Reconocimiento de Patrones
TP3 - Support Vector Machines

Nicolás San Martín
Sebastián Sujarchuk
Eric Brandwein

Agosto 2020

1 Algoritmo Pegasus

El clasificador SVM puede implementarse mediante el algoritmo *Pegasus*. En este tp realizamos dicha implementación y la utilizamos para clasificar datos sintéticos provenientes de distintas distribuciones gaussianas multivariadas en $\mathbb{R}^2$ usando *soft margin*, es decir que no se requiere que los puntos en su totalidad resulten correctamente clasificados, sino que puede haber puntos que no lo estén.

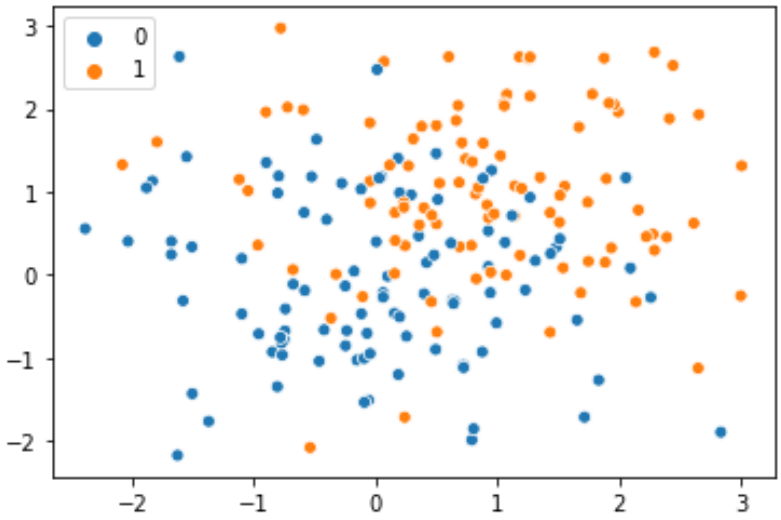

Figure 1: Dataset generado con dos gaussianas multivariadas con separación de 1 desvío estándar.

Probamos esta implementación con tres datasets diferentes, en los cuales las distribuciones gaussianas utilizadas para generar los puntos de las dos clases tienen distinto grado de separación. Dado que estos conjuntos de datos provienen de distribuciones gaussianas, la separación entre las clases se varió modificando las distancias entre las medias respectivas. Así, por ejemplo, en un caso la distancia entre las medias era de 1 desvío estándar, en otro de 3 y en el último caso de 5. Tales conjuntos pueden observarse en las Figuras 1, 2 y 3.

Para probar el algoritmo, entrenamos con un set de entrenamiento y luego clasificamos un set de validación, obteniendo el resultado que se observa en las Figuras 4, 5 y 6. En las mismas se puede ver la recta de separación y las rectas que junto a ella forman la "calle". En los gráficos de la columna izquierda se ven todos los puntos del dataset correspondiente coloreados según su clase, y en la columna derecha los resultados de la clasificación de un conjunto de testing. Los círculos representan los puntos bien clasificados, y las cruces, los mal clasificados. Además, el color representa la clase real. Por ejemplo, una cruz naranja representa un punto de la clase 1 que fue clasificado como la 0.

Asimismo se utilizaron distintos valores para el hiperparámetro C, siendo C el factor que multiplica las *slack variables* que se introducen en el modelo al incluir *soft margin* (y por ende afecta a la penalidad que se les da a los

