

Técnicas Multivariantes

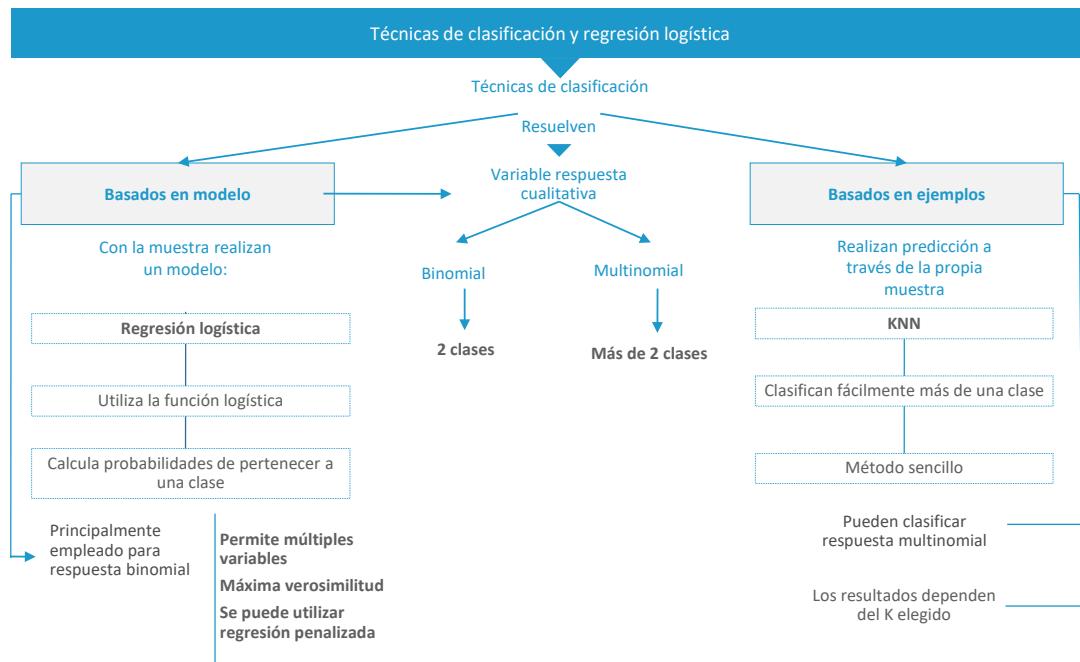
---

# Técnicas de clasificación

# Índice

<b>Esquema . . . . .</b>	<b>2</b>
<b>Ideas clave . . . . .</b>	<b>3</b>
7.1 Introducción y objetivos . . . . .	3
7.2 Problemas de clasificación . . . . .	4
7.3 Alternativas a la regresión lineal . . . . .	4
7.4 La regresión logística. . . . .	6
7.5 Método de los K vecinos más cercanos. . . . .	16
7.6 Aplicación en Python del método de los K vecinos más cercanos. . . . .	20
7.7 Referencias bibliográficas . . . . .	20
7.8 Ejercicios resueltos . . . . .	21

# Esquema



## 7.1 Introducción y objetivos

En este tema se va a abordar el problema de clasificación. La clasificación, es junto a la regresión uno de los principales problemas a estudiar del aprendizaje supervisado y por tanto forma uno de los pilares del aprendizaje automático o machine learning.

En los problemas de regresión, se estaba tratando de ajustar una variable que era cuantitativa, pero ésta también puede ser cualitativa. Por ejemplo, el color de los ojos es una variable cualitativa que puede tomar los valores de marrón, verde, azul, negro, etc. El proceso que predice este tipo de variables se conoce como **clasificación**. Predecir una variable cualitativa de una observación se puede definir como clasificarla, ya que estamos asignando una categoría o una clase a dicha variable. Por otro lado, a menudo, los métodos que se emplean para la clasificación, en un primer lugar predicen una probabilidad de pertenecer a cada una de las posibles categorías de la variable, como base para posteriormente realizar la clasificación. En este sentido, tienen un comportamiento similar a los métodos de regresión.

Existen multitud de técnicas distintas para realizar clasificación. En este tema, vamos a ver tres de las técnicas de clasificación sencillas más empleadas: la regresión logística y el método de los k-vecinos más cercanos, más conocido como KNN por sus siglas en inglés (*k-nearest neighbors*). Es interesante destacar, que tanto la regresión penalizada, que se vió en el Tema 6, como los árboles de decisión y métodos de ensamble asociados, que veremos en el Tema 8, se pueden emplear tanto para resolver problemas de regresión como de clasificación.

## 7.2 Problemas de clasificación

Los problemas de clasificación suceden muy a menudo. Es posible que incluso sean tanto o más frecuentes que los problemas de regresión. Veamos algunos ejemplos:

- ▶ Un paciente llega a la sala de urgencias de un hospital. En el triaje es necesario clasificarlo como urgente o no.
- ▶ Un servicio de banca online debe determinar si una transacción es fraudulenta o no.
- ▶ Un filtro anti-spam de un correo electrónico debe determinar si los correos entrantes son fraudulentos o no.

## 7.3 Alternativas a la regresión lineal

Para comprender porqué no es adecuada la regresión lineal podemos pensar en el siguiente ejemplo:

Una persona es llevada a un hospital por una ambulancia porque ha sufrido un desmayo. Basándose en sus síntomas esta persona puede ser categorizada como:

- ▶  $Y = 1$  si ha sufrido un ictus.
- ▶  $Y = 2$  si ha sufrido un infarto.
- ▶  $Y = 3$  si ha sufrido un ataque epiléptico.

Utilizando esta codificación, si realizaramos un ajuste por mínimos cuadrados para predecir  $Y$  a partir de un grupo de variables predictoras  $X_1, \dots, X_p$ , implicaría un orden

en la variable de respuesta, obteniéndose que el infarto está entre el ictus y el ataque epiléptico. Además, se estaría suponiendo que la distancia (numérica) entre el ictus y el infarto es la misma que entre el infarto y el ataque epiléptico, lo cual no tiene mucho sentido. De hecho, no existe ninguna razón para no haber ordenado los posibles valores de la variable respuesta  $Y$  como:

- ▶  $Y = 1$  si ha sufrido un ataque epiléptico.
- ▶  $Y = 2$  si ha sufrido un ictus.
- ▶  $Y = 3$  si ha sufrido un infarto.

Se obtendrían valores muy diferentes al caso anterior al realizar una regresión por mínimos cuadrados. En cambio, si la variable respuesta se puede ordenar de manera natural (por ejemplo, leve, moderado, severo) y se considera que la distancia entre categorías es la misma (la distancia entre leve y moderado es la misma que entre moderado y severo) sí que se podría codificar como 1, 2 y 3 la variable respuesta y aplicar una regresión lineal. Este ejemplo, rara vez se cumple, y por tanto, es necesario emplear otro tipo de modelos.

Cuando la variable respuesta es binaria (sólo tiene 2 valores posibles) se podría utilizar la estrategia de utilizar variables “dummy” (ver Tema 5). Por ejemplo, de manera similar al caso anterior, se podría codificar la variable respuesta  $Y$  como:

- ▶  $Y = 0$  si ha sufrido un ictus.
- ▶  $Y = 1$  si ha sufrido un infarto.

Entonces, se podría aplicar una regresión lineal a esta respuesta binaria y predecir como infarto los valores predichos de  $\hat{Y} > 0.5$  e ictus cuando  $\hat{Y} \leq 0.5$ . En este caso, la regresión lineal nos proporciona una estimación de la variable de salida. Además, si se intercambiara la codificación entre el ictus y el infarto, se seguirían obteniendo los

mismos resultados. No obstante, algunas de las estimaciones pueden estar fuera del intervalo  $[0, 1]$ , haciendo difícil la interpretación de los resultados como probabilidades. Asimismo, si la variable respuesta puede tomar más de 2 valores posibles, entonces no es posible ajustar la regresión lineal simple de manera sencilla para obtener un resultado satisfactorio. Por estas razones es necesario acudir a métodos alternativos más apropiados para clasificar respuestas cualitativas, siendo uno de los más sencillos y a la vez más utilizados la regresión logística.

## 7.4 La regresión logística

En el Tema 5 se presentaron brevemente los modelos lineales generalizados (GLM), que permitían relacionar mediante una función de enlace a un modelo de regresión lineal con una variable respuesta que no seguía una distribución normal, tal como se indica en la Ecuación 1.

$$f(Y) = \beta_0 + \beta_j \cdot X_j; \quad j = 1, \dots, m, \quad (1)$$

donde  $f(\cdot)$  es la función de enlace. Dentro de las aplicaciones de los GLM una de las más empleadas es la regresión logística, que se puede aplicar para resolver problemas de clasificación donde la variable respuesta es binomial (toma únicamente 2 valores) o multinomial (toma más de 2 valores).

### Regresión logística

Para explicar el funcionamiento de la regresión logística vamos a realizar un ejemplo real donde se emplea la regresión logística para resolver un problema de clasificación. El conjunto de datos *Default* (James et al. 2013), consiste en los datos obtenidos de un problema en el que se está interesado en predecir si un individuo fallará para pagar

la cuota de su tarjeta de crédito mediante unas variables predictoras. Las variables predictoras registradas en la base son:

- ▶ Registros de movimientos previos de una cuenta bancaria (Balance).
- ▶ Ingresos anuales (Ingresos).
- ▶ Si es estudiante o no lo es (Estudiante).

