Grado en Estadística

TRABAJO FIN DE GRADO

Aplicación de técnicas de clasificación a la detección de cáncer

Ignacio Cazorla Piñar

Sevilla, Junio de 2019

Índice general

Resumen

En este Trabajo Fin de Grado se realiza un estudio comparativo de diversos métodos de clasificación estadística, tanto desde el punto de vista teórico como aplicado.

La memoria se estructura en 3 capítulos. En el Capítulo 1 se realiza una breve introducción a las técnicas de machine learning, centrándonos en las técnicas de clasificación. Distinguimos entre técnicas paramétricas y no paramétricas.

En el Capítulo 2, se realiza una revisión metodológica de algunos de los más importantes clasificadores. Comenzamos con el estudio de los paramétricos: regresión logística y análisis discriminante. En el caso de la regresión logística, introducimos el modelo, estimación de los coeficientes, y realización de predicciones tanto en el caso simple como múltiple. En cuanto al Análisis Discriminante Lineal (LDA), este método se introduce como un clasificador basado en el Teorema de Bayes, y se trata tanto el caso de uno como de varios predictores. A continuación, recogemos el método de clasificación basado en las Máquinas de Véctor Soporte (SVM). Destacamos que es un método no paramétrico, en el que el problema de clasificación se reduce a un subconjunto potencialmente pequeño de las observaciones disponibles en el conjunto de entrenamiento. Frente a los clasificadores paramétricos, las máquinas de véctor soporte resultan se bastante robustos. Para finalizar el Capítulo 2, se recogen medidas para evaluar la calidad del clasificador aplicado: tasa de error y entrenamiento, equilibrio entre sesgo y varianza del modelo, métodos de remuestreo basadas en técnicas de validación cruzada, métodos de evaluación y selección del modelo, y medidas específicas de clasificación como son la sensibilidad, especificidad, curva ROC, y AUC.

En el Capítulo 3, se aplican los métodos y medidas anteriores al conjunto de datos Wisconsin, sobre diagnóstico de cáncer de mama, y que se encuentran disponibles en Kaggle. Se realiza un estudio descriptivo de estos datos, se detectan outliers, y se aplican métodos de selección de variables, para quedarnos con aquellas con mayor poder discriminatorio. Los datos se dividen en conjunto de entrenamiento y test. A ellos se les aplicarán los distintos clasificadores: regresión logística, análisis discriminante lineal, y máquinas de véctor soporte. Se obtienen y comparan las medidas de precisión obtenidas en ellos. El análisis estadístico se ha realizado utilizando el lenguaje y librerías de R.

Abstract

In this work a comparison of different statistical classification methods is carried out. Theoretical results and applications are given.

The work is divided in three chapters. In Chapter 1, machine learning and classification techniques are introduced. We distinguish between parametric and non-parametric methods.

In Chapter 2, a methodological review of most relevant classifiers is given. First, parametric methods are considered: logistic regression and linear discriminant analysis. As for logistic regression, the model is introduced, estimators of the coefficients, predictions for simple and multiple setting are studied. Second, Linear Discriminant Analysis (LDA) is introduced as a classifier based on Bayes theorem, results for one and several predictors are given. Next, classification methods based on Support Vector Machine (SVM) are studied. This is a nonparametric approach, in which the classification problem is reduced to a really small subset of data available in the training set. Support vector machines are more robust methods than the parametric ones. To conclude Chapter 2, measures to evaluate the quality of a classifier are given. These are: training and error rate, balance between bias and variance in a model, resampling methods based on cross validation, methods to evaluate and select a model, and tailored measures of classification such as sensitivity, specifity, ROC curve and AUC.

In Chapter 3, the previously methods and measures are applied to Wisconsin dataset, available at Kaggle. A descriptive study is carried out, techniques to detect outliers are applied, and methods to select predictor variables are considered in order to keep those explanatory variables with greater discriminatory power. The dataset is split into training and test set. The different classification methods, previously introduced, are applied, that is, logistic regression, LDA and SVM. The measures of quality of these classifiers are obtained. Comparison between them are given. R and libraries of this software have been used in our study.

Índice de figuras

Índice de cuadros

Capítulo 1

Machine learning para la detección de cancer.

1.1. Introducción.

El cáncer de mama es el más común entre las mujeres y una de las principales causas de muerte entre las mujeres en todo el mundo. Cada año, aproximadamente, 124 de cada 100,000 mujeres son diagnosticadas con cáncer de mama, y la estimación es que 23 de las 124 mujeres diagnosticadas morirán de esta enfermedad.
Cuando se detecta en sus primeras etapas, hay un 30 % de probabilidades de que el cáncer se pueda tratar de manera efectiva, pero la detección tardía de los tumores en etapa avanzada hace que el tratamiento sea más difícil. Actualmente, las técnicas más utilizadas para detectar el cáncer de mama en estadios tempranos son: mamografía (63 % a 97 % de precisión), FNA (aspiración con aguja fina) con interpretación visual (65 % a 98 % de precisión) y biopsia quirúrgica (aproximadamente 100 % de exactitud). Por lo tanto, la mamografía y la FNA con corrección de la interpretación visual varían ampliamente, y la biopsia quirúrgica, aunque fiable, es invasiva y costosa.
En este trabajo se analizará una técnica de diagnóstico que utiliza la FNA con interpretación computacional mediante aprendizaje automático y tiene como objetivo crear un clasificador que proporcione un alto nivel de precisión, con una tasa baja de falsos negativos.

La aspiración con aguja fina (FNA) es un procedimiento de diagnóstico que se usa para investigar bultos o masas. En esta técnica, se inserta una aguja delgada en la masa para tomar muestras de las células que, después de teñirse, se examinarán bajo un microscopio (biopsia). La muestra y la biopsia consideradas juntas se llaman biopsia por aspiración con aguja fina (FNAB) o citología por aspiración con aguja fina (FNAC). Las biopsias por aspiración con aguja fina son procedimientos quirúrgicos menores muy seguros. A menudo, se puede evitar una biopsia quirúrgica importante (por escisión o abierta) realizando una biopsia por aspiración con aguja, eliminando la necesidad de hospitalización, lo cual lo convierte en una técnica bastante más ventajosa en comparación al resto de técnicas mencionadas previamente, ya que es más segura, poco costosa y menos traumática para el paciente.

Actualmente, hay bastantes estudios tratando de lograr el mejor rendimiento para interpretación computacional para muestras obtenidas por FNA. En este trabajo, usa-

ramos 3 técnicas de clasificación para realizar dicha tarea: *Regresión logística*, *Análisis discriminante lineal (LDA)* y *Máquinas del vector soporte (SVM)*. Estos dos primeros métodos respectivamente, los distinguiremos cómo paramétricos mientras que este último como no paramétrico.

Capítulo 2

Técnicas de clasificación estadística.

En muchas situaciones reales disponemos de variables independientes, predictores, covariables o atributos a los que denotaremos por $X_1, X_2, ..X_p$. Estas variables son conocidas y las utilizaremos para predecir el valor de otra (u otras), a las que denotaremos por Y. La variable, Y, relacionada con las anteriores, se conoce como variable respuesta, dependiente u objetivo. Por simplicidad suponderemos sólo una variable objetivo: Y. Las covariables $\underline{X} = X_1, X_2, ..X_p$, y la variable objetivo Y están relacionadas, y supondremos que dicha relación se puede modelizar de forma aproximada por una función $f()$. Específicamente

$$Y = f(\underline{X}) + \epsilon$$

ϵ es un término de error aleatorio, independientes de las $\underline{X}'s$, y cuyo valor esperado es nulo.

$f(\circ)$: representa la información sistemática que las covariables $\underline{X}$ nos proporcionan sobre Y.
Las técnicas de aprendizaje estadístico nos proporcionan métodos para estimar f.
La estimación de f, $\widehat{f}(\underline{X})$, nos servirá, bien para predecir valores de la variable dependiente Y, o bien para inferir relaciones entre las covariables $\underline{X}$ e Y.

2.1. ¿Cómo estimaremos f?

A lo largo de este trabajo, exploramos varios enfoques lineales y no lineales para estimar f, es decir, una función que nos permita realizar predicciones. Sin embargo, estos métodos generalmente comparten ciertas características. En esta sección, proporcionamos una descripción general de las características comunes. Siempre asumiremos que tenemos un conjunto de datos formado por n puntos diferentes. Estas observaciones se denominan *conjunto o datos de entrenamiento* porque las usaremos para entrenar, o enseñar, a nuestro método como estimar f. Sea x_{ij} el valor de nuestro j-ésimo predictor para la i-ésima observación i, donde $i = 1, 2, ..., n$ y $j = 1, 2, ...p$. De forma análoga, sea y_i la variable respuesta para la i-ésima observación. Entonces nuestros datos de entrenamiento están formados por $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$ donde $x_i = (x_{i1}, x_{i2}, ..., x_{ip})^T$.
Nuestro objetivo es aplicar un método de aprendizaje estadístico a los datos de entrenamiento con el fin de estimar la función desconocida f. En terminos generales, la mayoría de métodos de aprendizaje estadístico para esta tarea se pueden clasificar como *paramétricos* o *no paramétricos*. Ahora discutiremos brevemente estos dos tipos de enfoques.

2.1.1. Métodos paramétricos versus no paramétricos.

Métodos paramétricos

Los métodos paramétricos implican un enfoque basado en un modelo que se construye en dos etapas.

1. Primero, estableceremos una hipótesis sobre la forma funcional o forma de f. Por ejemplo: una caso bastante simple es que f sea lineal en $\underline{X}$ siendo $\underline{X} = (X_1, ..., X_p)$

$$f(\underline{X}) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p.$$

   Este es el modelo lineal.

2. Después de que un modelo haya sido propuesto, necesitaremos un procedimiento que use nuestro *conjunto de entrenamiento* para ajustar el modelo. En el caso del modelo lineal, queremos encontrar valores para sus parámetros tales que:

$$Y \approx \widehat{\beta}_0 + \widehat{\beta}_1 X_1 + \widehat{\beta}_2 X_2 + ... + \widehat{\beta}_p X_p.$$

El modelo basado en este enfoque que se acaba de describir se conoce como paramétrico. Reduce el problema de estimar f a uno de estimar un conjunto de parámetros. Suponer una forma paramétrica para f simplifica el problema de estimar f, porque generalmente es mucho más fácil estimar un conjunto de parámetros, que ajustar una función completamente arbitraria. Sin embargo, la desventaja potencial de los modelos paramétricos es que el modelo que elegimos generalmente no coincidirá con la verdadera forma desconocida de f.

Métodos no paramétricos

Los métodos no paramétricos no hacen suposiciones explícitas sobre la forma funcional de f. En su lugar, buscan una estimación de f que se acerque lo más posible a los puntos de datos sin ser demasiado brusca. Tales enfoques pueden tener una gran ventaja sobre los enfoques paramétricos: al evitar el supuesto de una forma funcional particular para f, tienen el potencial de ajustarse con precisión a un rango más amplio de formas posibles para f. Cualquier enfoque paramétrico trae consigo la posibilidad de que la forma funcional utilizada para estimar f sea muy diferente de la verdadera f, en cuyo caso el modelo resultante no se ajustará bien a los datos. En contraste, los enfoques no paramétricos evitan completamente este peligro, ya que esencialmente no se hace ninguna suposición sobre la forma de f, pero los enfoques no paramétricos tienen una gran desventaja: ya que no reducen el problema de estimar f a un pequeño número de parámetros, se requiere un número muy grande de observaciones con el fin de obtener una estimación precisa de f.

2.2. Regresión logística.

Para ilustrar este primer punto, consideraremos el directorio de trabajo **Credit Card Default** en el que tenemos un conjunto de datos simulados que contiene información sobre 10.000 clientes. Estaremos interesados, en saber si un individuo cometerá un fraude o no en su pagos con tarjeta de crédito, a partir de las siguientes variables respuesta:

- *Default*: Un factor con niveles Sí y No que indica si el cliente incumplió con su deuda.
- *Student*: Un factor con niveles Sí y No que indica si el cliente es estudiante o no.
- *Balance*: El saldo promedio que el cliente tiene en su tarjeta de crédito después de realizar su pago mensual
- *Income*: Los ingresos anuales del cliente.

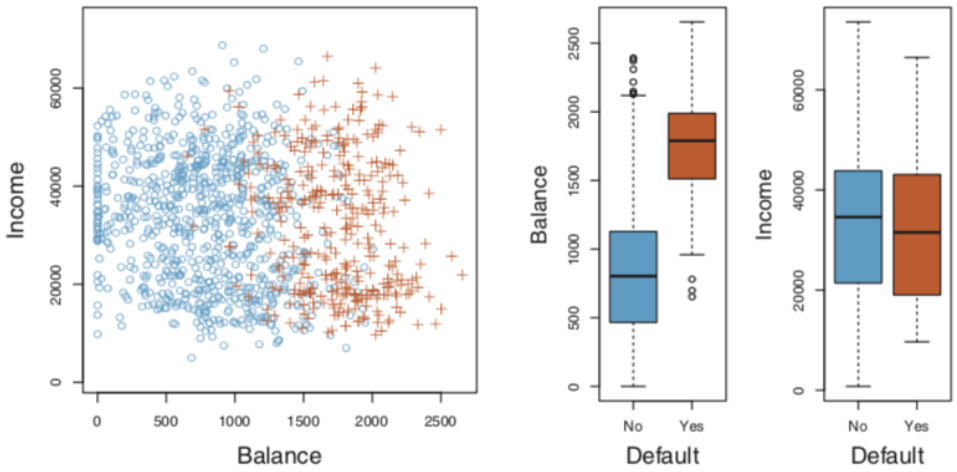

Figura 2.1: El conjunto de datos Default.

El conjunto de datos se muestra en la Figura 1.1. Hemos realizado un gráfico a partir del ingreso anual y el saldo mensual de la tarjeta de crédito. El gráfico del panel izquierdo en la Figura 1.1 muestra los individuos que cometieron fraude en un determinado mes en naranja, y aquellos que no en azul. La tasa general de fallo es del 3 %. En este gráfico, podemos observar que los individuos que cometen fraude tienden a tener mayores saldos de tarjeta de crédito que aquellos que no defraudaron.

En el gráfico del panel derecho de la Figura 1.1, se muestran dos pares de diagramas de caja. En el primero de muestra la distribución de *balance* dividido por la variable binaria fraude; el segundo es un gráfico similar para la variable ingresos *income*. En estos se observa que *Balance* discrimina mejor entre grupos que *income*, por lo que será más útil para realizar predicciones.

Cabe destacar que en la Figura 1.1 se muestra una relación muy pronunciada entre las variable predictora balance y la variable respuesta *default*. En la mayoría de casos reales, la relación entre la variables predictoras y la variables respuesta no será tan pronunciada. Sin embargo, con el fin de ilustrar de manera clara los pasos a seguir durante la clasificación, usaremos un ejemplo en el que la relación entre este tipo de variables es un tanto exagerada.

2.2.1. El modelo logístico simple.

La regresión logística es una extensión del modelo de regresión lineal pero en este caso, se modeliza la probabilidad de que Y pertenezca a una categoría particular de dos categorías existentes, cabe destacar que en el caso del modelo logístico simple consideramos una sola variable explicativa. Por tanto, se debe modelizar $p(X)$ usando una función con la que obtengamos resultados entre 0 y 1 para todos los valores de X. Luego, en el modelo de regresión logística simple, en el que sólo se considera una variable explicativa X vendrá dado por:

$$p(X) = \frac{e^{(\beta_0+\beta_1 X)}}{1 + e^{(\beta_0+\beta_1 X)}} \tag{2.1}$$

Para ajustar el modelo 2.1 se usa el método de *máxima verosimilitud*.

Después de manipular un poco 2.1, obtenemos que:

$$\frac{p(X)}{1 - p(X)} = e^{(\beta_0+\beta_1 X)} \tag{2.2}$$

La cantidad $\frac{p(x)}{1-p(x)}$ se denomina *odds* y puede tomar cualquier valor entre 0 e infinito. Valores del odds cercano a 0 e infinito indica valores muy bajos o muy altos respectiva y relativamente, de la probabilidad de que ocurra el suceso que se quiere predecir.

Tomando el logaritmo en 2.2, se tiene que:

$$\log\left(\frac{p(X)}{1 - p(X)}\right) = \beta_0 + \beta_1 X \tag{2.3}$$

En el modelo de regresión lineal simple , β_1 proporciona el cambio esperado en Y asociado al incremento de una unidad en X. Sin embargo, en un modelo de regresión logística, el incremento de una unidad en X cambia el *log odds* en β_1 como puede verse en (2.3). Debido a que la relación entre $p(X)$ y X dada en (2.2) no es una lineal, β_1 no corresponde al cambio en $p(X)$ asociado al incremento de una unidad en X. La cantidad que $p(X)$ cambia debida al incremento de una unidad en X dependerá del valor actual de X. Pero independientemente del valor de X, si β_1 es positivo, el aumento de X se asociará con el aumento de p(X), y si β_1 es negativo, el aumento de X se asociará con la disminución de $p(X)$.

2.2.2. Estimación de los coeficientes de regresión.

Los coeficientes β_0 y β_1 en (2.2) son desconocidos, y deben estimarse basándonos en el conjunto de *entrenamiento disponible.*

Se intentará estimar β_0 y β_1 de manera que al incluir estas estimaciones en el modelo para $p(x)$ en (2.1), se obtenga un número cercano a 1 para aquellos individuos que defraudaron, y un número cercano a 0 para aquellos individuos que no lo hicieron. Esta intuición, puede formalizarse usando la función de verosimilitud:

Cuadro 2.1: Para el conjunto de datos Default, los coeficientes estimados de la regresión logística que predice, la probabilidad de impago utilizando Balance.

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-10.6513306	0.3611574	-29.49221	0
balance	0.0054989	0.0002204	24.95309	0

$$\ell(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i':y_{i'}=0} (1 - p(x_{i'})) \tag{2.4}$$

La estimaciones β_0 y β_1 se eligen de forma que se maximice la función de verosimilitud.

En la Tabla 2.1 se muestran los coeficientes estimados y la información relacionada que resulta de ajustar un modelo de regresión logística en el conjunto de datos *Default* con el fin de predecir la probabilidad de *default=yes* usando la variable predictora *balance*. Observamos que $\beta_1 = 0.0055$; lo cual indica que el aumento en *balance* esta asociado con un aumento en la probabilidad de fraude. Para ser precisos, el aumento de una unidad en balance esta asociado al aumento en el log odds de fraude en 0.0055 unidades.

Muchos aspectos de la regresión logística mostrados en la Tabla 2.1 son similares a los de la regresión lineal. Por ejemplo, podemos medir la precisión de los coeficientes estimados obteniendo sus errores estándares. Destacamos el contraste

$$H_0 : \beta_1 = 0$$
$$H_1 : \beta_1 \neq 0.$$

Valores (absolutos) grandes del estadístico indican evidencias contra la hipótesis nula $H_0 : \beta_0 = 0$. Como el p-valor asociado con *balance* en la Tabla 2.1 es pequeño, podemos decir que probabilidad de *default* está relacionada con la variable predictora *balance*. En otras palabras, podemos concluir que existe relación positiva ($\beta_1 > 0$) entre el aumento de *balance* y la probabilidad de defraudar *Default*.

2.2.3. Predicciones.

Una vez los coeficientes han sido estimados, es posible calcular la probabilidad de *default* para cualquier valor de la variable *credit card balance*. Por ejemplo, a partir de la variable *student*, la cual indica si el estatus de un individuo es de estudiante o no. Luego para ajustar el modelo, simplemente crearemos una variable dummy que puede tomar los valores 1 si es estudiante ó 0 si no es estudiante. El modelo de regresión logística resultante para predecir la probabilidad de *default* del estatus *estudiante* puede verse en el Cuadro 2.2. El coeficiente asociado con la variable es positivo, y el p-valor asociado es estadísticamente significativo. Esto indica que los estudiantes tienden a tener mayor probabilidad de fraude que los no estudiantes:

$$Pr(default = yes|student = Yes) = \frac{e^{-3.5041+0.4048\times1}}{1 + e^{-3.5041+0.4048\times1}} = 0.0431,$$
$$Pr(default = yes|student = No) = \frac{e^{-3.5041+0.4048\times0}}{1 + e^{-3.5041+0.4048\times0}} = 0.0292,$$

Cuadro 2.2: Para el conjunto de datos Default, los coeficientes estimados de la regresión logística que predice, la probabilidad de impago a partir del status Student.