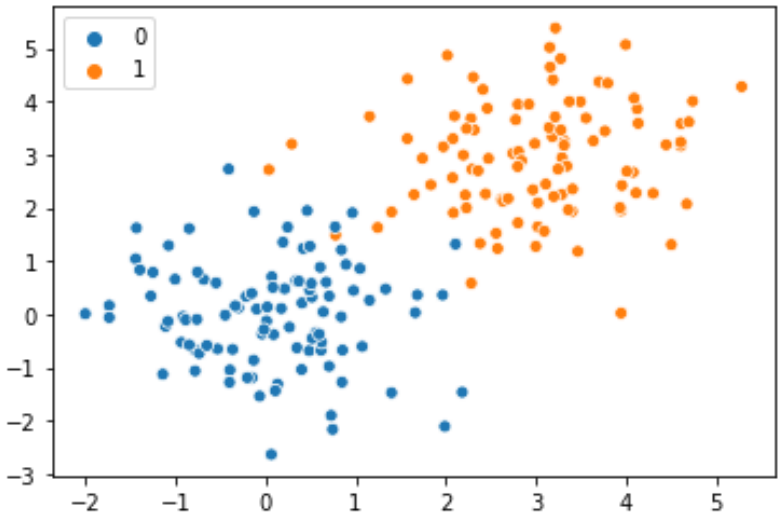

Figure 2: Dataset generado con dos gaussianas multivariadas con separacion de 3 desvíos estándar.

datos mal clasificados). También indicamos los valores de λ, donde $\lambda = \frac{1}{nC}$ y n el tamaño del dataset.

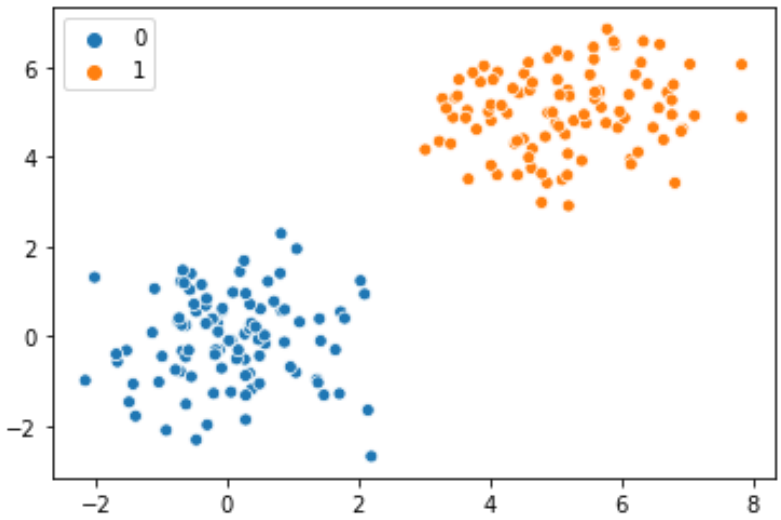

Figure 3: Dataset generado con dos gaussianas multivariadas con separacion de 5 desvíos estándar.

