Práctica 3: Clasificadores Bayesianos

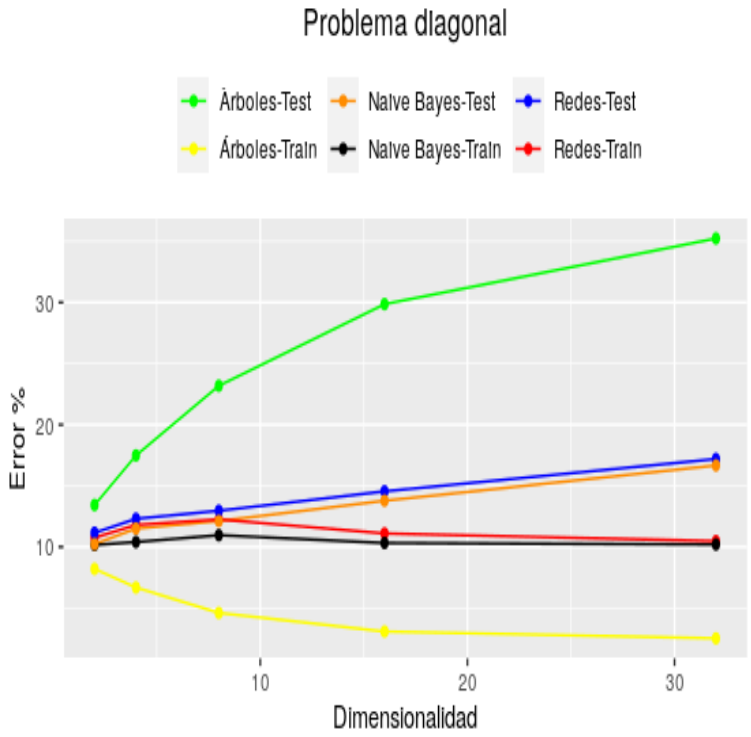
Alumno: Pablo Alonso

1)

Ver código fuente: Ej1/naive_bayes.c.

2)

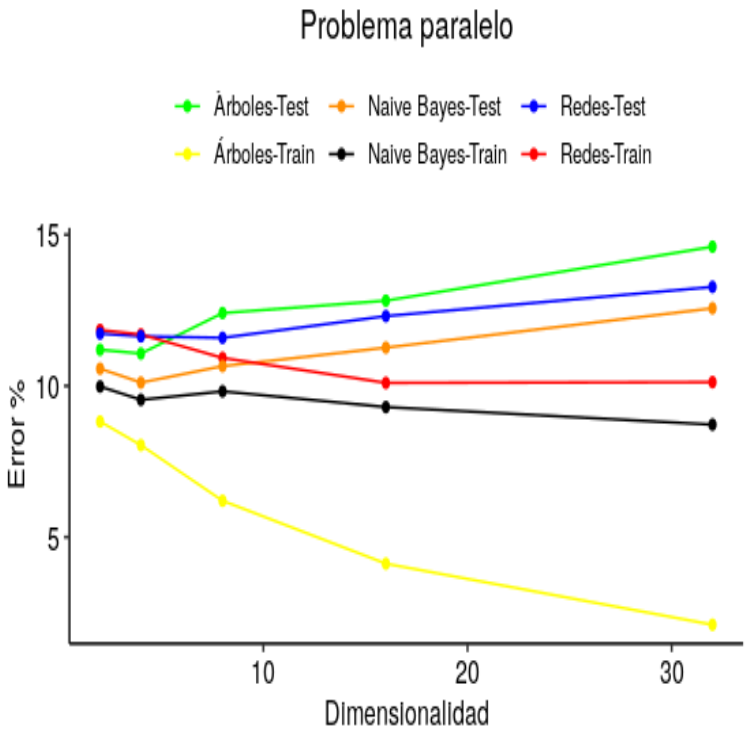
A continuación, se grafican los resultados obtenidos:



Lo primero que se observa es que Naive Bayes no tiene el problema de la dimensionalidad que encontramos en los árboles, ya que considera la probabilidad bajo cada una de las dimensiones de los datos para calcular la probabilidad final de una hipótesis mientras que, como ya hemos visto en prácticas anteriores, los

árboles deben aprender los cortes necesarios en cada una de las dimensiones y para esto necesitan una mayor cantidad de datos.