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-3.5041278	0.0707130	-49.554219	0.0000000
studentYes	0.4048871	0.1150188	3.520181	0.0004313

2.2.4. Regresión logística múltiple.

Consideremos ahora el problema de una variable respuesta binaria usando múltiples predictores. Podemos generalizar (2.3) de la siguiente manera

$$\log\left(\frac{p(\underline{X})}{1-p(\underline{X})}\right) = \beta_0 + \beta_1 X_1 + ... + \beta_p X_p \tag{2.5}$$

donde $X = (X_1, X_2, ..., X_p)$ son p variables predictoras. La ecuación (2.5), puede también expresarse como:

$$p(\underline{X}) = \frac{e^{(\beta_0+\beta_1 X_1+...+\beta_p X_p)}}{1+e^{(\beta_0+\beta_1 X_1+...+\beta_p X_p)}} \tag{2.6}$$

Al igual que en la sección previa, se usa el método de máxima verosimilitud para estimar $\beta_0, \beta_1, ..., \beta_p$.

2.2.5. Regresión logística para más de dos clases respuesta.

A veces deseamos clasificar una variable respuesta que tiene más de dos clases. Los modelos de regresión logística para variables respuesta con dos clases discutidos en las secciones anteriores tienen extensiones para múltiples clases, pero en la práctica tienden a no usarse con tanta frecuencia. Una de las razones es que el método que discutimos en la siguiente sección, *Análsis discriminante lineal*, es popular para la clasificación de múltiples clases. Por lo tanto, no profundizaremos en los detalles de la regresión logística de múltiples clases en este trabajo, sino que simplemente observamos que tal enfoque es posible y que el software para ello está disponible en R.

2.3. Análisis discriminante lineal (LDA).

La cuestión que nos planteamos ahora es la siguiente: ¿ Por qué necesitamos otro método si ya tenemos uno que efectúa buenas previsiones cómo lo es la regresión logística?. Existen varias razones:

- Cuándo las clases están bien separadas, las estimaciones de los parámetros para el modelo de regresión logística son sorprendentemente inestables. El análisis discriminante lineal no sufre este problema.
- Si el tamaño muestral es pequeño y la distribución de las variables predictoras $\underline{X}$ es aproximadamente normal en cada una de las clases, el análisis discriminante lineal es más estable que el modelo de regresión logística.
- Además, el análisis discriminante lineal es más popular cuando tenemos más de dos clases en la variable respuesta como ya comentamos previamente.

Ahora bien, en este diferente enfoque, modelizaremos la distribución de las variables predictoras $\underline{X}$ separadamente en cada una de las clases de la variables respuesta Y y posteriormente usaremos el *Teorema de Bayes* para convertirlas en estimaciones para $Pr(Y = k|\underline{X} = \underline{x})$. En el caso en que estas distribuciones son normales, resulta que el modelo es muy similar en forma al de regresión logística.

2.3.1. Uso del teorema de Bayes para la clasificación.

Supongamos que queremos clasificar una observación en una de las K clases pertenecientes a la variable respuesta con $K > 1$. Sea π_k la probabilidad previa general o a *priori* de que una nueva observación elegida al azar provenga de la k-ésima clase ; esto es la probabilidad de que una observación dada esté asociada con la k-ésima categoría de la variable respuesta Y. Sea $f_k(\underline{x}) = Pr(\underline{X} = \underline{x}|Y = k)$ la función de densidad de $\underline{X}$ para una observación procedente de la k-ésima categoría. En otras palabras, $f_k(\underline{x})$ será relativamente grande si hay una alta probabilidad de que una observación en la k-ésima clase sea $\underline{X} = \underline{x}$, y $f_k(\underline{x})$ es pequeño si es poco probable de que una observación en la k-ésima clase sea $\underline{X} = \underline{x}$. El teorema de Bayes establece que:

$$Pr(Y = k|\underline{X} = \underline{x}) = \frac{\pi_k f_k(\underline{x})}{\sum_{l=1}^{k} \pi_l f_l(\underline{x})} \tag{2.7}$$

En general, estimar π_k es fácil si tenemos una muestra aleatoria de $Y's$ de la población. Simplemente calculamos la fracción de las observaciones del conjunto de entrenamiento que pertenecen a la k-ésima clase. Sin embargo, estimar $f_k(\underline{x})$ tiende a ser más desafiante, a menos que supongamos algunas formas sencillas para estas densidades. Denotaremos como $p_k(\underline{x})$ a la probabilidad a *posteriori* de que una observación $X = x$ pertenezca a la k-ésima clase. El *Clasificador de Bayes*, clasifica una observación en la clase para la cual la probabilidad a posteriori $p_k(\underline{x})$ es mayor.Este clasificador tiene la tasa de error más baja entre todos los clasificadores, por supuesto, siempre y cuando todos los términos que aparecen en (2.7) estén correctamente especificados.

2.3.2. Análisis discriminante lineal para $p = 1$.

Considerando el caso de una sola variable predictora X que se supone continue, siendo p el número de variables predictoras. Bajo la hipótesis de que $f_k(x)$ es normal o Gaussiana. En la configuración unidimensional, la densidad normal toma la siguiente forma:

$$f_k(x) = \frac{1}{\sqrt{2\pi}\sigma_k} exp(-\frac{1}{2\sigma_k^2}(x-\mu_k)^2) \tag{2.8}$$

donde μ_k y σ_k^2 son la media y la varianza de los parámetros para la k-ésima clase. Además, suponemos también que $\sigma_1^2 = ... = \sigma_k^2$. Introduciendo (2.8) en (2.7) obtenemos que

$$p_k(x) = \frac{\pi_k f_x(k)}{\sum_{l=1}^{k} \pi_l f_l(x)} \tag{2.9}$$

.

El Clasificador de Bayes asignará una observación, $X = x$ a la clase para la cual (2.9) es mayor.

En algunas situaciones, no podemos calcular el clasificador de Bayes. En la práctica, incluso si estamos bastante seguros de nuestra hipótesis de que X se extrae de una distribución Gaussiana dentro de cada clase, todavía tenemos que estimar los parámetros $\mu_1, ..., \mu_k$, $\pi_1, ..., \pi_k$ y σ^2. En general, se utilizan las siguientes estimaciones para dichos parámetros

$$\widehat{\mu}_k = \frac{1}{n_k} \sum_{i:y_i=k} x_i.$$

$$\widehat{\sigma}_2 = \frac{1}{n-K} \sum_{k=1}^{K} \sum_{i:y_i=k} (x_i - \mu_k)^2.$$

En cuanto a la estimación de π_k, señalar que en algunas ocasiones se tiene conocimiento previo de las proporciones en las que se presentan cada una de las clases, y éstas se usan directamente. Si esto no ocurre, se suelen estimar por la proporción en que se presentan en el conjunto de entrenamiento.

$$\widehat{\pi}_k = n_k/n.$$

Los estimadores anteriores se sustituyen en 2.9. Se toma el logaritmo y operando se tiene que maximizar $\log p_k(x)$ que es equivalente a maximizar:

$$\delta_k(x) = x\frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k)$$

lineal en x??

Para reiterar, el clasificador de LDA opera bajo la hipótesis de que las observaciones de dentro de cada clase provienen de una distribución normal con un vector de media específico de clase y una varianza común σ^2, y agregando dichas estimaciones para estos parámetros en el clasificador de Bayes.

2.3.3. Análisis discriminante lineal para p>1.

Ahora extenderemos el clasificador LDA al caso de múltiples predictores. Para hacer esto, supondremos que $\underline{X} = (X_1, X_2, ..., X_p)$ se extrae de una distribución gaussiana multivariante, con un vector de medias específico para cada clase y una matriz de covarianzas común.

Formalmente, la densidad gaussiana multivariante se define como

$$f_k(x) = \frac{1}{2\pi^{(p/2)}|\Sigma|^{1/2}} exp\left(-\frac{1}{2}(\underline{x} - \underline{\mu})^T \Sigma^{-1}(\underline{x} - \underline{\mu})\right) \tag{2.10}$$

Para el caso de $p > 1$ predictores, como ya hemos comentado previamente, el clasificador LDA supone que las observaciones en la clase k-ésima se extraen de una distribución gaussiana multivariante $N(\underline{\mu}_k, \Sigma)$, dónde $\underline{\mu}_k$ es el vector de medias específico de clase, y Σ es la matriz de covarianzas común para las k clases. Introduciendo la función para la $k - ésima$ clase, $f_k(\underline{X} = \underline{x})$ en (1.7), obtendremos que el clasificar LDA asigna una observación $\underline{X} = \underline{x}$ a la clase para la cual

$$\delta_k(\underline{x}) = \underline{x}^T \Sigma^{-1} \underline{\mu}_k - \frac{1}{2} \underline{\mu}_k^T \Sigma^{-1} \underline{\mu}_k + \log \pi_k \tag{2.11}$$

es mayor.

Una vez más, necesitamos estimar los parámetros desconocidos $\underline{\mu}_1, ..., \underline{\mu}_k, \pi_1, ..., \pi_k$ y $\underline{\Sigma}$. Las fórmulas son similares a las usadas en el caso unidimensional.

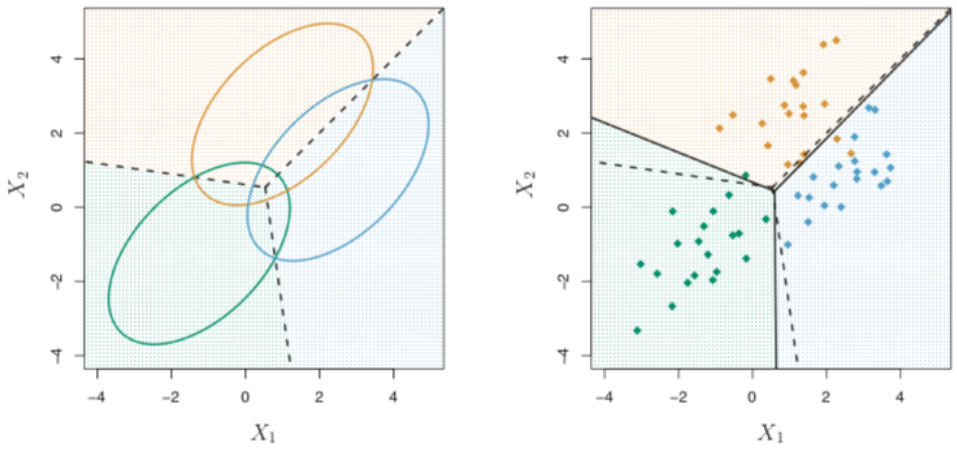

Figura 2.2: Un ejemplo con 3 clases.

En la Figura 1.2 las observaciones de cada clase se han extraido de una distribución gaussiana multivariable con p = 2, con un vector de medias específico de clase y una matriz de covarianza común. Panel izquierdo: se muestran las elipses que contienen el 95 % de la probabilidad para cada una de las tres clases. Las líneas discontinuas son los límites de decisión de Bayes. Panel derecho: se han generado 20 observaciones de cada clase y los límites de decisión de LDA correspondientes se indican mediante líneas negras continuas. Los límites de decisión de Bayes se muestran una vez más como líneas discontinuas.

2.4. Máquinas del vector soporte.

2.4.1. Clasificador de máximo margen.

En esta primera sección, definiremos un hiperplano e introducimos el concepto de un hiperplano de separación óptimo.

2.4.1.1. ¿Qué es un hiperplano?

La definición matemática de un hiperplano es bastante simple. En p dimensiones, un hiperplano se define como:

$$\beta_0 + \beta_1 x_1 + ... + \beta_p x_p = 0 \tag{2.12}$$

en el sentido de que si un punto $\underline{x} = (x_1, x_2, ..., x_p)^T$ de un espacio p-dimensional verifica (2.12), entonces $\underline{x}$ está sobre el hiperplano. Supongamos ahora que $\underline{x}$ no verifica (2.12), entonces si:

$$\beta_0 + \beta_1 x_1 + ... + \beta_p x_p > 0 \tag{2.13}$$

nos está diciendo que $\underline{x}$ cae a un lado del hiperplano, mientras que en la situación opuesta caerá en el otro lado.

2.4.1.2. Clasificación usando un hiperplano de separación.

Supongamos ahora que tenemos una matriz de datos de dimensiones $n \times p$ que se basa en n observaciones extraídas de un conjunto de entrenamiento en un espacio p dimensional, y que estas observaciones se dividen en dos clases, es decir, $y_1, ..., y_n \in$ {-1,1}, dónde -1 representa una clase y 1 la otra. También tendremos una observación extraída de un *conjunto test*, un vector de p características observadas $x^* = (x_1^*, ..., x_p^*)^T$. Nuestro objetivo es desarrolar un clasificador basado en el conjunto de entrenamiento que nos permite clasificar correctamente la observación extraída del conjunto de test en función de sus características.

Supongamos que es posible construir un hiperplano que separe las observaciones del conjunto de entrenamiento perfectamente en función de la clase a la que pertenecen, a dicho hiperplano lo llamaremos *hiperplano de separación*. Una vez conseguido nuestro hiperplano de separación, lo podemos usar para contruir un clasificador de una forma bastante natural: las observaciones del conjunto de test se asignarán a una clase u otra, dependiendo del lado del hiperplano en el que estén localizadas.

2.4.1.3. El clasificador de máximo margen.

Con el fin de construir un clasificador basado en un hiperplano de separación, debemos tener una manera razonable de decidir cuál de los infinitos posibles hiperplanos de separación es el más óptimo. Una posible elección es el *clasificador de máximo margen*,

que es el hiperplano de separación que más alejado está de las de las observaciones pertenecientes al conjunto de entrenamiento. Es decir, podemos calcular la distancia tomada perpendicularmente de cada observación de entrenamiento a un hiperplano de separación dado; la distancia más pequeña es la distancia mínima entre las observaciones y el hiperplano, y se conoce como *margen*. El hiperplano de máximo margen es el hiperplano de separación para el cual el margen es más grande, es decir, es el hiperplano que tiene la distancia mínima más lejana a las observaciones de entrenamiento. Luego, podemos clasificar una observación de prueba en función de a qué lado del hiperplano de margen máximo se encuentra. Si $\beta_0, \beta_1, ..., \beta_p$ son los coeficientes del hiperplano de máximo margen, entonces el clasificador de máximo margen clasificará una observación x^* pertenenciente al conjunto de test basándose en el signo de $f(x^*) = \beta_0 + \beta_1 x_1^* + ..., +\beta_p x_p^*$.

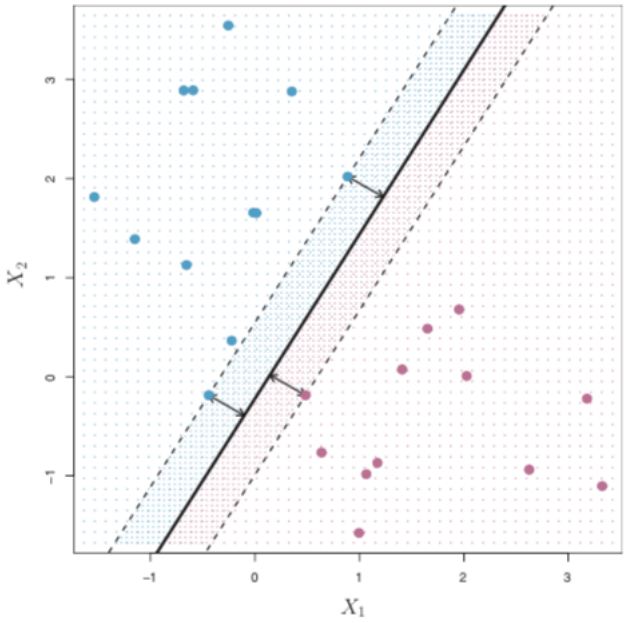

Figura 2.3: Ejemplo del clasificador de máximo margen

En la Figura 2.3 hay dos clases de observaciones, que se muestran en azul y en morado. El hiperplano de máximo margen se muestra como una línea continua. El margen es la distancia desde la línea continua a cualquiera de las líneas discontinuas. Los dos puntos azules y el punto púrpura que se encuentran en las líneas discontinuas forman el vector de soporte, y la distancia desde esos puntos al hiperplano se indica mediante flechas. La cuadrícula de color morado y azul indica la regla de decisión que obtenemos para un clasificador en base a este hiperplano de separación.

Observando la Figura 2.3, podemos ver que 3 observaciones del conjunto de entrenamineto son equidistantes del hiperplano de máximo margen y que se encuentran a lo largo de las líneas discontinuas que indican el ancho del margen. Estas tres observaciones son conocidas como *vectores de soporte*, ya que son observaciones en un espacio p-dimensional y "soportan" el hiperplano de máximo margen en el sentido de que si estos puntos se movieran ligeramente, el hiperplano de margen máximo también se movería, lo que significa que el hiperplano de máximo margen sólo depende de los vectores de soporte y no del resto de observaciones. Esta es una propiedad muy interesante que discutiremos más adelante en este capítulo.

2.4.1.4. Construcción del clasificador de máximo margen.

Ahora consideremos la tarea de construir el hiperplano de margen máximo en base a un conjunto de n observaciones de entrenamiento $x_1, ..., x_n \in \mathrm{R}^*$ y las etiquetas de clase asociadas $y_1, ..., y_n \in \{\text{-}1,1\}$. Brevemente, el hiperplano de máximo margen es la solución al problema de optimización:

$$maximize_{\beta_0,\beta_1...,\beta_p,M} \; M \tag{2.14}$$

$$\text{sujeto a} \sum_{j=1}^{p} \beta_j^2 = 1, \tag{2.15}$$

$$y_i(\beta_0 + \beta_1 x_{i1} + ..., +\beta_p x_{ip}) \geq M, \quad \forall i = 1, ..., n \tag{2.16}$$

Este problema (2.14) a (2.16) se puede resolver de manera eficiente, pero los detalles de su resolución están fuera del alcance de este trabajo.

2.4.2. Clasificador del vector soporte.

Un clasificador basado en un hiperplano de separación necesariamente clasificará perfectamente todas las observaciones de entrenamiento; esto puede conducir a problemas de sensibilidad en observaciones individuales.
En este caso, podríamos estar dispuestos a considerar un clasificador basado en un plano que no separa perfectamente las dos clases, en interés de obtener:

- Mayor robustez para observaciones individuales.
- Mejor clasificación de la mayoría de las observaciones pertenecientes al conjunto de entrenamiento.

Es decir, podría valer la pena clasificar erróneamente algunas observaciones de entrenamiento para hacer un mejor trabajo en la clasificación de las observaciones restantes.
El *clasificador del vector soporte* hace exactamente esto. En lugar de buscar el mayor margen posible para que cada observación no esté solo en el lado correcto del hiperplano, sino también en el lado correcto del margen, permitimos que algunas observaciones estén en el lado incorrecto del margen, o incluso en el lado incorrecto del hiperplano. Una observación puede estar no solo en el lado incorrecto del margen, sino también en el lado equivocado del hiperplano. De hecho, cuando no hay un hiperplano de separación, tal situación es inevitable. Las observaciones en el lado equivocado del hiperplano corresponden a las observaciones de entrenamiento clasificadas incorrectamente por el clasificador del vector soporte. El panel de la derecha de la Figura 2.4 ilustra este escenario.

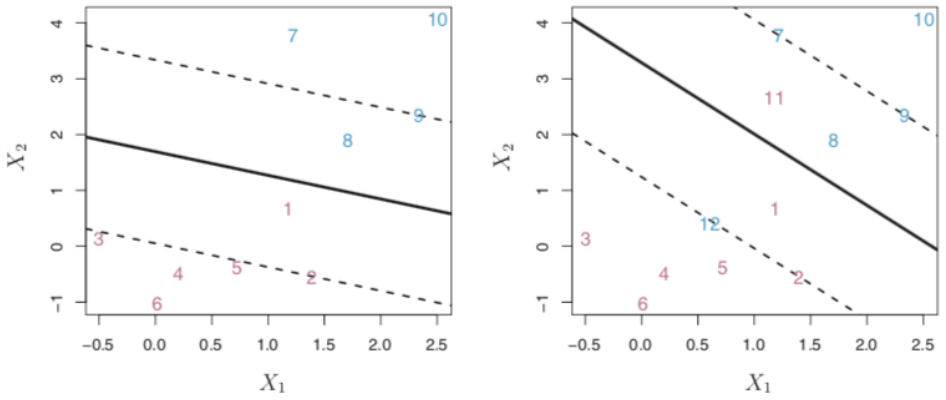

Figura 2.4: Ejemplo del clasificador del vector soporte. Solo las observaciones que caen dentro del margen influyen en la contrucción del clasificador.

