Ensamble de modelos basado en técnicas de muestreo

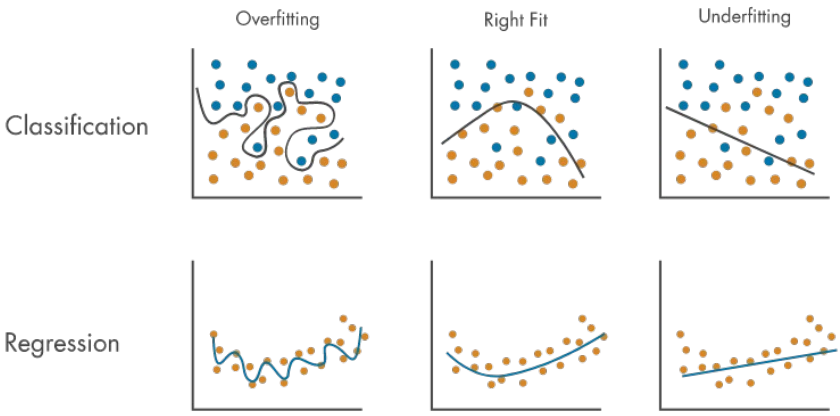
Inteligencia Artificial (IS) 2022/23 – Propuesta de trabajo

Juan Galán Páez

1. Introducción

A continuación se introducen una serie de conceptos, que serán de utilidad para contextualizar el trabajo:

1.1. Sobreajuste y generalización



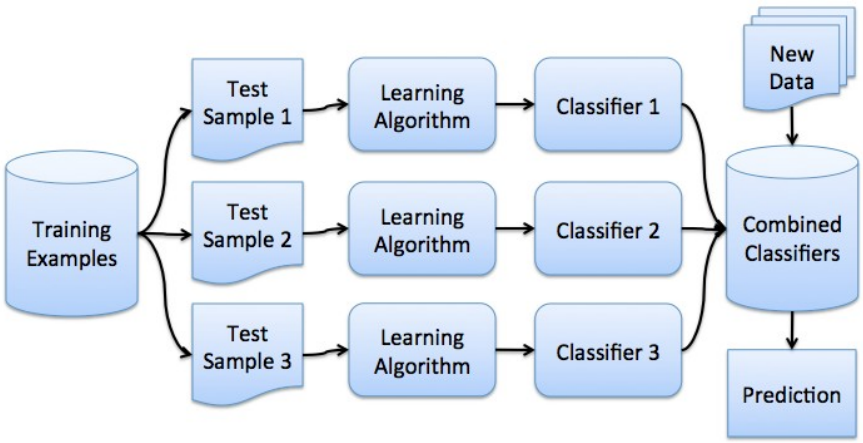
Recordemos que cuando entrenamos un algoritmo sobre un conjunto de entrenamiento, nuestro objetivo es que el modelo obtenido sea válido en toda la realidad del problema, es decir, más allá de nuestros datos de entrenamiento. Decimos que un modelo tiene capacidad de generalización si es capaz de clasificar correctamente ejemplos que no pertenecen al conjunto de entrenamiento (aunque pertenecen al dominio del problema). En cambio, un modelo sobreajustado es aquel que solo clasifica correctamente los ejemplos del conjunto de entrenamiento.

La flexibilidad de algunos algoritmos de aprendizaje supervisado (por ejemplo, los árboles de decisión) permite que, puedan acabar aprendiendo cada instancia del conjunto de entrenamiento, en lugar de patrones generales (que es lo que buscamos). Por ejemplo, en un caso extremo de sobreajuste en árboles de decisión, el árbol aprendido tendría una hoja por cada ejemplo del conjunto de entrenamiento.

