4. Búsqueda de hiperparámetros, validación cruzada y Evaluación

Ejercicio 4.1.
Sea GRID_SEARCH(X : LIST⟨TUPLE⟨ALGORITMO, HYPERS⟩⟩, D : DATA, M : METRICA) : TUPLE⟨ALGORITMO, HYPERS, FLOAT⟩ la función que ejecuta una búsqueda exhaustiva para encontrar la mejor configuración entre una lista de posibilidades y retorna el valor esperado para dicha combinación.
Detalles de los parámetros:

- X es una lista de TUPLE⟨A, HS⟩ : TUPLE⟨ALGORITMO, HYPERS⟩, en donde A es el nombre del algoritmo y HS representa un diccionario de hiperparámetros a valores.
  Por ej, un elemento de la lista puede ser $\langle arbol\_de\_decision, \{altura\_max : 10, medida : gini, atributos : todos\}\rangle$
  otro puede ser $\langle arbol\_de\_decision, \{altura\_max : 5, medida : entropy, atributos : todos\}\rangle$
- D, un dataset de $n \times m$ (instancias $\times$ atributos)
- M, una métrica (por ejemplo, ACCURACY)

(a) Escribir el pseudocódigo de la función. Para ello, suponga ya implementada la función CROSS_VAL(A : ALGORITMO, HS : HYPERS, D : DATA, M : METRICA) : FLOAT que devuelve el resultado de ejecutar validación cruzada para un algoritmo A e hiper-parámetros HS sobre la base de datos D y usando la métrica M. Retorna el valor obtenido.

(b) Escribir el pseudocódigo de la función CROSS_VAL(A : ALGORITMO, HS : HYPERS, D : DATA, M : METRICA) : FLOAT. ¿Qué tipo de validación cruzada elegiste para tu pseudocódigo?

(c) Definir ahora la función RANDOMIZE_SEARCH(..., N : INT) : TUPLE⟨ALGORITMO, HYPERS, FLOAT⟩ con los mismos parámetros que GRID_SEARCH más N, un parámetro que indica la cantidad de intentos a probar por cada entrada de la lista. Suponer para esta función que H : HYPERS es un diccionario en donde los valores también pueden ser distribuciones probabilísticas a las que se las puede muestrear con la función SAMPLE(D: DISTRIBUCIÓN) : FLOAT (suponer que SAMPLE(CONSTANTE) = CONSTANTE. Ej, $\langle arbol\_de\_decision, \{altura\_max : Binomial(n = 20, p = 0.5), medida : entropy, atributos : todos\}\rangle$

Ejercicio 4.2. Validación cruzada. Verdadero o Falso. Justificar

(a) Hacer validación cruzada evita el sobreajuste (overfitting) de los modelos sobre los datos.

(b) Hacer validación cruzada ayuda a obtener estimaciones más realistas de la performance de un modelo sobre nuevos datos que al hacerlo sobre los mismos datos de entrenamiento.

(c) En K-fold cross validation, conviene que K se acerque a N. De esta manera el resultado será lo más realista posible ya que se tiende a generar N modelos independientes. El problema es que hay que entrenar demasiados modelos.

(d) Evaluar un modelo sobre el conjunto de evaluación (control) resultará en un valor siempre peor o igual al conseguido en desarrollo.

(e) Una vez elegida la mejor configuración durante desarrollo, es conveniente hacer k-fold cross validation nuevamente, pero ahora incluyendo también el conjunto de evaluación.

(f) Luego de seleccionar la mejor configuración, es ideal volver a correr el algoritmo pero esta vez utilizando todos los datos de desarrollo para luego evaluarlo en los datos de evaluación.

(g) Se espera que al evaluar un modelo sobre el held-out set (conjunto de evaluación) el resultado sea peor al conseguido en desarrollo.

(h) En K-fold cross validation entrenamos K modelos en donde cada modelo sólo se entrena en un subconjunto disjunto al resto de los modelos.

(i) En K-fold cross validation validamos K modelos en donde cada modelo sólo evalúa a un subconjunto disjunto al resto de los modelos

Ejercicio 4.3. Preguntas para desarrollar.

(a) En la clase teórica vimos que al hacer cross validation los datos no siempre deben separarse al azar, ¿por qué?. Pensar ejemplos de al menos dos situaciones en las cuales no sea conveniente.

(b) ¿Por qué se deberían usar una sola vez los datos de control (held-out)?

(c) ¿Qué sería conveniente hacer si luego de un muy buen resultado en desarrollo, encuentro un pésimo resultado en evaluación?