2.4.2.1. Detalles del clasificador del vector soporte.

El clasificador de vectores de soporte clasifica una observación del conjunto test en función del lado del hiperplano en el que se encuentra. El hiperplano se elige para separar correctamente la mayoría de las observaciones de entrenamiento en las dos clases, pero puede clasificar erróneamente algunas observaciones. La solución al problema de optimización viene dado por:

$$maximize_{\beta_0,\beta_1...,\beta_p,\epsilon_0,\epsilon_1...,\epsilon_p,M} \; M \tag{2.17}$$

$$\text{con} \quad \sum_{j=1}^{p} \beta_j^2 = 1, \tag{2.18}$$

$$y_i(\beta_0 + \beta_1 x_{i1} + ..., +\beta_p x_{ip}) \geq M(1 - \epsilon_i) \quad \forall i = 1, ..., n \tag{2.19}$$

$$\text{con} \quad \sum_{j=1}^{p} \epsilon_i \leq C, \quad \epsilon_i \geq 0 \tag{2.20}$$

donde C es un parámetro de ajuste no negativo; M representa de nuevo el ancho del margen; Buscamos hacer esta cantidad lo más grande posible. En (2.19), $\epsilon_1, ..., \epsilon_p$ son variables de holgura que permiten que las observaciones individuales estén en el lado incorrecto del margen o del hiperplano. Una vez que hemos resuelto (2.17) a (2.20), clasificaremos una observación del conjunto de prueba x^* como antes, simplemente determinando de qué lado del hiperplano se encuentra. Es decir, clasificamos la observación de prueba en función del signo de $f(x^*) = \beta_0 + \beta_1 x_1^* + ..., +\beta_p x_p^*$.

Este problema de optimización tiene una propiedad muy interesante: resulta que solo las observaciones que se encuentran en el margen o que violan el margen afectarán al hiperplano y, por lo tanto, al clasificador obtenido.
Las observaciones que se encuentran directamente en el margen, o en el lado incorrecto del margen de su clase, se conocen como *vectores de soporte* (support vectors). Estas observaciones afectan al *clasificador de vectores de soporte.*
El hecho de que la regla de decisión del *clasificador del vector de soporte* se base solo en un subconjunto potencialmente pequeño de las observaciones de entrenamiento significa

que es bastante sólido para el comportamiento de las observaciones que están lejos del hiperplano de separación.

2.4.3. Máquinas del vector soporte.

Las máquinas del vector soporte son una extensión del clasificador del vector soporte que resulta de ampliar el 'feature space' de una manera específica, usando *kernels*. A continuación discutiremos esta extensión, aunque los detales están fuera del alcance de este trabajo. Sin embargo, la principal idea es la siguiente: queremos agrandar nuestro 'feature space' con el fin de establecer un límite no lineal entre las clases. Es decir, en lugar de ajustar un clasificador de vector soporte utilizando p características,

$$X_1, X_2, ..., X_p,$$

podríamos ajustar un clasificador del vector soporte usando $2p$ características

$$X_1, X_1^2, X_2, X_2^2 ..., X_p, X_p^2.$$

El sistema que habrá que optimizar sería una extensión del planteado en (2.17) a (2.20). El enfoque *kernel* que describimos a continuación es simplemente un enfoque computacionalmente eficiente para poner en práctica esta idea.
Ahora bien, resulta que la solución al problema del clasificador de vectores de soporte planteado en (2.17) a (2.20), involucra solo los productos escalares de las observaciones. El producto escalar de dos observaciones $x_i, x_{i'}$ viene dado por:

$$\langle x_i, x_{i'} \rangle = \sum_{j=1}^{p} x_{ij}, x_{i'j}. \tag{2.21}$$

A partir de esto, se puede demostrar que:

- El clasificador de vectores de soporte se puede representar como

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x_i, x_{i'} \rangle \tag{2.22}$$

  donde hay n parámetros α_i por observación perteneciente al conjunto de entrenamiento.

- Para estimar los parámetros $\alpha_1, ..., \alpha_n$ y β_0, todo lo que necesiramos son los $\begin{pmatrix} n \\ 2 \end{pmatrix}$ productos internos $\langle x_i, x_{i'} \rangle$ entre todos los pares de observaciones del conjunto de entrenamiento.

Ahora bien, resulta que α_i es cero para los vectores de soporte que no pertenecen a la solución, es decir, si una observación del conjunto de entrenamiento no es un vector soporte, entonces $\alpha_i = 0$. Entonces si ω es el conunto de índices de esos puntos de soporte, podremos reescribir la función(2.22) de la manera

$$f(x) = \beta_0 + \sum_{i \in \omega} \alpha_i \langle x_i, x_{i'} \rangle \tag{2.23}$$

En resumen, todo lo que necesitamos para representar el clasificador lineal $f(\underline{x})$ son los productos internos. Ahora supongamos que cada vez que un producto interno aparece en el cálculo para obtener la solución de un clasificador del vector soporte, lo reemplazamos con una generalización de las productos internos de la forma:

$$K(x_i, x_{i'}),$$

donde K es una función a la que nos referiremos como Kernel. Un *Kernel* es una función que cuantifica la similitud entre dos observaciones. Por ejemplo, podríamos tomar simplemente

$$K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j}, \tag{2.24}$$

Lo que simplemente nos devolvería el clasificador del vector de soporte. La ecuación (2.24) es conocida como *kernel* lineal. Sin embargo, podríamos elegir otra forma para (2.24). Por ejemplo, podríamos escoger

$$K(x_i, x_{i'}) = (1 + \sum_{j=1}^{p} x_{ij} x_{i'j})^d.$$

Esto se conoce como un *kernel* polinomial de grado d, donde d es un entero positivo. El uso de un *kernel* de este tipo con $d > 1$, en lugar del *kernel lineal* estándar , en el algoritmo de clasificación del vector de soporte conduce a un límite de decisión mucho más flexible. Cuándo el clasificador del vector de soporte se combina con un *kernel* no lineal , el clasificador resultante se conoce como *máquina del vector de soporte*.

Otra elección popular es el *kernel* radial, que toma la forma:

$$K(x_i, x_{i'}) = exp\left(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2\right). \tag{2.25}$$

Donde γ es una constante positiva.

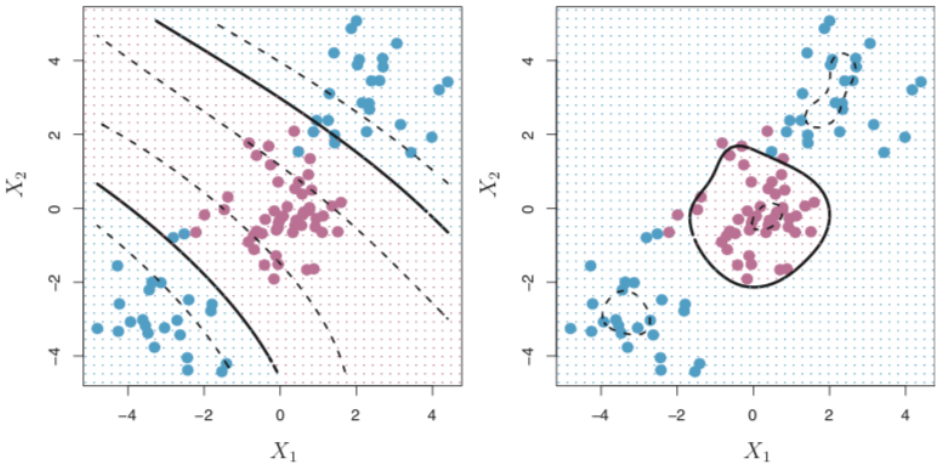

Figura 2.5: Ejemplo de máquinas del vector soporte para kernel polinomial y radial.

En el panel izquierdo de la Figura 2.5 se aplica una SVM con un kernel polinomial de grado 3 a los datos no lineales, lo que resulta en una regla de decisión bastante apropiada. En el panel derecho se aplica un SVM con kernel radial. En este ejemplo, cualquiera de los núcleos es capaz de capturar el límite de decisión.

Algunas de estas figuras en la presentación están tomadas de "An Introduction to Statistical Learning, with applications in R" (Springer, 2013) con el permiso de los autores: G. James, D. Witten, T. Hastie and R. Tibshirani[8]

2.5. Evaluación de la precisión del modelo.

Una vez explicados el conjunto de métodos de clasificación, nos planteamos la siguiente pregunta. ¿Por qué es necesario introducir tantos métodos de aprendizaje estadísticos diferentes, en lugar de un único método y además que sea el más eficiente? La respuesta es que ningún método es superior a los demás en todos los posibles conjuntos de datos existentes. En un conjunto de datos particular, un método específico puede funcionar bien, pero puede haber otros métodos que funcionen mejor en un conjunto de datos similar pero diferente. Seleccionar el método que mejor interprete los datos, es una de las partes más difíciles en la práctica del aprendizaje estadístico. En esta sección, analizaremos algunos de los conceptos más importantes que surgen al seleccionar un método de aprendizaje estadístico para conjuntos de datos con variables respuesta cualitativas.

2.5.1. Medición de la calidad del ajuste.

Para evaluar la calidad del ajuste de un método de aprendizaje estadístico en un conjunto de datos dado, necesitamos medir de alguna forma cómo de bien sus predicciones realmente coinciden con los datos observados. Es decir, debemos cuantificar en que medida el valor de respuesta predicho para una observación dada se aproxima al valor de respuesta real para esa observación. Supongamos que queremos estimar $\widehat{f}$ para un conjunto de observaciones $(\underline{x}_1, y_1), ..., (\underline{x}_n, y_n)$, donde $y_1, ..., y_n$ es cualitativa. El enfoque más común para cuantificar la calidad de nuestra estimación para $\widehat{f}$, es la *tasa de error*, ésta es la proporción de errores que se cometen si aplicamos nuestra estimación $\widehat{f}$ al conjunto de entrenamiento

$$\text{tasa de error de entrenamiento} = \frac{1}{n}\sum_{i=1}^{n} I(y_i \neq \widehat{y}_i) \tag{2.26}$$

donde $\widehat{y}_i$ es la clase predicha para la i-ésima observación usando $\widehat{f}_i$. $I(y_i \neq \widehat{y}_i)$ es una variable indicadora que es igual a 1 si $y_i \neq \widehat{y}_i$ e igual a 0 si $y_i = \widehat{y}_i$. Si $I(y_i \neq \widehat{y}_i) = 0$ entonces la i-ésima observación fue clasificada correctamente por nuestro método de clasificación; mientras que en el caso contario, se habrá clasificado incorrectamente. La ecuación (2.26) se conoce como la *tasa de error de entrenamiento*, ya que se calcula en función de los datos que se usaron para entrenar a nuestro clasificador. Pero en general, no nos interesa medir la calidad del ajuste de nuestro método de clasificación en nuestro conjunto de entrenamiento, ya que obtendríamos unos resultados bastante buenos, pero lejanos a la realidad. Por lo que las nuevas mediciones de calidad las realizaremos en función del conjunto de test.
Para expresarlo de manera más matemática, supongamos que ajustamos nuestro método de clasificación en nuestro conjunto de entrenamiento $(\underline{x}_1, y_1), (\underline{x}_2, y_2), ..., (\underline{x}_n, y_n)$ y obtenemos nuestra estimación $\widehat{f}$. Luego podemos calcular $\widehat{f}_1, \widehat{f}_2, ..., \widehat{f}_n$. Si estos son iguales a $y_1, ..., y_n$ entonces la *tasa de error de entrenamiento* será baja. Sin embargo, realmente no estamos interesados en saber si $\widehat{f}(\underline{x}_i) = y_0$; en cambio, queremos saber si $\widehat{f}(\underline{x}_0) = y_0$ donde (x_0, y_0) es una observación de prueba perteneciente al conjunto de test, el cual no se ha usado para entrenar el método de aprendizaje estadístico. En resumen, queremos elegir el método para el cual la *tasa de error de test* sea más baja, que viene dada por

$$\text{tasa error test} = \frac{1}{n} Ave(I(y_0 \neq \widehat{y_0})) \tag{2.27}$$

donde $\widehat{y}_0$ es la clase en la que se clasifica la observación del conjunto de test pertenece después de haber realizado la predicción. Un buen clasificador es aquel en el que el error (2.27) es más pequeño.
¿Cómo podemos intentar seleccionar el método que minimice (2.27)? En algunos momentos, es posible que tengamos un conjunto de observaciones que no se usaron para entrenar el método de clasificación. Luego podemos simplemente evaluar (2.27) a partir de las observaciones del conjunto de test y seleccionar el método para el cual la tasa de error del conjunto test sea menor. Pero, ¿qué pasa si no hay observaciones pertentecientes al conjunto de test disponibles?. En este caso, uno puede haber seleccionado aquel método de clasificación para el cual la *tasa de error de entrenamiento* sea menor. Desafortunadamente, hay un problema fundamental con esta estrategia: no hay garantía de que el método con la tasa de error de entrenamiento más baja también tenga la tasa de error de test más baja.
Por lo que en la práctica, se puede calcular la tasa de error de entrenamiento con relativa facilidad, pero la tasa de error de prueba usualmente es más difícil ya que no suele haber conjuntos de test disponibles.

2.5.2. El equilibrio entre sesgo-varianza.

Es posible demostrar que la tasa de error de test esperada, para un valor dado $\underline{x}_0$, puede siempre descomponerse en la suma de tres cantidades fundamentales: la varianza de $\widehat{f_0}$, el sesgo al cuadrado de $\widehat{f_0}$ y la varianza de los errores irreducibles ϵ.

¿ A qué nos referimos con la varianza y el sesgo de un método de aprendizaje estadístico? La varianza se refiere a la cantidad en la que $\widehat{f}$ cambiaría si la estimaramos usando conjuntos de datos diferentes. Dado que los conjuntos de entrenamiento se utilizan para ajustar nuestro método de aprendizaje estadístico, diferentes conjuntos de entrenamientos resultarán en una $\widehat{f}$ diferente cada vez. Pero lo ideal es que la estimación de $\widehat{f}$ no varíe mucho para diferentes conjuntos de datos. En general, los métodos estadísticos más flexibles tienden a tener mayores varianzas. Por otra parte, nos referiremos al sesgo como el error existente que hay al querer estimar cualquier problema de la vida real. Por ejemplo, la regresión logística supone que hay una relación existente entre Y y $X_1, X_2, ..., X_p$. Es poco usual que en cualquier problema se presente una relación tan simple, por lo que la regresión logística dará lugar a cierto sesgo a la hora de estimar $\widehat{f}$. Generalmente, los métodos estadísticos más flexibles tienden a menores sesgos. Como regla general, a medida que usamos métodos más flexibles, la varianza aumentará y el sesgo disminuirá.

Nos referiremos a la relación entre el sesgo, la varianza y la tasa de error de test como equilibrio entre sesgo-varianza. Para que un método de aprendizaje estadístico presente buen rendimiento y fiabilidad en sus predicciones, se requerirá tanto baja varianza como bajo sesgo. Esto se conoce como un equilibrio sesgo-varianza ya que es fácil obtener un método con sesgo extremadamente bajo pero con una varianza alta o un método con una varianza muy baja pero con un sesgo alto. El desafío radica en encontrar un método para el que tanto la varianza como el sesgo al cuadrado sean bajos. Este equilibrio es uno de los puntos más recurrentes y a tener en cuenta a la hora de ajustar cualquier modelo

de aprendizaje estadístico. En una situación de la vida real en la que f es desconocida, generalmente no es posible computar explícitamente la tasa de error de test , el sesgo o la varianza de un método de aprendizaje estadístico. Sin embargo, siempre se debe tener en cuenta dicho equilibrio que hemos comentado previamente. A continuación, discutiremos la validación cruzada que es un método para calcular la tasa de error de test a partir del conjunto de entrenamiento.

2.6. Técnicas de remuestreo: Validación Cruzada.

Los métodos de remuestreo son una herramienta indispensable en las estadísticas modernas. Implican extraer muestras de un conjunto de entrenamiento y volver a reajustar un modelo de interés en cada muestra para obtener información adicional sobre el modelo ajustado. Este enfoque puede permitirnos obtener información que no estaría disponible si se ajusta el modelo solo una vez utilizando la muestra de entrenamiento original.
Los enfoques de remuestreo pueden ser computacionalmente costosos, ya que implican ajustar el mismo método estadístico varias veces utilizando diferentes subconjuntos de los datos de entrenamiento. Sin embargo, debido a los recientes avances en la capacidad de computación, los requisitos computacionales de los métodos de remuestreo generalmente no son prohibitivos. En esta sección, analizaremos uno de los métodos de remuestreo más usados, la *validación cruzada*. Este método es una herramienta muy importante en la aplicación práctica de muchos procedimientos de aprendizaje estadístico.Por ejemplo, la validación cruzada se puede usar para estimar el error de test asociado a método de aprendizaje estadístico determinado para evaluar su desempeño o para seleccionar el nivel apropiado de flexibilidad. El proceso de evaluar el rendimiento de un modelo se conoce como *evaluación de modelo*, mientras que el proceso de selección del nivel de flexibilidad de un modelo se conoce como *selección de modelo*.

2.6.1. Enfoque del conjunto de validación.

Supongamos que nos gustaría estimar la tasa de error de test asociada al ajuste de un método de aprendizaje estadístico particular en un conjunto de observaciones. El enfoque del conjunto de validación, es una estrategia muy simple para esta tarea. Implica dividir aleatoriamente el conjunto de observaciones disponibles en dos partes, un conjunto de entrenamiento y un conjunto de validación. El modelo se ajusta al conjunto de entrenamiento, y el modelo ajustado se usa para predecir las respuestas para las observaciones en el conjunto de validación. La tasa de error del conjunto de validación resultante, generalmente se evalúa utilizando la *tasa de error de test* para una variable respuesta cualitativa. El enfoque del conjunto de validación es conceptualmente simple y fácil de implementar. Pero tiene dos inconvenientes potenciales:

- La tasa de error del conjunto de validación puede ser muy variable, dependiendo de qué observaciones se incluyen en el conjunto de entrenamiento y qué observaciones se incluyen en el conjunto de validación.
- En el enfoque de validación, sólo un subconjunto de las observaciones, aquellas que se incluyen en el conjunto de entrenamiento en lugar de en el conjunto de validación, se utilizan para ajustar el modelo. Debido a que los métodos estadísticos tienden

a empeorar cuando se entrenan con menos observaciones, esto sugiere que la tasa de error del conjunto de validación puede tender a sobreestimar la tasa de error de prueba para el ajuste del modelo en todo el conjunto de datos.

2.6.2. Leave-One-Out Cross-Validation

El método *Leave-one-out-cross-validation(LOOCV)* está estrictamente relacionado con el enfoque del conjunto de validación visto previamente pero intentando abordar los incovenientes que este método expone. Al igual que se ha visto previamente, LOOCV implica dividir el conjunto de observaciones en dos partes. Sin embargo, en vez de crear dos subconjuntos de tamaño similar, se utiliza sólo una observación (x_1, y_1) para el conjunto de validación y el resto de observaciones $(x_2, y_2), ..., (x_n, y_n)$ para el conjunto de entrenamiento. El método de aprendizaje estadístico se ajusta a las $n-1$ observaciones de entrenamiento, y se realiza una predicción $\widehat{y}_1$ para la observación excluída, utilizando su valor $\underline{x}_1$. Dado que $(\underline{x}_1, y_1)$ se usó en el proceso de ajuste del modelo, proporcionará una tasa de error de test relativamente libre de sesgos. Pero aunque dicha tasa está libre de sesgos, es una estimación muy pobre ya que es muy variable, pues se basa en una sola observación $(\underline{x}_1, y_1)$. Podemos repetir el procedimiento seleccionando $(\underline{x}_2, y_2)$ para los datos de validación, entrenando el método en el conjunto de $n-1$ observaciones restantes $(x_1, y_1), (x_3, y_3), ..., (x_n, y_n)$ y calcular nuestra tasa de error de test. Repetir este proceso n veces, produce n tasas de error. La LOOCV nos proporciona el promedio para estas n estimaciones:

$$CV_n = \frac{1}{n} \sum_{i=1}^{n} Err_i \tag{2.28}$$

donde $Err_i = I(y_i \neq \widehat{y}_i)$.

LOOCV tiene un par de ventajas importantes sobre el enfoque del conjunto de validación. Primero, tiene mucho menos sesgo. En LOOCV, ajustamos repetidamente el método de aprendizaje estadístico utilizando conjuntos de entrenamiento que contienen $n-1$ observaciones, casi tantas como están en el conjunto de datos completo. Esto contrasta con el enfoque del conjunto de validación, en el que el conjunto de entrenamiento suele ser aproximadamente la mitad del tamaño del conjunto de datos original. En consecuencia, el enfoque LOOCV tiende a no sobreestimar la tasa de error de prueba tanto como lo hace el enfoque de conjunto de validación. En segundo lugar, a diferencia del enfoque de validación que dará resultados diferentes cuando se aplique repetidamente debido a la aleatoriedad en las divisiones del conjunto de entrenamiento/validación, la ejecución de LOOCV varias veces siempre dará los mismos resultados: ya que no hay aleatoriedad en las divisiones del conjunto de entrenamiento/validación.