En la Figura 1 se muestran los valores de la base *Default*. Un primer análisis visual nos permite hacernos una idea de que cuanto mayor sea el balance más posibilidades hay de que se produzca un impago de la cuota de la tarjeta de crédito.

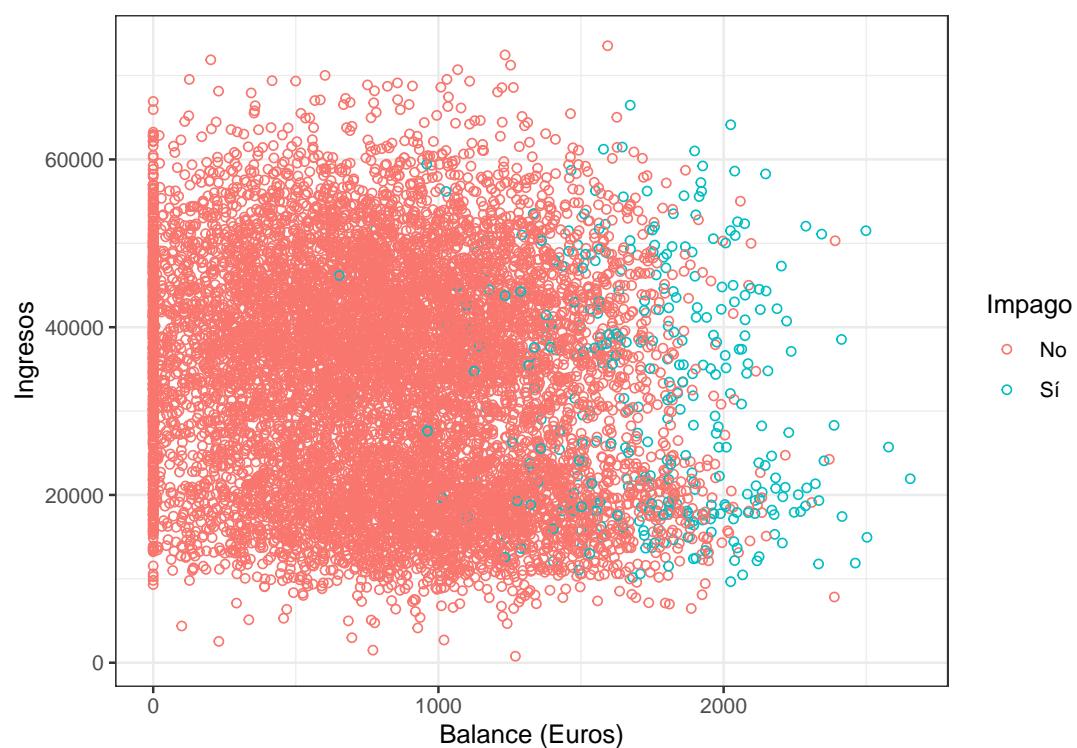


Figura 1: Conjunto de datos Default

En concreto, se quiere predecir el *Impago* ( $Y$ ) a partir del *Balance* ( $X_1$ ) y de los *Ingresos* ( $X_2$ ). Como  $Y$  no es una variable cuantitativa, la regresión lineal simple, como hemos visto en la sección anterior, no es apropiada. En su lugar, lo adecuado sería realizar una regresión logística. Podemos observar las diferencias entre aplicar un modelo de

regresión lineal simple y un modelo de regresión logística, ajustando en ambos casos únicamente por el *Balance*. En el modelo de regresión lineal simple se pueden obtener valores negativos de la **probabilidad** de que se produzca un impago, mientras que empleando la regresión logística, los posibles valores están contenidos en el intervalo  $[0, 1]$  tal y como se muestran en la Figura 2.

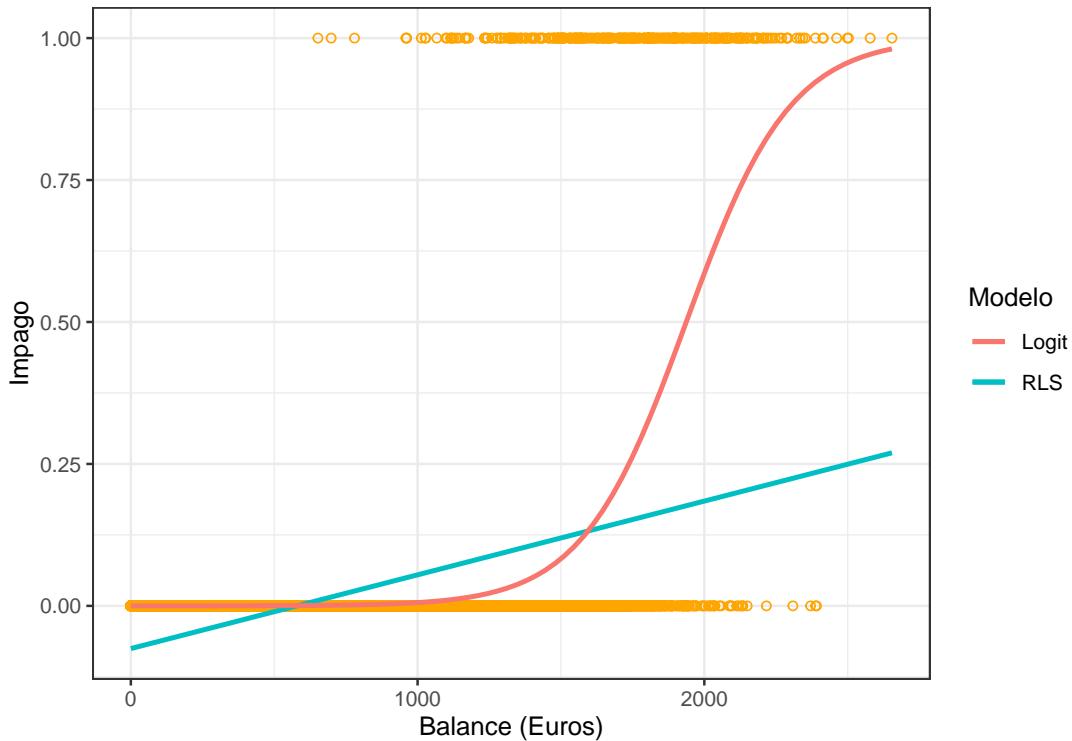


Figura 2: Modelos de regresión aplicados al conjunto de datos Default

Por lo tanto, la regresión logística sí que permite modelar la probabilidad de que se produzca el *Impago* para los diferentes valores del *Balance*. De este modo, los valores de  $Pr(\text{Impago} = S | \text{Balance})$ , los cuales los podemos abreviar con  $p(\text{Balance})$  siempre tomarán valores entre 0 y 1. Además, para cualquier valor de *Balance* se puede obtener una estimación de la probabilidad de *Impago* y también es posible clasificar a los distintos individuos mediante algún criterio escogido. Por ejemplo, se puede tomar el criterio de que para una determinada probabilidad, p.e.  $p(\text{Balance}) > 0.5$  se van a clasificar a los individuos como *Impago*. Si queremos ser capaces de capturar más individuos en los cuales se vaya a producir un *Impago*, este umbral se puede bajar, p.e.  $p(\text{Balance}) > 0.1$ , con lo cual se predecirán más sujetos como *Impago*.

## La función logística

Para conseguir obtener valores de probabilidad (entre 0 y 1) de la variable respuesta la regresión logística emplea la función logística, presentada en la Ecuación 2.

$$p(X) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}} \quad (2)$$

Operando un poco sobre la Ecuación 2 se puede obtener la Ecuación 3.

$$\frac{p(X)}{1 - p(X)} = e^{\beta_0 + \beta_1 X} \quad (3)$$

donde  $\frac{p(X)}{1 - p(X)}$  son los **odds**, y pueden tomar un valor entre 0 y  $\infty$ . Valores de los **odds** cercanos a 0 indican una baja probabilidad del suceso y valores de los **odds** muy altos indican una alta probabilidad. Por ejemplo, si la probabilidad de *Impago* fuese del 0.2, los **odds** valdrían:  $\frac{0.2}{1-0.2} = 1/4$ . De manera similar, si la probabilidad fuese de 0.9, los **odds** valdrían  $\frac{0.2}{1-0.2} = 9$ . Tomando logaritmos a ambos lados de la Ecuación 3 se obtiene la Ecuación 4.

$$\log\left(\frac{p(X)}{1 - p(X)}\right) = \beta_0 + \beta_1 X \quad (4)$$

donde la parte de la izquierda de la igualdad son los **log-odds**, o más comúnmente llamado **logit**. Si recordamos lo visto en el Tema 4, la interpretación del coeficiente  $\beta_1$  era el cambio que se producía en la variable respuesta  $Y$  ante un incremento unitario de la variable  $X$ . De manera similar, en la regresión logística, se puede interpretar al coeficiente  $\beta_1$  como el cambio que se produciría en el logit( $X$ ) ante el incremento de una unidad en  $X$ , o equivalentemente, que se multiplicarían los **odds** por  $e^{\beta_1}$ . Así pues, en la regresión logística  $\beta_1$  no se corresponde con el cambio de  $p(X)$  asociado con el

incremento de una unidad en  $X$ . La cantidad que  $p(X)$  cambia debido al cambio de una unidad en  $X$  depende del propio valor de  $X$ , ya que la relación entre  $X$  y  $p(x)$  no es lineal.