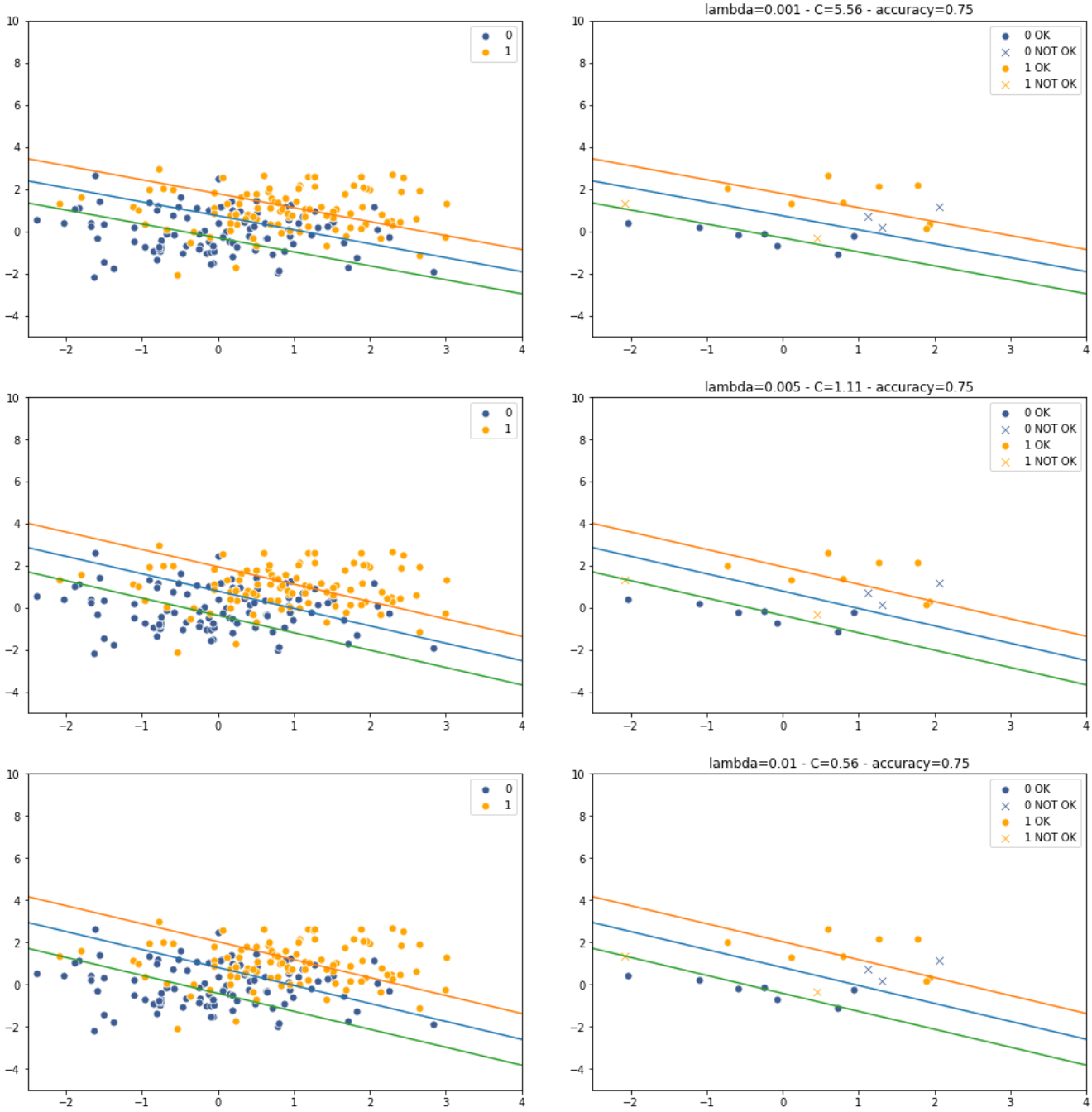

Figure 4: Resultado dataset con separación 1.