Ejercicio 4.4. Consideremos K-fold cross validation con las siguientes modificaciones:

- Para cada instancia x_i dentro de un conjunto de datos $\mathcal{D}$ se tiene un mapeo G que le asigna un único grupo $g \in \mathcal{G}$ tal que $G(x_i) = g$ (pueden pensarlo como un DICC(INSTANCIA, GRUPO)).
- Al dividir $\mathcal{D}$ en los k folds, se asegura que cada grupo g este contenido únicamente en un fold.

(a) Construir el algoritmo GROUP_K_CROSS_VAL(A : ALGORITMO, HS : HYPERS, D : DATA, M : MÉTRICA, G: GRUPOS, K: INT) : FLOAT que devuelve el resultado de evaluar el algoritmo dado con los hiperparámetros dados y algún valor de k respetando los grupos dados.

(b) ¿En que escenarios podría tener sentido utilizar este procedimiento? Desarrolle.

(c) ¿Qué cambiaría en su implementación si hubiese más folds que grupos?

Ejercicio 4.5. Luego de correr un grid search (búsqueda de hiperparámetros) en donde evaluamos miles de configuraciones utilizando cross validation, encontramos un modelo que funciona mejor que el resto con un 93 % de accuracy en los folds para validación. Este resultado, comparado contra datos nuevos de la realidad, será generalmente:

(a) demasiado optimista (porque el modelo puede haber sobreajustado a los datos de entrenamiento de cada fold).

(b) demasiado pesimista (al probar tantas combinaciones de modelos, el modelo tiene que esforzarse más que lo que haría en eval para conseguir un buen resultado).

(c) acertado ya que si hicimos bien los splits de cross validation, podemos garantizar que no hay información que se esté filtrando entre el entrenamiento y la validación de los modelos.

(d) demasiado optimista (porque habremos usado los datos de validación para tomar decisiones respecto al sistema).

Ejercicio 4.6. La técnica de SOBREMUESTREO (oversampling) consiste en crear copias de instancias al azar (a veces agregando mínimas modificaciones en alguno de sus atributos) y asignarle la etiqueta original.

Este proceso se utiliza mucho cuando las clases están muy desbalanceadas y no es suficiente la cantidad de instancias de alguna clase para entrenar un clasificador. En estos casos es normal, por ejemplo, sobremuestrear hasta lograr la misma cantidad de instancias para cada clase.

Por ejemplo, dado el problema de clasificar entre GATOS, PERROS y CONEJOS, y dado que la cantidad de instancias es 100, 2000 y 20 respectivamente, generamos un nuevo dataset de 2000, 2000, y 2000 instancias, en donde las instancias agregadas son copias de instancias de la clase correspondiente elegidas al azar y a las que se le agrega un poco de ruido a sus columnas numéricas.

Describir los pros y contras de aplicar la técnica en los siguientes pasos del proceso. Justificar en términos de posibilidades de sub o sobreestimar la performance de nuestros modelos y concluir cuál es la opción más segura:

(I) Antes de partir en desarrollo - evaluación

(II) Antes de partir en Folds en desarrollo.

(III) Luego de seleccionar los folds que se utilizarán para entrenar el modelo dentro de las iteraciones de K-fold cross val. Aplicarlo sólo a los datos de entrenamiento.

(IV) Luego de seleccionar los folds que se utilizarán para entrenar el modelo dentro de las iteraciones de K-fold cross val. Aplicarlo tanto a los datos de entrenamiento como a los de validación.

Ejercicio 4.7. Ordenar los siguientes pasos del desarrollo y evaluación de un modelo de manera apropiada (indentar si hay loops y especificar sobre qué datos se haría)

(a) Mandar reporte final a toda la empresa y submitear paper.

(b) Partir los datos en desarrollo - evaluación

(c) Entrenar un modelo usando una configuración, testear su performance en datos que no fueron utilizados para entrenar. Guardar el resultado.

(d) Partir en 10 folds.

(e) Correr un proceso que lista los atributos según qué tanto se correlacionan con las etiquetas y conservar sólo el top-10 para entrenar el modelo.

(f) Definir la métrica a utilizar (ej: accuracy pesada por clase)

(g) Mirar / escuchar / leer instancias del dataset para pensar buenos atributos.

(h) No me dio tan bien como esperaba. Empiezo de nuevo repensando algunas cosas.

(i) Armar una grilla de configuraciones a probar

(j) Después de correr lo anterior me di cuenta que podría haber probado tal modelo / tal hiperparámetro. Agregar a las configuraciones y repetir.

(k) Graficar la distribución de las etiquetas.

Ejercicio 4.8. Completar el notebook `notebook_seleccion_modelos.ipynb`