## Estimación de los coeficientes

Los coeficientes  $\beta_0$  y  $\beta_1$  son desconocidos y deben ser estimados utilizando datos de entrenamiento. Para realizar esta estimación, uno de los métodos más empleados es el de **máxima verosimilitud**. Este método se basa en estimar unos coeficientes de tal modo que la probabilidad predicha  $\hat{p}(x_i)$  para cada individuo se acerque lo máximo posible a su valor observado. Es decir, se buscan unos coeficientes  $\hat{\beta}_0$  y  $\hat{\beta}_1$  de modo que los individuos que su verdadero valor es 0 van a obtener un valor bajo de  $\hat{p}(x_i)$ , mientras que los individuos que su verdadero valor es 1 van a obtener un valor cercano a 1 de  $\hat{p}(x_i)$ . Matemáticamente, la función de la verosimilitud ( $\ell$ ) se representa tal y como se muestra en la Ecuación 5.

$$\ell(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i':y'_i=0} (1 - p(x'_i)). \quad (5)$$

donde se escoge la estimación de los coeficientes  $\hat{\beta}_0$  y  $\hat{\beta}_1$  de tal modo que se maximice la Ecuación 5. El método de máxima verosimilitud es un método muy general que se utiliza para ajustar una gran cantidad de modelos no lineales. De hecho, la técnica de ajuste por mínimos cuadrados ordinarios para la regresión lineal se puede considerar un caso particular de estimación por máxima verosimilitud.

Para el ejemplo que estamos tratando, los coeficientes estimados de la regresión logística utilizando máxima verosimilitud se muestran en la Tabla 1.

Tabla 1: Estimación de los coeficientes por máxima verosimilitud de la regresión logística para el modelo de *Impago* mediante la variable predictora *Balance*

Coeficiente	Estimación puntual	Error estándar	p. valor
Interceptación $\beta_0$	-10.6513	0.3611	<0.0001
Balance $\beta_1$	0.0055	0.0002	<0.0001

Se observa que el coeficiente  $\hat{\beta}_1 = 0.0055$ . Así pues, un incremento en el *Balance* supone un incremento la probabilidad de *Impago*. Más específicamente, un incremento de una unidad en el *Balance* repercute en un incremento de 0.0055 en el **logit** de *Impago*. Los valores del error estándar y del p.valor de los coeficientes se pueden interpretar del mismo modo que se explicó en el Tema 4. En este caso, se puede concluir que el *Impago* sí que está asociado con el *Balance*, puesto que el p.valor asociado es muy pequeño. Por otro lado, la interceptación del modelo, ajusta las probabilidades ajustadas promedio a la proporción de valores de *Impago* observados con valor = 1.

## Predicciones

Una vez que se han estimado los coeficientes del modelo de regresión logística es posible realizar una predicción sobre nuevos individuos. Por ejemplo, si quisieramos predecir la probabilidad de *Impago* de una persona con un *Balance* de 1000 Euros, se sustituiría los valores en los coeficientes y de X en la Ecuación 2 obteniéndose:

$$p(X) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}} = \frac{e^{-10.6513 + 0.0055 \cdot 1000}}{1 + e^{-10.6513 + 0.0055 \cdot 1000}} = 0.00576,$$

Lo cual significa que tendría una probabilidad de impago del 0.57 %. En cambio, para un *Balance* de 2000 Euros se obtendría una probabilidad de *Impago* de 0.586, lo que supondría un 58.6 %.

En un modelo de regresión logística, además de utilizar variables predictoras continuas, como el *Balance*, también se pueden emplear variables cualitativas, empleando la estrategia de convertirlas a variables del tipo **dummy**, tal y como se vió en el Tema 5. Por ejemplo, en el Conjunto de datos **default**, existe la variable cualitativa *Estudiante*, que se puede codificar como una variable **dummy** asignando el valor 1 para los *estudiantes* y el valor 0 para los *no estudiantes*. En la Tabla 2, se muestran los coeficientes obtenidos de aplicar un modelo de regresión logística con la variable predictora *Estudiante*.

Tabla 2: Estimación de los coeficientes por máxima verosimilitud de la regresión logística para el modelo de Impago mediante la variable predictora Estudiantes

Coeficiente	Estimación puntual	Error estándar	p. valor
Interceptación $\beta_0$	-3.5041	0.0707	<0.0001
Estudiante [Sí] $\beta_1$	0.4049	0.1150	0.0004

El coeficiente obtenido es positivo, lo cual nos indica que la asociación de ser estudiante con el *Impago* es positiva. Además, el p.valor es estadísticamente significativo. Este ajuste indica que los estudiantes tienden a tener una mayor probabilidad de impago que los no estudiantes:

$$p(\text{Impago}|\text{estudiante}) = \frac{e^{-3.5041+0.4049 \cdot 1}}{1 + e^{-3.5041+0.4049 \cdot 1}} = 0.0431,$$

$$p(\text{Impago}|\text{no estudiante}) = \frac{e^{-3.5041+0.4049 \cdot 0}}{1 + e^{-3.5041+0.4049 \cdot 0}} = 0.0292.$$

## La regresión logística múltiple

Una vez que ya se ha visto que se pueden ajustar tanto variables predictoras cuantitativas como cualitativas, el siguiente paso es ajustar más de una variable predictora en

el modelo. Análogamente a la regresión lineal múltiple, es posible ajustar un modelo mediante la Ecuación 6.

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X_1 + \dots + \beta_p X_p, \quad (6)$$

donde  $X = (X_1, \dots, X_p)$  son las  $p$  variables predictoras del modelo. La Ecuación 6 puede ser reescrita tal y como se muestra en la Ecuación 7.

$$p(X) = \frac{e^{\beta_0 + \beta_1 X_1 + \dots + \beta_p X_p}}{1 + e^{\beta_0 + \beta_1 X_1 + \dots + \beta_p X_p}}. \quad (7)$$

Para obtener el valor de la estimación de los coeficientes  $\beta_0, \beta_1, \dots, \beta_p$  se puede utilizar el método de máxima verosimilitud. Si aplicamos el modelo de regresión logística múltiple al ejemplo de **default**, se obtienen los resultados que se muestran en la tabla 3.

Tabla 3: Estimación de los coeficientes por máxima verosimilitud de la regresión logística para el modelo de Impago mediante las variables predictoras Balance, Ingresos y Estudiantes

Coeficiente	Estimación puntual	Error estándar	p. valor
Interceptación $\beta_0$	-10.8690	0.4923	<0.0001
Balance $\beta_1$	0.0057	0.0002	<0.0001
Ingresos $\beta_2$	0.0030	0.0082	0.7115
Estudiante [Sí] $\beta_3$	-0.6468	0.2362	0.0062

Los resultados indican que las variables *Balance* y *Estudiante* son estadísticamente significativas mientras que la variable *Ingresos* no lo es. Además, aparece un resultado que puede parecer un tanto sorprendente: ahora el valor de *Estudiante* es negativo, al contrario de lo que sucedía en el caso en el que únicamente se ajustaba por esta variable (ver Tabla 2). ¿Cómo puede ser esto posible? En la Figura 3 se ilustra esta aparente paradoja. En el panel de la izquierda, las líneas rosa y azul representan la

tasa de *Impago* promedio para estudiantes y para no estudiantes, respectivamente, en función de su *Balance*. El coeficiente negativo para *Estudiantes* en el modelo de regresión logística múltiple significa que para un valor fijo del *Balance* y del *Ingreso*, un estudiante tiene una menor probabilidad de producir impago que un no estudiante. Sin embargo, si nos fijamos en el panel de la derecha de la Figura 3, se observa que los estudiantes, en promedio tienen valores más altos de *Balance*, por lo que cuando ajustamos un modelo de regresión logística únicamente por la variable *Estudiante*, el ser estudiante tiene asociado una mayor probabilidad de producir *Impago*, pero realmente esto no es debido a ser estudiante, si no a tener un mayor *Balance*, que en ese modelo actuaría como una variable de confusión. Así pues, antes de ajustar un modelo (ya sea de regresión lineal o logística o de cualquier otro tipo) tenemos que asegurarnos de que hemos tenido en cuenta todas las posibles fuentes de variación de la variable de salida.

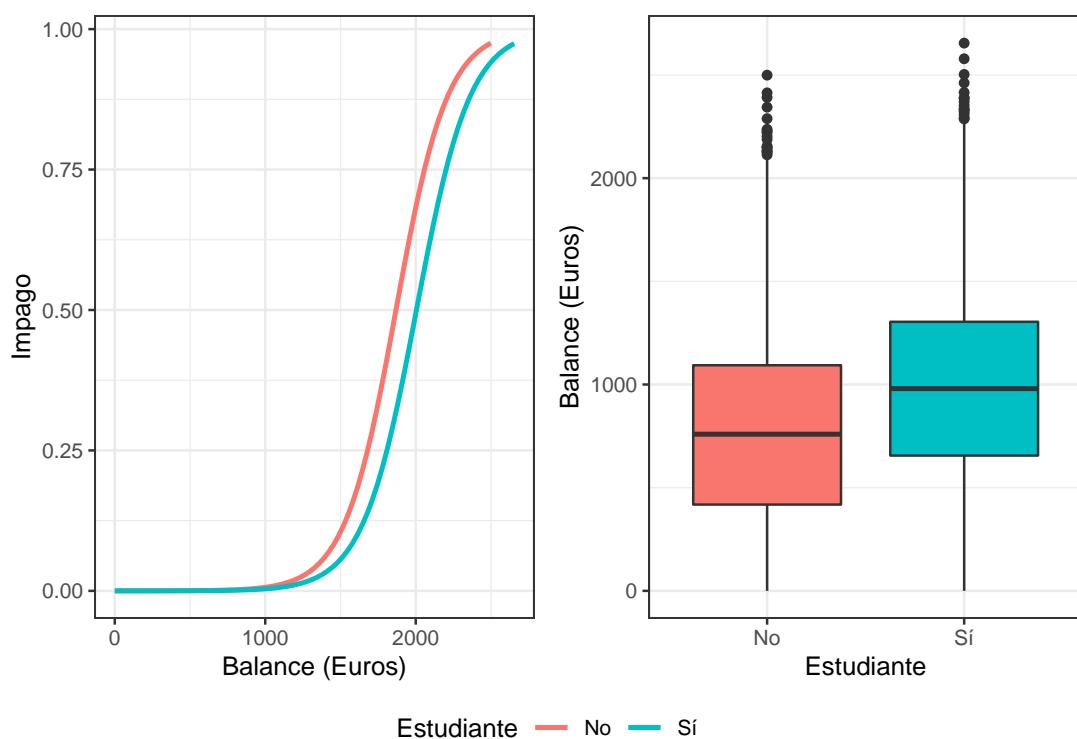


Figura 3: El fenómeno de la confusión en el conjunto de datos “Default”. A la izquierda la tasa de impago predicha frente al balance para estudiantes y no estudiantes. A la derecha un gráfico de cajas y bigotes del balance para estudiantes y no estudiantes

Para realizar predicciones, simplemente se deben sustituir los coeficientes por los estimados en el modelo y se pueden obtener valores predichos para nuevos individuos.

