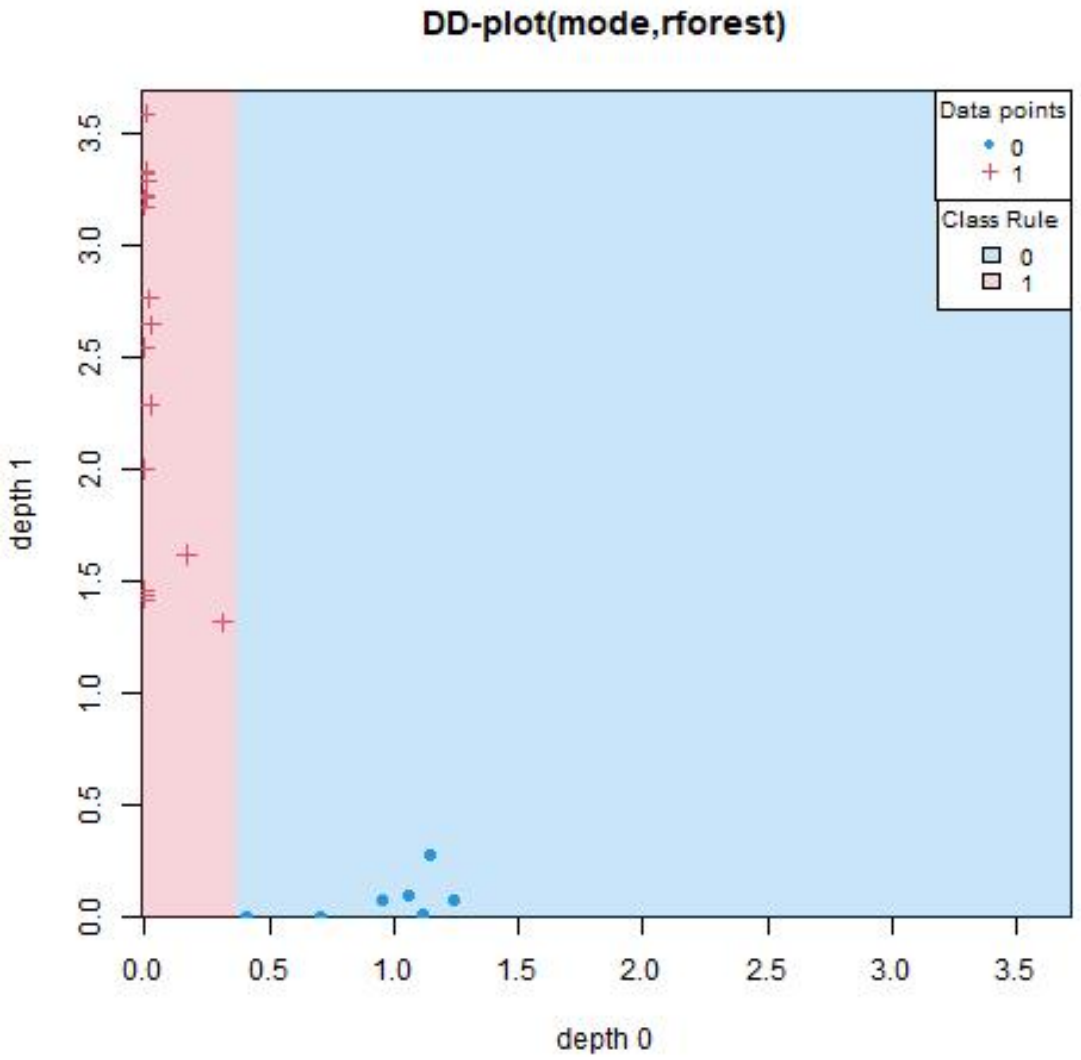

Figure 17. Random Forest Entrenamiento

		Train			**Test**	
Prob.		1	1		1	1
		0	1		0	1
Conf M.	0	7	0	0	1	0
	1	0	18	1	0	9

Table 12. Confusion Matrix and classification probabilities Random Forest

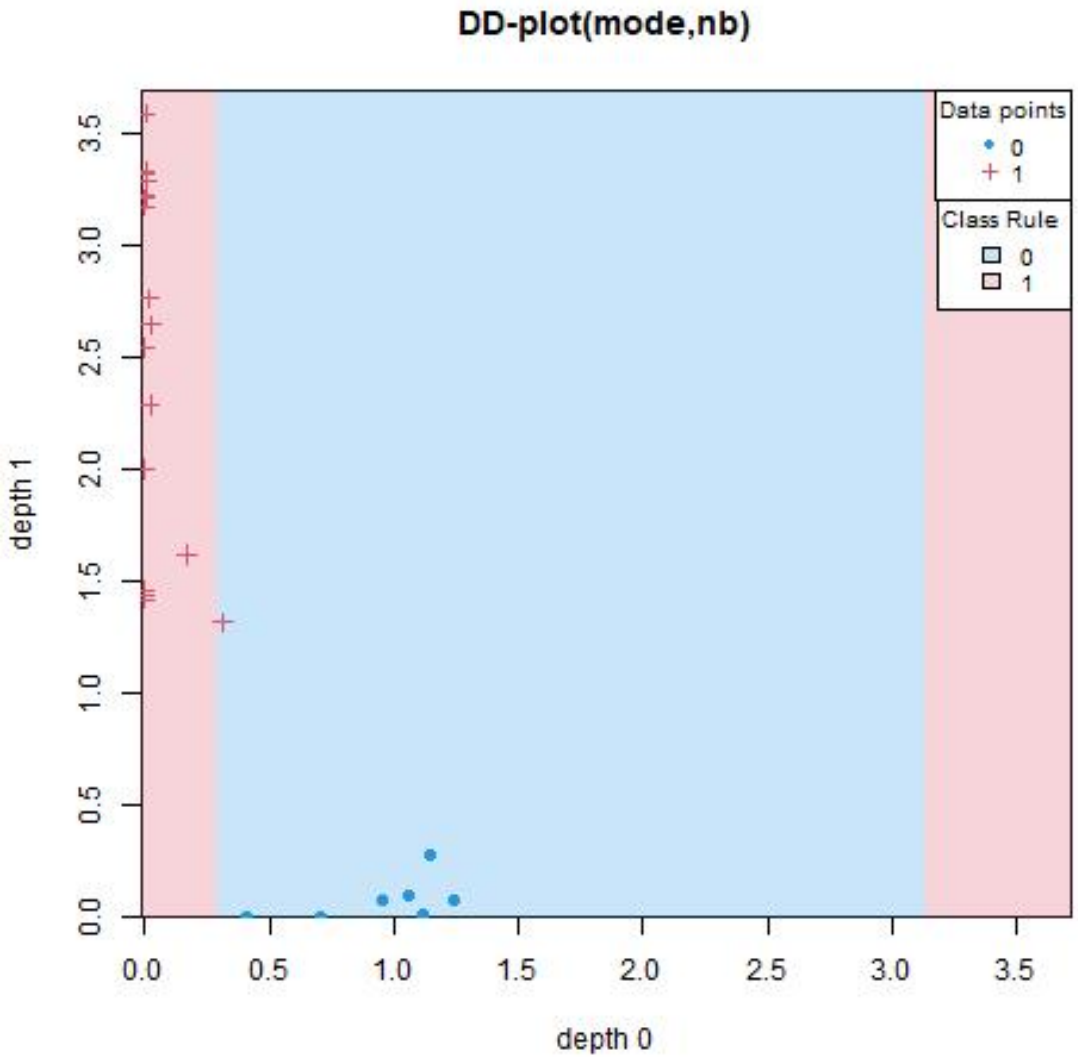

Figure 18. Naive Bayes Entrenamiento

	Train		**Test**	
Prob.	1	0.94	1	1

Conf M.		0	1
	0	7	0
	1	1	17

	0	1
0	1	0
1	0	9

Table 13. Confusion Matrix and classification probabilities Naive Bayes

4.4. **Canadian Weather Multivariate.** Para las figuras 13 a 16 se observa una semejanza entre lo arrojado por Naive Bayes y lo obtenido por Random Forest, ambos con buen desempeño y solo viendose obstaculizados por la baja cantidad de datos que quedaron en la clase 0 del conjunto de prueba. Nuevamente el ganador fue el KNN y el peor el SVM de kernel lineal.