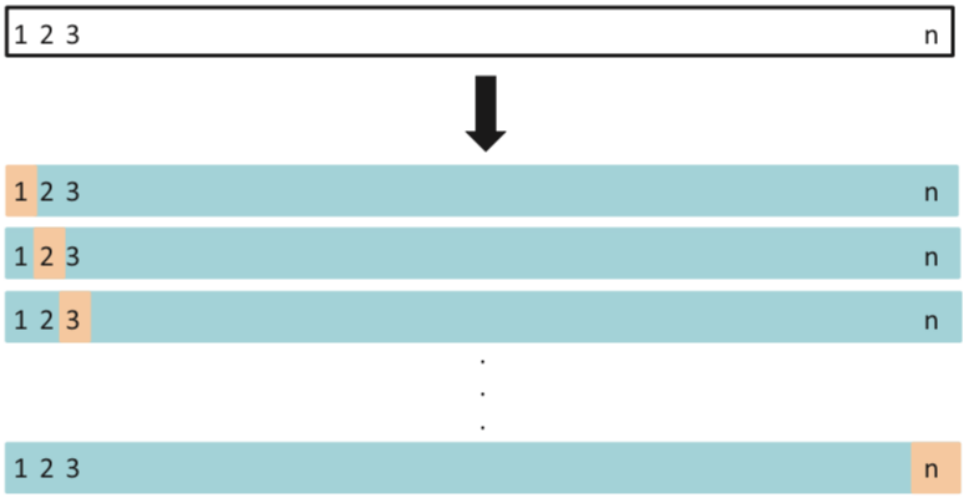

Figura 2.6: Un ejemplo esquemático de LOOCV

En la Figura 2.6 un conjunto de n observaciones se divide repetidamente en un conjunto de entrenamiento (mostrado en azul) que contiene todas las observaciones menos una, y un conjunto de validación que contiene solo esa observación (que se muestra en color beige). El error de la prueba se calcula promediando las n tasas de error resultantes. El primer conjunto de entrenamiento contiene todos menos la observación 1, el segundo conjunto de entrenamiento contiene todos menos la observación 2, y así sucesivamente.

2.6.3. K-Fold Cross-Validation.

Una alternativa a LOOCV es la validación cruzada en k capas (k-Fold Cross-Validation). Este enfoque implica dividir aleatoriamente el conjunto de observaciones en k grupos, o pliegues, de aproximadamente el mismo tamaño. El primer pliegue se trata como un conjunto de validación, y el método se ajusta a los $k-1$ restantes. La tasa de error de test, se calcula luego sobre las observaciones en el pliegue retenido. Este procedimiento se repite k veces. Cada vez, un grupo diferente de observaciones se trata como un conjunto de validación. Este proceso da como resultado k estimaciones de la tasa de error de test. La estimación de k-fold CV se calcula promediando estos valores,

$$CV_n = \frac{1}{k}\sum_{i=1}^{k} Err_i \tag{2.29}$$

donde $Err_i = I(y_i \neq \widehat{y}_i)$.

No es difícil ver que LOOCV es un caso especial de k-fold CV en el que $k = n$. En la práctica, uno normalmente realiza k-fold CV usando k = 5 o k = 10. ¿Cuál es la ventaja de usar k = 5 o k = 10 en lugar de k = n? La ventaja más obvia es computacional. LOOCV requiere ajustar el método de aprendizaje estadístico n veces. Esto tiene la desventaja de ser computacionalmente caro. Pero la validación cruzada es un enfoque muy general que se puede aplicar a casi cualquier método de aprendizaje estadístico. Algunos métodos de aprendizaje estadístico tienen procedimientos de ajuste de computación intensivos, por lo que la ejecución de LOOCV puede plantear problemas computacionales, especialmente si n es extremadamente grande. Por el contrario, realizar 10 veces k-fold CV requiere

que el procedimiento de aprendizaje se ajuste diez veces, lo que puede ser mucho más factible. También puede haber otras ventajas no computacionales al realizar k-fold CV k=5 o k=10, lo que implica el equilibrio entre sesgo-varianza.

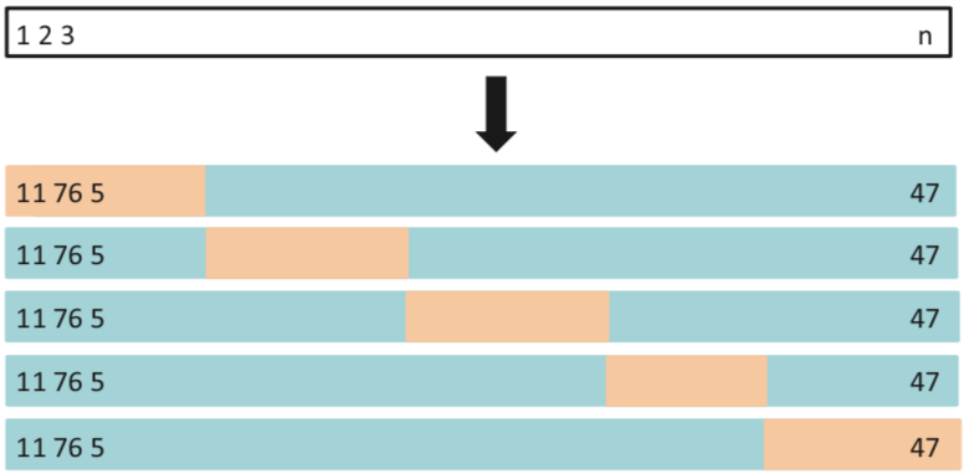

Figura 2.7: Un ejemplo esquemático de 5-Fold Cross-Validation

Un conjunto de n observaciones es divido aleatoriamente en cinco grupos no superpuestos. Cada una de estos grupos actúa como un conjunto de validación (mostrado en color beige) y el resto como un conjunto de entrenamiento (mostrado en azul). El error de la prueba se estima promediando las cinco estimaciones de la tasa de error resultante.

2.6.4. El equilibrio entre sesgo-varianza para K-Fold Cross-Validation.

Como ya hemos mencionado previamente, K-Fold CV presenta una ventaja computacional sobre LOOCV. Pero dejando de lado los problemas informáticos, una ventaja menos obvia, pero potencialmente más importante del K-Fold CV es que, a menudo proporciona estimaciones más precisas de la tasa de error de test que LOOCV. Esto tiene que ver con el equilibrio entre sesgo-varianza.

Por otra parte, como se mencionó en secciones anteriores, el enfoque del conjunto de validación puede llevar a sobreestimar la tasa de error de test , ya que en este enfoque el conjunto de validación utilizado para ajustar el método de aprendizaje estadístico contiene sólo un porcentaje dado de las observaciones de todo el conjunto de datos. Usando esta lógica, no es difícil ver que LOOCV proporcionará estimaciones aproximadamente imparciales de la tasa de error de test, ya que cada conjunto de entrenamiento contiene $n-1$ observaciones, que es casi la cantidad de observaciones en el conjunto de datos completo. Y realizar k-fold CV para, digamos, $k=5$ o $k=10$ llevará a un nivel intermedio de sesgo, ya que cada conjunto de entrenamiento contiene $\frac{(k-1)n}{k}$ observaciones, menos que en el enfoque LOOCV, pero sustancialmente más que en el planteamiento del conjunto de validación. Por lo tanto, desde la perspectiva de la reducción del sesgo, está claro que LOOCV debe preferirse a K-fold CV. Sin embargo, sabemos que el sesgo no es la única fuente de preocupación en el proceso de estimación de un modelo. También debemos considerar la varianza del procedimiento. Resulta que LOOCV tiene varianza más alta que

k-fold CV con $k < n$. ¿Por qué se da esta situación? Cuando realizamos LOOCV, estamos promediando las salidas de los n modelos ajustados, cada uno de los cuales está ajustado en un conjunto casi idéntico de observaciones; por lo tanto, estas salidas están altamente correlacionadas entre sí. En contraste, cuando realizamos k-fold CV con $k < n$, estamos promediando las salidas de k modelos ajustados que están algo menos correlacionados entre sí, ya que la coincidencia entre los conjuntos de entrenamiento en cada modelo es menor. Dado que la media de muchas cantidades altamente correlacionadas entre sí tiene una varianza mayor que la media de muchas cantidades que no están tan correlacionadas, la estimación de la tasa de error de test resultante de LOOCV tiende a tener una varianza mayor que la estimación de la tasa de error de test resultante de k-fold CV. Por lo general, dadas estas consideraciones, se realizará usualmente K-fold CV con $k = 5$ o $k = 10$, ya que se ha demostrado empíricamente que estos valores producen estimaciones de tasa de error de test que no sufren sesgos excesivamente altos, ni una varianza muy alta.

2.7. Medidas para clasificación.

En la práctica, un clasificador binario como el que trataremos durante este trabajo puede cometer dos tipos de errores: puede asignar incorrectamente a una persona con cáncer benigno a la categoría de malignos , o puede asignar incorrectamente a una persona con cáncer maligno a la categoría de benignos. Obviamente, nos interesa asignar una persona con cáncer benigo a la categoría maligno que la situación opuesta, ya que esta última podría acarrear consecuencias mucho mas graves, por lo que resulta interesante determinar que tipos de errores se pueden estar cometiendo. Una forma conveniente de mostrar esta información es la matriz de confusión mostrada en la Figura 2.8.

El rendimiento específico de clase es importante en medicina y biología, donde los términos sensibilidad y especificidad caracterizan el rendimiento de un clasificador. Dicho esto, es evidente que un buen clasificador será aquel que determina resultado positivos en enfermos y resultados negativos en sanos. Por lo tanto, las condiciones que debemos exigir para obtener un clasificador fiable son [3]:

- *Validez*: Nos referiremos a la validez de una prueba diagnóstica como la capacidad para detectar correctamente la presencia o ausencia de la enfermedad que se estudia.
- *Reproductividad*: Nos referiremos a la reproductividad como la capacidad del clasificador de ofrecer los mismos resultados para conjuntos de test similares, pero al mismo tiempo diferentes.
- *Seguridad*: La seguridad viene determinada por el valor predictivo de un resultado positivo o negativo. ¿Con qué seguridad un test predecirá la presencia o ausencia de enfermedad? Ante un resultado positivo de un test ¿qué probabilidad existe de que este resultado indique la presencia de la enfermedad?.

La validez de un modelo puede obtenerse calculando los valores de la sensibilidad y la especificidad:

- *Sensibilidad*: probabilidad de clasificar correctamente a un individuo enfermo, es decir, la probabilidad de que para un sujeto enfermo se obtenga en la prueba un resultado positivo. La sensibilidad es, por lo tanto, la capacidad del test para detectar la enfermedad.
$$sensibilidad = \frac{VP}{VP + FN}$$
- *Especificidad*: probabilidad de clasificar correctamente a un individuo sano, es decir, la probabilidad de que para un sujeto sano se obtenga un resultado negativo. En otras palabras, se puede definir la especificidad como la capacidad para detectar a los sanos.
$$especificidad = \frac{VN}{VN + FP}$$

		Predicción	
		Positivos	**Negativos**
Observación	**Positivos**	Verdaderos Positivos (VP)	Falsos Negativos (FN)
	Negativos	Falsos Positivos (FP)	Verdaderos Negativos (VN)

Figura 2.8: Matriz de confusión.

Hasta el momento estamos abordando el problema de medición de la prueba con resultados dicotómicos (positivo o negativo) pero esta metodología en muchas situaciones no nos proporciona suficiente información, por lo que la confirmación de un diagnóstico deberá hacerse a partir de un parámetro numérico.
Para ello utilizaremos la curva ROC con su AUC correspondiente. Este es un gráfico popular para mostrar simultáneamente los dos tipos de errores(1-especificidad y sensibilidad) para todos los umbrales posibles. El rendimiento general de un clasificador, resumido en todos los umbrales posibles, viene dado por el área bajo la curva ROC (AUC) por lo que de esta manera se convierte en el mejor indicador de la capacidad predictiva del test, independientemente de la prevalencia de la enfermedad de la población. Una curva ROC ideal abarcará la esquina superior izquierda, por lo que cuanto más grande sea el valor AUC, mejor será el clasificador. Los valores posibles para el valor AUC están entre [0,1]. En resumen, la curva ROC nos proporciona una representación global de la exactitud diagnóstica[2][4].

Para interpretar los resultado de AUC se han establecido unos valores predeterminados:

- **[0.5]**: Test totalmente aleatorio
- **(0.5,0.6)**: Test malo.
- **[0.6,0.7)**: Test regular.
- **[0.7,0.8)**: Test bueno.
- **[0.8,0.9)**: Test muy bueno.
- **[0.9,1]**: Test excelente.

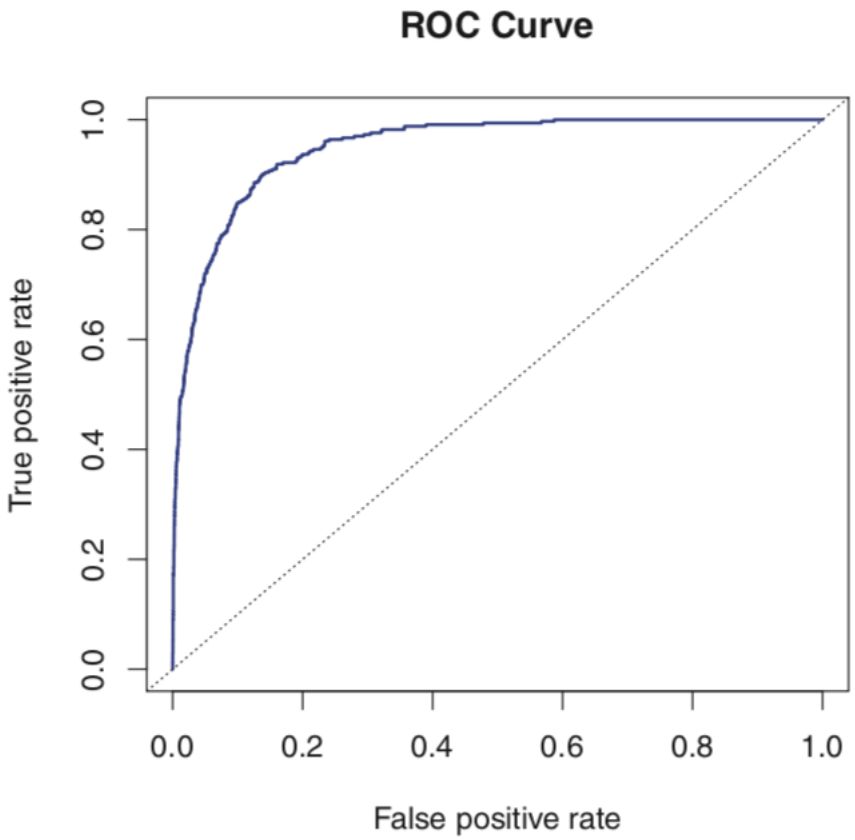

Figura 2.9: Curva ROC.

En la Figura 2.9 observamos una curva ROC con un AUC superior a 0.8. El eje de abscisas viene representado por la tasa de falso positivos o (1-especificidad) mientra que el eje de ordenadas por la tasa de verdaderos positivos o sensitividad. La curva ROC es necesariamente creciente, lo cual refleja la relación existen entre la sensibilidad y la especificidad: si modificamos el umbral de decisión para obtener mayor sensibilidad, un efecto inmediado sería una disminución en la especificidad.

La sensibilidad, la especifidad y el AUC son valores intrínsecos al test diagnóstico (no varían entre poblaciones), nos permiten medir la validez de nuestra clasificación pero por sí solos carecen de utilidad en la práctica. Dada esta situación, nos debemos plantear el siguiente raciocinio: ante un resultado positivo en la clasificación, es decir, que el paciente esté enfermo, ¿cuál es la probabilida de que el individuo presente realmente la enfermedad?. Para abordar este problema usaremos los valores predictivos[3]:

- *Valor predictivo positivo*: podemos definirlo como la probabilidad de padecer la enfermedad si se obtiene un resultado positivo en el test.

$$VPP = \frac{VP}{VP + FP}$$

- *Valor predictivo negativo*: será la probabilidad de que un sujeto con un resultado negativo en la prueba esté realmente sano.

$$VPN = \frac{VN}{VN + FN}$$

Sin embargo, a diferencia de la sensibilidad y la especificidad estos valores no son intrínsecos y se verán afectados por la prevalencia. La prevalencia viene dada por el número de individuos en la población que presentan la enfermedad, es decir, como de frecuente es la enfermadad a diagnosticar en la población. Dicho esto, habrá que tener en cuenta dos posibles situaciones:

- Si la prevalencia de una determinada enfermedad en una población es baja, el valor predictivo positivo tiende a ser bajo ya que, al haber una mayor número de personas sanas, se incrementa el número de falsos positivos.
- Si la prevalencia de una enfermedad es muy elevada, el valor predictivo negativo tiende a bajar pues, al haber un mayor número de personas enfermas, aumenta el número de falsos negativos.

Por último, tenemos el índice de concordancia de Kappa:

- *Índice de concordancia de Kappa*: Es una medida estadística que ajusta el efecto del azar en la proporción de la concordancia observada para para variables categóricas. Asociada a una matriz de confusión, los programas estadísticos proporcionan dicho índice. Para una matriz de confusión 2×2 como la recogida en la Figura 2.8, este índice se calcularía como

$$K = \frac{P_0 - P_e}{1 - P_e}$$

siendo $P_0 = \frac{VP+VN}{VP+FN+FP+VN}$ y $P_e = \frac{1}{n^2} \sum_{j=1}^{2} n_{j.} n_{.j}$

Capítulo 3

Estudio del conjunto de datos Wisconsin.

Analizaremos el conjunto de datos Wisconsin (diagnóstico) de cáncer de mama, obtenido de Kaggle. Este conjunto de datos fue creado por el Dr. William H. Wolberg, médico del Hospital de la Universidad de Wisconsin en Madison, Wisconsin, EE. UU. Para crear el conjunto de datos utilizó muestras de líquidos tomadas de pacientes con masas mamarias sólidas y un programa informático gráfico fácil de usar llamado Xcyt, que es capaz de realizar el análisis de las características citológicas basadas en un escaneo digital. El programa utiliza un algoritmo de ajuste de curva, para calcular diez características de cada una de las celulas de la muestra, luego calcula el valor medio, el valor extremo y error estándar de cada característica para la imagen, devolviendo un vector de 30 características.

Información sobre la variable respuesta:

- 1) Numero de identificación del paciente 2) Diagnosis(M = maligno, B = benigno)

Información sobre las variables dependientes, se calculan 10 variables de valor real para cada núcleo celular:

1. Radius (la media de las distancia desde el centro a los puntos en el perímetro)
2. texture (desviación estandar de los valores en la escala de grises)
3. perimeter (perimetro)
4. area (area)
5. smoothness (variación local de las longitudes del radio)
6. compactness($perimeter^2/area - 1.0$)
7. concavity(severidad de las porciones cóncavas del contorno)
8. Concave points(número de porciones cóncavas del contorno)
9. Simmetry (Simetría)
10. Dimensión fractal(aproximación al límite-1)

La media, el error estándar y el "peor" o el más grande (la media de los tres valores más grandes) de estas características se computaron para cada imagen, resultando en 30 variables. El conjunto de datos puede considerarse libre de ruido, es decir, ya depurado.

El objetivo del análisis para este conjunto de datos será clasificar correctamente las células como malignas o benignas en función de sus características, intentando reducir lo máximo posible la tasa de falsos negativos, es decir, clasificar un tumor maligno cómo benigno.