La mayoría de los algoritmos de aprendizaje proporcionan mecanismos que permiten reducir el sobreajuste, como limitar la profundidad máxima o el número de hojas en el caso de los árboles de decisión, tal y como vimos en clase. Los mecanismos para combatir el sobreajuste mencionados, son específicos de cada algoritmo de aprendizaje. En este trabajo nos centramos en los métodos de ensamble [1] que consisten en combinar varios modelos con capacidad predictiva variable para obtener uno con buena capacidad predictiva sin caer en el sobreajuste. Esta técnica puede ser aplicada a cualquier algoritmo de aprendizaje supervisado, sin embargo, funciona mejor con unos algoritmos que con otros.

1.2. Ensamble de modelos predictivos

La idea del ensamble de modelos consiste en entrenar varios modelos sobre un mismo problema y luego combinarlos para obtener un modelo final (al que llamaremos meta-modelo) con mayor capacidad predictiva y menor sobreajuste que los modelos que lo componen. Existen numerosas formas de agregar (técnicas de ensamble) estos modelos. Las más sencillas consisten en calcular la predicción media o la más frecuente entre las predicciones de cada uno de los modelos.



Uno de los requisitos necesarios para obtener un buen ensamble de modelos es que estos sean diferentes entre si. Esto se puede conseguir de dos formas: 1) combinando modelos obtenidos a partir de algoritmos diferentes (redes neuronales, regresión lineal, Knn, árboles de decisión, etc.) sobre el mismo conjunto de entrenamiento 2) Combiando modelos obtenidos entrenados mediante el mismo algoritmo sobre diferentes conjuntos de entrenamiento. En este trabajo nos centraremos en la segunda opción.

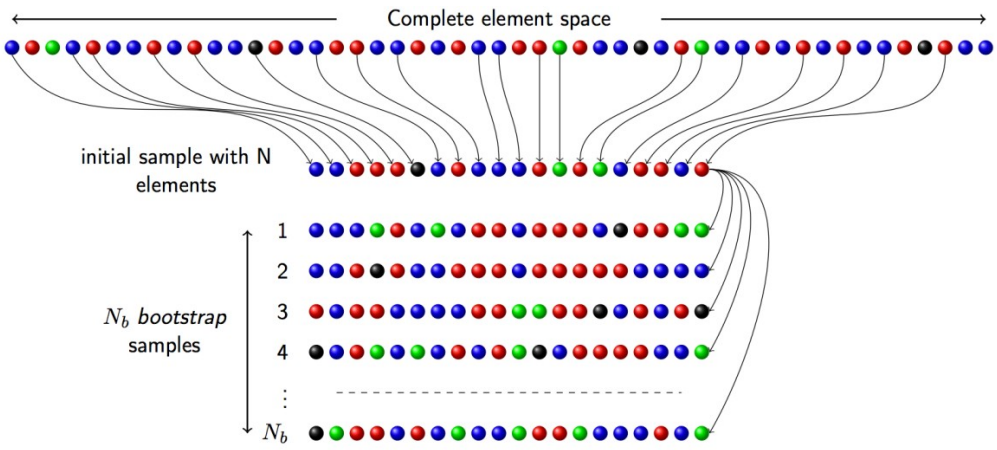
La idea detrás de esto es que, en vez de obtener un modelo que intente capturar toda la casuística del problema, obtenemos varios modelos, cada uno especializado en una parte del problema, que al ponerlos en común nos proporciona una solución más robusta.

Se ha comentado que una de las opciones existentes para obtener modelos diferentes entre si, consiste en entrenar nuestro algoritmo con diferentes conjuntos de datos. Esto plantea un problema y es que normalmente la cantidad de ejemplos que contiene nuestro conjunto de entrenamiento es limitada. Por tanto dividir el conjunto de entrenamiento en muchos subconjuntos y entrenar un algoritmo con cada subconjunto no es una opción. Vamos a generar conjuntos de entrenamiento diferentes aplicando técnicas de muestreo (tanto de columnas como de filas) sobre el conjunto de datos original. A continuación introducimos dos técnicas que nos ayudaran a resolver este problema.