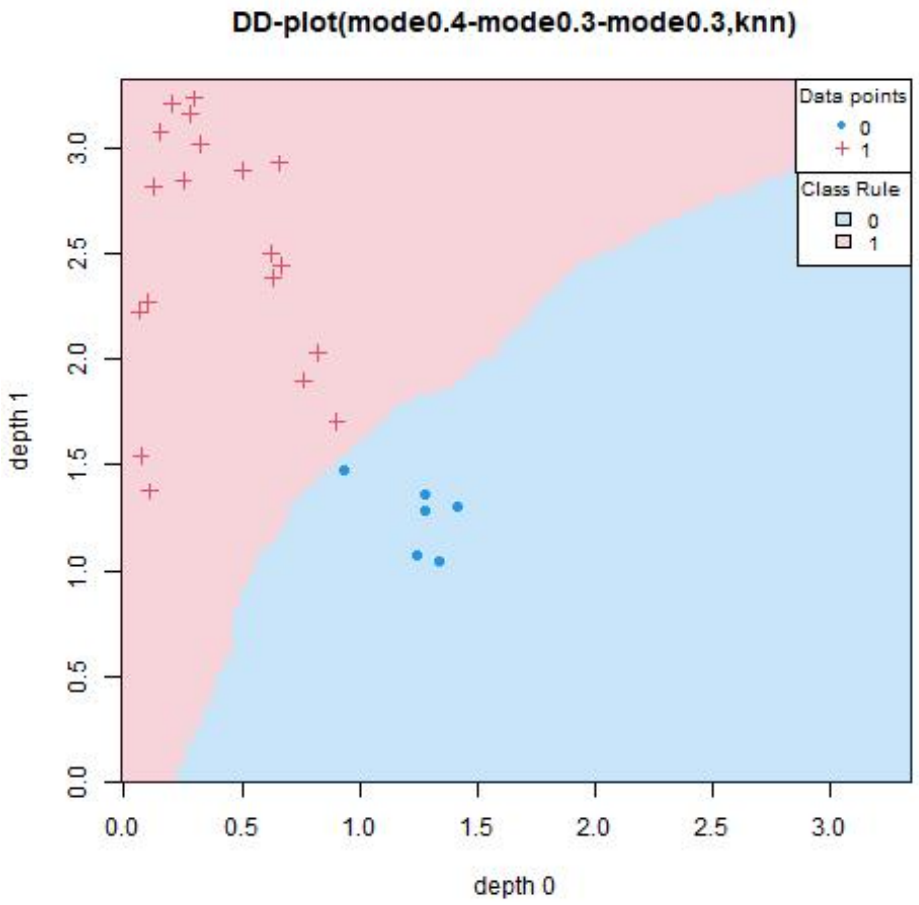

Figure 19. KNN Entrenamiento

	Train			**Test**		
Prob.		1	1		1	1
		0	1		0	1
Conf M.	0	6	0	0	2	0
	1	0	19	1	0	8

Table 14. Confusion Matrix and