Se nota que las curvas de train y test de las redes se aproximan bastante. Esto se puede explicar con el hecho de que las redes buscan una hipótesis que reduzca la suma de errores cuadrados y esto, como se explica en Mitchell, es equivalente a encontrar la hipótesis más probable que explique los datos que es lo que hace Naive Bayes. Es decir que intentan resolver el mismo problema. Las curvas de Naive Bayes son aún mejores que las de las redes porque este modelo tiene la ventaja de modelar la distribución subyacente de los datos y los únicos datos que no va a poder clasificar correctamente son aquellos que se encuentra más cerca del centro de la distribución a la que no pertenecen. En otras palabras, los errores de Naive Bayes en este caso surgen en las intersecciones de las dos distribuciones.



En la segunda gráfica se vuelve a observar lo explicado anteriormente. Además se nota que tanto para redes como para Naive Bayes, el problema diagonal y el paralelo son el mismo problema, mientra que para los árboles que ambas distribuciones estén separadas por el eje y causa que el error se duplique.

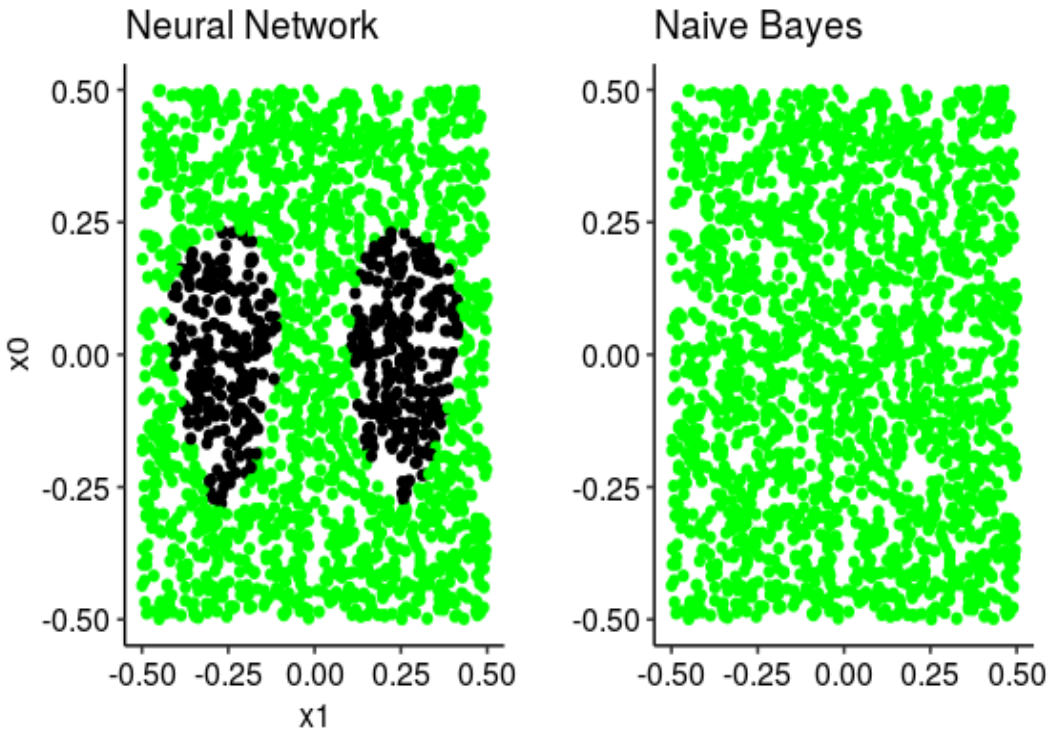
3)

Primero resolvemos el problema de dos_elipses:

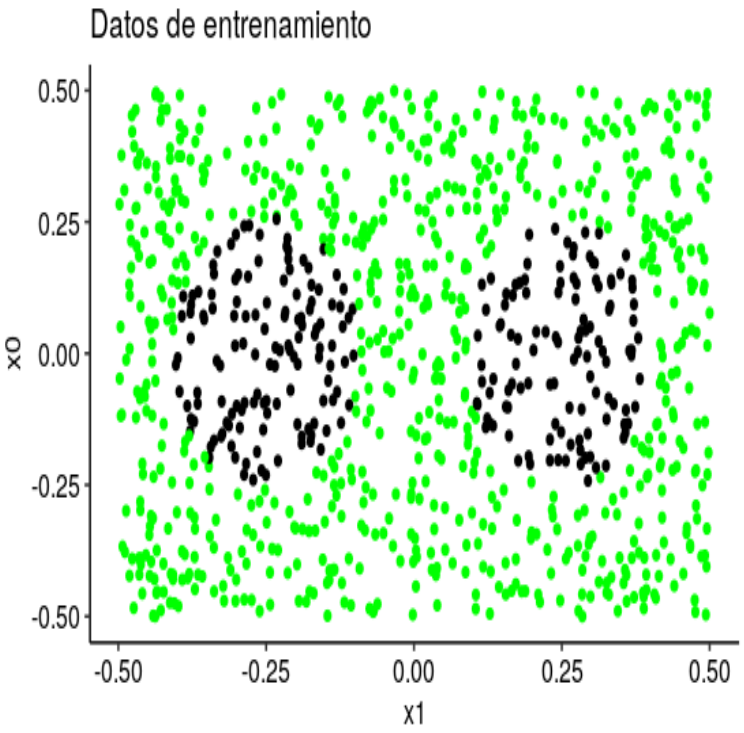
Para correr la red se usaron los siguientes parámetros:

- Learning-rate: 0.01
- Momentum: 0.5
- Nodos en la capa intermedia: 6
- Validación: 20% de los patrones del .data
- Épocas: 40000

Ahora vamos a ver las predicciones hechas por ambos modelos:



Para las redes se alcanza una precisión del 96% y para Naive Bayes apenas un 75%. El problema de Naives Bayes aquí es que ,por un lado, la probabilidad a priori de no pertenecer a las elipses es muy alta comparado con la probabilidad de pertenecer a las mismas. Por otro lado, nuestro algoritmo de Naive Bayes calcula la probabilidad normal de forma independiente para cada feature. Este cálculo se ve afectado por 2 factores: el promedio y la varianza.

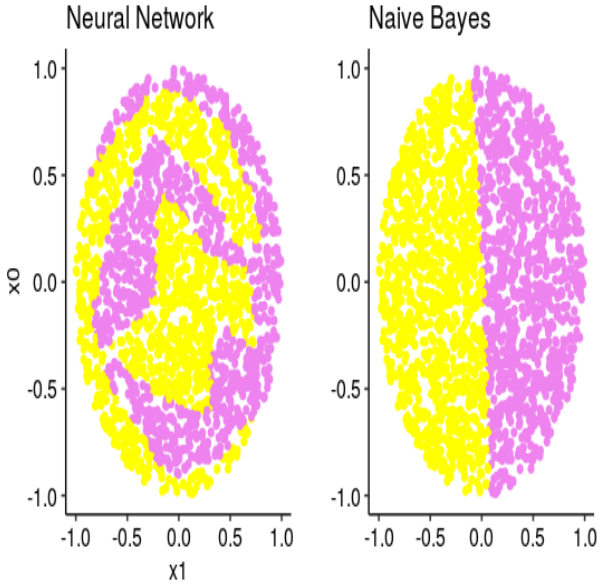


Prestando atención a los datos que usamos para entrenar nuestro modelo Naive Bayes, se puede observar que la media para ambas clases es prácticamente el origen de coordenadas mientras que la varianza en los puntos fuera de las elipses es mayor que la de aquellos que están por adentro. En consecuencia, dado el desbalance que causa la probabilidad a priori y la influencia que la varianza causa la probabilidad normal de no pertenecer a las elipses, causa que todos los puntos se clasifiquen como fuera de las elipses.