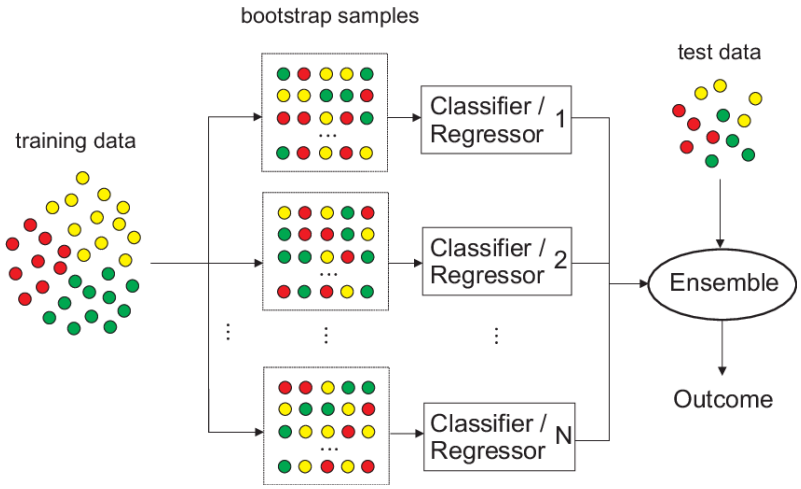
1.3. Técnicas de muestreo para ensamble de modelos

1.3.1 Muestreo de registros o filas:

Una de las técnicas más populares para derivar nuevos conjuntos de ejemplos a partir de la muestra original se denomina **Bootstrapping** [3]. Esta es una técnica muy usada en estadística con el objetivo de conocer con fiabilidad la distribución de una población completa a partir de una muestra.
Boostrapping consiste en, a partir de un conjunto con N ejemplos (filas), obtener muchas muestras de tamaño N realizando muestreo con reemplazamiento sobre N.

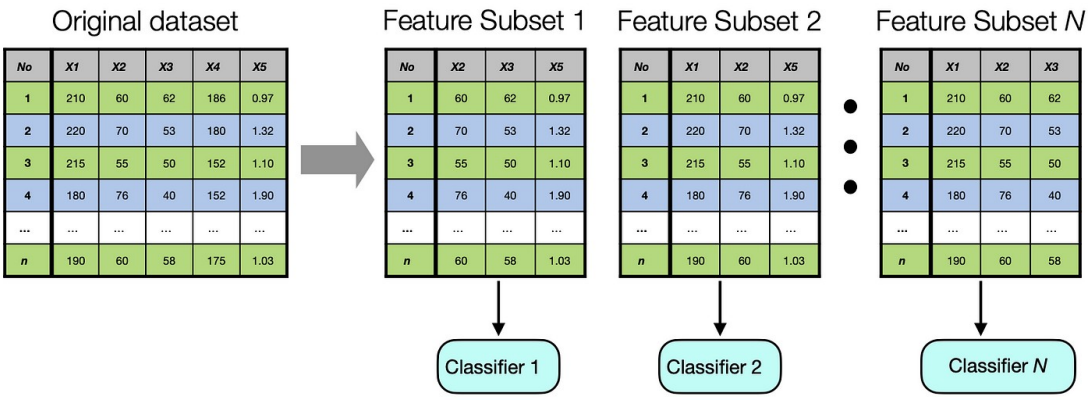


Es decir, a partir de un conjunto de entrenamiento con N ejemplos, creamos nuevos conjuntos de entrenamiento diferentes (de tamaño N) tras realizar muestreos con reemplazamiento de N muestras sobre el original. Estos conjuntos obtenidos, serán parecidos pero no idénticas ya que, como resultado del muestreo con reemplazamiento, algunos ejemplos del conjunto original aparecerán repetidos y otros no estarán presentes. Esto nos permitirá entrenar modelos ligeramente diferentes usando siempre un conjunto de datos de tamaño N.