3.1. Método de trabajo.

- Se realiza un estudio descriptivo para ver que variables tienen un mayor poder predictivo.
- Se incluye un estudio sobre la colinealidad y multicolinealidad para realizar un proceso adecuado de la selección de variables.
- Tratamiento de las observaciones consideradas como outliers.
- Aplicación de las técnicas de clasificación.
- Los paquetes estadísticos que se han usado durante el análisis son:

1. Library(caret)[9]
2. Library(corrplot)[13]
3. Library(car)[5]
4. Library(e1071)[11]
5. Library(ROCR)[12]

3.1.1. Obtención de los datos.

Obtenemos el conjunto de datos:

```
datos<-read.csv("data.csv",header=T, stringsAsFactors=F)
head(datos)
```

```
##         id diagnosis radius_mean texture_mean perimeter_mean area_mean
## 1   842302         M       17.99        10.38         122.80    1001.0
## 2   842517         M       20.57        17.77         132.90    1326.0
## 3 84300903         M       19.69        21.25         130.00    1203.0
## 4 84348301         M       11.42        20.38          77.58     386.1
## 5 84358402         M       20.29        14.34         135.10    1297.0
## 6   843786         M       12.45        15.70          82.57     477.1
##   smoothness_mean compactness_mean concavity_mean concave.points_mean
## 1         0.11840          0.27760         0.3001             0.14710
## 2         0.08474          0.07864         0.0869             0.07017
## 3         0.10960          0.15990         0.1974             0.12790
## 4         0.14250          0.28390         0.2414             0.10520
## 5         0.10030          0.13280         0.1980             0.10430
## 6         0.12780          0.17000         0.1578             0.08089
##   symmetry_mean fractal_dimension_mean radius_se texture_se perimeter_se
## 1        0.2419                0.07871    1.0950     0.9053        8.589
## 2        0.1812                0.05667    0.5435     0.7339        3.398
## 3        0.2069                0.05999    0.7456     0.7869        4.585
## 4        0.2597                0.09744    0.4956     1.1560        3.445
## 5        0.1809                0.05883    0.7572     0.7813        5.438
## 6        0.2087                0.07613    0.3345     0.8902        2.217
##   area_se smoothness_se compactness_se concavity_se concave.points_se
## 1  153.40      0.006399        0.04904      0.05373           0.01587
## 2   74.08      0.005225        0.01308      0.01860           0.01340
```

```
## 3   94.03      0.006150        0.04006      0.03832           0.02058
## 4   27.23      0.009110        0.07458      0.05661           0.01867
## 5   94.44      0.011490        0.02461      0.05688           0.01885
## 6   27.19      0.007510        0.03345      0.03672           0.01137
##   symmetry_se fractal_dimension_se radius_worst texture_worst
## 1     0.03003             0.006193        25.38         17.33
## 2     0.01389             0.003532        24.99         23.41
## 3     0.02250             0.004571        23.57         25.53
## 4     0.05963             0.009208        14.91         26.50
## 5     0.01756             0.005115        22.54         16.67
## 6     0.02165             0.005082        15.47         23.75
##   perimeter_worst area_worst smoothness_worst compactness_worst
## 1          184.60     2019.0           0.1622            0.6656
## 2          158.80     1956.0           0.1238            0.1866
## 3          152.50     1709.0           0.1444            0.4245
## 4           98.87      567.7           0.2098            0.8663
## 5          152.20     1575.0           0.1374            0.2050
## 6          103.40      741.6           0.1791            0.5249
##   concavity_worst concave.points_worst symmetry_worst
## 1          0.7119               0.2654         0.4601
## 2          0.2416               0.1860         0.2750
## 3          0.4504               0.2430         0.3613
## 4          0.6869               0.2575         0.6638
## 5          0.4000               0.1625         0.2364
## 6          0.5355               0.1741         0.3985
##   fractal_dimension_worst  X
## 1                 0.11890 NA
## 2                 0.08902 NA
## 3                 0.08758 NA
## 4                 0.17300 NA
## 5                 0.07678 NA
## 6                 0.12440 NA
```

El conjunto de datos ya está depurado, el único cambio que deberemos realizar es eliminar la columna X, un vector con datos nulos.

```
datos$X<-NULL
```

Renombramos la variable respuesta para hacer más facil la visualización de los resultados y eliminamos la primera variable, que corresponde al ID de los pacientes.

```
datos<-datos[,-1]
datos$diagnosis <-factor(ifelse(datos$diagnosis=="B","Benign","Malignant"))
```

3.1.2. Análisis descriptivo.

A continuación comenzamos realizando un análisis descriptivo, visualizando las medias y varianzas que presentan los datos para las distintas variables explicativas distinguiendo por los grupos **Maligno** y **Benigno**.

Cuadro 3.1: Medias por grupos

	Maligno	Benigno
radius_mean	17.4628	12.1465
texture_mean	21.6049	17.9148
perimeter_mean	115.3654	78.0754
area_mean	978.3764	462.7902
smoothness_mean	0.1029	0.0925
compactness_mean	0.1452	0.0801
concavity_mean	0.1608	0.0461
concave.points_mean	0.0880	0.0257
symmetry_mean	0.1929	0.1742
fractal_dimension_mean	0.0627	0.0629
radius_se	0.6091	0.2841
texture_se	1.2109	1.2204
perimeter_se	4.3239	2.0003
area_se	72.6724	21.1351
smoothness_se	0.0068	0.0072
compactness_se	0.0323	0.0214
concavity_se	0.0418	0.0260
concave.points_se	0.0151	0.0099
symmetry_se	0.0205	0.0206
fractal_dimension_se	0.0041	0.0036
radius_worst	21.1348	13.3798
texture_worst	29.3182	23.5151
perimeter_worst	141.3703	87.0059
area_worst	1422.2863	558.8994
smoothness_worst	0.1448	0.1250
compactness_worst	0.3748	0.1827
concavity_worst	0.4506	0.1662
concave.points_worst	0.1822	0.0744
symmetry_worst	0.3235	0.2702
fractal_dimension_worst	0.0915	0.0794

De manera general, podemos observar mayores medias y varianzas de las variables relacionadas a individuos con tumores malignos que benignos, excepto para variables textura y suavidad, que muestran resultados similares. Cabe destacar, que las variables están en diferentes unidades, lo cuál es lógico, dada la naturaleza de los datos.

Observemos a continuación con que frecuencia se presentan las clases:

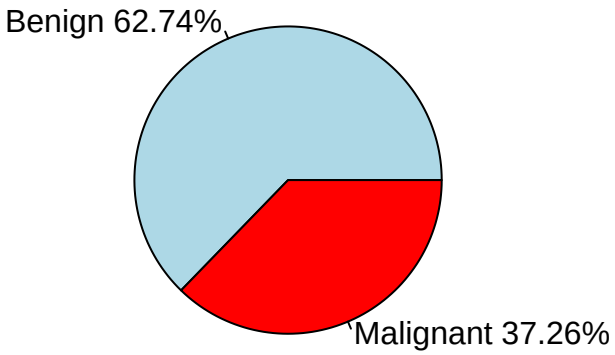

Figura 3.1: Frecuencias.

Es decir, un 62.74 % de las observaciones tienen una diagnosis de benigno y un 37.25 % de maligno. Cuando se estudian problemas con datos no balanceados, es crucial ajustar el clasificador o el balance de los datos pertenecientes al conjunto de entrenamiento, para evitar obtener un modelo de clasificación inexacto. Existen varios métodos para tratar

Cuadro 3.2: Varianzas por grupos

	Maligno	Benigno
radius_mean	10.2654	3.1702
texture_mean	14.2844	15.9610
perimeter_mean	477.6259	139.4156
area_mean	135378.3554	18033.0301
smoothness_mean	0.0002	0.0002
compactness_mean	0.0029	0.0011
concavity_mean	0.0056	0.0019
concave.points_mean	0.0012	0.0003
symmetry_mean	0.0008	0.0006
fractal_dimension_mean	0.0001	0.0000
radius_se	0.1191	0.0127
texture_se	0.2335	0.3471
perimeter_se	6.5974	0.5947
area_se	3764.4690	78.2070
smoothness_se	0.0000	0.0000
compactness_se	0.0003	0.0003
concavity_se	0.0005	0.0011
concave.points_se	0.0000	0.0000
symmetry_se	0.0001	0.0000
fractal_dimension_se	0.0000	0.0000
radius_worst	18.3490	3.9258
texture_worst	29.5371	30.1835
perimeter_worst	867.7181	182.9822
area_worst	357565.4218	26765.4259
smoothness_worst	0.0005	0.0004
compactness_worst	0.0290	0.0085
concavity_worst	0.0329	0.0197
concave.points_worst	0.0021	0.0013
symmetry_worst	0.0056	0.0017
fractal_dimension_worst	0.0005	0.0002

conjuntos de datos no balanceados ("downsampling","upsampling") pero algunos estudios demuestran que estas prácticas no tienen un efecto positivo en el rendimiento predictivo de los clasificadores entrenados, especialmente cuando el conjunto de datos obtenidos no es excesivamente grande, ya que se pierde información.

Luego, por esta razón, además de que el balanceo de los datos no es totalmente desproporcionado y que usaremos algunas técnicas de aprendizaje estadístico que son casi insensibles a este tipo de problemas (SVM), podremos considerar que existe un balanceo razonable que no afectará a los modelos de clasificación. Por otro lado, debemos considerar que algunas otras técnicas si que se verán penalizadas por dicho balanceo, ya que estas utilizan las probabilidades a priori de cada clase para asignar una observación a una clase específica, esto ocurre por ejemplo en el Análisis Discriminante Lineal.

Aunque se ha realizado un análisis para los tres bloques de variables que muestra el conjunto de datos ("Mean","SE","Worst"), solo se mostrarán los resultados para las variables relacionadas con *Mean*. Los resultados que muestran el resto de variables son similares.

A continuación, obtenemos diagramas de caja y bigotes para los datos relacionados con las medidas del grupo *mean*:

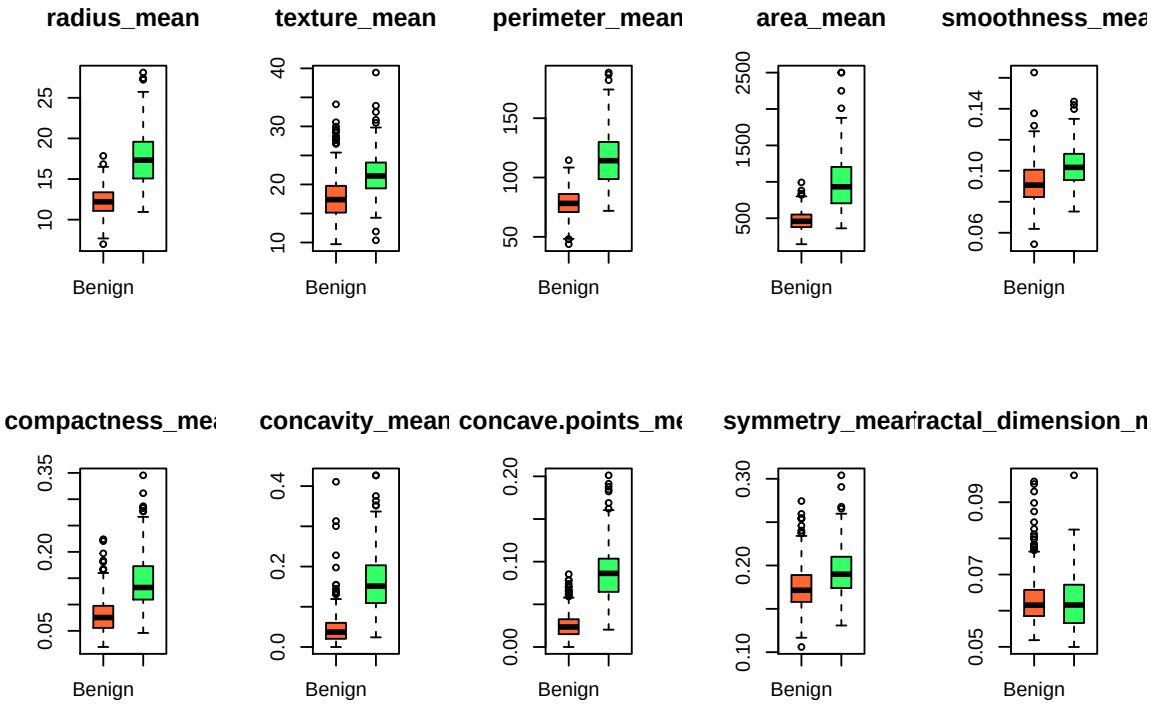

Figura 3.2: Diagrama de caja y bigotes para el conjunto Mean.

Observamos que las variables que mejor discriminan entre los grupos maligno y benigno son aquellas relacionadas con el radio, el perímetro, el área, la compacidad , la concavidad y los puntos cóncavos. Luego son aquellas que intentaremos conservar a la hora de eliminar las variables que estén altamente correlacionadas entre sí para evitar problemas de colinealidad, ya que son más útiles para realizar predicciones. Además, de manera general, las variables relacionadas con tumores malignos paracen mostrar más dispersión que las relacionadas con tumores benignos.

3.1.3. Colinealidad y multicolinealidad.

Nos referiremos a colinealidad a la situación en la que dos o más variables predictoras están estrechamente relacionadas entre sí. Una forma sencilla de detectar la colinealidad es observar la matriz de correlación de las variables predictoras. Un elemento de esta matriz que es grande en valor absoluto indica un par de variables altamente correlacionadas y, por lo tanto, un problema de colinealidad en los datos.

A continuación, visualizamos la correlación existente entre las variables de manera global:

```
cor <- cor(datos[, -1])
head(cor)
```

```
##                  radius_mean texture_mean perimeter_mean area_mean
## radius_mean        1.0000000   0.32378189      0.9978553 0.9873572
## texture_mean       0.3237819   1.00000000      0.3295331 0.3210857
## perimeter_mean     0.9978553   0.32953306      1.0000000 0.9865068
## area_mean          0.9873572   0.32108570      0.9865068 1.0000000
## smoothness_mean    0.1705812  -0.02338852      0.2072782 0.1770284
```

```
## compactness_mean   0.5061236   0.23670222      0.5569362 0.4985017
##                  smoothness_mean compactness_mean concavity_mean
## radius_mean           0.17058119        0.5061236      0.6767636
## texture_mean         -0.02338852        0.2367022      0.3024178
## perimeter_mean        0.20727816        0.5569362      0.7161357
## area_mean             0.17702838        0.4985017      0.6859828
## smoothness_mean       1.00000000        0.6591232      0.5219838
## compactness_mean      0.65912322        1.0000000      0.8831207
##                  concave.points_mean symmetry_mean fractal_dimension_mean
## radius_mean                0.8225285    0.14774124            -0.31163083
## texture_mean               0.2934641    0.07140098            -0.07643718
## perimeter_mean             0.8509770    0.18302721            -0.26147691
## area_mean                  0.8232689    0.15129308            -0.28310981
## smoothness_mean            0.5536952    0.55777479             0.58479200
## compactness_mean           0.8311350    0.60264105             0.56536866
##                  radius_se  texture_se perimeter_se   area_se
## radius_mean      0.6790904 -0.09731744    0.6741716 0.7358637
## texture_mean     0.2758687  0.38635762    0.2816731 0.2598450
## perimeter_mean   0.6917650 -0.08676108    0.6931349 0.7449827
## area_mean        0.7325622 -0.06628021    0.7266283 0.8000859
## smoothness_mean  0.3014671  0.06840645    0.2960919 0.2465524
## compactness_mean 0.4974734  0.04620483    0.5489053 0.4556529
##                  smoothness_se compactness_se concavity_se
## radius_mean       -0.222600125      0.2060000    0.1942036
## texture_mean       0.006613777      0.1919746    0.1432931
## perimeter_mean    -0.202694026      0.2507437    0.2280823
## area_mean         -0.166776667      0.2125826    0.2076601
## smoothness_mean    0.332375443      0.3189433    0.2483957
## compactness_mean   0.135299268      0.7387218    0.5705169
##                  concave.points_se  symmetry_se fractal_dimension_se
## radius_mean             0.3761690 -0.104320881         -0.042641269
## texture_mean            0.1638510  0.009127168          0.054457520
## perimeter_mean          0.4072169 -0.081629327         -0.005523391
## area_mean               0.3723203 -0.072496588         -0.019886963
## smoothness_mean         0.3806757  0.200774376          0.283606699
## compactness_mean        0.6422619  0.229976591          0.507318127
##                  radius_worst texture_worst perimeter_worst area_worst
## radius_mean         0.9695390     0.2970076       0.9651365  0.9410825
## texture_mean        0.3525729     0.9120446       0.3580396  0.3435459
## perimeter_mean      0.9694764     0.3030384       0.9703869  0.9415498
## area_mean           0.9627461     0.2874886       0.9591196  0.9592133
## smoothness_mean     0.2131201     0.0360718       0.2388526  0.2067184
## compactness_mean    0.5353154     0.2481328       0.5902104  0.5096038
##                  smoothness_worst compactness_worst concavity_worst
## radius_mean            0.11961614        0.4134628       0.5269115
## texture_mean           0.07750336        0.2778296       0.3010252
## perimeter_mean         0.15054940        0.4557742       0.5638793
## area_mean              0.12352294        0.3904103       0.5126059
```

```
## smoothness_mean        0.80532420          0.4724684        0.4349257
## compactness_mean       0.56554117          0.8658090        0.8162752
##                   concave.points_worst symmetry_worst
## radius_mean                  0.7442142      0.1639533
## texture_mean                 0.2953158      0.1050079
## perimeter_mean               0.7712408      0.1891150
## area_mean                    0.7220166      0.1435699
## smoothness_mean              0.5030534      0.3943095
## compactness_mean             0.8155732      0.5102234
##                   fractal_dimension_worst
## radius_mean                   0.007065886
## texture_mean                  0.119205351
## perimeter_mean                0.051018530
## area_mean                     0.003737597
## smoothness_mean               0.499316369
## compactness_mean              0.687382323
```

```
det(cor)
```

```
## [1] 2.081724e-31
```

La salida previa muestra el determinante obtenido para la matriz de correlaciones global. Un determinante $det(R) \approx 0$ indicará que existe correlación entre las variables.

Para obtener el siguiente gráfico se ha usado la librería corrplot[13].

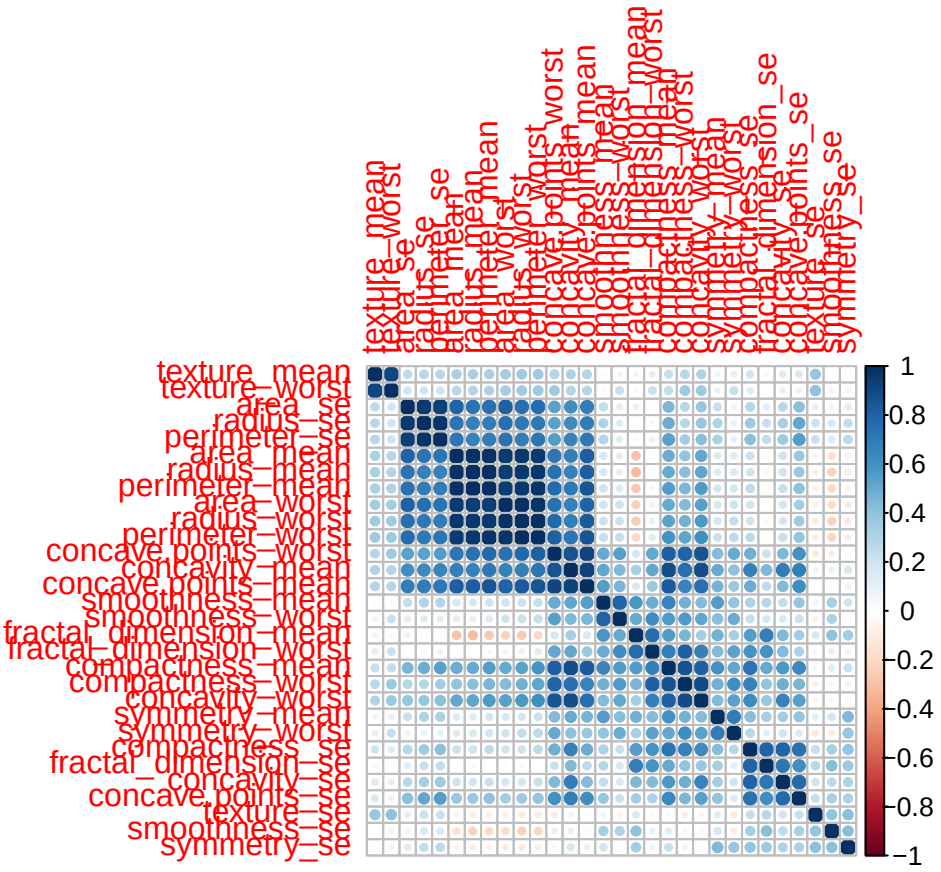

Figura 3.3: Matriz de correlación para el conjunto global de variables.

Eliminamos las variables que presentan mayor coeficiente de correlación en valor absoluto, intentando conservar al mismo tiempo, aquellas que mejor discriminan, como ya se indicó previamente. Se puede observar dos grupos claros entre variables correlacionadas. El primero entre radius, perimeter y area (las variables son funciones unas de otras) y el segundo entre compactness, concavity y concave points. Además, se puede observar que de manera general los variables pertenecientes al conjunto "mean" están altamente correlacionadas con las pertenecientes al conjunto "worst", esto es debido a que las variables de este último grupo han sido obtenidas a partir de los datos del primero.