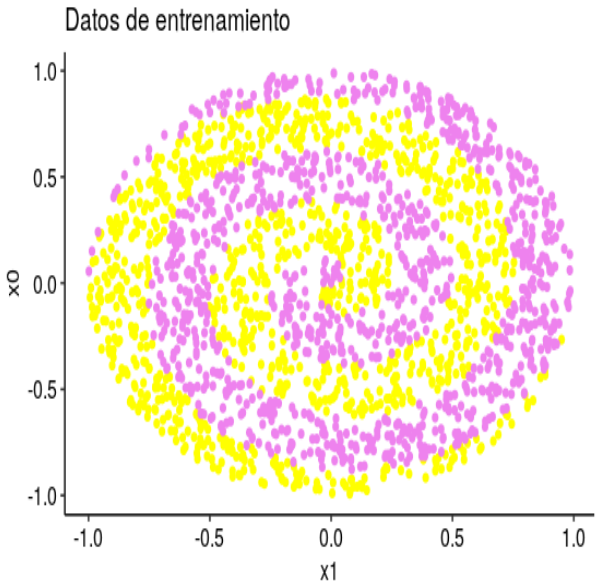
Ahora vamos a resolver el problema de las espirales anidadas.
Para correr la red se usaron los siguientes parámetros:

- Learning-rate: 0.01
- Momentum:0.5
- Nodos en la capa intermedia: 10
- Validación: 20% de los patrones del .data
- Épocas: 40000

En la siguiente gráfica se observan las predicciones realizadas por ambos modelos:



En este caso las redes alcanzan una precisión del 85% mientras que Naive Bayes tiene una precisión del 58 % apróx. Lo que llama la atención es el corte que hace Naive Bayes en el círculo de radio 1. Acá nuevamente la elección del algoritmo se basa en el cálculo de la probabilidad normal.



Nuevamente mirando los datos de entrenamiento, se puede deducir que aunque la varianza en este caso es la misma, el promedio no lo es, ya que

en una zona del círculo la densidad de la clase 0 (puntos amarillos) es mayor que la clase 1 (puntos violetas) y en la otra zona se da el caso contrario. Se deduce entonces que la decisión del algoritmo se ve influenciada en este caso por la distancia a los promedios de cada clase, tomando como clase aquella cuyo promedio es más cercano y produciendo el corte ya mencionado. Dado que la probabilidad a priori es la misma para ambas clases, no resulta en un factor influyente.

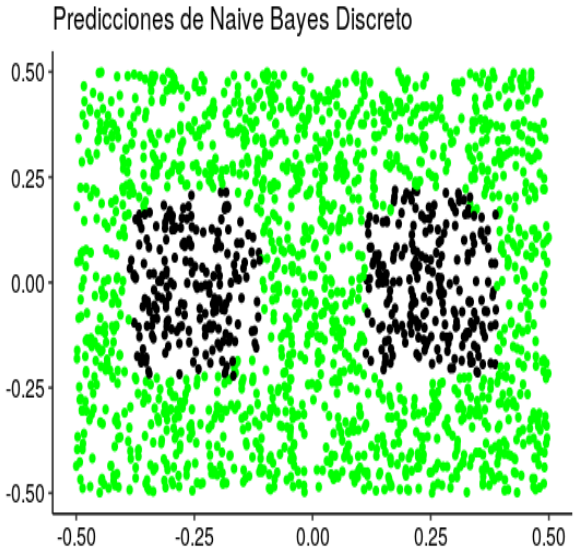
En conclusión de lo que se vio hasta ahora, aproximando con funciones normales, Naive Bayes no funciona bien si la distribución de los datos no es normal sino uniforme como en estos problemas.