classification probabilities KNN

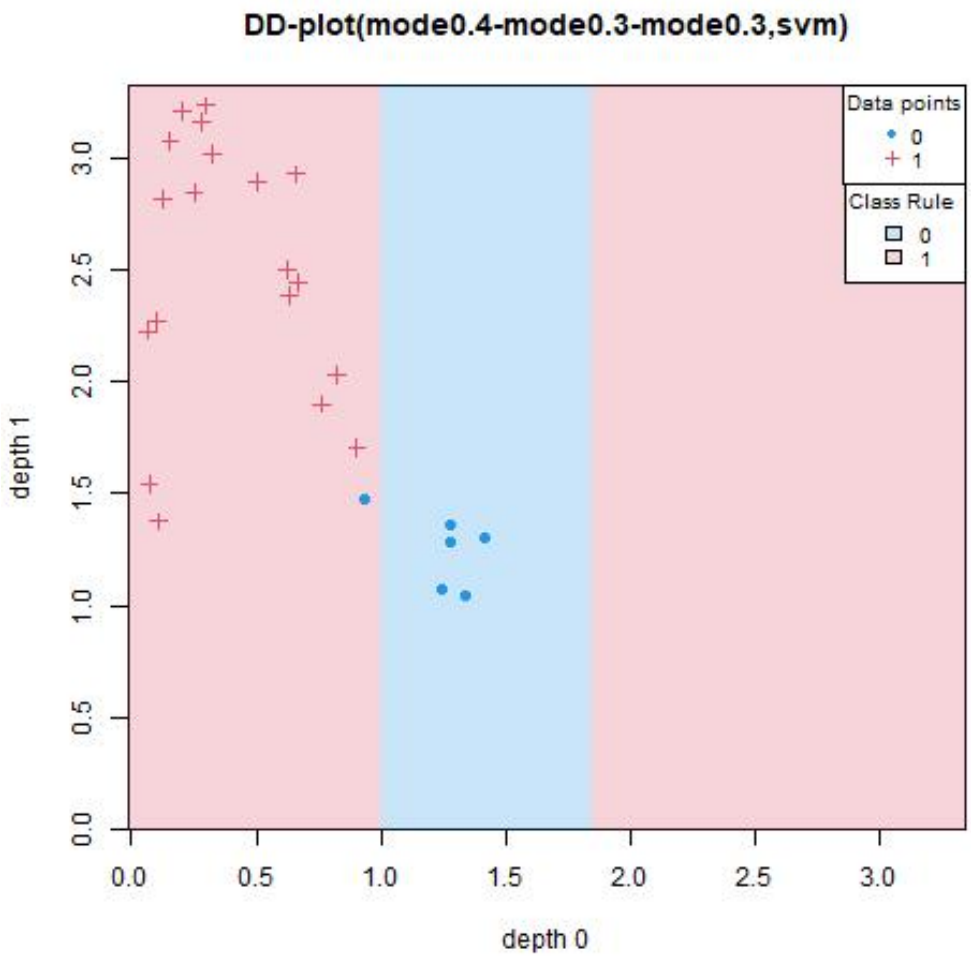

Figure 20. Svm Entrenamiento

		Train			**Test**	
Prob.		0.83	1		0.5	1
		0	1		0	1
Conf M.	0	5	1	0	1	1
	1	0	19	1	0	8