Por tanto, para evitar problemas de multicolinealidad eliminamos las variables que presentan un coeficiente de correlación cuyo valor absoluto sea superior o igual a 0.7.

```
library(caret)
correlacionadas <- colnames(datos[,-1])[findCorrelation(cor,
cutoff = 0.7, verbose = FALSE)]
datos.I <- datos[, which(!colnames(datos) %in% correlacionadas)]
colnames(datos.I)
```

```
##  [1] "diagnosis"              "texture_mean"
##  [3] "area_mean"              "symmetry_mean"
##  [5] "texture_se"             "smoothness_se"
##  [7] "symmetry_se"            "fractal_dimension_se"
##  [9] "smoothness_worst"       "symmetry_worst"
## [11] "fractal_dimension_worst"
```

Finalmente, se han eliminado 19 variables conservándose por lo tanto 10 variables predictivas que parece que discriminan mejor y por tanto aportan más información. Éstas son las que se muestran en la salida anterior. Volvamos a visualizar la correlación existente entre las variables que hemos conservado y calculamos el determinante de la matriz de correlaciones:

```
## [1] 0.006750918
```

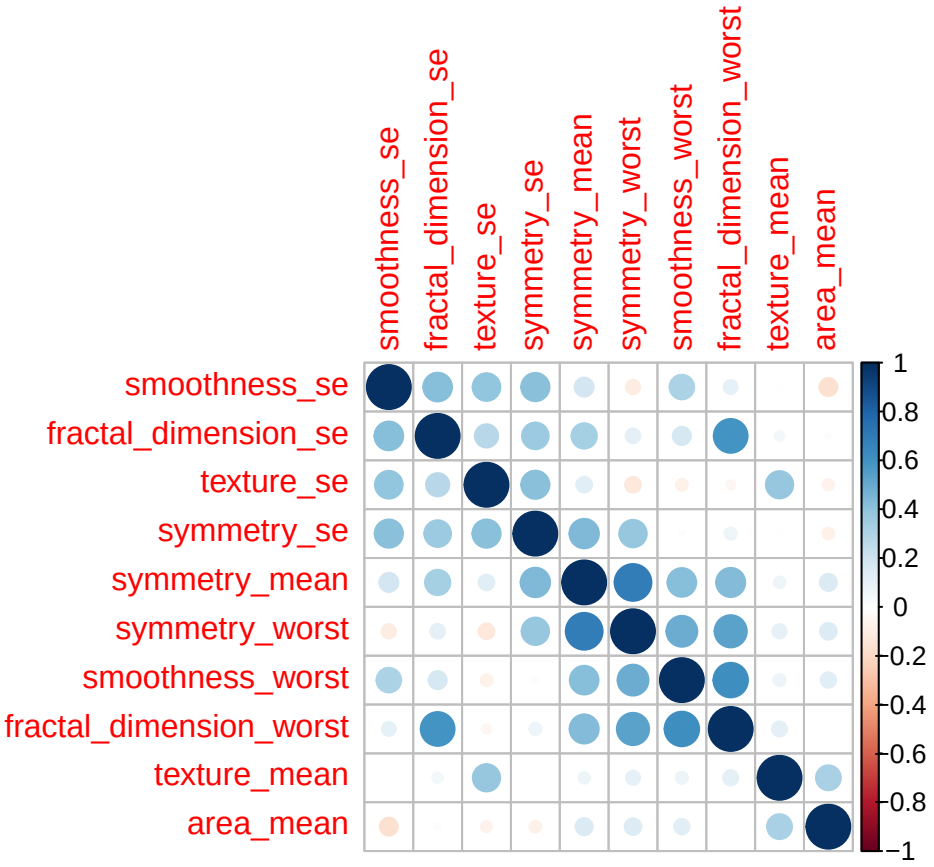

Figura 3.4: Matriz de correlación para las variables seleccionadas.

Desafortunadamente, no todos los problemas de colinealidad pueden detectarse mediante la inspección de la matriz de correlación: es posible que exista colinealidad entre tres o más variables, incluso si ningún par de variables tiene una correlación particularmente alta. A esta situación la llamamos multicolinealidad. En lugar de inspeccionar la matriz de correlación, una mejor manera de evaluar la multicolinealidad es calcular el factor de inflación de la varianza (VIF). El VIF es la relación de la varianza de β_j cuando se ajusta el modelo completo dividido por la varianza de β_j si se ajustará por si sola. El valor más pequeño posible para VIF es 1, lo que indica la ausencia total de colinealidad. Normalmente, en la práctica siempre hay una pequeña cantidad de colinealidad entre las variables predictoras. Como regla general, un valor VIF que exceda de 5 o 10 indica una cantidad problemática de colinealidad.
Por tanto, obtendremos el VIF para el resto de variables que hemos conservado y eliminaremos aquellas con un valor superior a 10. Con la función vif() podemos obtener el valor VIF relacionado con cada coeficiente.

```
##            texture_mean                 area_mean           symmetry_mean
##                1.925156                  2.985237                  2.987645
##              texture_se              smoothness_se                symmetry_se
##                2.310204                  8.219535                  3.272657
##    fractal_dimension_se          smoothness_worst            symmetry_worst
##                10.509265                  5.333537                  6.015961
## fractal_dimension_worst
##                10.912085
```

Observamos valores VIF altos, relacionados con las variables *fractal_dimension_worst* y *fractal_dimension_se*. Si observamos la Figura 3.4, estas son las variables que mayores coeficientes de correlación representan respecto al resto de variables. Eliminamos *fractal_dimension_worst*, ya que presenta mayores indices de correlación que fractal_dimension_se y volvemos a obtener los valores VIF relacionados a cada coeficiente.

```
##          texture_mean            area_mean       symmetry_mean
##              1.923607             2.634878            2.890959
##           texture_se       smoothness_se          symmetry_se
##              2.117158             3.400808            2.959994
## fractal_dimension_se   smoothness_worst      symmetry_worst
##              2.223812             3.160107            5.046096
```

Obtenemos unos resultados excelentes, todos los VIF obtenidos son menores o iguales a 5. Parece que hemos conseguido solventar el problema de la multicolinealidad.

Por tanto, elimamos la variable *fractal_dimension_worst* del conjunto de datos, conservando un total de 9 variables predictoras.

Veamos como se comportan las variables que hemos conservado. Para ello en primer lugar realizamos un gráfico de dispersión para el conjunto de variables conservadas, tanto de manera global como por grupos. Para obtener dichos gráficos se ha usado la librería *library(car)*[5].

```
library(car)
```

En la Figura 3.5 se han representado las variables para todas las variables conservadas.

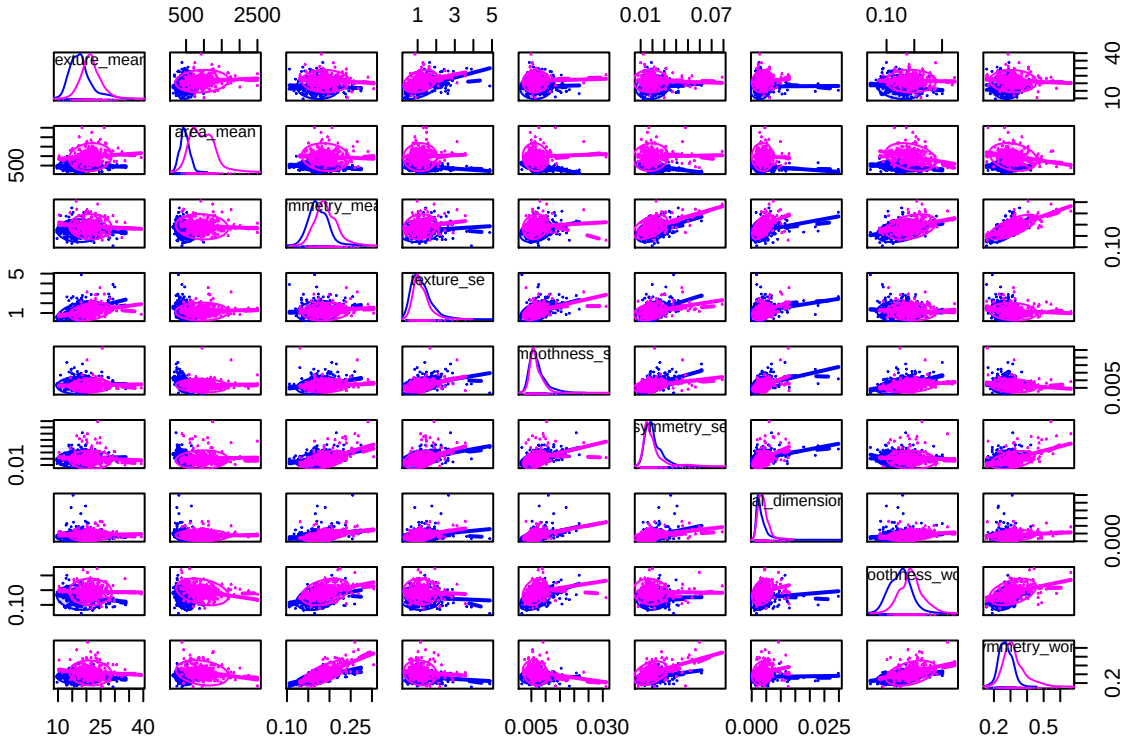

Figura 3.5: Gráfico de dispersión para el conjunto global de datos.

En la Figura 3.6 se han representado las variables del grupo *Mean*

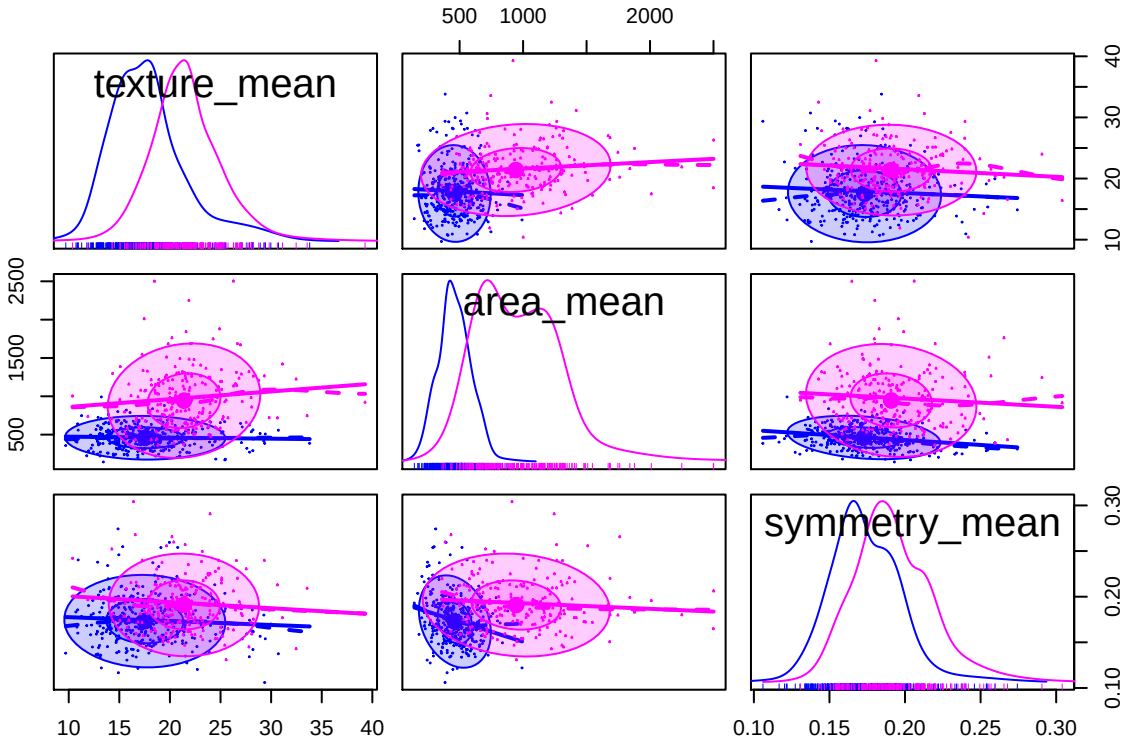

Figura 3.6: Gráfico de dispersión para el grupo Mean.

En la Figura 3.7 se han representado las variables del grupo *SE*

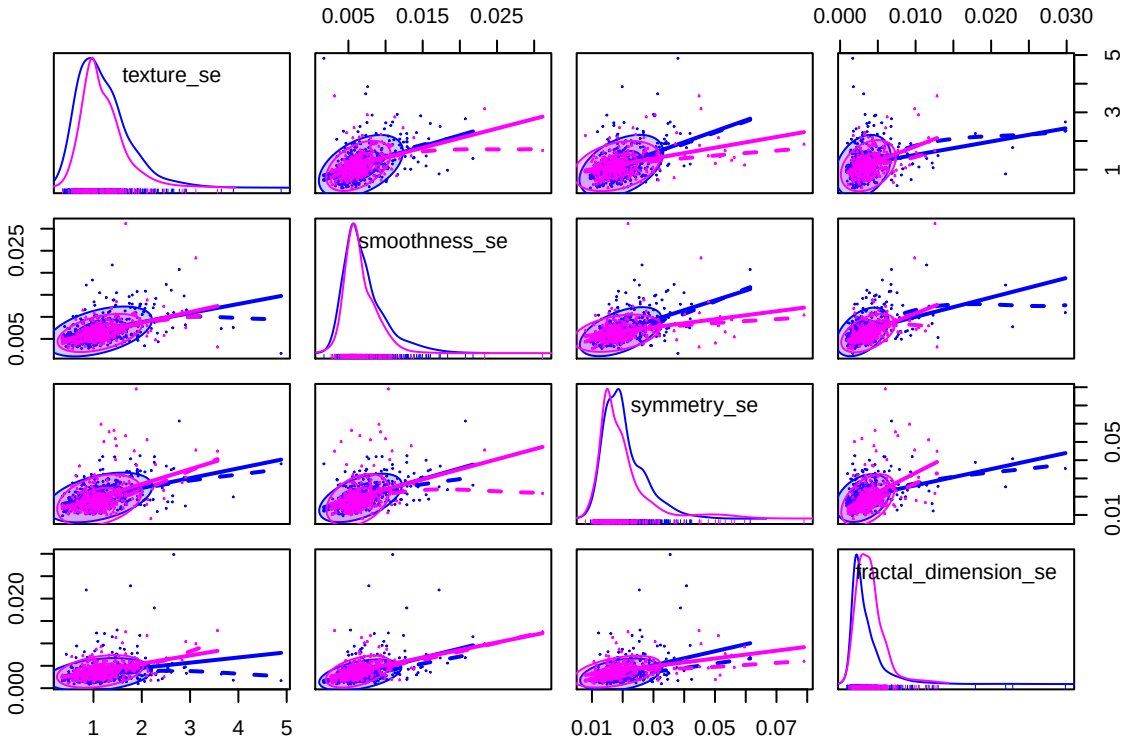

Figura 3.7: Gráfico de dispersión para el grupo SE.

En la Figura 3.8 se han representado las variables del grupo *Worst*

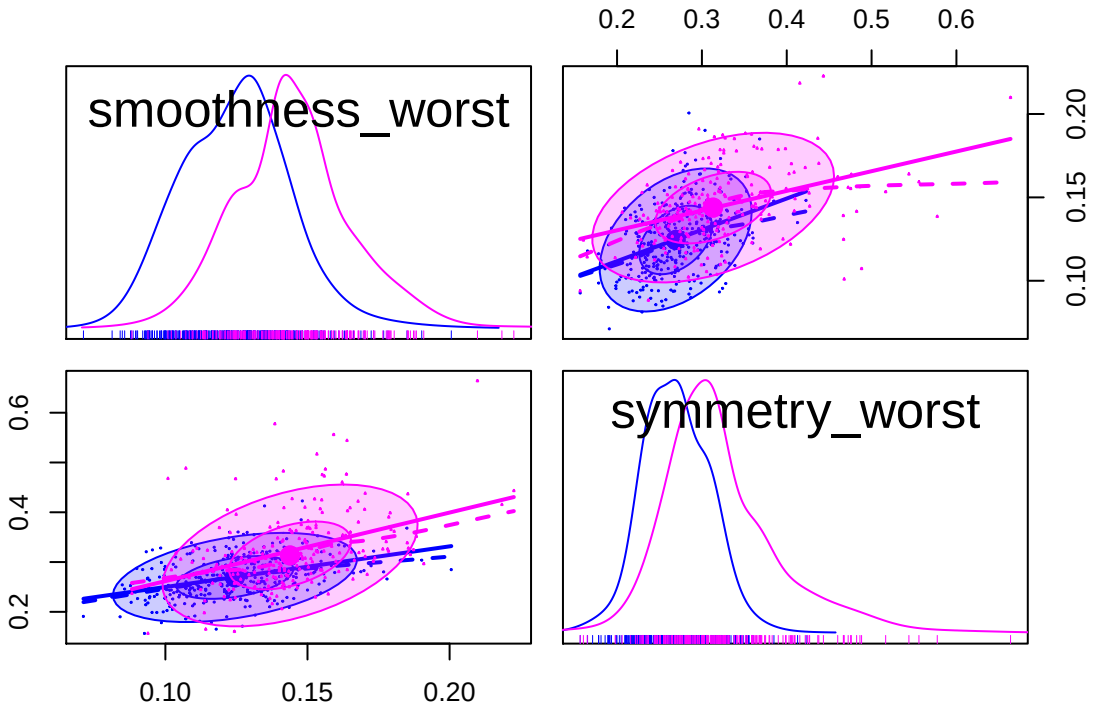

Figura 3.8: Gráfico de dispersión para el grupo Worst.

Observamos mezclas de distribuciones tanto en los gráficos de la diagonal (mixtura de distribuciones) como fuera de la diagonal.

Por defecto, se muestran dos elipses de densidad normal bivariada al 95 % en cada diagrama de dispersión. Suponiendo que cada par de variables tiene una distribución normal bivariada, esta elipse engloba aproximadamente el 95 % de los puntos. La estrechez de la elipse refleja el grado de correlación de las variables. Si obtenemos un círculo y no está orientada en diagonal, las variables no están correlacionadas. Si la elipse es estrecha y está orientada diagonalmente, las variables están correlacionadas.
Las líneas continuas representan la línea que mejor se ajusta, podemos definirla como la línea que mejor expresa la relación entre cada par de variables. Las líneas discontinuas representan la línea Lowess, que es una herramienta que se utiliza en regresión para visualizar la relación entre las variables de una forma no paramétrica.

Además se muestra en las Figuras 3.6 y 3.8 que las variables pertenecientes al grupo Mean y Worst serán mas útiles para discriminar. En cuanto al grupo de variables que corresponden al grupo SE recogidas en la Figura 3.7 proporcionan información de que en general dichas variables que medimos tienen mayor dispersión en el grupo Maligno que en Benigno.

Por último, en el conjunto de todos los gráficos se observan observaciones dispersas y alejadas del centro de las elipses, lo que puede indicar la presencia de *Outliers*.

3.1.4. Outliers.

Los outliers multivariantes son observaciones que se consideran extrañas no por el valor que toman en una determinada variable, sino en el conjunto de aquellas. Su presencia tiene efectos todavía más perjudiciales que en el caso unidimensional, porque distorsionan

no sólo los valores de la medida de posición (media) o de dispersión (varianza), sino muy especialmente, las correlaciones entre las variables.

Los conjuntos de datos que muestran múltiples valores atípicos, están sujetos a los siguientes efectos[6]:

Efecto de enmascaramiento. Se dice que un outlier enmascara a un segundo outlier, si el segundo outlier puede considerarse como un valor extremo sólo por sí mismo, pero no en presencia del primer outlier. Así, después de la eliminación del primer outlier, en una segunda instancia, el otro punto se convierte en un valor atípico. El enmascaramiento se produce cuando un grupo de observaciones extremas sesga las estimaciones de la media y de la covarianza hacia él, y la distancia resultante del valor extremo a la media es pequeña.

Efecto de empantanamiento. Se dice que un outlier empantana una segunda observación, si esta última puede ser considerada como un valor extremo sólo bajo la presencia de la primera. En otras palabras, después de la eliminación del primer outlier, la segunda observación se convierte en un no-outlier. El empantanamiento ocurre cuando un grupo de valores extremos sesga las estimaciones de la media y de la covarianza hacia él y lejos de otros valores no periféricos, y la distancia resultante de estos casos a la media es grande, haciéndolos parecer como outliers.

Existen varias técnicas de eliminación de outliers multivariantes, en este trabajo usaremos la *distancia de Mahalanobis*.

La distancia de Mahalanobis es un criterio muy conocido que depende de los parámetros estimados de la distribución multivariada. Ésta describe la distancia entre cada punto de datos y el centro de masa. Cuando un punto se encuentra en el centro de masa, la distancia de Mahalanobis es cero y cuando un punto de datos se encuentra distante del centro de masa, la distancia es mayor a cero. Por lo tanto, los puntos de datos que se encuentran lejos del centro de masa se consideran valores atípicos.

Una vez detectados indicios, en nuestro conjunto de datos, de la existencia de posibles outliers, el procedimiento usual a seguir sería ver si ha ocurrido algún tipo de error con estas observaciones y comporbar si sería posible volver a medirlas. Al no ser posible en este caso, hemos optado por eliminar aquellos que, a nuestro juicio, pueden ser outliers. Habiendo comparado los resultados obtenidos con y sin ellos en los análisis realizados.