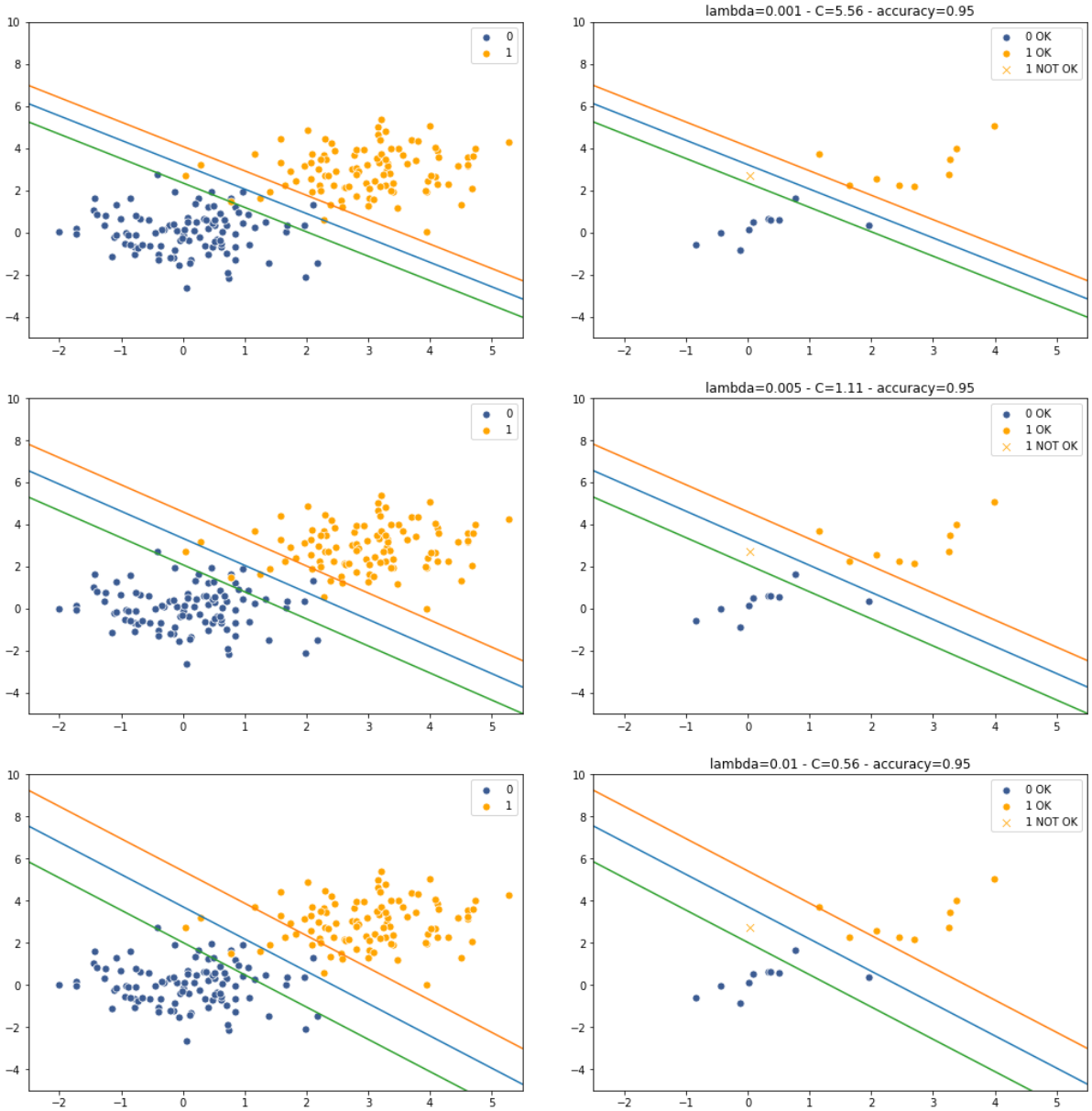

Figure 5: Resultado dataset con separación 3.

