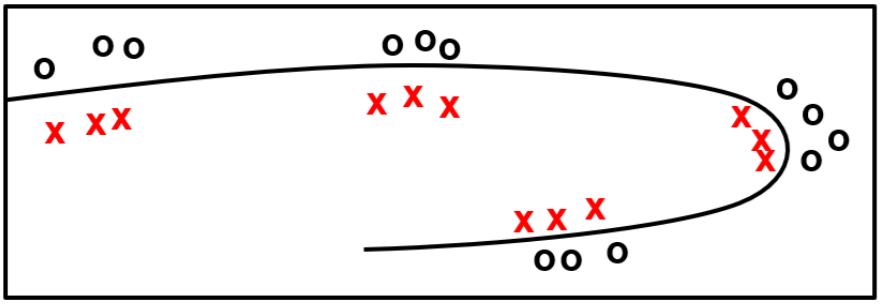
Modelos híbridos

Modelos Avanzados de Aprendizaje Automático I

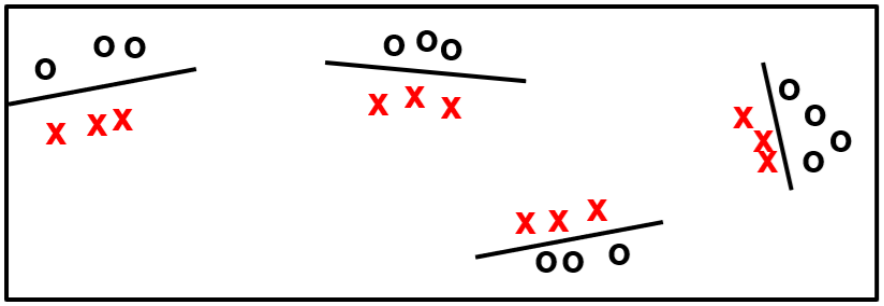
Los modelos híbridos de Aprendizaje Automático combinan diferentes técnicas y algoritmos de AA para aprovechar las fortalezas individuales de cada uno y mitigar sus debilidades. Los modelos híbridos buscan mejorar la precisión, la generalización y la capacidad de adaptación a distintos tipos de datos y problemas, y han destacado como una estrategia poderosa para mejorar el rendimiento y la robustez de los sistemas predictivos.

A pesar de estudiar más modelos híbridos en clase de teoría, en este ejercicio se implementará un único modelo, para que no suponga una gran carga de trabajo al ser el último ejercicio. Este modelo, también explicado en clase de teoría, resulta de la combinación de kNN y SVM.

La idea de este modelo es que, para un problema concreto, conseguir un modelo global puede ser complicado, es decir, es posible que este necesite una región de separación que sea muy compleja; sin embargo, a nivel local las regiones de separación pueden ser muy sencillas. Por ejemplo, como se muestra en teoría, para este *dataset* la región de separación puede ser complicada:



Sin embargo, trabajando localmente, las regiones de separación a desarrollar podrían ser lineales:



Por lo tanto, se está suponiendo que los modelos locales a desarrollar son de baja complejidad. Esto, además, viene avalado por el hecho de que localmente el número de instancias es muy bajo, por lo que un modelo complejo se sobreajustaría muy rápidamente.

En este modelo a desarrollar, dada una instancia a clasificar, kNN se utiliza para hallar los vecinos

más cercanos, que definen la localidad de la instancia. Una vez que se tiene, se realiza una separación lineal utilizando un SVM. Es importante tener en cuenta que, al igual que en kNN, en realidad no existe un modelo que se entrena, sino que se crea un modelo nuevo para cada instancia, y este se elimina una vez hecha la predicción.

Para hacer esto, nos basaremos en las funciones relativas a kNN desarrolladas en los ejercicios anteriores.

En este ejercicio, se pide realizar las siguientes funciones:

- *predictKNN_SVM*. Dado un *dataset* y una instancia, aplica el modelo kNN-SVM en esa instancia. Esta función recibe como parámetros de entrada:
    - *dataset*, de tipo *Batch*, con el conjunto de datos que define el problema.
    - *instance*, de tipo *AbstractArray{<:Real,1}*, con la instancia a clasificar.
    - *k*, de tipo *Int*, con el número de vecinos a tomar para definir la localidad.
    - *C*, de tipo *Real*, con el valor del hiperparámetro C de la SVM a desarrollar.