Table 15. Confusion Matrix and classification probabilities SVM

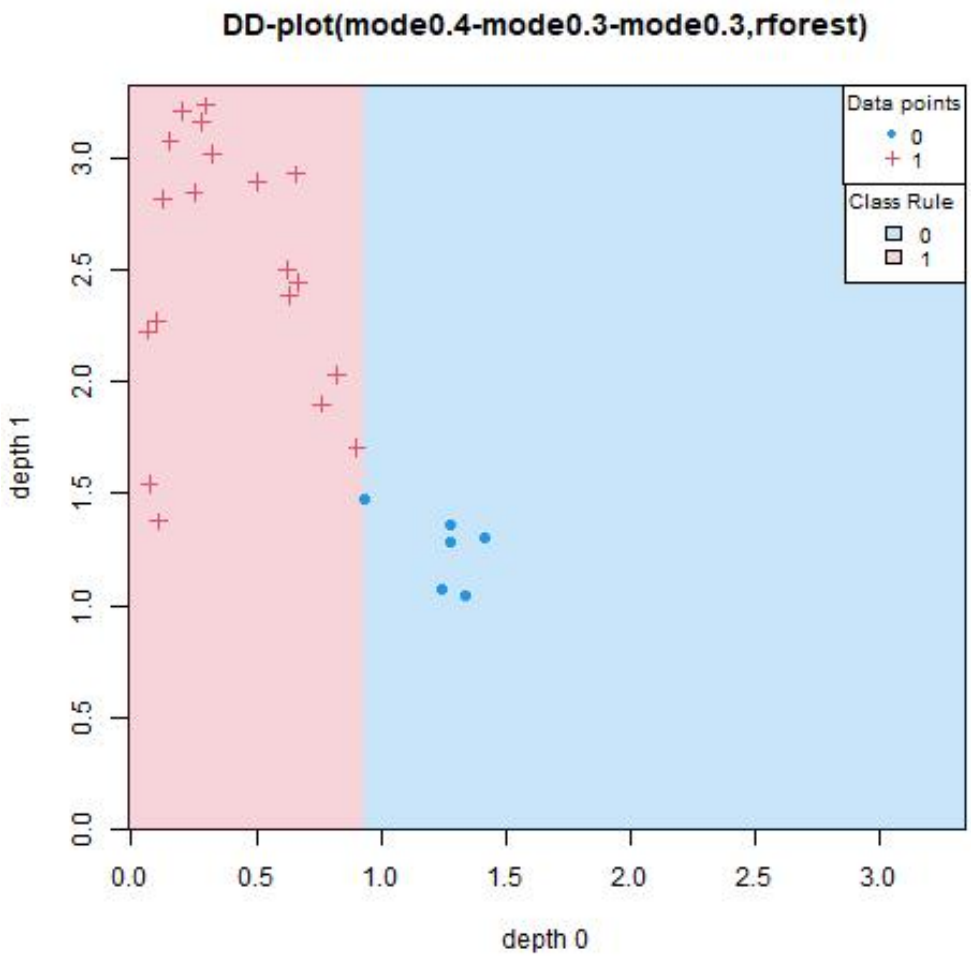

Figure 21. Random Forest Entrenamiento

	Train			**Test**		
Prob.	1	1		0.5	1	
		0	1		0	1
Conf M.	0	6	0	0	1	1
	1	0	19	1	0	8

