Support Vector Machine (SVM)

En este apartado se aborda el modelo predictivo llamado Support Vector Machine (SVM). Se trata de un algoritmo de machine learning supervisado, y usado para funciones de clasificación y regresión.

El objetivo de SVM es el de encontrar un hiperplano que mejor separe las clases sobre los datos de nuestra variable respuesta. Para encontrar este hiperplano óptimo, el cúal separa bien nuestros datos y a la vez maximiza el margen (distancia entre hiperplano y puntos más cercanos a él de cada clase), en muchos casos hay que aumentar la dimensionalidad, llegando a dimensiones que no pueden representarse gráficamente, pero que sí permiten una correcta discriminación entre clases.

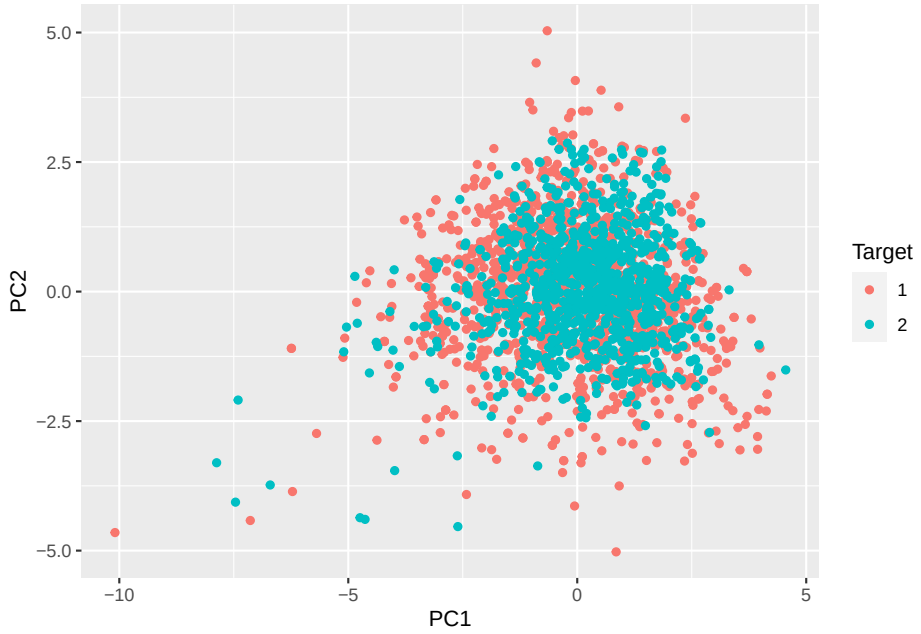
Obtener datos

Primero, se carga la base de datos balanceada, y se estandariza para evitar problemas derivados de la diferencia de escalas entre las variables. También se convierte la variable respuesta a factor, ya que se trata de un problema de clasificación.

Es trivial el hecho de que en este caso, los datos no pueden ser separados linealmente, dado el número de variables presentes, que son 16. Por esta razón, y dependiendo de la función kernel utilizada (y de sus parámetros), SVM utilizará espacios dimensionales transformados a partir del número de variables de la base de datos inicial. En otras palabras, el kernel define la manera como los datos se transforman en el nuevo espacio dimensional, y la dimensionalidad de dicho espacio resultante quedará determinada por los parámetros del kernel.

Para escoger el kernel se grafica las dos primeras dimensiones del PCA.

Figura 140: PC1 y PC2 respeto la variable Target



Como se puede ver, haciendo una representación del PCA de la primera dimensión y la segunda (PC1 y PC2) separando por colores la variable respuesta *TARGET*, siendo rojo (1) los clientes considerados como no morosos y azul (2) los potencialmente morosos, se puede concluir que parece que el grupo de no morosos

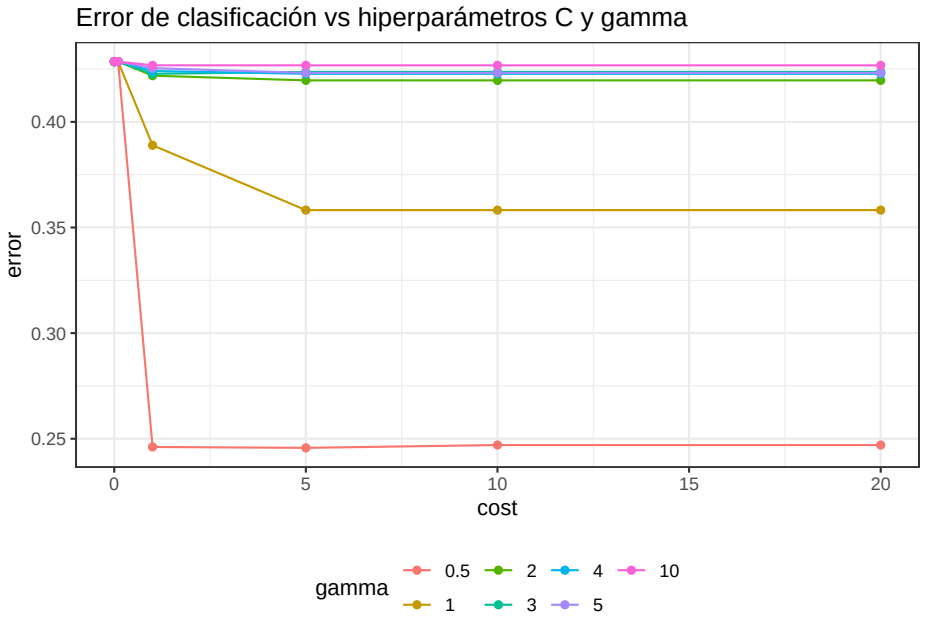
se encuntra dentro del grupo de morosos formando practicamente dos formas redondeadas en ambos casos. Al ver esta relación, se ha concluido que el kernel más adecuado en este caso era el radial.