Por ejemplo, un estudiante con un *Balance* de 1500 Euros y unos *Ingresos* de 40 mil Euros obtendría una probabilidad de *Impago* de:

$$\hat{p}(X) = \frac{e^{-10.8690+0.00574\cdot1500+0.0030\cdot40-0.6468\cdot1}}{1+e^{-10.8690+0.00574\cdot1500+0.0030\cdot40-0.6468\cdot1}} = 0.058,$$

Y un no estudiante con las mismas características tendría una probabilidad:

$$\hat{p}(X) = \frac{e^{-10.8690+0.00574\cdot1500+0.0030\cdot40-0.6468\cdot0}}{1+e^{-10.8690+0.00574\cdot1500+0.0030\cdot40-0.6468\cdot0}} = 0.105.$$

Como se ha visto, la regresión logística es un método adecuado cuando se necesita ajustar un modelo con diferentes variables predictoras. No obstante, cabe recordar que cuando el número de variables predictoras es muy elevado, o estas presentan colinealidad, deberíamos acudir a otro tipo de técnicas, como puede ser la regresión logística penalizada aplicando a la regresión logística las técnicas que vimos en el Tema 6.

## Regresión logística multinomial

La regresión logística también permite resolver problemas cuando la variable respuesta es multinomial. La estrategia que se utiliza es que se fija una de las posibles clases de la variable respuesta (*la clase de referencia*) y se ajustan regresiones logísticas binomiales enfrentando cada una de las clases restantes con la clase de *referencia* (se tendrán  $n =$  número de clases - 1 regresiones en total). Entonces, para una observación se asigna la clasificación de la clase que tenga en la predicción que tenga un mayor valor de probabilidad. Aunque es posible aplicar esta extensión de la regresión logística, existen otros métodos más sencillos para tratar con la clasificación de variables multinomiales. En este tema se va a ver el método de los K vecinos más cercanos.

## 7.5 Método de los K vecinos más cercanos.

En algunas ocasiones, en vez de emplear una técnica basada en un modelo, se puede utilizar una técnica basada en las propias observaciones, es decir, utilizar una técnica basada en los ejemplos. En los problemas de clasificación, podemos emplear el método de los K-vecinos más cercanos, más conocido como KNN, por sus siglas en inglés.

El método, clasifica las nuevas observaciones basándose en las observaciones ya existentes, es decir, las observaciones de entrenamiento. Para ello, si se define  $K$  como un número entero positivo y siendo  $x_0$  una nueva observación que se quiere clasificar. Lo primero que se realiza en el método es identificar los  $K$  puntos más cercanos a  $x_0$ , representados por  $\mathcal{N}_0$ . Entonces, la probabilidad de que el valor de una nueva observación  $x_0$  sea  $j$  viene determinado por la fracción de los  $K$  puntos más cercanos que sean de la clase  $j$ . Es decir:

$$Pr(Y = j|X = x_0) = \frac{1}{K} \sum_{i \in \mathcal{N}_0} I(y_i = j), \quad (8)$$

donde la función  $I(\cdot)$  cuenta los valores de  $y_i$  que toman el valor  $j$ . Por último, se asigna a la observación  $x_0$  la clase que ha obtenido una mayor probabilidad. Este método se puede aplicar, por tanto, para problemas en los que la variable salida sea multinomial y existan más de 2 clases posibles.

Para comprender mejor como funciona el método, supongamos el siguiente ejemplo sencillo. Se tienen 2 categorías posibles: “apto”, “no apto” y 2 variables predictoras continuas ( $X_1$  y  $X_2$ ) que se corresponden con los resultados normalizados obtenidos en un test psicotécnico. En este caso, se tienen 12 observaciones (6 apto y 6 no apto) de las cuales sabemos su verdadero valor (ver Figura 4 ¿Cómo clasificaríamos una nueva observación  $x_0 : X_1 = 0.3, X_2 = 0.6$  aplicando el método de KNN con  $K = 3$ ?

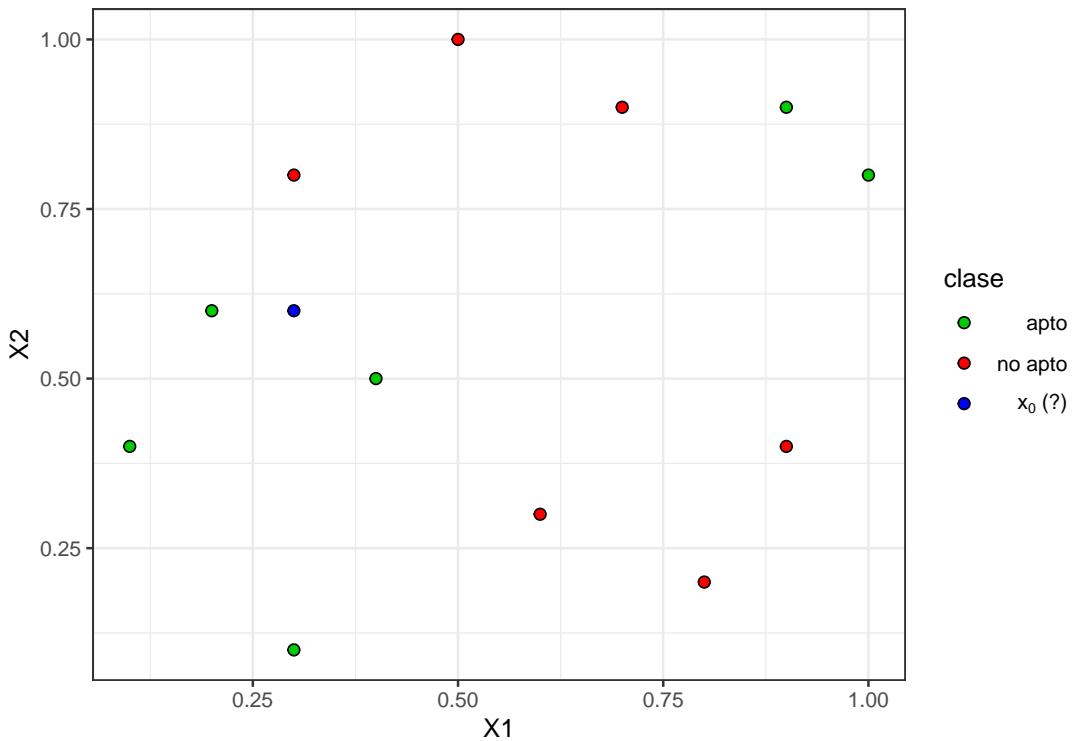


Figura 4: Aplicación del método de KNN para un ejemplo sencillo con  $K=3$

Si observamos la Figura, vemos que con  $K = 3$  deberíamos escoger los  $K = 3$  puntos más cercanos a  $x_0$  y calcular las probabilidades. En este caso, dado que dos de los tres puntos más cercanos son “apto” y el otro “no apto” se tendría que:

$$Pr(Y = \text{apto}|X = x_0) = 2/3;$$

$$Pr(Y = \text{no apto}|X = x_0) = 1/3.$$

Escogiendo el valor que obtiene una mayor probabilidad, decidiríamos que la nueva observación  $x_0$  sería apta, tal y como se muestra en la figura 5.

