2023/2024 1Q

Lista de Problemas 4

APA

Javier Béjar

Departament de Ciències de la Computació

Grau en Enginyeria Informàtica - UPC

UPC FIB Facultat d'Informàtica de Barcelona

UNIVERSITAT POLITÈCNICA DE CATALUNYA

DEPARTAMENT DE CIÉNCIES DE LA COMPUTACIÓ

FACULTAT D'INFORMÁTICA DE BARCELONA

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Primera edición, septiembre 2021

Esta edición, Septiembre 2023

Instrucciones:

Para la entrega de grupo debéis elegir un problema del capítulo de problemas de grupo.

Para la entrega individual debéis elegir un problema del capítulo de problemas individuales. **Cada miembro del grupo debe elegir un problema diferente**.

Debéis hacer la entrega subiendo la solución al racó.

Evaluación:

La nota de esta entrega se calculará como 1/3 de la nota del problema de grupo más 2/3 de la nota del problema individual.

Al realizar el informe correspondiente a los problemas explicad los resultados y las respuestas a las preguntas de la manera que os parezca necesaria. Se valorará más que uséis gráficas u otros elementos para ser más ilustrativos.

La parte que no es de programación la podéis hacer a mano y escanearla a un archivo **PDF**. Comprobad que **sea legible**.

Para la parte de programación podéis entregar los resultados como un notebook (Colab/Jupyter). Alternativamente, podéis hacer un documento explicando los resultados como un PDF y un archivo python con el código

También, si queréis, podéis poner las respuestas a las preguntas en el notebook, este os permite insertar texto en markdown y en latex.

Objetivos:

1. Conocer el análisis de problemas usando máquinas de soporte vectorial, àrboles de decisión y conjuntos de clasificadores
2. Interpretar modelos de árboles de decisión
3. Usar técnicas de relevancia de atributos sobre clasificadores no lineales

CAPÍTULO 1

Problemas de Grupo

1. **Quiero andar en bicicleta**

   Si trabajasteis con el conjunto de datos de Bike Sharing con modelos lineales, probablemente notasteis la falta de linealidad en los datos y no quedasteis totalmente satisfechos con los resultados[1]. Ahora tenemos una nueva oportunidad para mejorar el modelo.

   El conjunto de datos de Bike Sharing del repositorio de conjuntos de datos de UCI recopila estadísticas agregadas del uso de bicicletas con información adicional relevante. Podéis descargar los datos desde aquí `https://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset`.

   El objetivo de este problema es predecir cuántas bicicletas se usarán diariamente (el archivo `day.csv`). Podéis leer en el `Readme.txt` los detalles sobre las variables.

   a) El primer paso es preprocesar y preparar los datos antes de ajustar cualquier modelo. Hay algunas variables que no son útiles para el problema o que no tiene sentido usar. Eliminadlas del conjunto de datos. Dividid los datos en conjuntos de entrenamiento y test[2] (60 % /40 %). Tened en cuenta que los modelos basados en árboles de decisión no necesitan ninguna normalización de los datos.

   b) Ajustad un árbol de regresión explorando sus hiperparámetros adecuadamente[3]. Registrad la puntuación de la validación cruzada para el mejor modelo y la calidad del modelo con los datos del test. Representad el árbol del modelo final utilizando la librería `viztree`[4] e intentad explicar cómo el árbol toma las decisiones a partir de los atributos en la parte superior del árbol. Representad las predicciones del árbol contra los valores reales. ¿Hay algo raro en la gráfica?

   c) Ajustad un regresor de random forest y un regresor gradient boosting a los datos explorando sus hiperparámetros adecuadamente. Registrad la puntuación de la validación cruzada para los mejores modelos y su calidad con los datos de test. ¿Son estos modelos mejores

[1] Si no resolvisteis el problema con estos datos, ahora podéis hacerlo

[2] Fijad el parámetro `random_state` en la función `train_test_split` para que los resultados no cambien cuando repitáis el experimento.

[3] Podéis usar la búsqueda en cuadrícula de `scikit-learn` o la búsqueda bayesiana de la biblioteca `scikit-optimize` como hemos visto en los notebooks de clase.