A la hora de extraer outliers debemos de tener en cuenta que nuestro conjunto de datos no es excesivamente grande por lo que la extracción de un alto número de observaciones podría afectar negativamente al estudio. Después de realizar varias pruebas hemos obtenido un mejor rendimiento de los clasificadores (teniendo en cuenta un número bajo de observaciones) para la extracción del 2 % de outliers sobre el conjunto total de datos, lo que equivale a 11 observaciones.

```
porcentaje.outliers <- 2

numero.outliers <- trunc(nrow(datos.I[,-1])
                  * porcentaje.outliers / 100)

maha.dist <- mahalanobis(datos.I[,-1],
      colMeans(datos.I[,-1]), cov(datos.I[,-1]))

maha.dist.order <- order(maha.dist, decreasing=TRUE)
```

```
filas.sin.outliers <- maha.dist.order[(numero.outliers+1)
                          :nrow(datos.I[,-1])]

es.outlier<- rep(TRUE, nrow(datos.I[,-1]))

es.outlier[filas.sin.outliers] <- FALSE

pch <- es.outlier * 1

outliers<- which(pch==1)
outliers
```

```
##  [1]   4  13  72  79 123 153 193 213 214 291 562
```

```
outliers.clasif<-datos.I[outliers,]
table(outliers.clasif[,"diagnosis"])
```

```
##
##    Benign Malignant
##         5         6
```

```
datos<-datos.I[-outliers,]
```

La salida anterior muestra las observaciones a la que corresponden los outliers. Por tanto se han extraído 11 outliers de los cuales, 5 corresponden a una diagnosis de benigno y 6 a una diagnosis de maligno.

3.2. Técnicas de clasificación.

Dividimos los datos en conjuntos de entrenamiento y test, asignando el 70 % de las observaciones al conjunto de entrenamiento y al 30 % restante al conjunto test. Hay que tener en cuenta que posteriormente, a la hora contruir los clasificadores, el conjunto de entrenamiento volverá a dividirse entre conjuntos de entrenamiento y validación, por eso este es considerablemente mayor que el conjunto de test.

```
dim(datos)
```

```
## [1] 558  10
```

```
nrows <- NROW(datos)
set.seed(1234)
index <- sample(1:nrows, 0.7 * nrows)

conjunto.entrenamiento <- datos[index,]
conjunto.test <- datos[-index,]
conjunto.entrenamiento<-as.data.frame(conjunto.entrenamiento)
conjunto.test<-as.data.frame(conjunto.test)
```

Volvemos a comprobar la frecuencia de las clases en los conjuntos obtenidos.

```
##
##    Benign Malignant
## 0.6384615 0.3615385
##
##    Benign Malignant
## 0.6130952 0.3869048
```

En el conjunto de entrenamiento el 63.84 % de las observaciones tienen una diagnosis benigno mientras que el 36.16 % restante tienen una diagnosis maligno. Para el conjunto de test el 61.30 % de las observaciones tienen una diagnosis benigno mientras que el 38.70 % restante tienen una diagnosis maligno.

3.2.1. Regresión logística.

Construiremos nuestro modelo de Regresión Logística a paritr de las funciones **trainControl()** y **train()** perteneciente a la librería *library(caret)*[9].

Con la función **trainControl** controlaremos los matices computacionales asociados de la función **train**. Entre los argumentos que tiene disponible dicha funcion usaremos:

- *method*: El método de remuestreo a llevar a cabo. En este caso usaremos K-Fold Cross-Validation por motivos que se indican previamente.
- *number*: El número de pliegues o el número de iteracciones.
- *savePredictions*= un indicador de cuantas de las predicciones de cada remuestreo se deben guardar. Los valores pueden ser “all”, “end” o “none”. También se puede usar un valor lógico que se convierte en “all” (para TRUE) o “none” (para FALSE). “end” guarda las predicciones para los parámetros de ajuste óptimos.
- *p*: el porcentaje del conjunto de entrenamiento.

La función *train* configura una cuadrícula de parámetros de ajuste, para un conjunto dado de métodos clasificación y regresión, se ajusta a cada modelo y calcula una medida de rendimiento basada en el remuestreo. Viene dada por la siguiente forma

$$train(x, y, method = "rf", ..., trCtrol = "")$$

donde los argumentos utilizados indican los siguiente:

- x: un cojunto de datos que contiene datos de entrenamiento donde las observaciones están en filas y las variables están en columnas.
- y: un vector numérico o factorial que contiene el resultado observado para cada muestra.
- *method:* un argumento que especifica que modelo de clasificación o modelo de regresión usar. En este caso el valor que le daremos será “lda” que corresponde al Análisis Discriminante Lineal.
- *trControl:* Una lista de valores que definen cómo actúa esta función, la cual definimos previamente.

Una vez aplicada la función *train*, obtendremos una lista de valores de la clase *train* que contiene:

- *modelType*: Un identificador del tipo del modelo.
- *results*: un marco de datos con la tasa de error y los valores de los parámetros de ajuste.
- *call*: la llamada de la misma función.
- *metric*: una cadena que especifica que indicador métrico se utilizará para seleccionar el modelo óptimo.
- *trControl*: la lista de los parámetros de control.
- *finalModel*: el modelo ajustado utilizando los mejores parámetros.
- *trainingData*: El conjunto de entrenamiento utilizado.

Para construir un modelo de regresión logística a partir de dichas funciones, indicaremos *method="glm"* y *family=binomial*. Si no indicaramos este último argumento, obtendríamos un modelo de regresión lineal.
Podemos usar la función *summary ()* para acceder a aspectos particulares del modelo ajustado, como los p-valores para los coeficientes del modelo.

```
ctrl <- trainControl(method = "repeatedcv",
number = 10, savePredictions = TRUE,
repeats = 1)

mod_reg_log <- train(diagnosis~.,  data=conjunto.entrenamiento,
method="glm", family="binomial", trControl = ctrl)
summary(mod_reg_log)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9300  -0.1115  -0.0131   0.0040   4.1828
##
## Coefficients:
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)           -4.381e+01  7.361e+00  -5.951 2.67e-09 ***
## texture_mean           4.463e-01  1.107e-01   4.033 5.51e-05 ***
## area_mean              1.974e-02  3.238e-03   6.096 1.09e-09 ***
## symmetry_mean          2.631e+00  2.406e+01   0.109   0.9129
## texture_se             1.012e+00  9.162e-01   1.104   0.2695
## smoothness_se         -1.734e+02  3.167e+02  -0.547   0.5841
## symmetry_se           -8.033e+01  9.978e+01  -0.805   0.4208
## fractal_dimension_se -2.096e+01  3.246e+02  -0.065   0.9485
## smoothness_worst       1.076e+02  3.573e+01   3.012   0.0026 **
## symmetry_worst         3.011e+01  1.579e+01   1.906   0.0566 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 510.353  on 389  degrees of freedom
## Residual deviance:  70.635  on 380  degrees of freedom
## AIC: 90.635
##
## Number of Fisher Scoring iterations: 9
```

Hemos obtenido 3 variables estadísticamente significativas las cuales son: *texture_mean, area_mean, smoothness_worst*. Por lo tanto, podemos concluir que existe una asociación entre dichas variables y la probabilidad de diagnosis. Por otra parte, para *symmetry_worst* hemos obtenido un p-valor=0.0566, ya que dicho valor está muy cercano al nivel de significación establecido $\alpha = 0.05$ la incluiremos también en el modelo.

Volvemos a obtener el modelo, eliminando las variables que no son estadísticamente significativas:

```
ctrl <- trainControl(method = "repeatedcv",
number = 10, savePredictions = TRUE, repeats = 1)

mod.lr <- train(diagnosis~texture_mean+
  area_mean+smoothness_worst+symmetry_worst ,
data=conjunto.entrenamiento, method="glm", family="binomial",
                 trControl = ctrl)
summary(mod.lr)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9050  -0.1274  -0.0189   0.0049   4.2996
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -41.78930    6.09522  -6.856 7.08e-12 ***
## texture_mean       0.46238    0.09440   4.898 9.67e-07 ***
## area_mean          0.01929    0.00300   6.430 1.28e-10 ***
## smoothness_worst  97.81244   21.35747   4.580 4.65e-06 ***
## symmetry_worst    23.86452    6.47462   3.686 0.000228 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 510.353  on 389  degrees of freedom
## Residual deviance:  74.346  on 385  degrees of freedom
## AIC: 84.346
##
```

```
## Number of Fisher Scoring iterations: 8
```

En este nuevo modelo, hemos obtenido que todas las variables son estadísticamente significativas. Tambíen hemos obtenido un AIC=84.346 para este segundo modelo ajustado mientras que para el primero obtuvimos un AIC=90.635. El criterio de información de Akaike (AIC) es una medida de la calidad relativa de un modelo estadístico, para un conjunto dado de datos. Como tal, el AIC proporciona un medio para la selección del modelo. Sin embargo, el criterio AIC no nos puede aportar información acerca de la calidad del modelo en un sentido absoluto. Si todos los modelos candidatos no se ajustan bien, AIC no dará ningún aviso de ello.

Interpretemos pues los coeficientes de la regresión:

Vemos que $\beta_1 = 0.46238$, lo cual indica que un aumento en *texture_mean* estará asociado con un aumento en la probabilidad de diagnosis como maligna. Para ser precisos, el aumento en una unidad de dicha variable estará asociado con el aumento en el log odds de diagnosis en 0.46238 unidades. Estos resultados pueden interpretarse de manera mecánica para el resto de variables. De manera general podemos ver que un aumento en el valor la variables estará asociado con el aumento en la probabilidad de diagnosis maligna.

La función confusionMatrix() nos proporciona una matriz de confusión para las clases observadas y predichas, ofreciéndonos a parte un conjunto de métricas para estas predicciones. Estas métricas han sido descritas previamente en el Punto 2.7 del Capítulo 2.
Obtenemos pues la matriz de confusión para un umbral de decisión de 0.5.

```
prediccionLR <- predict(mod.lr, conjunto.test, type="prob")
predclase= function(p,u)
{ ifelse(p>=u,"Malignant","Benign")
}
prediccion<-predclase(prediccionLR,0.5)
prediccion<-prediccion[,2]
prediccion<- as.factor(prediccion)
length(prediccion)
```

```
## [1] 168
```

```
confusionMatrix(prediccion,conjunto.test$diagnosis, positive = "Malignant")
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  Benign Malignant
##   Benign        97         2
##   Malignant      6        63
##
##                Accuracy : 0.9524
##                  95% CI : (0.9083, 0.9792)
##     No Information Rate : 0.6131
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9008
```

```
##  Mcnemar's Test P-Value : 0.2888
##
##             Sensitivity : 0.9692
##             Specificity : 0.9417
##          Pos Pred Value : 0.9130
##          Neg Pred Value : 0.9798
##              Prevalence : 0.3869
##          Detection Rate : 0.3750
##    Detection Prevalence : 0.4107
##       Balanced Accuracy : 0.9555
##
##        'Positive' Class : Malignant
##
```

Obtenemos la curva ROC a partir de la librería library(ROCR)[12]:

```
library(ROCR)
prediobjLR<-prediction(prediccionLR[,2],conjunto.test$diagnosis)
```

```
plot(performance(prediobjLR, "tpr","fpr"),
     main="Cuva ROC: Regresión Logística",
     xlab="Tasa de falsos positivos", ylab="Tasa de verdaderos positivos")
abline(a=0,b=1,col="blue",lty=2)
auc<- as.numeric(performance(prediobjLR,"auc")@y.values)
legend("bottomright",legend=paste("AUC=",round(auc,3)))
```

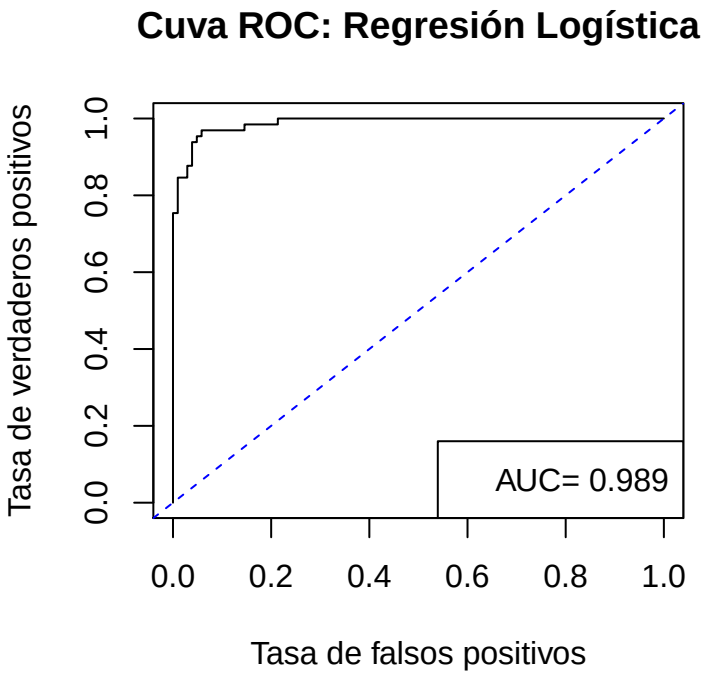

Figura 3.9: Curva ROC para el modelo de Regresión Logística.

La capacidad predictiva del modelo se ilustra elaborando la curva ROC. Nos interesa conocer la capacidad discriminativa de nuestro test diagnóstico, es decir, la habilidad para distinguir entre pacientes sanos y enfermos. Para ello, el parámetro que usaremos será el

área bajo la curva ROC (AUC), medida intrínseca al test diagnóstico. En la Figura 3.9 observamos un AUC=0.989, lo que nos indica que hemos obtenido un test muy bueno para discriminar pacientes con y sin la enfermedad a lo largo de todo el rango de umbrales de decisión posibles.

La curva ROC también nos permite obtener que umbral de decisión determina la sensibilidad y la especificidad más alta. Gráficamente, éste corresponde al punto de la curva ROC más cercano al ángulo superior-izquierdo del gráfico (punto 0,1). En la siguiente salida se muestra el umbral de decisión óptimo:

```
# Cálculo del punto de corte óptimo
pred <- prediction(prediccionLR[,2],conjunto.test$diagnosis)
perf <- performance(prediobjLR, "tpr","fpr")

cost.perf <- performance( pred, measure = "cost" )
opt.cut <- pred@cutoffs[[1]][which.min(cost.perf@y.values[[1]])]
opt.cut
```

```
## [1] 0.6334776
```

Por tanto el umbral de decisón óptimo es 0.6334.

Una de las ventajas que presentan los métodos paramétricos que estamos utilizando, es que podemos cambiar el umbral de decisión de los modelos, en otras palabras, podemos reducir uno de los dos tipos de errores que puede cometer un clasificador binario como este. Por lo tanto, nos interesa reducir el error de clasificar una célula maligna como benigna. El incoveniente que presenta este tipo de prácticas, es que el otro error se incrementará pero en este tipo de casos vale la pena. Para ello, usaremos el argumento *type="prob"* en la función *predict()* para que nos proporcione las probabilidades a posteriori de cada individuo. Posteriormente, crearemos una función que clasifique en función de nuestro umbral establecido. A pesar de haber obtenido que el umbral de decisión óptimo es 0.6334, nos interesa reducir todo lo posible la tasa de falsos negativos sin incrementar excesivamente la tasa de falsos positivos. Por esta razón, se ha decidido usar un umbral de decision igual a 0.5. Es decir, si la probabilidad a posteriori obtenido para el individuo es mayor o igual que 0.5 se clasificará como maligno, en caso contrario, como benigno. Como ya se eligió un umbral de decisón predeterminado igual a 0.5 , ya solo nos faltará analizar la matriz de confusión obtenida previamente.

Con una accuracy=0.9524, la probabilidad de clasificar correctamente un observación sobre el total es 0.9524. Luego con una sensitividad= 0.9692 y una especificidad=0.9417, la probabilidad de clasificar correctamente a un individuo enfermo es 0.9692 mientras que la probabilida de clasificar correctamente a un individuo sano es 0.9417. Una vez clasificados los individuos, la probabilidad de que un individuo con una diagnosis maligna este realmente enfermo (TFN) es 0.9130 mientras que la probabilidad de que un individuo con una diagnosis benigna este realmente sano (TVP) es 0.9798.
En relación a otros estudios, en este caso podemos considerar una prevalencia de la enfermedad para esta población bastante elevada, en consecuencia, el valor predictivo negativo tenderá a bajar pues al haber un mayor número de personas enfermas, aumenta el número de falsos negativos.

3.2.2. Análisis discriminante lineal

El Análisis Disciminante Lineal también lo llevaremos a cabo con las funciones **Control()** y **train()**, indicando en los argumentos de la función **train()** *method="lda"*.

```
ctrl <- trainControl(method = "repeatedcv", number = 10,
      savePredictions = TRUE, repeats = 1)

mod_fit <- train(diagnosis~.,  data=conjunto.entrenamiento,
  method="lda",  trControl = ctrl)
mod_fit
```

```
## Linear Discriminant Analysis
##
## 390 samples
##   9 predictor
##   2 classes: 'Benign', 'Malignant'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 1 times)
## Summary of sample sizes: 351, 351, 351, 351, 351, 352, ...
## Resampling results:
##
##   Accuracy  Kappa
##   0.961336  0.9124357
```

La salida previa indica el método utilizado para construir el modelo de clasificación. El tamaño del conjunto de observaciones usado, 390, el número de variables predictoras, 9, y el número de clases de la variables respuesta, 2. Por otro lado tenemos el método de remuestreo utilizado, en este caso validación cruzada con 10 capas, indicando el número de observaciones contenidas en cada conjunto de entrenamiento para entrenar el modelo, 351 aproximadamente, el resto de observaciones compondrían el conjunto de validación. Por último, tenemos métricas de la clasificación, obtenidas a partir de la validacion cruzada.

En la siguiente salida podemos ver como se han obtenido dichas métricas:

```
mod_fit$resample
```

```
##     Accuracy     Kappa    Resample
## 1  0.9230769 0.8245877 Fold01.Rep1
## 2  0.9743590 0.9433962 Fold02.Rep1
## 3  1.0000000 1.0000000 Fold03.Rep1
## 4  0.9743590 0.9433962 Fold04.Rep1
## 5  0.9487179 0.8849558 Fold05.Rep1
## 6  0.9210526 0.8224299 Fold06.Rep1
## 7  1.0000000 1.0000000 Fold07.Rep1
## 8  0.9743590 0.9433962 Fold08.Rep1
## 9  1.0000000 1.0000000 Fold09.Rep1
## 10 0.8974359 0.7621951 Fold10.Rep1
```

En esta salida podemos ver como para cada iteracción se ha obtenido una Accuracy y un índice de Kappa diferentes y posteriormente se ha obtenido la media de todos ellos,

que es el resultado que muestra la salida anterior.

```
mod_fit$finalModel
```

```
## Call:
## lda(x, grouping = y)
##
## Prior probabilities of groups:
##    Benign Malignant
## 0.6384615 0.3615385
##
## Group means:
##           texture_mean area_mean symmetry_mean texture_se smoothness_se
## Benign        17.45960  464.9004     0.1741478   1.208808   0.007181518
## Malignant     21.50638  951.0348     0.1910156   1.169859   0.006508780
##           symmetry_se fractal_dimension_se smoothness_worst symmetry_worst
## Benign     0.02032012          0.003456783        0.1256243      0.2686012
## Malignant  0.01927120          0.003767858        0.1442161      0.3193546
##
## Coefficients of linear discriminants:
##                                LD1
## texture_mean           0.090456380
## area_mean              0.004234128
## symmetry_mean         -2.125967490
## texture_se             0.041013710
## smoothness_se          1.092574803
## symmetry_se          -14.196726679
## fractal_dimension_se  53.874496507
## smoothness_worst      22.396329683
## symmetry_worst         8.555807851
```

Esta salida nos indica que $\widehat{\pi}_1 = 0.6385$ y $\widehat{\pi}_2 = 0.3615$; en otras palabras, el 63.85 % de las observaciones corresponden a una diagnosis benigna. También nos proporciona las medias por grupo para cada variables, y como ya se indicó previamente en el análisis descriptivo, se puede ver que existe cierta tendencia en las celulas malignas a tener mayores medidas de las medias. La salida de los *coeficientes discriminantes lineales* nos proporcionan la combinación lineal entre las variables predictoras que han sido usadas para crear la regla de decisión del Análisis Discriminante Lineal. En otras palabras, estos son los multiplicadores de "X=x" en (1.11)

$$\delta_k(\underline{x}) = \underline{x}^T \Sigma^{-1} \underline{\mu}_k - \frac{1}{2} \underline{\mu}_k^T \Sigma^{-1} \underline{\mu}_k + \log \pi_k$$

Si dicha multiplicación es relativamente grande, entonces LDA clasificará la observación como maligna, y si es relativamente pequeña, entonces la clasificará como benigna.