4)

Sobre el código:

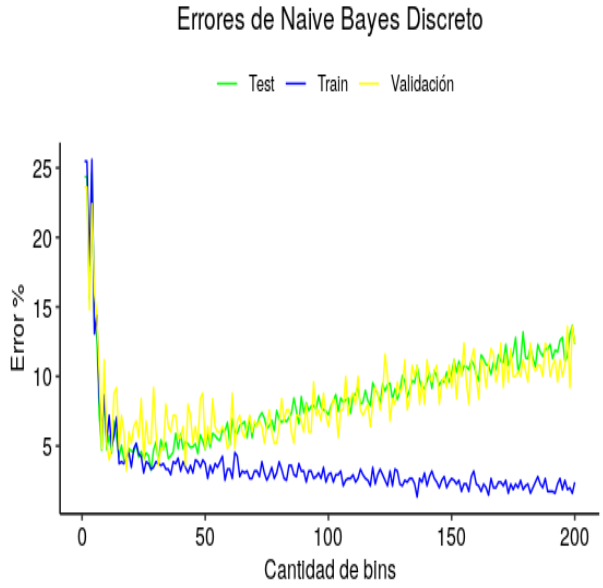
- Ver comentarios en implementación.
- Se usó un 20% de los datos para validación.
- Se asumió que todas las variables tenián la misma cantidad de valores discretos.
- Se agregó un parámetro LÍMITE (máxima distancia al 0 que puede tomar una variable).

Primero se discuten los resultados para dos elipses.

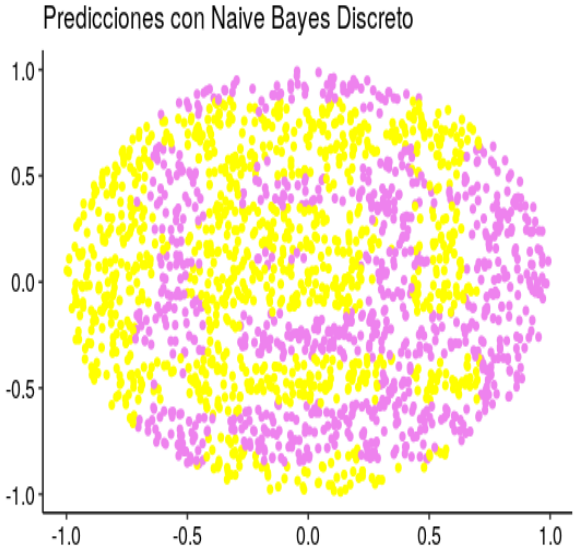


Se logra una precisión del 95%, comparado con el 75% alcanzado en el punto anterior. Al optimizar el número de bins, se obtiene una gran precisión en

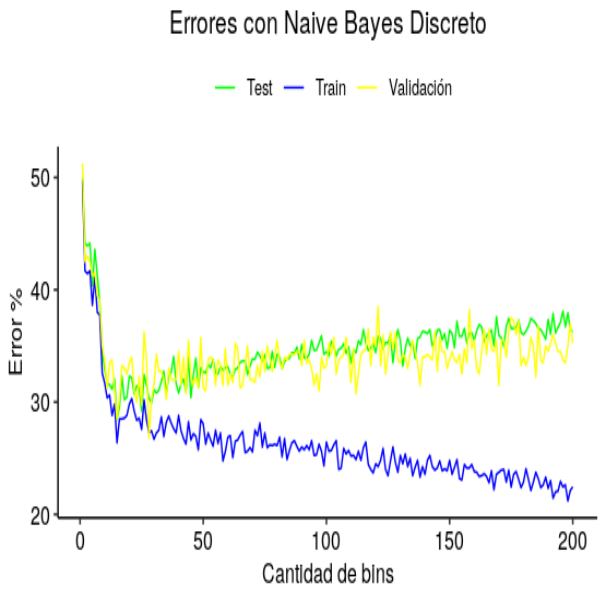
las estadísticas de los histogramas construidos. En vez de usar el cálculo de la función normal en la cual influian el promedio y la varianza , en este caso las elecciones se hacen de acuerdo a estos histogramas, de los cuales se extrae una mejor información, ya que al crear segmentos en las variables continuas se calcula una probabilidad de acuerdo al segmento de la variable y a la clase.