Encontrar los valores de los hiperparámetros C (coste) y Gamma

Para poder determinar los valores óptimos de los hiperparámetros, se debe hacer una *validación cross-fold*. Una vez se encuentren estos valores óptimos, se ejecutará el SVM con ellos (es decir, el mejor modelo) para el conjunto de datos de validación. Esto nos permite obtener las métricas de rendimiento de este modelo sobre los datos, y poder compararlas con las de otros algoritmos de clasificación.

A continuación se usa la función *tune* para encontrar, dentro de una lista predefinida de valores para cada hiperparámetro, la combinación que resulte con el modelo con mejor rendimiento. En el gráfico siguiente se muestra una comparativa del error para cada combinación de hiperparámetros:

Figura 141: Representación del error en función de C y Gamma



Con el gráfico anterior se debería poder tener una idea, a nivel visual, del valor del hiperparámetro gamma que minimiza el error.En nuestro caso, la conclusión es muy clara, siendo el valor de 0.5 el que minimiza el error con mucha diferencia respecto a los demás valores. También se pueden obtener los valores óptimos con la siguiente instrucción seleccionando el objeto `best.parameters`, de la función `tune()`. El resultado es el siguiente:

Cuadro 59: Hiperparámetros óptimos

	cost	gamma
Valores	5	0.5

Por tanto, los valores óptimos para los dos hiperparámetros son los anteriores. A continuación se puede ver más información sobre el mejor modelo encontrado en el *cross-fold validation*.

Se puede apreciar como el número de vectores de soporte, aquellos puntos más cercanos al límite de decisión y que definen la posición del hiperplano, es elevado, contando con 2042.

Este fenómeno puede estar causado por varios motivos. Podría ser que la frontera de decisión entre las clases es inherentemente compleja, por lo que se necesitan muchos datos para representar de forma precisa la separación entre las dos clases. También podría haber ocurrido por *overfitting*, situación en la que el modelo tiene muy buen rendimiento con el conjunto de datos de entrenamiento, pero mal rendimiento con datos nuevos. El *overfitting* ocurre cuando el modelo es demasiado complejo en relación con la cantidad de datos disponibles para entrenarlo.

A pesar de esto, con el mejor modelo procedemos a hacer la predicción con la base de datos de validación, para obtener las métricas del rendimiento del modelo obtenido.

A continuación se muestra la matriz de confusión para evaluar dichas métricas y la capacidad predictiva del modelo. Hace falta aclarar que en esta matriz, el 0 representa a los "No morosos" y el 1 a los "Morosos". En los comentarios subsecuentes, "negativo" es 0 y "positivo" es 1.

Cuadro 60: Matriz de confusión del conjunto de validación

	Realidad	
	No moroso	Potencial moroso
Predicción		
No moroso	784	10
Potencial moroso	135	70

Las matrices de confusión se utilizan comunmente comúnmente en problemas de clasificación para evaluar el rendimiento de un modelo. En este caso, se trata con un modelo que clasifica clientes en dos categorías: no morosos (clase 0) y potencialmente morosos (clase 1). A continuación se interpretarán los elementos que componen esta matriz:

- **Verdaderos positivos (TP): 70.** Esto significa que 70 clientes fueron correctamente clasificados como morosos.
- **Verdaderos negativos (TN): 784.** Indica que 784 clientes fueron correctamente clasificados como no morosos.
- **Falsos positivos (FP): 135.** Representa a clientes que fueron incorrectamente clasificados como morosos cuando en realidad no lo eran.
- **Falsos negativos (FN): 10.** Muestra la cantidad de clientes que fueron incorrectamente clasificados como no morosos cuando en realidad sí lo eran .

Además, también vemos valores que nos indican la Sensibilidad , que es la proporción de positivos reales que se identificaron correctamente.

$Sensitivity = \frac{TP}{TP+FN} = \frac{70}{70+10} \approx 0{,}8750$

Tmbién la proporción de negativos reales que se identificaron correctamente como tales o especificidad.

$Specificity = \frac{TP}{TP+FP} = \frac{70}{70+135} \approx 0{,}8531$

Por último, destacar el valor de Accuracy o exactitud, que es la proporción de predicciones que el modelo clasificó correctamente.

$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} = \frac{70+784}{70+784+10+135} \approx 0{,}8549$

Para detectar si existe la presencia de overfitting en el modelo se debe comparar los valores de accuracy para el test y el train con los datos balanceados. El accuracy con los valores de training toma un valor de 0.7521, mientras que el accuracy con los valores de test toma el valor de 0.7286. Este hecho nos demuestra que, pese a realizar cross-validation, el modelo presenta un ligero overfitting sobre los datos train. Sin embargo, los resultados obtenidos son muy positivos para un modelo clasificatorio.