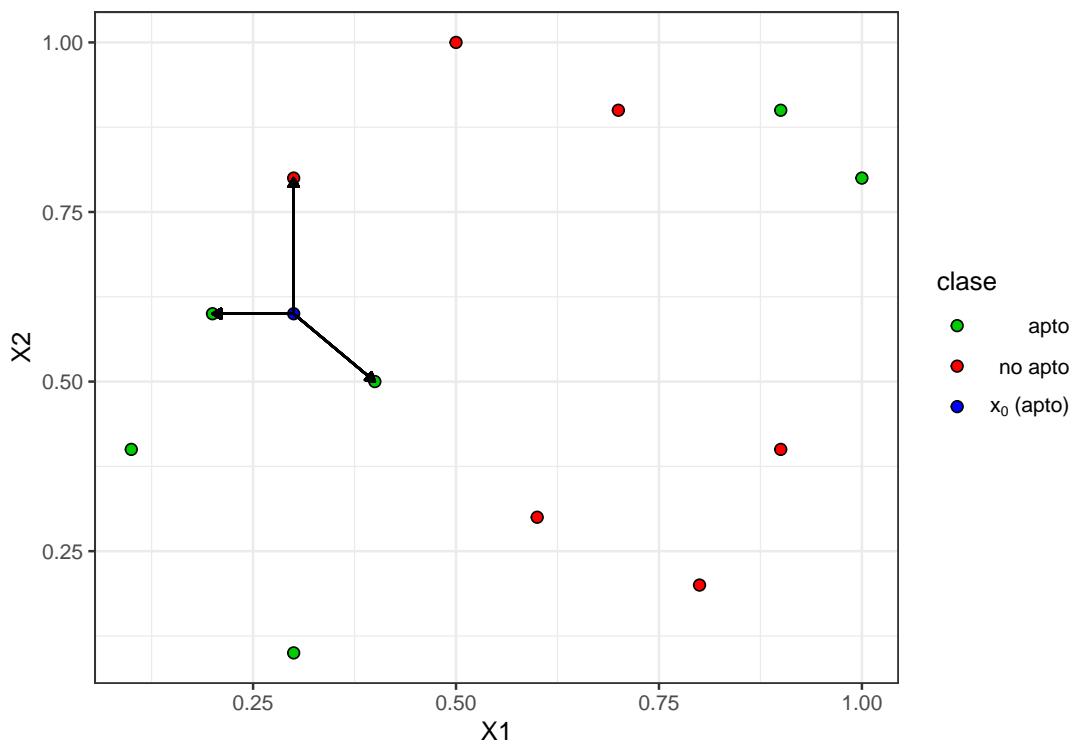


Figura 5: Aplicación del método de KNN para un ejemplo sencillo con  $K=3$ . Resolución

Además, es posible determinar las distintas regiones de clasificación sin necesidad de tener una observación nueva. En la figura 6 se ha estimado el valor de  $Y$  para todos los valores posibles de  $X_1$  y  $X_2$ .

```
## # A tibble: 2 x 2
##   clase     n
##   <fct> <int>
## 1 apto    4704
## 2 no apto 5509
```

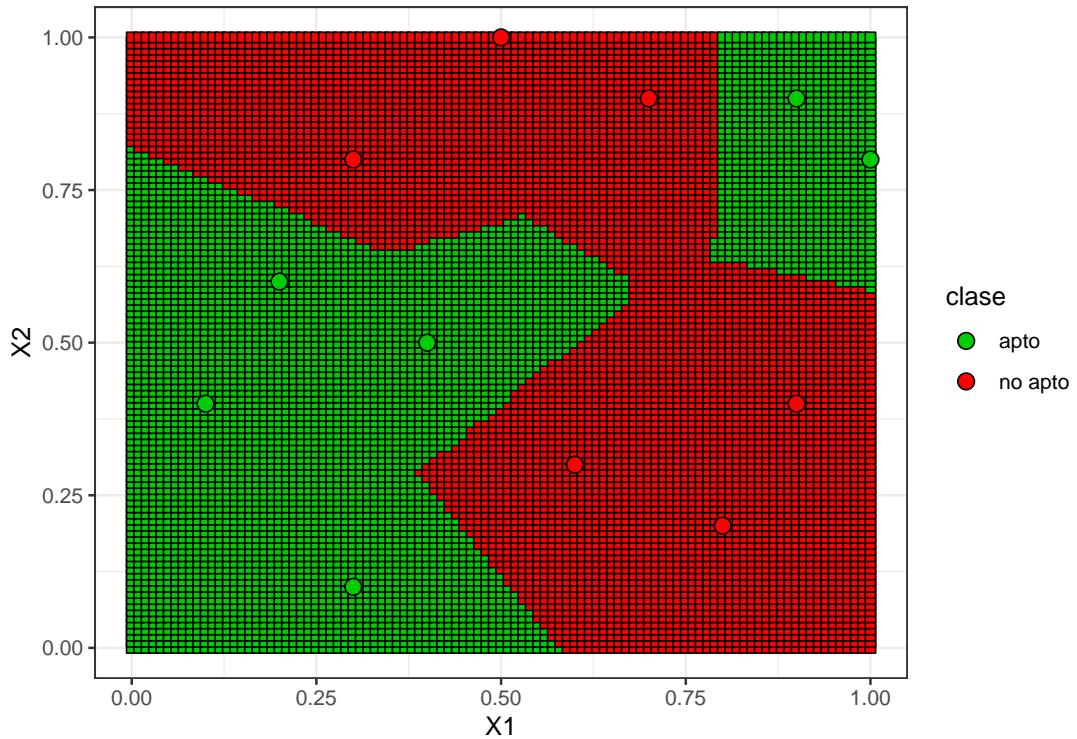


Figura 6: Obtención de las regiones “apto” y “no apto”

El valor de  $K$ , tiene un gran efecto sobre el clasificador: cuando  $K$  es un valor bajo, las fronteras entre las regiones pueden ser más flexibles, y en cambio, cuando  $K$  es grande, estas fronteras son más rígidas. Un buen modo de saber que  $K$  escoger es emplear VC para obtener al  $K$  óptimo que minimice el error de clasificación en nuestro problema.

Además, cuando la variable de salida es binomial, es conveniente que el valor de  $K$  seleccionado sea un número impar, para que no haya puntos en los que las probabilidades de ser de la clase  $j$  sea la misma. En cambio, cuando estamos ante un problema multinomial, es posible que se produzca algún empate. Para resolver este inconveniente se puede asignar un peso a los vecinos de modo que tengan un mayor peso los vecinos más cercanos en la estimación de la categoría de la observación.

## 7.6 Aplicación en Python del método de los K vecinos más cercanos.

Para aplicar el método de los K vecinos más cercanos es necesario realizar los siguientes pasos:

- ▶ *Estandarizar las variables predictoras.* Como se van a definir las distancias a las observaciones de entrenamiento es necesario que todas las variables tengan una misma escala para que sean comparables entre sí. Esto se va a conseguir estandarizando las variables predictoras. En Python, una de las maneras de realizar esto es mediante la función **preprocessing.scale()** de la librería *sklearn*.
- ▶ *Ajustar el parámetro  $K$*  que minimice el error de clasificación utilizando validación cruzada. Para ello se emplea la función **KNeighborsClassifier()** de la librería *sklearn.neighbors* donde el parámetro  $K$  se introduce como el argumento **n\_neighbors**.
- ▶ *Construir el clasificador* con el valor de  $K$  óptimo para poder predecir nuevas observaciones.

### Material audiovisual



Accede al vídeo: Técnicas de clasificación

## 7.7 Referencias bibliográficas

Geron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. 2 edition.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The elements of statistical learning: data mining, inference and prediction*. Springer, 2 edition.

James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). *An introduction to statistical learning : with applications in R*. New York : Springer, [2013] ©2013.

Peña, D. (1987). *Regresión y diseño de experimentos*.

## 7.8 Ejercicios resueltos

### Ejercicio 1.

En este ejercicio vamos a poner en práctica la regresión logística. Carga el dataset **California housing** y define una nueva variable: **oceano** que tomará el valor 1 si la variable **ocean\_proximity** es: **<1H OCEAN o NEAR OCEAN**, y 0 en cualquier otro caso (**INLAND, ISLAND o NEAR BAY**). Una vez creada esta variable:

- a) Ajusta un modelo de regresión logística con el precio de la casa como predictor de esta variable binomial que has creado (**oceano**).
- b) Si una nueva casa costase 250000\$, ¿en qué categoría de la variable **oceano** la clasificarías?

## Solución

- a) El primer paso, como viene siendo habitual es importar las librerías que se van a emplear en la resolución de los ejercicios:

```
# cargar librerias-----
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import linear_model
from sklearn.metrics import mean_squared_error, r2_score
from tabulate import tabulate
from plotnine import *
```

Cargamos el dataset.

```
# path que se va a crear en nuestro sistema-----
HOUSING_PATH = os.path.join("datasets", "housing")

# definir una funcion que cargue el csv en un dataframe-----
def load_housing_data(housing_path=HOUSING_PATH):
    csv_path = os.path.join(housing_path, "housing.csv")
    return pd.read_csv(csv_path)

# cargar base
housing = load_housing_data()
```

Una vez que el dataset se ha cargado, se crea la variable **oceano**.

```
housing['oceano'] = [1 if x == '<1H OCEAN' else
                     1 if x == 'NEAR OCEAN' else
                     0 for x in housing['ocean_proximity']]
```

# comprobacion de la variable oceano-----

```
print(housing.info())
```

```

## <class 'pandas.core.frame.DataFrame'>
## RangeIndex: 20640 entries, 0 to 20639
## Data columns (total 11 columns):
##   #   Column           Non-Null Count  Dtype  
##   --  --  
##   0   longitude        20640 non-null   float64 
##   1   latitude         20640 non-null   float64 
##   2   housing_median_age 20640 non-null   float64 
##   3   total_rooms      20640 non-null   float64 
##   4   total_bedrooms   20433 non-null   float64 
##   5   population       20640 non-null   float64 
##   6   households       20640 non-null   float64 
##   7   median_income    20640 non-null   float64 
##   8   median_house_value 20640 non-null   float64 
##   9   ocean_proximity  20640 non-null   object  
##   10  oceano            20640 non-null   int64  
## dtypes: float64(9), int64(1), object(1)
## memory usage: 1.7+ MB
## None

print(housing["ocean_proximity"].value_counts())