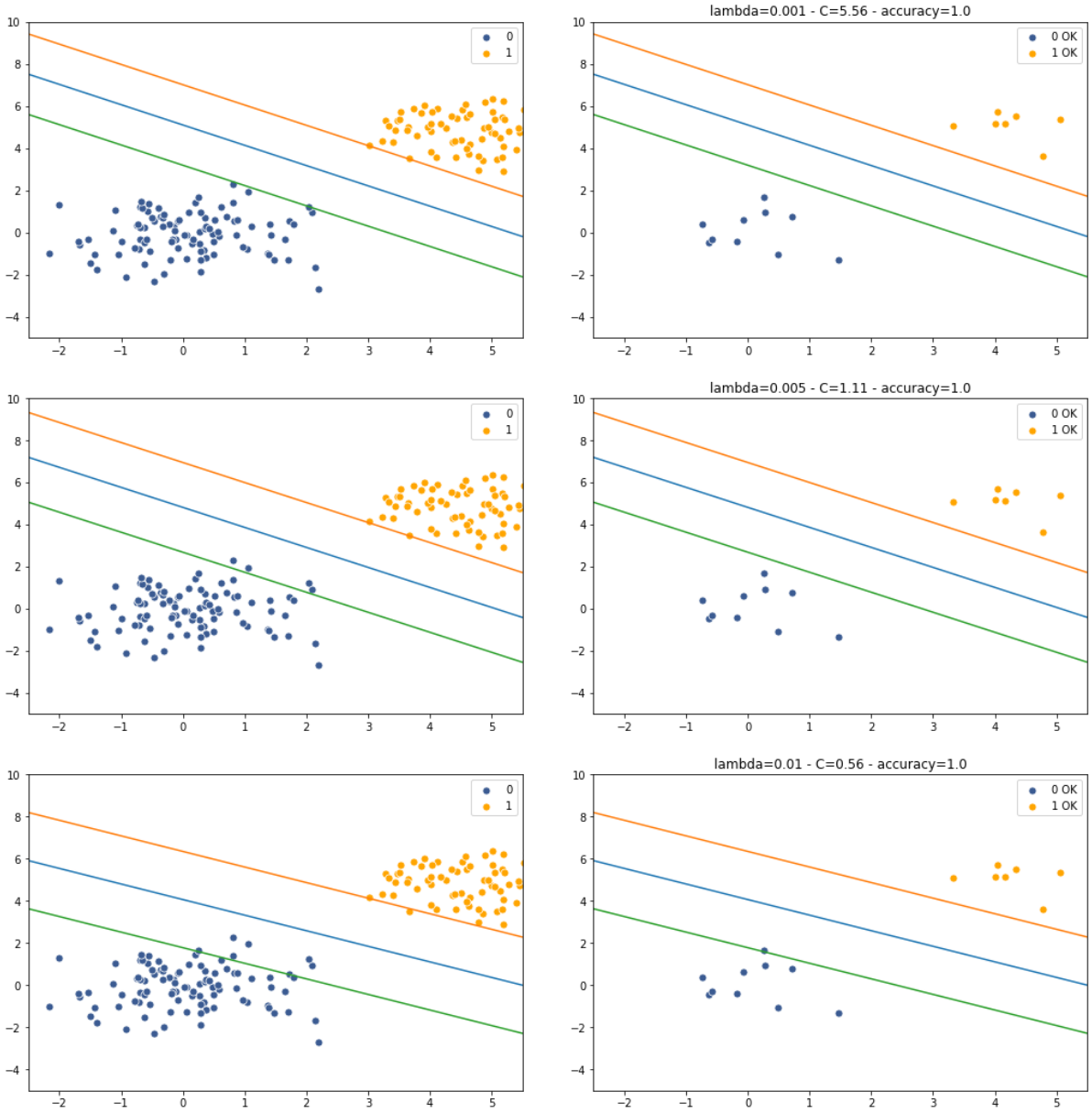

Figure 6: Resultado dataset con separación 5.

2 SVM con kernel RBF

Para este punto generamos datos no linealmente separables. Es decir, generamos dos clases cuyos puntos no se pueden separar por un hiperplano. Para eso se generaron dos clases, una de las cuales tiene puntos en una distribución gaussiana con covarianza isotrópica, mientras que la otra tiene puntos distribuidos uniformemente a lo largo de un círculo de radio r más una distancia generada con una distribución gaussiana con media 0.

Para generar los datos de la segunda clase mencionada, se generaron, en primer lugar, las coordenadas polares de los puntos, con ángulo generado con distribuciones uniformes y radio generado con una distribución $\mathcal{N}(\mathbf{0}, \sigma\mathbf{I})$ y luego se transformaron dichas coordenadas a coordenadas cartesianas.

A este dataset lo partimos en dos: un set de training y otro de testing. Entrenamos un modelo SVM con kernel RBF con $C = 1$ y $\gamma = 1$ con el set de training, y luego clasificamos el set de testing para ver la accuracy del modelo, que fue de 0.925. El dataset y el resultado de clasificar se pueden ver en la Figura 7.

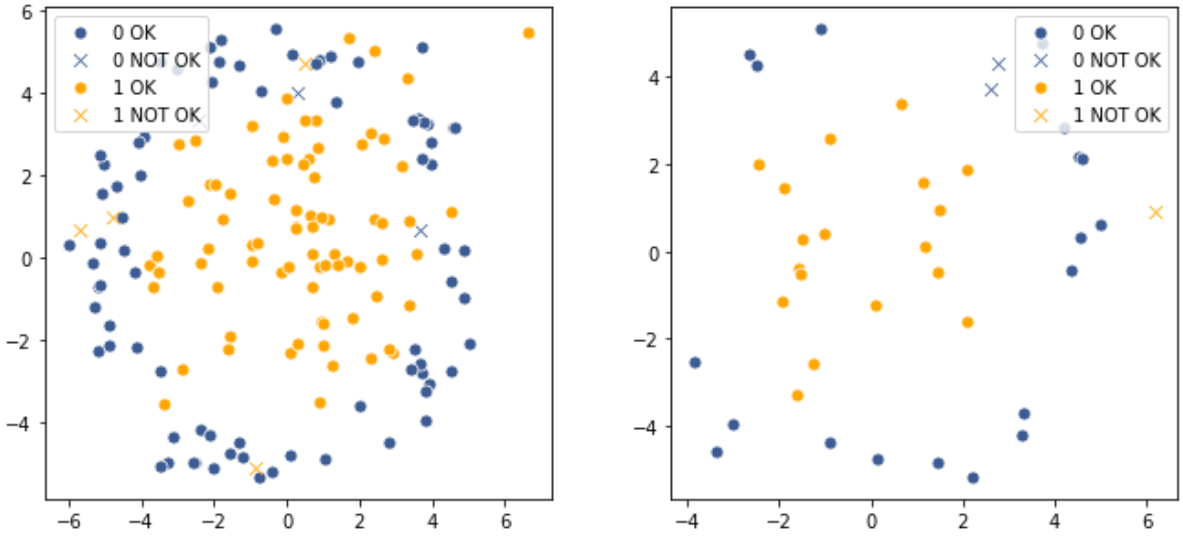

Figure 7: Dataset y resultado para SVM con kernel RBF.