Analizando las curvas de aprendizaje de Naive Bayes Discreto, se observa sobreajuste cuando la cantidad de bins es lo suficientemente grande. Esto quiere decir que al construir histogramas con demasiados intervalos, la probabilidad que el histograma almacena para estos intervalos solo se ajusta a los datos con el cual entrenamos el modelo, aumentando la probabilidad de predecir de forma errónea nuevos datos que no hallan sido vistos antes.

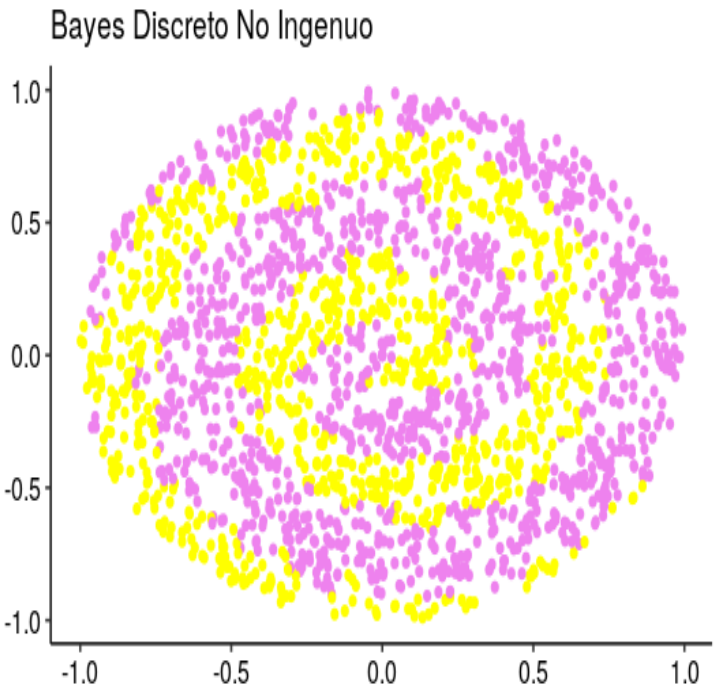


En las espirales se repite el panorama que ya analizamos antes. Naive Bayes Discreto obtiene una precisión del 70% comparado con el 58% obtenido con Naive Bayes Continuo. Cabe destacar que si bien hay una mejora significativa, la discretización pierde eficacia cuando las clases tienen curvas como en este caso. Eso se explica con el hecho de que al hacer particiones en ambas dimensiones, se divide el espacio de puntos en cuadrados, a partir de los cuales hacemos la clasificación. Al hacerlo sobre clases que representan curvas en el espacio hay una obvia pérdida de información dentro de cada cuadrado.

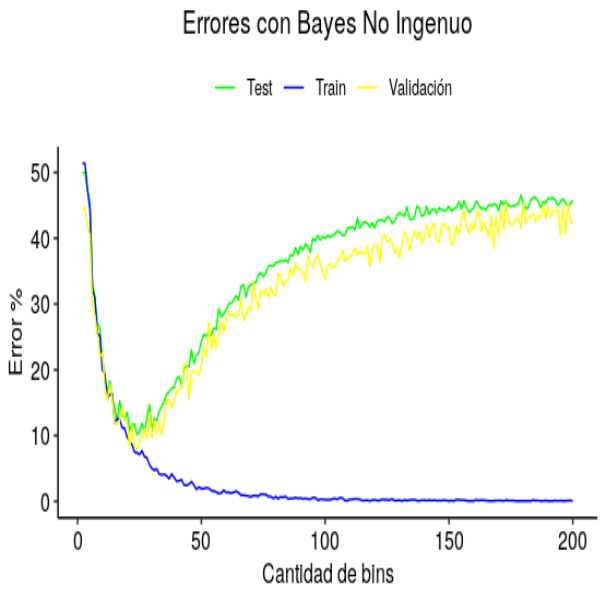


Al analizar los errores, nuevamente vemos el panorama que observamos en 2 elipses: cuando el número de bins es demasiado grande, los cuadrados en el espacio se vuelven demasiado pequeños y se ajustan demasiado a los puntos con los cuales entrenamos el modelo, por lo que no clasifican correctamente puntos que no sean los ya mencionados.