```

```

## <1H OCEAN      9136
## INLAND        6551
## NEAR OCEAN    2658
## NEAR BAY      2290
## ISLAND         5
## Name: ocean_proximity, dtype: int64

print(housing["oceano"].value_counts())

```

```

## 1    11794
## 0    8846

```

```
## Name: oceano, dtype: int64
```

En este caso, se va a definir como variable respuesta la variable que acabamos de crear **oceano**. Ahora se define el modelo de regresión logística en el que utilizaremos como variable predictor a la variable **median\_house\_value**. Para ello emplearemos una clase de tipo *LogisticRegression* de la librería *sklearn*. Además, la variable predictor se va a estandarizar. En el caso de la regresión simple no es necesario, pero se estandariza para que el coeficiente obtenido sea comparable con el obtenido en la regresión múltiple del ejercicio 2.

```
# importar clase-----
from sklearn.linear_model import LogisticRegression
# importar estandarizador.-----
from sklearn.preprocessing import StandardScaler
#definir variable respuesta
x = housing["median_house_value"].values.reshape(-1,1)
# estandarizamos x-----
scaler = StandardScaler()
scaler.fit(x)

## StandardScaler()

x_prepared = scaler.transform(x)
# separar variable respuesta del dataset-----
y = housing["oceano"].values.reshape(-1,1).ravel()
# ajustar el modelo-----
logistic_reg = LogisticRegression()
logistic_reg.fit(x_prepared, y)
# obtener coeficientes del modelo-----
# intercepto

## LogisticRegression()
```

```

print("El intercepto es = %.3f" % logistic_reg.intercept_)
# coeficientes de regresion

## El intercepto es = 0.377

print("El coeficiente es = %.3f" % logistic_reg.coef_)

## El coeficiente es = 0.895

```

El coeficiente es positivo, lo que indica que el precio de las casas está asociado con ser de la categoría 1 para la variable **oceano**.

- b) Se predice para una nueva casa con un valor de 250000 dólares. El primer paso es transformar este valor en su medida estandarizada.

```

# predecir clase-----
x_nueva = [[250000]]

# estandarizamos la observacion
x_nueva_prepared = scaler.transform(x_nueva)

# predecir nueva casa-----
clase_predicha = logistic_reg.predict(x_nueva_prepared)
print(clase_predicha)

## [1]

```

Para la casa con un valor de 250000 dólares se obtiene que la clase predicha es 1, es decir, que es más probable según el modelo de regresión logística que **oceano** sea 1. En concreto, se puede obtener esa probabilidad con la función **logistic\_reg.predict\_proba()**.

```

# calcular probabilidad-----
clase_predicha_prob = logistic_reg.predict_proba(x_nueva_prepared)

# sacar por pantalla las probabilidades de 0 y de 1-----

```

```

print("Pr(oceano = 0|median_housing_value = 250000) =
%.4f" % clase_predicha_prob[0,0],",
"\nPr(oceano = 1|median_housing_value = 250000) =
%.4f" % clase_predicha_prob[0,1],")

```

## Pr(oceano = 0|median\_housing\_value = 250000) = 0.3294 ,  
## Pr(oceano = 1|median\_housing\_value = 250000) = 0.6706 .

Aplicando la función **predict\_proba()**, se observa que la probabilidad de que la clase obtenida sea 1 es del 67 %.

## Ejercicio 2

A partir del conjunto de datos empleado en el ejercicio 1:

- Ajusta un modelo de regresión logística múltiple con todas las variables de **California housing** excepto **ocean\_proximity**, **longitude** y **latitude**, como predictores de la variable **oceano**.
- Que probabilidad tendría una casa de con las siguientes características de ser **oceano**.
  - ▶ **housing\_median\_age**: 35 años
  - ▶ **total\_rooms**: 5000
  - ▶ **total\_bedrooms**: 1200
  - ▶ **population**: 100000
  - ▶ **households**: 1000
  - ▶ **median\_income**: 10 ( decenas de miles de dólares)
  - ▶ **median\_house\_value**: 250000 dólares

## Solución

- a) Ahora, debemos construir la matriz de diseño eliminando la variable **ocean\_proximity** para poder realizar la regresión lineal múltiple.

```
# CONSTRUIR MATRIZ X-----
# quitar variable ocean_proximity-----
housing_num = housing.drop("ocean_proximity", axis=1)
housing_num = housing_num.drop("oceano", axis=1)
housing_num = housing_num.drop("longitude", axis=1)
housing_num = housing_num.drop("latitude", axis=1)
# importar el "imputador"-----
from sklearn.impute import SimpleImputer
# importar el "estandarizador"-----
from sklearn.preprocessing import StandardScaler
# importar la clase pipeline"-----
from sklearn.pipeline import Pipeline
# definir el pipeline-----
num_pipeline = Pipeline([
    ("imputador", SimpleImputer(strategy="median")),
    ("std_scaler", StandardScaler()),
])
# aplicar el pipeline-----
housing_num_tr = num_pipeline.fit_transform(housing_num)
# importar clases-----
from sklearn.compose import ColumnTransformer
# atributos de las variables numericas-----
num_attribs = list(housing_num)
# definir full pipeline-----
full_pipeline = ColumnTransformer([
    ("num", num_pipeline, num_attribs)
])
```

```

housing_prepared = full_pipeline.fit_transform(housing_num)

# definir matriz X
X = housing_prepared

# separar variable respuesta del dataset-----
y = housing["oceano"].values.reshape(-1,1).ravel()

# ajustar el modelo-----
logistic_reg_m = LogisticRegression()
logistic_reg_m.fit(X, y)

# obtener coeficientes del modelo-----
# intercepto

## LogisticRegression()

print("El intercepto es", logistic_reg_m.intercept_)

# coeficientes de regresion
# print("Los coeficientes de las variables:",
# list(housing_num.columns.values.tolist()), "son:",
# logistic_reg_m.coef_)

# podemos poner en una tabla estos valores-----
# poner nombres de las variables en una lista-----

## El intercepto es [0.40640537]

variables = housing_num.columns.values.tolist()

# poner los coeficientes en otra lista-----
coefs = logistic_reg_m.coef_.tolist()[0]

# definir las filas de la tabla-----
table = zip(variables, coefs)

# imprimir las tablas con nombres de las columnas (headers)-----
print(tabulate(table, headers = ["Variable", "Coeficiente"]))

## Variable          Coeficiente

```

```

## -----
## housing_median_age      -0.0732381
## total_rooms              -2.01059
## total_bedrooms           0.167547
## population               1.37391
## households                0.472922
## median_income             0.0478714
## median_house_value         1.17594

```

- b) Una vez que se ha obtenido el modelo de regresión logística múltiple, se pasa a predecir la clase de **oceano** de la nueva casa. Para ello, se debe estandarizar dicha observación.

```

# obtener observacion-----
housing_new = pd.DataFrame(np.array([[35, 2000, 1000, 1200, 900,
10, 250000]]), columns = variables)

```

```

# estandarizar-----
scaler2 = StandardScaler()
scaler2.fit(housing_num)

```

```

## StandardScaler()

housing_new_prepared = scaler2.transform(housing_new)
# valor estandarizado-----
print(housing_new_prepared)

```

```

## [[ 0.50539419 -0.29142558  1.09671839 -0.19910795  1.04744666
##     3.22634352  0.37388967]]

```

Se introduce la observación estandarizada en el modelo de regresión logística múltiple.

```

# predecir clase-----
clase_predicha=logistic_reg_m.predict(housing_new_prepared)
print(clase_predicha)

```

```
## [1]
```

Para esta casa se sigue obteniendo un valor predicho de 1, es decir, que es más probable según el modelo de regresión logística múltiple que **oceano** sea 1. En concreto, se puede obtener esa probabilidad con la función **logistic\_reg.predict\_proba()**.

```
# calcular probabilidad-----  
clase_predicha_prob=logistic_reg_m.predict_proba(housing_new_prepared)  
# sacar por pantalla las probabilidades de 0 y de 1-----  
print("Pr(oceano=0|housing_new)=%.4f" % clase_predicha_prob[0,0],  
",","\\nPr(oceano=1|housing_new)=%.4f" % clase_predicha_prob[0,1],  
".")  
  
## Pr(oceano = 0|housing_new) = 0.1240 ,  
## Pr(oceano = 1|housing_new) = 0.8760 .
```

Al añadir más variables a la predicción, la nueva observación ha aumentado su probabilidad de ser del tipo 1. Se observa que con el modelo de regresión logística múltiple tiene una probabilidad de ser de clase 1, obtenida con la función **predict\_proba()**, del 87.6 %.

### Ejercicio 3

Partiendo del conjunto de datos utilizado en los ejercicios 1 y 2, realiza:

- ▶ Un modelo de KNN con  $K = 10$  para clasificar la variable **oceano** a partir de las variables predictoras **median\_house\_value** y **total\_rooms**. Representa gráficamente las regiones obtenidas.
- ▶ ¿Qué sucede si utilizamos  $K = 1$ ? ¿Y si utilizamos  $K = 100$ ?
- ▶ ¿Cuál sería el valor de  $K$  óptimo?