[4] Podéis ver como se usa en la última sesión de laboratorio.

que el árbol de regresión? Calculad la relevancia de sus atributos utilizando la importancia de permutación (permutation importance) con los datos de test. ¿Son los atributos relevantes de los modelos los que aparecen en los primeros niveles del árbol de regresión? Representad sus predicciones contra los valores reales. ¿Ha desaparecido lo raro del gráfico anterior? ¿Por qué?

d) Ajustad una máquina de vectores de soporte para regresión usando un kernel RBF, normalizad adecuadamente los datos para este modelo. Explorad los hiperparámetros del modelo adecuadamente (observad que la SVM de regresión tiene los parámetros `C` y `epsilon` además de los hiperparámetros específicos para el kernel). Registrad la puntuación de la validación cruzada para el mejor modelo y su calidad con los datos del test. ¿Es mejor este modelo? ¿Son diferentes las importancias de permutación? Considerando la calidad de los modelos y el coste de computar las predicciones, ¿hay algún modelo que preferirías utilizar en una aplicación que tiene que predecir en tiempo real? ¿Por qué?

2. **Pérdida de clientes (Churn)**

   Churn es un término de marketing que se refiere a la cantidad de clientes que abandonarán una empresa durante un período específico. Obviamente, si se puede predecir qué clientes probablemente van a cancelar su servicio, se puede intentar evitarlo mediante una acción específica, como proponer al cliente servicios adicionales u ofrecer un mejor precio.

   Vamos a usar el conjunto de datos `telco-customer-churn` del repositorio de datos Open ML que se puede obtener usando la función `fetch_openml` de scikit-learn usando ese nombre[5].

   a) Para este conjunto de datos necesitaréis un procesamiento previo. Transformad las características categóricas en características binarias (podéis usar la función `get_dummies` de Pandas). También faltan algunos valores en las características numéricas codificadas como espacios en blanco, debéis imputarlos adecuadamente. Dividid los datos en entrenamiento y test (70 %/30 %). Tened en cuenta que los modelos basados en árboles de decisión no necesitan ninguna normalización de los datos.

   b) En este problema no tiene sentido predecir el objetivo si no se puede explicar en términos simples cómo predice el modelo. Los árboles de decisión parecen la elección obvia. Ajustad este modelo explorando sus hiperparámetros adecuadamente[6]. Calculad la calidad del modelo. Representad el árbol de decisión del modelo final con la librería `viztree`[7] e intentad explicar cómo el árbol toma las decisiones a partir de los atributos en la parte superior del árbol.

   c) El conjunto de datos está claramente desbalanceado dado que la fracción de clientes que abandonan una empresa es menor que la de los que se quedan. Usad el parámetro `class_weights` del modelo para que las clases se equilibren durante el entrenamiento y repetid la exploración de hiperparámetros para encontrar el mejor modelo. Comentad las diferencias entre la calidad de este modelo y la del árbol de decisión final de la pregunta anterior. ¿Es este árbol mejor? ¿La información del árbol es mejor para determinar qué clientes abandonarán?

   d) Random forest aprovecha muchos árboles de decisión para aumentar la calidad del modelo. Ajustad este modelo equilibrando y sin equilibrar las clases. Explorad sus hiperparámetros adecuadamente y obtened la calidad de los modelos. ¿Son mejores estos modelos? Considerad no solo la calidad de los modelos, sino también el coste de la intervención para tratar de evitar el abandono.

[5] Podéis descargar los datos como un marco de datos de pandas y en el objeto que obtenéis encontraréis marcos de datos con las variables de entrada y la salida.

[6] Podéis utilizar la búsqueda en cuadrícula de `scikit-learn` o la búsqueda bayesiana de la biblioteca `scikit-optimize` como hemos visto en los notebooks de clase.

[7] Podéis ver como se usa en la última sesión de laboratorio.

3. **Sumemos de nuevo**

   Nota: Este problema solo puede ser resuelto por grupos que resolvieron el problema de grupo 3 de la lista 2.

   Los resultados del experimento de suma usando k-vecinos más cercanos fueron un poco decepcionantes incluso transformando los atributos para una mejor representación. Las relaciones no lineales entre algunos atributos fueron las culpables de estos resultados no óptimos. Ahora que conocemos otros clasificadores, podemos probar si pueden hacer un mejor trabajo.