Table 16. Confusion Matrix and classification probabilities Random Forest

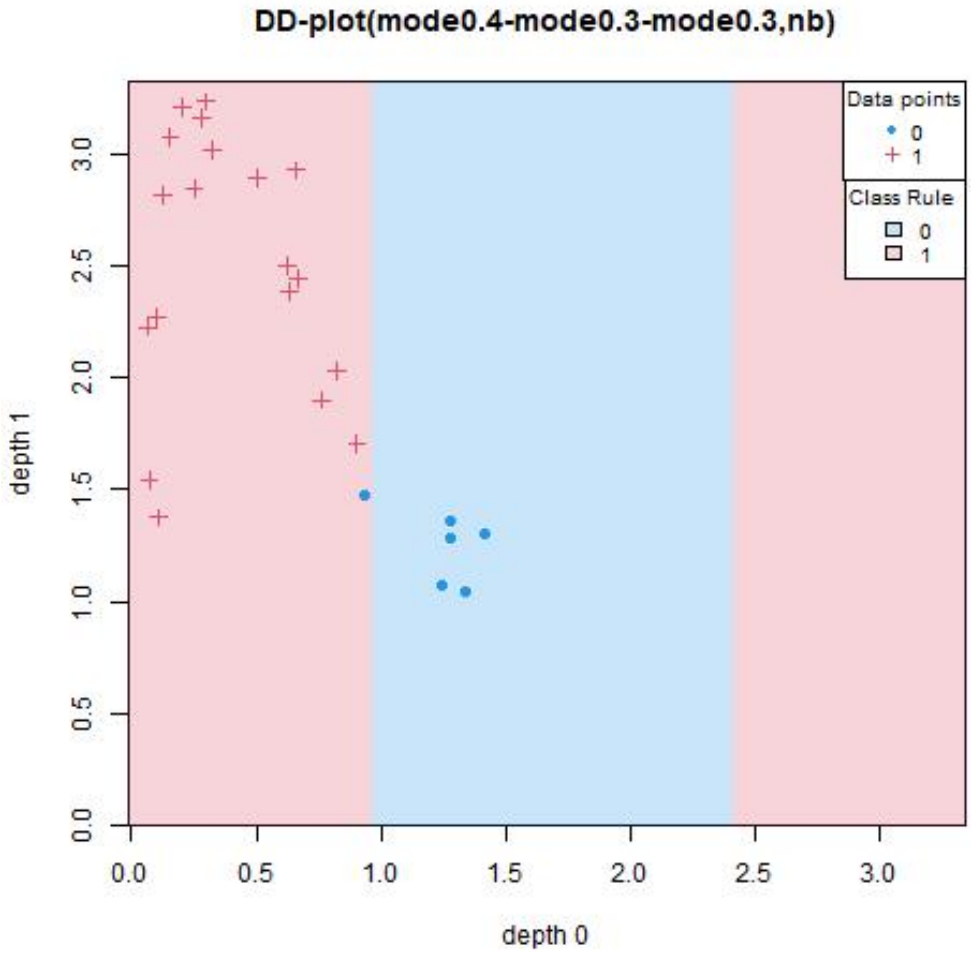

Figure 22. Naive Bayes Entrenamiento

		Train			Test	
Prob.		0.83	1		0.5	1
		0	1		0	1
Conf M.	0	5	1	0	1	1
	1	0	19	1	0	8

Table 17. Confusion Matrix and classification probabilities Naive Bayes

4.5. **Comparasión de Probabilidades de clasificación.** Según lo observado en las siguientes tablas, el mejor desempeño en entrenamiento lo tenía el Random Forest, seguido del Knn, sin embargo al momento de evaluar en los datos de prueba, el más consistente resultaba ser el KNN.

Tecator

	Train		Test	
	0	1	0	1
KNN	0.7	0.75	0.7	0.74
SVM	0.76	0.6	0.64	0.61
R. Forest	1	1	0.76	0.64
N. Bayes	0.8	0.57	0.72	0.55

Table 18. Classification probabilities Tecator

Phoneme

	Train					Test				
	0	1	2	3	4	0	1	2	3	4
KNN	0.97	1	0.97	0.8	0.79	1	1	0.86	0.73	0.76
SVM	0.97	0.9	0.83	1	0	1	0.72	0.86	0.93	0
R. Forest	1	1	1	1	1	1	0.72	0.86	0.33	0.47
N. Bayes	0.94	0.9	0.89	0.63	0.38	1	0.82	0.86	0.8	0.33

Table 19. Classification probabilities Phoneme

Canadian Univariate

	Train		Test	
	0	1	0	1
KNN	1	1	1	1
SVM	0.86	1	0	1
R. Forest	1	1	1	1
N. Bayes	1	0.94	1	1

Table 20. Classification probabilities Canadian Univariate

Canadian Multivariate

	Train		Test	
	0	1	0	1
KNN	1	1	1	1
SVM	0.83	1	0.5	1
R. Forest	1	1	0.5	1
N. Bayes	0.83	1	0.5	1

Table 21. Classification probabilities Canadian Multivariate

5. Conclusiones

Basados en la descripción previa de los conjuntos de datos y con el ánimo de dar una interpretación a los resultados observados se concluye lo siguiente del trabajo.