  Para hacer esto, dar los siguientes pasos:
    - En primer lugar, calcular la distancia de la instancia al *batch* pasado como parámetro utilizando la función *euclideanDistances* desarrollada en el ejercicio anterior.
    - Tomar los índices de las instancias con distancia más pequeña. Para hacer esto, utilizar la función *partialsortperm*.
    - Si las salidas deseadas de las instancias correspondientes a estos índices son todas iguales, es decir, si pertenecen a la misma clase, se devuelve esa clase como predicción. Para saber si las salidas deseadas son iguales, utilizar la función *unique*.
    - Se crea un modelo de SVM con *kernel* lineal, el valor de C pasado como argumento y el *keyword random_state* con un valor fijo de 1, para garantizar la repetibilidad de los experimentos, mediante la función SVC, y se entrena mediante la función *fit!*, ambas de Scikit-Learn. A la hora de entrenar, no se entrena con todas las instancias del *batch*, sino solamente con aquellas correspondientes a los índices de instancias más cercanas.
    - Se utiliza este modelo para hacer una predicción sobre la instancia pasada como parámetro, utilizando la función *predict* de Scikit-Learn. En este aspecto, es necesario tener en cuenta dos cuestiones:
        - Esta función tiene, como segundo parámetro, no una instancia sino un

conjunto de instancias, es decir, una matriz, estando cada una en una fila de la matriz. Para pasarle una instancia (un vector), este se puede convertir a una matriz con una fila mediante la función *reshape*.

- Dado que la función recibe un conjunto de instancias, devuelve un conjunto de predicciones, es decir, un vector que, en este caso, tendrá un único valor. Por lo tanto, será necesario tomar el valor del único elemento de este vector.

- Como resultado, esta función devolverá la predicción realizada en este último paso. El tipo de esta predicción será igual al tipo de dato que tengan las salidas deseadas del *batch* de entrenamiento.

Para el desarrollo de esta función no se permite el uso de bucles.

- *predictKNN_SVM*. Con el mismo nombre que la función anterior, está sobrecargada, y tiene como diferencia que esta, en lugar de recibir una instancia, recibe un conjunto de instancias y hace llamadas a la función anterior para clasificarlas. Esta función recibe como parámetros de entrada:
  - *dataset*, de tipo *Batch*, con el conjunto de datos que define el problema.
  - *instances*, de tipo *AbstractArray{<:Real,2}*, con el conjunto de instancias a clasificar, estando cada una en una fila de esta matriz.
  - *k*, de tipo *Int*, con el número de vecinos a tomar para definir la localidad.
  - *C*, de tipo *Real*, con el valor del hiperparámetro C de la SVM a desarrollar.

  Esto se hará fácilmente mediante una *comprehension* que itere sobre las filas de *instances* utilizando la función *eachrow*, y haciendo llamadas a la función anterior, de la misma manera que en el ejercicio anterior se hizo en la función *predictKNN*.

  Para el desarrollo de esta función no se permite el uso de bucles.

Para experimentar con estas funciones, se pueden utilizar las funciones desarrolladas en este ejercicio y los anteriores con algunos de los *datasets* disponibles, como "agaricus-lepiota", "twonorm", "ring", "hypothyroid", "tokyo1", "mushroom" o "ionosphere". Después de cargar el *dataset*, se puede hacer un *hold out* (partición simple del conjunto de datos en entrenamiento y test), y entrenar los distintos modelos mediante las funciones *predictKNN*, *predictKNN_SVM* y *trainSVM*, aplicando el modelo entrenado en el conjunto de test para comparar los resultados.

- ¿El modelo kNN-SVM devuelve siempre mejores resultados que kNN?
- ¿Cuál de los 3 modelos devuelve los mejores resultados?

Firmas de las funciones:

A continuación se muestran las firmas de las funciones a realizar en los ejercicios propuestos. Tened en cuenta que, dependiendo de cómo se defina la función, esta puede contener o no la palabra reservada *function* al principio:

```
function predictKNN_SVM(dataset::Batch, instance::AbstractArray{<:Real,1}, k::Int, C::Real)
function predictKNN_SVM(dataset::Batch, instances::AbstractArray{<:Real,2}, k::Int, C::Real)
```