   El escenario será el mismo, aprendiendo a sumar números de tres dígitos con ejemplos definidos por seis atributos, los tres primeros atributos corresponden a los tres dígitos del primer número y los tres últimos a los dígitos del segundo número. La salida tiene cuatro valores, uno para cada dígito para cada número del resultado de la suma. Por ejemplo:

X1	X2	X3	X4	X5	X6	Y1	Y2	Y3	Y4
0	1	6	4	7	7	0	4	9	3

   a) Dado que vamos a utilizar clasificadores más potentes, probablemente necesitemos menos ejemplos para funcionar mejor, así que generad dos conjuntos de datos de 25.000 y 50.000 ejemplos. Dividid los conjuntos de datos en un conjunto de test y entrenamiento con una proporción de 90 %/10 %.

   b) Como recordaréis, este es un problema de clasificación de salida múltiple, por lo que necesita usar el objeto `MultiOutputClassifier` de scikit learn. Esto envolverá los clasificadores para que pueda realizar este tipo de clasificación.

   Usad `SVC` para la clasificación con un kernel `rbf` con diferentes valores para `C`. Podéis usar diferentes potencias de 10 para ese parámetro para ver dónde está el mejor acierto para los datos de test en lugar de usar validación cruzada, os tardará mucho si la usáis. Aseguraos de usar el parámetro `n_jobs` de los objetos de clasificación de salida múltiple para que las cosas se hagan en paralelo.

   c) Usad `GradientBoostingClassifier` con diferentes valores para el número de estimadores y la profundidad máxima de los árboles. La cantidad de estimadores no tiene que ser muy grande, pero probablemente necesitaréis alrededor de 50 árboles con la tasa de aprendizaje predeterminada. Además, los árboles demasiado poco profundos darán malos resultados. Evaluad el acierto sobre el test como en el apartado anterior.

   Podéis acceder a los clasificadores individuales del clasificador de salida múltiple con el atributo `estimator_`. Obtened la importancia de los atributos para cada clasificador que encontraréis en el atributo `feature_importances` e interpretad los resultados.

   d) ¿Por qué creéis que los resultados de estos clasificadores son mejores que los del clasificador k vecinos más cercanos? ¿Por qué creéis que necesitamos menos ejemplos para estos resultados?

CAPÍTULO 2

Problemas Individuales

Para obtener los datos para estos problemas necesitaréis instalaros la última versión de la libreria `apafib`. La podéis instalar localmente haciendo:

```
pip install -user -upgrade apafib
```

Para usar las funciones de carga de datos solo tenéis que añadir su importación desde la librería, en vuestro script o notebook, por ejemplo

```
from apafib import load_BCN_cesta
```

La función os retornara un `DataFrame` de Pandas o arrays de numpy con los datos y la variable a predecir, cada problema os indicará que es lo que obtendréis.

1. **Comprender o predecir, ¿es esa la cuestión?**

   En muchas aplicaciones, obtener una representación que sea fácil de entender puede ayudar a modificar el proceso subyacente que genera los datos o a comprender el procedimiento que utilizan los expertos humanos para tomar decisiones.

   En el ámbito de seguros, los actuarios deciden cuál es el riesgo que corresponde a los diferentes objetos asegurables analizando datos históricos. En este problema, vamos a usar datos de seguros de automóviles del repositorio de datos Open ML que se puede obtener usando la función `fetch_openml` de scikit-learn usando el nombre `autos`[1]. El objetivo es la columna `symboling` que indica el riesgo de un automóvil dado por los actuarios.

   a) Este conjunto de datos necesitará un procesamiento previo. Transforma el atributo objetivo en dos categorías, 0 si el valor es menor o igual que 0 (riesgo bajo) y 1 si es mayor que 0 (riesgo alto). Separa los atributos según su tipo, categórico o continuo para formar dos conjuntos de datos. Binariza los atributos categóricos usando la función de pandas `get_dummies` e imputa los valores faltantes en los atributos continuos. Divide los conjuntos de datos en entrenamiento y test (60 %/40 %). La división debe estratificarse, utiliza el

[1] Podéis descargar los datos como un dataframe de pandas y en el objeto que obtendréis encontraréis dataframes con las variables de entrada y el objetivo.

mismo valor para el parámetro `random_state` para que la división de los dos conjuntos de datos sea la misma. No se necesita más preprocesamiento.

b) Ajusta un árbol de decisión empleando los datos categóricos explorando los hiperparámetros del modelo adecuadamente. Obtén la calidad del modelo. Visualiza el árbol de decisión con la librería `viztree`[2] y la importancia de los atributos usando la importancia de permutación (permutation importance) con los datos de test.

c) Ajusta un árbol de decisión usando los atributos continuos explorando los hiperparámetros del modelo adecuadamente. Obtén la calidad del modelo. Visualiza el árbol de decisión con la librería `viztree` y la importancia de los atributos usando la importancia de permutación (permutation importance) con los datos de test. ¿Qué modelo preferirías de los dos según lo fácil que sea explicar las decisiones del modelo o su calidad? ¿Por qué?

d) Junta todos los atributos y ajusta un árbol de decisión explorando los hiperparámetros del modelo adecuadamente. Evalúa su calidad. ¿Cambiarías tu decisión anterior dados los resultados de este modelo? ¿Por qué?