La función predict() nos proporcionará una lista predicciones sobre el conjunto de test a partir del modelo ajustado.

```
prediccionLDA1 <- predict(mod_fit, conjunto.test)
prediccionLDA1<-as.factor(prediccionLDA1)
prediccionLDA1
```

```
##   [1] Malignant Malignant Malignant Benign    Malignant Malignant Malignant
##   [8] Malignant Malignant Malignant Malignant Benign    Benign    Malignant
##  [15] Malignant Benign    Malignant Benign    Malignant Benign    Benign
##  [22] Malignant Benign    Malignant Malignant Benign    Benign    Benign
##  [29] Malignant Benign    Malignant Benign    Malignant Benign    Malignant
##  [36] Malignant Benign    Benign    Benign    Benign    Malignant Benign
##  [43] Benign    Malignant Benign    Malignant Malignant Benign    Malignant
##  [50] Malignant Malignant Malignant Malignant Benign    Benign    Malignant
##  [57] Benign    Benign    Malignant Benign    Benign    Benign    Malignant
##  [64] Benign    Malignant Malignant Malignant Benign    Benign    Malignant
##  [71] Benign    Malignant Benign    Benign    Benign    Benign    Benign
##  [78] Benign    Malignant Benign    Benign    Benign    Benign    Benign
##  [85] Benign    Benign    Malignant Benign    Malignant Benign    Benign
##  [92] Benign    Benign    Malignant Benign    Benign    Benign    Malignant
##  [99] Malignant Benign    Benign    Benign    Benign    Benign    Malignant
## [106] Benign    Benign    Benign    Benign    Benign    Benign    Malignant
## [113] Benign    Benign    Benign    Benign    Benign    Benign    Benign
## [120] Benign    Benign    Malignant Malignant Benign    Benign    Benign
## [127] Malignant Benign    Benign    Benign    Benign    Benign    Benign
## [134] Malignant Benign    Benign    Benign    Benign    Malignant Benign
## [141] Benign    Benign    Benign    Benign    Benign    Malignant Benign
## [148] Benign    Benign    Benign    Benign    Benign    Malignant Benign
## [155] Benign    Benign    Benign    Benign    Benign    Benign    Benign
## [162] Benign    Benign    Benign    Malignant Malignant Malignant Benign
## Levels: Benign Malignant
```

Obtenemos pues la matriz de confusión para un umbral de decisión de 0.5:

```
confusionMatrix(prediccionLDA1,conjunto.test$diagnosis, positive="Malignant")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  Benign Malignant
##   Benign       100        11
##   Malignant      3        54
##
##                Accuracy : 0.9167
##                  95% CI : (0.8641, 0.9537)
##     No Information Rate : 0.6131
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.8203
##  Mcnemar's Test P-Value : 0.06137
##
##             Sensitivity : 0.8308
##             Specificity : 0.9709
##          Pos Pred Value : 0.9474
##          Neg Pred Value : 0.9009
##              Prevalence : 0.3869
```

```
##          Detection Rate : 0.3214
##    Detection Prevalence : 0.3393
##       Balanced Accuracy : 0.9008
##
##        'Positive' Class : Malignant
##
```

Obtenemos por último la curva ROC[12]:

```
library(ROCR)
prediccionLDA <- predict(mod_fit, conjunto.test, type="prob")
prediobjLDA<-prediction(prediccionLDA[,2],conjunto.test$diagnosis)
```

```
plot(performance(prediobjLDA, "tpr","fpr"),
     main="Cuva ROC para LDA",
     xlab="Tasa de falsos positivos", ylab="Tasa de verdaderos positivos")
abline(a=0,b=1,col="blue",lty=2)
auc<- as.numeric(performance(prediobjLDA,"auc")@y.values)
legend("bottomright",legend=paste("AUC=",round(auc,3)))
```

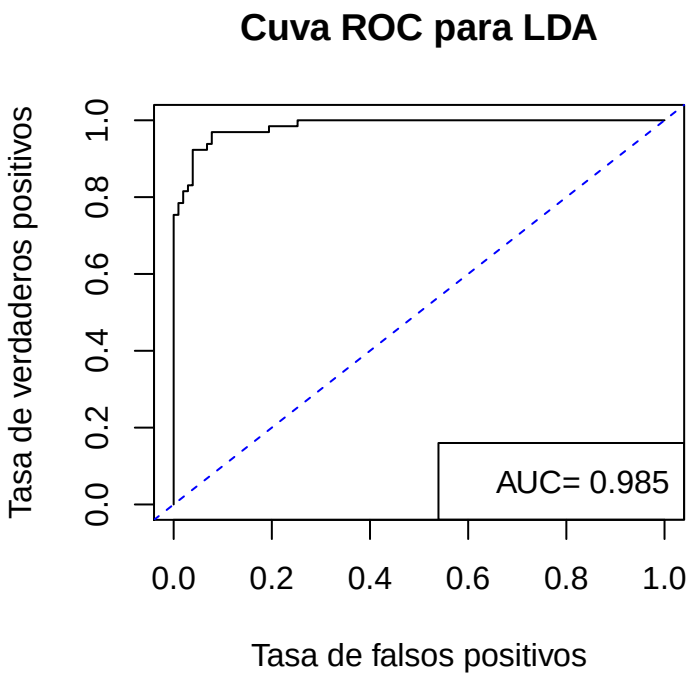

Figura 3.10: Curva ROC para Análisis Discriminante Lineal.

Volveremos a medir la capacidad discriminativa de nuestro test diagnóstico mediante el área bajo la curva ROC (AUC). En la Figura 3.10 observamos un AUC=0.985, lo que nos indica que hemos obtenido un test muy bueno para discriminar pacientes con y sin la enfermedad a lo largo de todo el rango de umbrales de decisión posibles. En la siguiente salida se muestra el umbral de decisión óptimo:

```
## [1] 0.275405
```

Por tanto el umbral de decisión óptimo es 0.2754.

A pesar de haber obtenido un umbral de decisión óptimo igual a 0.2754, hemos elegido un umbral igual a 0.15 por la misma razón indicada previamente en la sección del Modelo de Regresión Logística. Para dicho umbral, obtendríamos la siguiente salida:

```
prediccionLDA <- predict(mod_fit, conjunto.test, type="prob")
predclase= function(p,u)
{ ifelse(p>=u,"Malignant","Benign")
}
prediccion<-predclase(prediccionLDA,0.15)
prediccion<-prediccion[,2]
prediccion<- as.factor(prediccion)
confusionMatrix(prediccion,conjunto.test$diagnosis, positive = "Malignant")
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  Benign Malignant
##   Benign        94         2
##   Malignant      9        63
##
##                Accuracy : 0.9345
##                  95% CI : (0.8859, 0.9669)
##     No Information Rate : 0.6131
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.8647
##  Mcnemar's Test P-Value : 0.07044
##
##             Sensitivity : 0.9692
##             Specificity : 0.9126
##          Pos Pred Value : 0.8750
##          Neg Pred Value : 0.9792
##              Prevalence : 0.3869
##          Detection Rate : 0.3750
##    Detection Prevalence : 0.4286
##       Balanced Accuracy : 0.9409
##
##        'Positive' Class : Malignant
##
```

En la primera predicción con un umbral de decisión de 0.5 hemos obtenido una precisión=0.9167 , una sensibilidad=0.8308 y una especificidad=0.9709. Lo cual se puede traducir en que la probabilidad de clasificar una observación correctamente de manera global es 0.9167, la probabilidad de clasificar correctamente a un individuo enfermo es 0.8308y la probabilidad de clasificar correctamente a un individio sano es 0.9709.

Por otra parte, hemos obtenido un valor predictivo positivo= 0.9474 lo cual indica que para un individuo con una diagnosis maligna la probabilidad de qué realmente padezca la enfermedad es 0.9844 y un valor predictivo negativo=0.9009, es decir, la probabilidad de que un individuo con una diagnosis benigna no padezca la enfermedad es 0.9421.

Sin embargo, estos resultados para este tipo de estudios deberían mejorarse. Al tratarse de vidas humanas no podemos permitirnos clasificar incorrectamente a un individuo enfermo. Por lo tanto, nos resultará de mas interés usar el segundo modelo obtenido con un umbral de decisión de 0.15 a la hora de realizar futuras predicciones.

En la salida para un umbral de decisón de 0.15 hemos obtenido accuracy= 0.9345, una sensitividad=0.9692 y una especifidad=0.9126.

Por último, tenemos que la probabilidad de padecer la enfermedad si obtenemos un resultado positivo es 0.8750 y la probabilidad de estar sano es de 0.9792 si se ha obtenido un resultado negativo.

3.2.3. Support Vectors Machine.

Debido a que en este conjunto de datos, hay una gran cantidad de variables en relación con la cantidad de observaciones usaremos un kernel lineal, porque la flexibilidad adicional que resultaría del uso de un kernel polinomial o radial es innecesaria.

La desventaja que contamos en este caso con el uso de SVM es que, a diferencia de los métodos anteriores, la clasificación no se basa en un umbral de decisión sino que, clasificará una observación según el lado del hiperplano en el que se encuentre. Por tanto, no podremos reducir el error de clasificar una diagnosis maligna como una benigna como se hizo en el Análisis Discriminante Lineal y en la Regresión Logística.

Podemos realizar una validación cruzada con k capas utilizando la función *tune()* perteneciente a la librería library(e1071)[11]) para seleccionar la mejor opción de los parámetros *gamma* y *cost* para un SVM con kernel lineal.

```
library(caret)
library(e1071)
svm.tune <- tune(svm, diagnosis~., data = conjunto.entrenamiento,
                      ranges = list(gamma = 2^(-8:1), cost = 2^(0:4)),
                      kernel="linear")

svm.tune$best.model
```

```
##
## Call:
## best.tune(method = svm, train.x = diagnosis ~ ., data = conjunto.entrenamiento,
##     ranges = list(gamma = 2^(-8:1), cost = 2^(0:4)), kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##       gamma:  0.00390625
##
## Number of Support Vectors:  45
```

```
mean(svm.tune$performances[,3]) #Tasa de error
```

```
## [1] 0.03487179
```

```
1-mean(svm.tune$performances[,3]) #Accuracy
```

```
## [1] 0.9651282
```

La mejor elección de parámetros implica **cost** $= 1$ y **gamma** $= 0.00390625$. Cuánto mayor sea el parámetro gamma más curvo será el hiperplano de separación, esto podría delinear los datos demasiado bien y dar lugar a un sobreajuste. Por otra parte, el parámetro cost es el responsable del margen de SVM. Los observaciones que caen dentro de este margen no se clasifican como ninguna de las dos categorías. Cuanto menor sea el valor del parámetro cost, mayor será el margen.

Podemos acceder fácilmente a los errores de validación cruzada para cada uno de estos modelos usando el comando summary ():

```
erroresSVM<-summary(svm.tune)
erroresSVM
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##       gamma cost
##  0.00390625    1
##
## - best performance: 0.03076923
##
## - Detailed performance results:
##         gamma cost      error dispersion
## 1  0.00390625    1 0.03076923 0.02648194
## 2  0.00781250    1 0.03076923 0.02648194
## 3  0.01562500    1 0.03076923 0.02648194
## 4  0.03125000    1 0.03076923 0.02648194
## 5  0.06250000    1 0.03076923 0.02648194
## 6  0.12500000    1 0.03076923 0.02648194
## 7  0.25000000    1 0.03076923 0.02648194
## 8  0.50000000    1 0.03076923 0.02648194
## 9  1.00000000    1 0.03076923 0.02648194
## 10 2.00000000    1 0.03076923 0.02648194
## 11 0.00390625    2 0.03589744 0.02756327
## 12 0.00781250    2 0.03589744 0.02756327
## 13 0.01562500    2 0.03589744 0.02756327
## 14 0.03125000    2 0.03589744 0.02756327
## 15 0.06250000    2 0.03589744 0.02756327
## 16 0.12500000    2 0.03589744 0.02756327
## 17 0.25000000    2 0.03589744 0.02756327
## 18 0.50000000    2 0.03589744 0.02756327
## 19 1.00000000    2 0.03589744 0.02756327
## 20 2.00000000    2 0.03589744 0.02756327
## 21 0.00390625    4 0.03589744 0.02756327
## 22 0.00781250    4 0.03589744 0.02756327
## 23 0.01562500    4 0.03589744 0.02756327
## 24 0.03125000    4 0.03589744 0.02756327
## 25 0.06250000    4 0.03589744 0.02756327
## 26 0.12500000    4 0.03589744 0.02756327
## 27 0.25000000    4 0.03589744 0.02756327
## 28 0.50000000    4 0.03589744 0.02756327
## 29 1.00000000    4 0.03589744 0.02756327
## 30 2.00000000    4 0.03589744 0.02756327
```

```
## 31 0.00390625    8 0.03589744 0.02756327
## 32 0.00781250    8 0.03589744 0.02756327
## 33 0.01562500    8 0.03589744 0.02756327
## 34 0.03125000    8 0.03589744 0.02756327
## 35 0.06250000    8 0.03589744 0.02756327
## 36 0.12500000    8 0.03589744 0.02756327
## 37 0.25000000    8 0.03589744 0.02756327
## 38 0.50000000    8 0.03589744 0.02756327
## 39 1.00000000    8 0.03589744 0.02756327
## 40 2.00000000    8 0.03589744 0.02756327
## 41 0.00390625   16 0.03589744 0.02756327
## 42 0.00781250   16 0.03589744 0.02756327
## 43 0.01562500   16 0.03589744 0.02756327
## 44 0.03125000   16 0.03589744 0.02756327
## 45 0.06250000   16 0.03589744 0.02756327
## 46 0.12500000   16 0.03589744 0.02756327
## 47 0.25000000   16 0.03589744 0.02756327
## 48 0.50000000   16 0.03589744 0.02756327
## 49 1.00000000   16 0.03589744 0.02756327
## 50 2.00000000   16 0.03589744 0.02756327
```

Utilizamos de nuevo la función predict() junto con confusionMatrix() para obtener predicciones y evaluarlas:

```
pred.svm <- predict(svm.tune$best.model, conjunto.test)
confusion.svm <- confusionMatrix(pred.svm, conjunto.test$diagnosis, positive="Maligna
confusion.svm
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  Benign Malignant
##   Benign        96         4
##   Malignant      7        61
##
##                Accuracy : 0.9345
##                  95% CI : (0.8859, 0.9669)
##     No Information Rate : 0.6131
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8632
##  Mcnemar's Test P-Value : 0.5465
##
##             Sensitivity : 0.9385
##             Specificity : 0.9320
##          Pos Pred Value : 0.8971
##          Neg Pred Value : 0.9600
##              Prevalence : 0.3869
##          Detection Rate : 0.3631
##    Detection Prevalence : 0.4048
```

```
##       Balanced Accuracy : 0.9353
##
##        'Positive' Class : Malignant
##
```

Como en los modelos anteriores, vemos que se ha cometido un número muy bajo de errores. De hecho, esto no es sorprendente, debido a que hay bastantes variables en relación con la cantidad de observaciones lo cual implica que es fácil encontrar hiperplanos que separen completamente las clases.

Los resultados obtenidos se pueden interpretar de igual manera que en el Análisis Discriminante Lineal: hemos obtenido una accuracy igual a 0.9345 lo que se puede interpretar como que la probabilidad de clasificar correctamente un observación sobre el total es 0.9345.
La sensitividad tomar el valor 0.9385 mientras que la especificidad es 0.9320, es decir, la probabilidad de clasificar correctamente a un individuo enfermo 0.9385 mientras que la probabilida de clasificar correctamente a un individuo sano es 0.9320. Una vez clasificados los individuos, la probabilidad de que un individuo con una diagnosis maligna este realmente enfermo (VPN) es 0.8971 mientras que la probabilidad de que un individuo con una diagnosis benigna este realmente sano (VPN) es 0.9600.

Obtenemos por último la curva ROC:

```
svm1<-svm(diagnosis~., data = conjunto.entrenamiento,
                       gamma = 0.00390625, cost = 1,
                       kernel="linear",probability=TRUE)
pred.svm <- predict(svm1, conjunto.test,probability = T)
Scores.svm<-attributes(pred.svm)$probabilities[,1]
prediobjSVM<-prediction(Scores.svm,conjunto.test$diagnosis)
```

```
plot(performance(prediobjSVM, "tpr","fpr"),
     main="Curva ROC para SVM",
     xlab="Tasa de falsos positivos", ylab="Tasa de verdaderos positivos")
abline(a=0,b=1,col="blue",lty=2)
auc<- as.numeric(performance(prediobjSVM,"auc")@y.values)
legend("bottomright",legend=paste("AUC=",round(auc,3)))
```

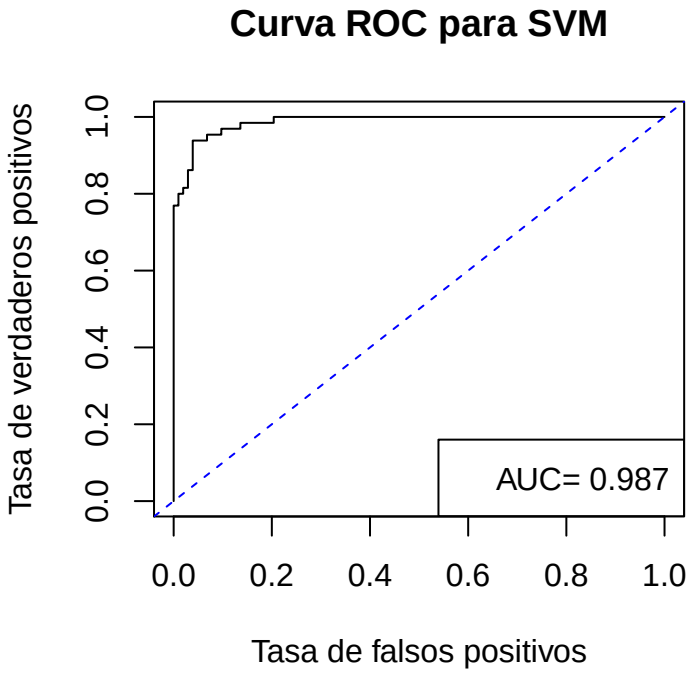

Figura 3.11: Curva ROC para SVM.

Al igual que en los modelos anteriores, obtenemos un bastante alto AUC=0.987. Lo que indica que la capacidad discriminativa de nuestro modelo es muy buena.

Conclusión:

Para comparar la capacidad discriminativa de dos tests diagnósticos es importante verificar un concepto metodológico de suma importancia: los tests a comparar deben ser medidos simultáneamente y aplicados sobre los mismos sujetos. Verificados estos requisitos, para comparar la capacidad discriminativa de dos tests diagnósticos deben compararse sus respectivas AUC, siendo más discriminativo el test con la mayor AUC[2].

Bibliografía

[1] Allaire, J., Xie, Y., McPherson, J., Luraschi, J., Ushey, K., Atkins, A., Wickham, H., Cheng, J., Chang, W. and Iannone, R. 2019. *Rmarkdown: Dynamic documents for r.*

[2] Cerda, J. and Cifuentes, L. 2011. *Revista chilena de infectología: Uso de curvas roc en investigación clínica.* Disponible en https://scielo.conicyt.cl/scielo.php?script=sci_arttext&pid=S0716-10182012000200003.

[3] Fernández, S. and Díaz, S. 2010. *Pruebas diagnósticas: Sensibilidad y especificidad.* Disponible en https://www.fisterra.com/mbe/investiga/pruebas_diagnosticas/pruebas_diagnosticas.asp.

[4] Fernández, S. and Ullibarri, G.L. de 1998. *Unidad de epidemología clínica y bioestadística: Curvas roc.* Disponible en https://www.fisterra.com/mbe/investiga/pruebas_diagnosticas/pruebas_diagnosticas.asp.

[5] Fox, J., Weisberg, S. and Price, B. 2019. *Car: Companion to applied regression.*

[6] García, J.A.M. and Uribe, I.A. 2013. *Técnicas para detección de outliers multivariantes.* Disponible en https://revistas.upb.edu.co/index.php/telecomunicaciones/article/viewFile/3308/2909.

[7] Hastie, T., Tibshirani, R. and Friedman, J. *The elements of statistical learning.* Springer.

[8] James, G., Witten, D., Hastie, T. and Tibshirani, R. *An introduction to statistical learning.* Springer.

[9] Kuhn, M. et al. 2019. *Caret: Classification and regression training.*

[10] Luque-Calvo, P.L. 2017. *Escribir un trabajo fin de estudios con r markdown.* Disponible en http://destio.us.es/calvo.

[11] Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., Leisch, F., Chang, C.-C. and Lin, C.-C. 2005. *E1071: Misc functions of the department of statistics, probability theory group.*

[12] Sing, T., Sander, O. and Lengauer, N.B. andThomas 2015. *ROCR:Visualizing the performance of scoring classifiers.*

[13] Taiyun Wei and, V.S., Levy, M., Xie, Y., Jin, Y. and Zemla, J. 2017. *Corrplot: Visualization of a correlation matrix.*

[14] Xie, Y. 2019. *Knitr: A general-purpose package for dynamic report generation in r.*