## Solución

En este problema se trata de obtener un modelo que permita clasificar la variable oceano a través de las variables predictoras  $X_1 = \text{median\_house\_value}$  y  $X_2 = \text{total\_rooms}$ . El valor real de la variabre respuesta se muestra en la Figura 7.

```
#seleccionar predictores-----
predictores = housing_num[["median_house_value",
                           "total_rooms"]]

# pegar la clase a la base con los predictores estandarizados---
puntos = predictores.copy()
puntos["oceano"] = housing[["oceano"]]
puntos["oceano"] = puntos["oceano"].astype(object)

puntos.head()

##      median_house_value  total_rooms  oceano
## 0          452600.0        880.0      0
## 1          358500.0       7099.0      0
## 2          352100.0       1467.0      0
## 3          341300.0       1274.0      0
## 4          342200.0       1627.0      0

puntos.info()

# pintar la clase-----
```

```
## <class 'pandas.core.frame.DataFrame'>
## RangeIndex: 20640 entries, 0 to 20639
## Data columns (total 3 columns):
## #   Column           Non-Null Count  Dtype  
## --- 
## 0   median_house_value  20640 non-null   float64
## 1   total_rooms         20640 non-null   float64
```

```

##  2    oceano          20640 non-null   object
## dtypes: float64(2), object(1)
## memory usage: 483.9+ KB

(
  ggplot(puntos, aes(x = "median_house_value",
  y = "total_rooms", fill = "oceano")) +
  geom_point() +
  ylab("X2") +
  xlab("X1") +
  theme_bw() +
  theme(legend_position = "right",
subplots_adjust={'right': 0.8})
)

```

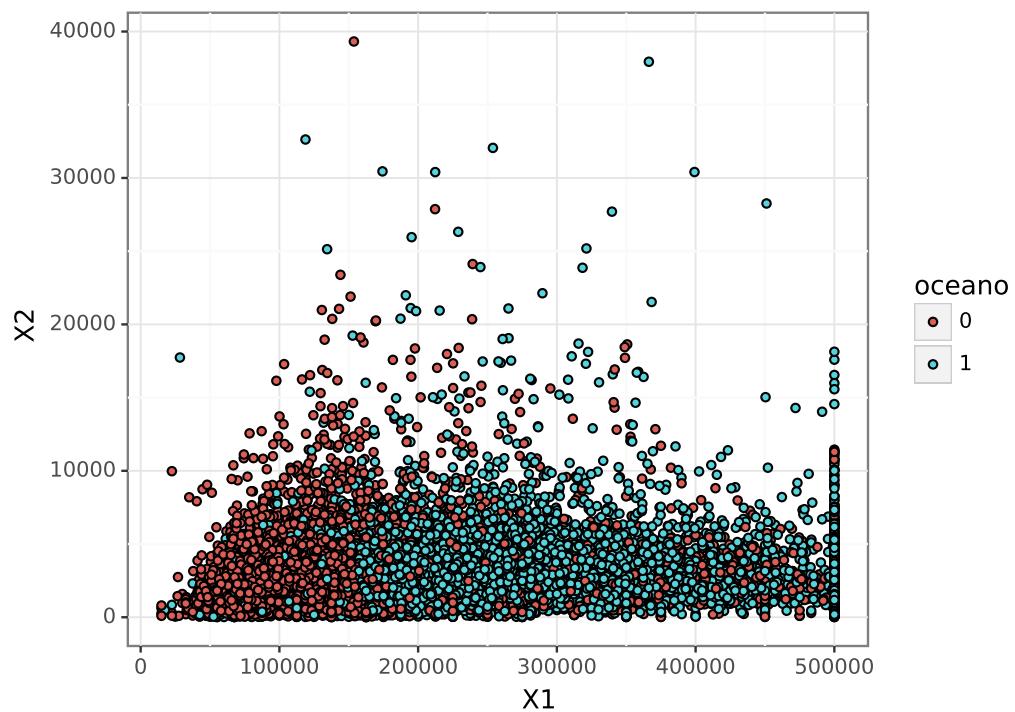


Figura 7: clases de la variable oceano

Para aplicar el método KNN es necesario definir las distancias a los puntos más cercanos de modo que todas las variables tengan una escala similar. Por lo tanto, se estandarizan

las variables predictoras.

```
# estandarizar predictores-----
scaler3 = StandardScaler()
scaler3.fit(predictores)

## StandardScaler()

predictores_prepared = scaler3.transform(predictores)
```

En el siguiente paso, se ajusta un modelo con  $K = 10$ .

```
from sklearn.neighbors import KNeighborsClassifier

# crear modelo con K = 10-----
model_KNN_10 = KNeighborsClassifier(n_neighbors = 10)

# utilizamos un modelo donde usamos todos los puntos-----
model_KNN_10.fit(predictores_prepared, y)

# definimos una rejilla de puntos
# valores de x1

## KNeighborsClassifier(n_neighbors=10)

min_x1 = min(predictores_prepared[:,0])
max_x1 = max(predictores_prepared[:,0])
x1_values = np.linspace(min_x1, max_x1, 101)

# valores de x2
min_x2 = min(predictores_prepared[:,1])
max_x2 = max(predictores_prepared[:,1])
x2_values = np.linspace(min_x2, max_x2, 101)

x1_grid, x2_grid = np.meshgrid(x1_values, x2_values)
x1_grid = x1_grid.flatten()
x2_grid = x2_grid.flatten()
x_grid = pd.DataFrame({'x1':x1_grid, 'x2':x2_grid})
```

```

# tipos-----
pred_10 = model_KNN_10.predict(x_grid)

regiones_10 = x_grid.copy()
# pegar los tipos predichos a la rejilla-----
regiones_10["clase"] = pred_10
regiones_10["clase"] = regiones_10["clase"].astype(object)
# ver cuantos hay de cada en la rejilla
print(regiones_10["clase"].value_counts())

# pintar la rejilla-----
## 1      7301
## 0      2900
## Name: clase, dtype: int64

(
    ggplot(regiones_10, aes(
        x = "x1",
        y = "x2",
        fill = "clase")) +
    geom_point(size = 1) +
    ylab("X2") +
    xlab("X1") +
    theme_bw() +
    theme(legend_position = "right",
subplots_adjust={'right': 0.8})
)

```

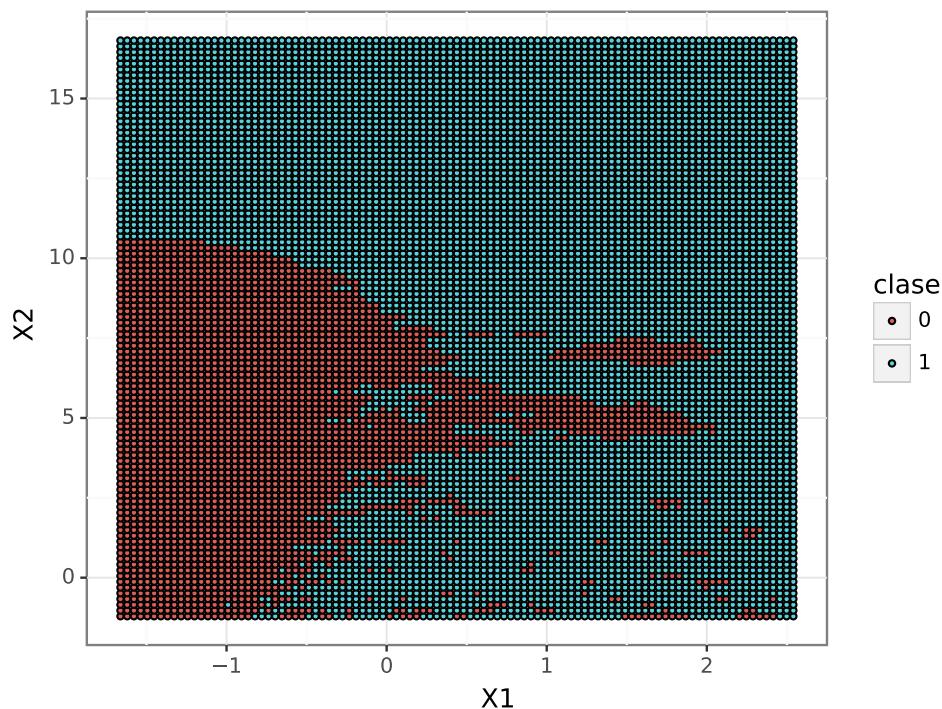


Figura 8: regiones con  $K = 10$

Se observa que existe una zona en la esquina inferior derecha donde la clase predicha sería 0 y el resto, con pequeñas excepciones serían regiones donde se predicaría la clase 1.

b) Se comprueba que regiones se obtendrían cambiando la  $K$  a  $K = 1$  y  $K = 100$ .