2. **La influencia de la cesta de la compra**

Uno de los datos que recolecta la web de *la ciutat al día* del ayuntamiento de Barcelona es la evolución del precio de los alimentos (verduras, fruta, carne, pescado). Una cosa que nos podemos preguntar es en que otra información puede tener influencia esta evolución y cuál de estos productos es el que tiene más relación. Igual que en otros problemas que usan este conjunto de datos, correlación no significa causalidad, la relación suele corresponden a otras variables no observadas, pero a falta de conocerlas podemos usar las relaciones que aparecen para formular hipótesis. En este caso nos podemos preguntar si esta evolución de precios tiene relación con la evolución de reservas hoteleras internacionales en Barcelona.

Puedes obtener estos datos mediante la función `load_BCN_cesta` de la librería `apafib`. Resuelve los siguientes apartados ilustrando los resultados de la manera que te parezca adecuada.

a) Divide el conjunto de datos en entrenamiento y test (80 %/20 %). Haz una exploración mínima del conjunto de datos de entrenamiento observando las relaciones entre las variables, especialmente con la variable objetivo. Describe las cosas que hayas visto que te parezcan interesantes. Transforma las variables adecuadamente según el modelo que utilices para poder ajustar modelos de regresión, tanto el conjunto de entrenamiento como el de test.

b) Ajusta una SVM de regresión con kernel polinómico y con kernel RBF explorando los hiperparámetros adecuadamente. Elige adecuadamente el modelo que parezca mejor y usa el método de *permutation importance* sobre el test para determinar qué atributos son más importantes en el modelo para predecir.

c) Ajusta los modelos random forest y gradient boosting para regresión explorando los hiperparámetros adecuadamente. Elige adecuadamente el modelo que parezca mejor y usa el método de *permutation importance* sobre el test para determinar qué atributos son más importantes en el modelo para predecir.

d) Como hemos visto en teoría cuando tenemos modelos diferentes podemos combinarlos usando diferentes estrategias. Entrena un `StackedRegressor` y un `VotingRegressor` usando los dos mejores modelos que has encontrado en los apartados anteriores con sus mejores hiperparámetros. ¿Es mejor alguno de estos modelos combinados? Calcula la *permutation importance* de los atributos sobre el test con estos modelos combinados ¿ha cambiado qué usan los modelos para obtener las predicciones?

[2] Puedes ver como se usa en la última sesión de laboratorio.

3. **¿Quién hace ruido en Barcelona por la noche?**

   Barcelona es una ciudad turística, pero no siempre recibe los mejores turistas. El ruido por la noche es un problema recurrente (no siempre debido a los turistas obviamente). Dentro de los datos que recolecta la web de *la ciutat al día* del ayuntamiento de Barcelona está el número de personas que hay diariamente en la ciudad desglosado en diferentes procedencias, tanto desde municipios cercanos, provincias o países[3]. Otra información que está disponible es el nivel de ruido en la ciudad (contaminación acústica) tomado a diferentes horas, el que nos interesa el de las 4 de la mañana. En este caso podemos averiguar si el que haya más personas de ciertos lugares en la ciudad tiene relación con el nivel de ruido que hay por la noche.