3 Cross Validation

Utilizamos cross validation para determinar el mejor par (C, γ) para el dataset generado. Utilizamos K-fold con $K = 5$ para la validación. Los valores de C probados fueron $(0.01, 1, 5, 10)$, y los de γ, $(0.1, 0.001, 1e-5)$. Los puntajes resultantes para cada posible combinación de hiperparámetros se pueden ver en la Tabla 1. Los puntajes fueron calculados con el promedio del accuracy en cada paso de validación.

Vemos que el par de mayor puntaje es el $C = 1, \gamma = 0.1$, con un puntaje de 0.9. Utilizando estos parámetros para clasificar los elementos de testing, llegamos a una accuracy del 0.95, que es mejor que el 0.925 que alcanzamos en el punto anterior. El siguiente gráfico muestra las clasificaciones del algoritmo para un conjunto de testing.

C	γ	Puntaje
0.01	0.1	0.57
1	0.1	0.9
5	0.1	0.89
10	0.1	0.89
0.01	0.001	0.51
1	0.001	0.51
5	0.001	0.51
10	0.001	0.57
0.01	1e-05	0.5
1	1e-05	0.5
5	1e-05	0.5
10	1e-05	0.5

Table 1: Puntaje de cross validation para cada par probado de C y γ.

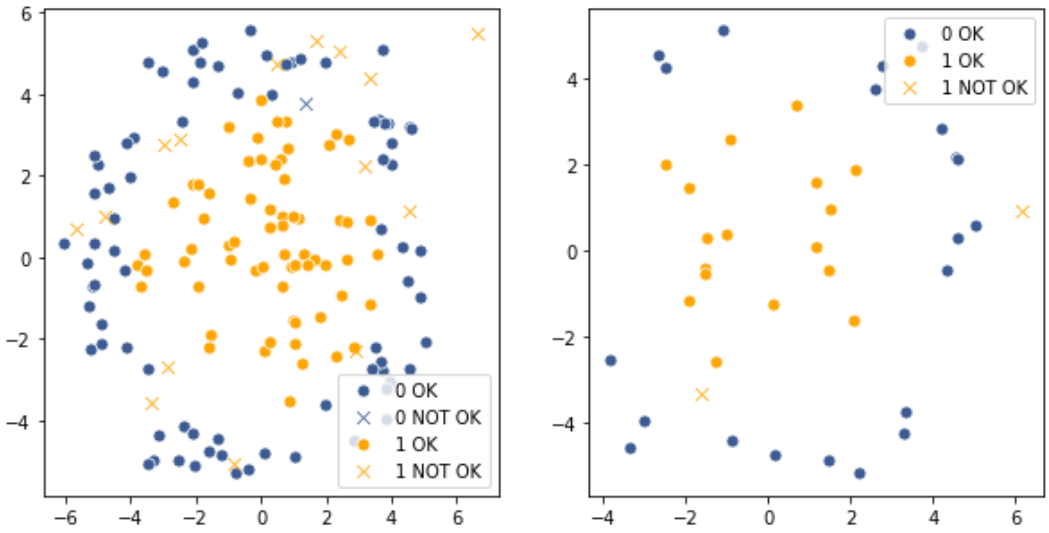

Figure 8: Resultado de clasificación de un conjunto de testing para SVM con kernel RBF.

A Apéndice de notación

El color de los puntos en cada gráfico define la clase verdadera del punto correspondiente. En donde corresponde, la forma define si fue bien clasificado o no, siendo un círculo para bien clasificado o una cruz para mal clasificado. Mal clasificado es NOT OK, bien clasificado es OK. Por ejemplo, un punto 1 NOT OK es un punto de la clase 1 que fue mal clasificado.