- Para los datos en tecator, la similitud en sus profundidades hace necesario usar métodos no lineales que se ajusten mejor, por esta razón los modelos que mejor se ajustan son Knn y Random forest.
- Los otros métodos son mucho más rápido que KNN debido a la ejecución en tiempo real de KNN.
- SVM se preocupa por los valores atípicos mejor que KNN.
- Si los datos de entrenamiento son mucho más grandes que el número de variables ($m >> n$), KNN es mejor que SVM. SVM supera a KNN cuando hay funciones grandes y datos de entrenamiento menores.
- Random forest admite la interacción automática de características, mientras que KNN no puede.
- Random forest puede descuidar algunos valores clave en los datos de entrenamiento, lo que puede afectar la precisión.
- SVM usa el kernel para resolver problemas no lineales, mientras que Random forest deriva hiper-rectángulos en el espacio de entrada para resolver el problema.
- Random forest es mejor para datos categóricos y manejan la colinealidad mejor que SVM.

6. Trabajo Futuro

De lo aprendido al momento de realizar este trabajo me queda la enorme utilidad que proporciona el DD-Plot como base para aplicar métodos de clasificación supervisada convencionales. Dicho esto, en el paquete del software R para utilizar la función classif.DD existe una limitación para implementar métodos distintos a los que vienen de forma predeterminada, así como algunos fallos en el uso de medidas de profundidad para datos funcionales multivariados, por lo cual es obligatorio utilizar una de las alternativas propuestas enJuan A. Cuesta-Albertos, 2016. Como consecuencia a lo mencionado previamente, existe un deseo implementar en un futuro las siguientes mejoras:

- Las medidas de profundidad que no funcionan correctamente.
- Más métodos de clasificación supervisada.
- Métricas por defecto distintas a la matriz de confusión y las probabilidades de clasificación de cada clase.
- Comparaciones con métodos no supervisados para datos funcionales.
- Probar los modelos en conjuntos de datos no prefabricados en R.

References

Cuevas A, F. R., Febrero M. (2007). Robust estimation and classification for functional data via projection-based depth notions. *Comput Stat 22(3)*, 481–496.

Delaigle, A., & Hall, P. (2010). Defining probability density for a distribution of random functions. *The Annals of Statistics, 38*, 1171–1193.

Donoho D. L., G. M. (1992). Breaking properties of location estimates based on halfspace depth and projected outlyingness. *Annuals of Statistics*.

Fabrice Rossi, N. V. (2006). Support vector machine for functional data classification. *Neurocomputing, Elsevier, 69 (7-9)*, 730–742.

Ghosh AK, C. P. (2005). On maximum depth and related classifiers. *Scand J Stat 32(2)*, 327–350.

Green PJ, S. B. (1994). Nonparametric regression and generalized linear models: A roughness penalty approach.

J, T. (1975). Mathematics and the picturing of data. *Proceedings of the International Congress of Mathematician*.

J. O. RAMSAY, B. S. (2005). Functional data analysis.

Juan A. Cuesta-Albertos, M. O. d. l. F., M. Febrero-Bande. (2016). Dd$G-classifierinthefunctionalsetting$. *Sociedad de Estadística e Investigación Operativa-Springer*, 119–142.

Jun Li, R. Y. L., Juan A. Cuesta-Albertos. (2012). Dd-classifier: Nonparametric classification procedure based on dd-plot. *Journal of the American Statistical Association*, 737–753.

Liu RY, S. K., Parelius JM. (1999). Multivariate analysis by data depth: Descriptive statistics, graphics. *Ann Stat 27(3)*, 783–858.

Neto, M. R. (n.d.). The concept of depth in statistics.

Pokotylo, O. (2016). Depth- and potential-based supervised learning.

Ramsay JO, D. C. (1991). Some tools for functional data analysis. *BMC Medical Research Methodology*.

Shahid Ullah1, C. F. F. (2013). Applications of functional data analysis: A systematic review. *BMC Medical Research Methodology*.

Tatjana Lange, P. M., Karl Mosler. (2012). Fast nonparametric classification based on data depth. *SEMINAR OF ECONOMIC AND SOCIAL STATISTICS UNIVERSITY OF COLOGNE*.

Xiongtao Dai, F. Y., Hans-Georg M¨uller. (2016). Optimal bayes classifiers for functional data and density ratios.