5)



La precisión alcanzada con este versión del algoritmo de Bayes es entre 87% y 90%. El avance que hicimos para lograr esta muy significativa mejora fue medir estadísticamente los valores discretos de las variables como haciamos antes, pero no solo de acuerdo a las clases sino considerando todos los atributos al mismo tiempo. La ventaja con respecto al método anterior, es que se guarda la información de todos los aspectos del problema. Es decir, al fijar un valor discreto antes solo considerábamos el valor que tomaba la clase, ahora consideramos todos los posibles valores que toman el resto de los atributos y las clases también, guardando una probabilidad para cada caso particular. La ganancia de esto es que nuestro modelo se flexibiliza y, como vemos en la gráfica, con esto se supera bastante el problema de las curvas que surge cuando consideramos que los atributos son independientes.

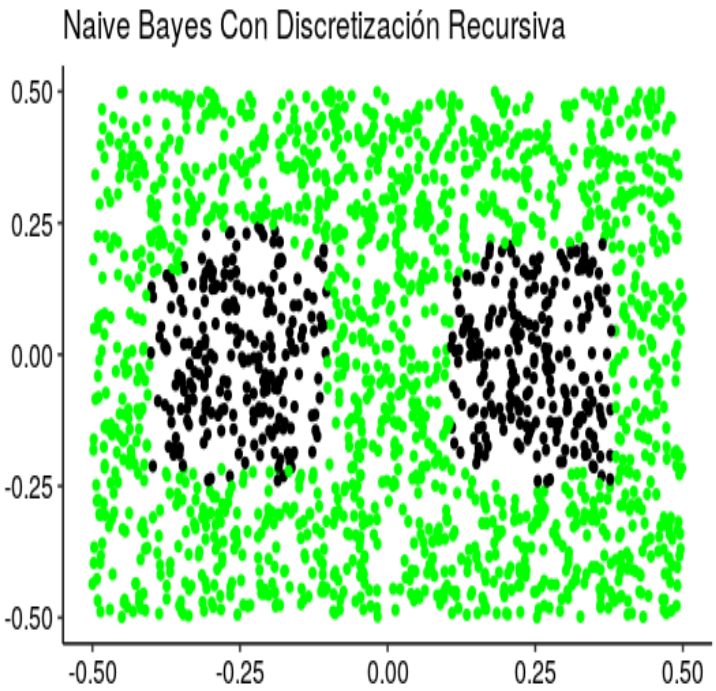


En las curvas de aprendaje se observa mayor sobreajuste aún que en el ejercicio anterior. Esto también es esperable pues medimos más aspectos que en el ejercicio anterior.

Un problema de implementación puede ser como elegir representar una combinación de valores discretos que condicionen a un determinado feature. Por ejemplo para $\mathrm{P}(X_1/X_2,..X_n,h)$, ¿cómo representar las posibles combinaciones de valores de $X_1..X_n,h$? Si se implementa un modelo que indexe por cada uno de los valores discretos de cada variable, se puede obtener un modelo que solo sirva para cierta dimensionalidad. Lo que hice en mi caso fue asumir que todas las variables tenian un número fijo de valores discretos posibles y pensar $X_2..X_nh$ como una cifra en una base de ese número, calcular su equivalente en base decimal y usarlo para hacer la última indexación en un arreglo de 3 dimensiones. De esta forma se soluciona el problema de la indexación.

Otro problema es la cantidad de veces que se necesita recorrer los patrones de entrenamiento para poder calcular las probabilidades condicionales (1 recorrido por cada atributo). Para problemas de grandes dimensiones esto puede resultar muy lento.

6)



El resultado obtenido tuvo una presición del 96.3% superando lo que habíamos logrado en ejercicios anteriores. La mejora que hace este algoritmo de discretización es dividir los rangos de valores de cada variable de forma que cada segmento tenga la menor incertidumbre posible. De esta forma la probabilidad calculada para cada valor discreto que toma un feature es más pura porque hay menos pérdida de información.

Llama la atención que usar Bayes No Ingenuo con discretización recursiva dé malos resultados.