   Puedes obtener estos datos mediante la función `load_BCN_ruido` de la librería `apafib`. Resuelve los siguientes apartados ilustrando los resultados de la manera que te parezca adecuada.

   a) Divide el conjunto de datos en entrenamiento y test (80 %/20 %). Haz una exploración mínima del conjunto de datos de entrenamiento observando las relaciones entre las variables, especialmente con la variable objetivo. Describe las cosas que hayas visto que te parezcan interesantes. Transforma las variables adecuadamente según el modelo que utilices para poder ajustar modelos de regresión, tanto el conjunto de entrenamiento como el de test.

   b) Ajusta un modelo K-nearest neighbours de regresión y una SVM de regresión con kernel RBF explorando los hiperparámetros adecuadamente. Elige adecuadamente el modelo que parezca mejor y usa el método de *permutation importance* sobre el test para determinar qué atributos son más importantes en los modelos para predecir.

   c) Ajusta los modelos random forest y gradient boosting para regresión explorando los hiperparámetros adecuadamente. Elige adecuadamente el modelo que parezca mejor y usa el método de *permutation importance* sobre el test para determinar qué atributos son más importantes en el modelo para predecir.

   d) Como hemos visto en teoría, cuando tenemos varios modelos podemos combinarlos usando diferentes estrategias. Entrena un `StackedRegressor` y un `VotingRegressor` usando los dos mejores modelos que has encontrado en los apartados anteriores con sus mejores hiperparámetros. ¿Es mejor alguno de estos modelos combinados? Calcula la *permutation importance* de los atributos sobre el test con estos modelos combinados ¿ha cambiado qué utilizan los modelos combinados para obtener las predicciones? ¿Hay un consenso entre los modelos sobre quienes son los más ruidosos?

4. **La imprenta es más poderosa que la espada**

   Los procesos industriales son complejos y dependen de muchas variables que hacen que incluso a expertos en el dominio les sea difícil dar diagnósticos suficientemente fiables. En este tipo de dominios se pueden aplicar técnicas de aprendizaje automático para generar un modelo que posteriormente puede ser interpretado para ayudar a comprender las causas de los problemas. El conjunto de datos Cylinder Bands Data Set[4] del repositorio de conjuntos de datos de UCI recopila información sobre problemas en la impresión en hueco grabado. Las imprentas hacen grandes tiradas y deben parar las máquinas cuando surge algún problema. Un problema habitual es que aparezcan bandas en la impresión, lo que arruina toda la tirada. En este conjunto de datos se miden los valores de una treintena de atributos para casos en los que hubo o no un problema de bandas. El objetivo es obtener un modelo que permita obtener un diagnóstico que determine en que caso puede suceder el problema.

[3] Estamos en una era de vigilancia perpetua, el gran hermano siempre te está observando.

[4] Podéis encontrar una descripción del conjunto de datos en https://archive.ics.uci.edu/ml/datasets/Cylinder+Bands.

Puedes obtener estos datos mediante la función `load_bands` de la librería `apafib`. Resuelve los siguientes apartados ilustrando los resultados de la manera que te parezca adecuada.

a) Divide el conjunto de datos en entrenamiento y test (70 %/30 %). Tienes variables categóricas y continuas, transforma las variables adecuadamente según el modelo que utilices, para poder ajustar modelos de clasificación tanto el conjunto de entrenamiento como el de test.

b) Ajusta una SVM para clasificación con kernel polinómico y una con kernel RBF a los datos explorando los hiperparámetros adecuadamente. Elige adecuadamente el modelo que parezca mejor y usa el método de *permutation importance* sobre el test para determinar qué atributos son más importantes en el modelo para predecir.

c) Ajusta los modelos random forest y gradient boosting para clasificación explorando los hiperparámetros adecuadamente. Elige el modelo que parezca mejor y usa el método de *permutation importance* sobre el test para determinar qué atributos son más importantes en el modelo para predecir.

d) A veces el tener una idea detallada de como se diagnostica en el problema puede ayudarnos a aprender sobre él. Esto solo lo podemos hacer con un modelo que sea directamente interpretable. Ajusta un árbol de decisión a los datos explorando los hiperparámetros adecuadamente. Evalúa la calidad del modelo comparado con la de los otros dos de los apartados anteriores y representa el árbol de decisión obtenido usando la librería `viztree`[5]. ¿Coinciden los atributos más importantes de los mejores modelos obtenidos en los apartados anteriores calculados mediante *permutation importance* con las decisiones que hace el árbol en sus primeros niveles? Comparando la calidad de todos los modelos ¿qué modelo elegirías?

5. **Diagnóstico por la escritura**

El diagnóstico temprano de las enfermedades neuro degenerativas es un problema importante, ya que puede permitir tomar medidas preventivas y realizar tratamientos que prolonguen la calidad de vida de las personas. El hacer este diagnóstico sin tener que recurrir a métodos de neuroimagen (MRI, PET) es interesante, porque aparte de evitar exponer a los pacientes con frecuencia a radiaciones es más económico. En la enfermedad de Alzheimer una posibilidad es efectuar diagnóstico mediante la escritura. El conjunto de datos DARWIN[6] del repositorio de conjuntos de datos de UCI recoge medidas tomadas mediante un lápiz electrónico de la escritura de pacientes sanos y pacientes con Alzheimer. El objetivo es obtener un modelo que pueda distinguirlos.