►  $K = 1$

En la Figura 9 se muestran las regiones obtenidas con  $K = 1$ .

```
from sklearn.neighbors import KNeighborsClassifier
# crear modelo con K = 10-----
model_KNN_1 = KNeighborsClassifier(n_neighbors = 1)
# utilizamos un modelo donde usamos todos los puntos-----
model_KNN_1.fit(predictores_prepared, y)
```

```

# tipos-----  

## KNeighborsClassifier(n_neighbors=1)  

pred_1 = model_KNN_1.predict(x_grid)  

regiones_1 = x_grid.copy()  

# pegar los tipos predichos a la rejilla-----  

regiones_1["clase"] = pred_1  

regiones_1["clase"] = regiones_1["clase"].astype(object)  

print(regiones_1["clase"].value_counts())  

# pintar la rejilla-----  

## 1      6592  

## 0      3609  

## Name: clase, dtype: int64  

(  

    ggplot(regiones_1, aes(  

        x = "x1",  

        y = "x2",  

        fill = "clase")) +  

    geom_point(size = 1) +  

    ylab("X2") +  

    xlab("X1") +  

    theme_bw() +  

    theme(legend_position = "right",  

        subplots_adjust={'right': 0.8})  

)

```

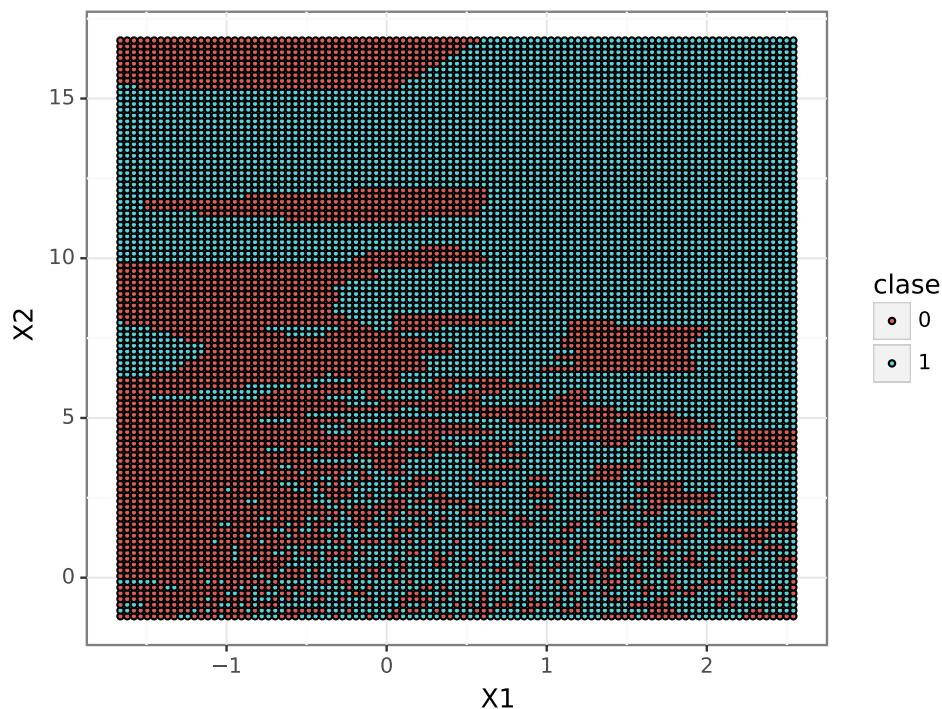


Figura 9: regiones con  $K = 1$

►  $K = 100$ .

En la Figura 10 se muestran las regiones obtenidas con  $K = 100$ .

```
from sklearn.neighbors import KNeighborsClassifier
# crear modelo con K = 10-----
model_KNN_100 = KNeighborsClassifier(n_neighbors = 100)
# utilizamos un modelo donde usamos todos los puntos-----
model_KNN_100.fit(predictores_prepared, y)

# tipos-----
```

```
## KNeighborsClassifier(n_neighbors=100)

pred_100 = model_KNN_100.predict(x_grid)

regiones_100 = x_grid.copy()
```

```

# pegar los tipos predichos a la rejilla-----
regiones_100["clase"] = pred_100
regiones_100["clase"] = regiones_100["clase"].astype(object)
print(regiones_100["clase"].value_counts())

# pintar la rejilla-----

## 1      7573
## 0      2628
## Name: clase, dtype: int64

(
    ggplot(regiones_100, aes(
        x = "x1",
        y = "x2",
        fill = "clase")) +
    geom_point(size = 1) +
    ylab("X2") +
    xlab("X1") +
    theme_bw() +
    theme(legend_position = "right",
    subplots_adjust={'right': 0.8})
)

```

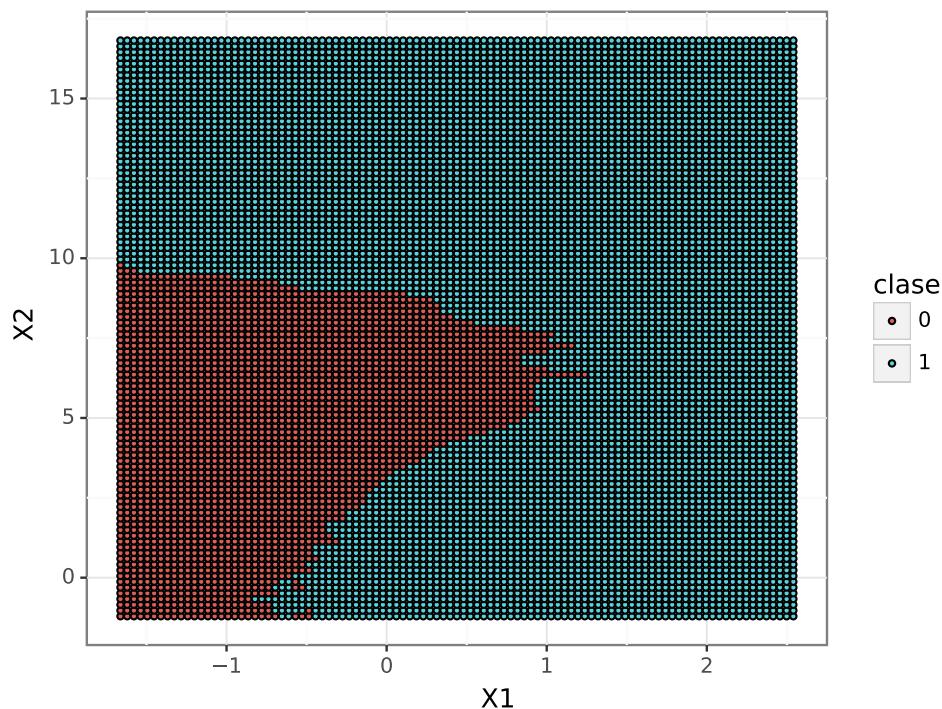


Figura 10: regiones con  $K = 100$

Se observa que cuando la  $K$  es baja se permite mucha flexibilidad en las regiones, pudiendo aparecer muchas regiones pequeñas. En cambio, cuando  $K$  es grande, las regiones están más definidas.

c)

Para escoger el valor de  $K$  óptimo emplearemos VC. Se entrenará al modelo para un número de observaciones y se comprobará con el resto cuál es la que minimiza el error. Para ellos se utilizará la función `cross_val_score` del módulo `sklearn.cross_validation` que permite obtener estas puntuaciones a partir del modelo. En este ejemplo, se va a coger como medida de validación cruzada el **accuracy**, que determina la proporción de casos bien clasificados sobre el total del modelo.

```
# Encontrar el valor optimo de K
from sklearn.model_selection import cross_val_score
# valores de K que vamos a probar-----
```

```

k_range = range(1, 100)

# inicializar vector de puntuaciones-----
k_scores = []

# bucle

for k in k_range:

    # ajustar el modelo con k vecinos

    knn = KNeighborsClassifier(n_neighbors=k)

    # obtener puntuaciones de VC

    scores = cross_val_score(knn, predictores_prepared, y,
        cv = 10, scoring = "accuracy")

    # Guardar puntuaciones en el vector

    k_scores.append(scores.mean())

```

En la Figura 11 se muestra como varía el **accuracy** con K. Se observa que a partir de 50, la **accuracy es prácticamente la misma. El máximo global (obtenido con la función np.argmax()\*\*)** se encuentra en  $K = 94$ .

```

# print(k_scores)

# pintar valores-----
plt.plot(k_range, k_scores)

plt.xlabel('Valor de K para KNN')

plt.ylabel('Accuracy de VC')

plt.show()

```

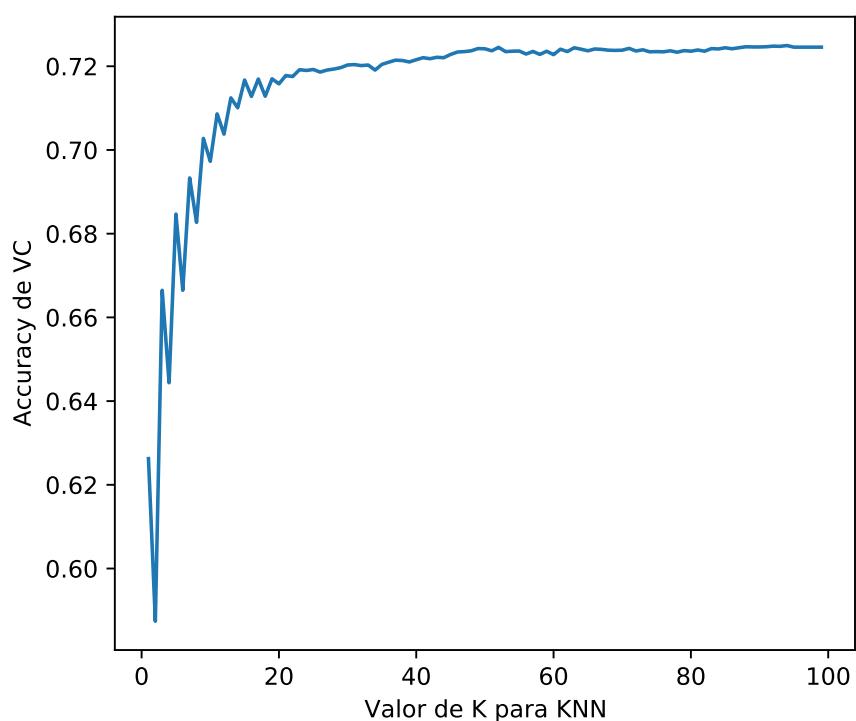


Figura 11: Accuracy vs valores de K

```
print("El valor de K que maximiza la accuracy es",
np.argmax(k_scores) + 1)
```

```
## El valor de K que maximiza la accuracy es 94.
```