Puedes obtener estos datos mediante la función `load_darwin` de la librería `apafib`. Resuelve los siguientes apartados ilustrando los resultados de la manera que te parezca adecuada.

a) Divide el conjunto de datos en entrenamiento y test (70 %/30 %). Transforma las variables adecuadamente según el modelo que utilices para poder ajustar modelos de clasificación tanto el conjunto de entrenamiento como el de test. Para poder comparar la eficacia de los métodos estableceremos Naïve Bayes como modelo base. Ajusta los hiperparámetros de este modelo adecuadamente y evalúa su calidad.

b) Ajusta una SVM para clasificación con kernel lineal, polinómico y RBF a los datos explorando los hiperparámetros adecuadamente. Elige adecuadamente el modelo que parezca mejor y compara sus resultados con el de Naïve Bayes.

[5] Puedes ver como se usa en la última sesión de laboratorio.

[6] Podéis encontrar una descripción del conjunto de datos en https://archive-beta.ics.uci.edu/ml/datasets/darwin.

c) Ajusta los modelos random forest y gradient boosting para clasificación explorando los hiperparámetros adecuadamente. Elige adecuadamente el modelo que parezca mejor y compara sus resultados con el de Naïve Bayes y el mejor modelo del apartado anterior.

d) Como hemos visto en teoría, cuando tenemos varios modelos podemos combinarlos usando diferentes estrategias. Entrena un `VotingClassifier` usando el modelo Naïve Bayes y los dos mejores modelos que has encontrado en los apartados anteriores con sus mejores hiperparámetros. ¿Es mejor el modelo combinado?

6. **Siéntate recto**

El tener una buena postura al sentarse evita muchos problemas de salud. A medida que nos hacemos mayores van apareciendo diversos problemas de espalda que han de diagnosticarse. El conjunto de datos Vertebral Column Data Set[7] del repositorio de conjuntos de datos de UCI recoge medidas tomadas de la forma y orientación de la pelvis y la parte lumbar de la columna vertebral para un conjunto de pacientes sin problemas y con dos diferentes diagnósticos (hernia discal y Spondylolisthesis). El objetivo es obtener un modelo que pueda distinguir los pacientes sin problemas y los dos diagnósticos.

En este conjunto de datos hay algo de desbalance entre las clases. La mayoría de los métodos de scikit-learn tienen un parámetro `class_weight` que permite darle el peso a las clases. Usa el valor `"balanced"` (se le da un peso a las clases de manera que parezca que tienen el mismo número de ejemplos) a la hora de ajustar los modelos que lo permitan.

Puedes obtener estos datos mediante la función `load_column` de la librería `apafib`. Resuelve los siguientes apartados ilustrando los resultados de la manera que te parezca adecuada.

a) Divide el conjunto de datos en entrenamiento y test (70 %/30 %). Transforma las variables adecuadamente para poder ajustar un modelo de clasificación tanto el conjunto de entrenamiento como el de test. Para poder comparar la eficacia de los métodos estableceremos la regresión logística como modelo base. Ajusta los hiperparámetros de este modelo adecuadamente y evalúa su calidad.

b) Ajusta una SVM para clasificación con kernel polinómico y RBF a los datos explorando los hiperparámetros adecuadamente. Elige adecuadamente el modelo que parezca mejor y compara sus resultados con el de regresión logística.

c) Ajusta los modelos random forest y gradient boosting para clasificación explorando los hiperparámetros adecuadamente. Elige adecuadamente el modelo que parezca mejor y compara sus resultados con el de regresión logística y el mejor modelo del apartado anterior.

d) Como hemos visto en teoría, cuando tenemos varios modelos podemos combinarlos usando diferentes estrategias. Entrena un `VotingClassifier` usando el modelo regresión logística y los dos mejores modelos que has encontrado en los apartados anteriores con sus mejores hiperparámetros. ¿Es mejor el modelo combinado?

[7] Podéis encontrar una descripción del conjunto de datos en http://archive.ics.uci.edu/ml/datasets/vertebral+column.