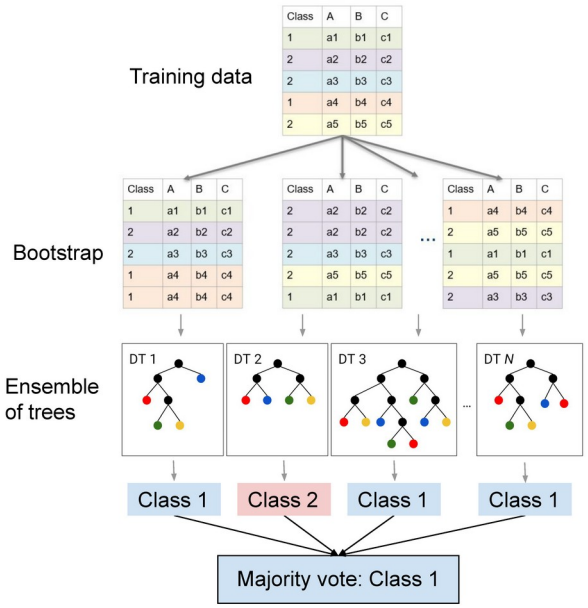
1.3.2 Muestreo de atributos o columnas:

La segunda técnica que usaremos en este trabajo se denomina **Random Subspace method**. Esta técnica es otra idea sencilla pero igualmente potente. Si en el caso anterior trabajamos a nivel de ejemplos (filas), en este caso lo haremos con las características o variables (columnas) de nuestro conjunto de entrenamiento. Para esto, el método de los subespacios aleatorios consiste en entrenar cada uno de los modelos usando un subconjunto aleatorio de variables o características, para que cada modelo centre su aprendizaje en solo una parte de los datos.



En este trabajo combinaremos las dos técnicas descritas para generar, a partir del original, nuevos conjuntos de entrenamiento. Es decir, para generar cada nuevo conjunto, aplicaremos primero boostraping (muestrear N filas con reemplazamiento) y luego el método de los subespacios aleatorios (seleccionar aleatoriamente un subconjunto de columnas). Con cada conjunto entrenaremos un modelo que aprenderá lo máximo posible de una versión diferente del conjunto de datos original. Es decir, unos modelos serán especialistas en unas dimensiones del problema y otros en otras. De la combinación de estos modelos especializados se espera obtener un modelo más robusto. A la hora de combinar los modelos entrenados y obtener la predicción final (es decir, la predicción del meta-modelo) a partir de las predicciones de los modelos individuales, tomamos la clase más frecuente en el caso de problemas de clasificación y la media de las predicciones en el caso de problemas de regresión.

Existen algoritmos muy potentes que están basados en este tipo de ensamble de modelos. Aunque no lo usaremos en este trabajo, uno de los más populares se llama Random Forest (bosques aleatorios) que es un ensemble de árboles de decisión..



2. Objetivos

Objetivo principal: desarrollar nuestro propio ensamble de modelos, es decir, aplicar las técnicas Bootstrapping y Random Subspace Method para obtener múltiples conjuntos de de entrenamiento sobre los que los que entrenar modelos para luego obtener la predicción mayoritaria o promedio. A partir de ahora llamaremos a nuestro ensamble, el **meta-modelo**.

Junio:

- Realizar un ensamble de modelos de **clasificación**.
- El ensamble, podrá configurarse para usar, al menos, los siguientes **algoritmos** de Scikit-learn:
  - Árboles de decisión (DecisionTreeClassifier)
  - Clasificador lineal mediante descenso por el gradiente (SGDClassifier)
- **Métricas** para evaluar el rendimiento del meta-modelo:
  - balanced_accuracy_score
  - f1_score
- **Conjuntos de datos:**
  - **Pendientes de publicación.**

Julio y noviembre:

- Realizar un ensamble de modelos de **regresión**.
- El ensamble, podrá configurarse para usar, al menos, los siguientes **algoritmos** de Scikit-learn:
  - Árboles de decisión (DecisionTreeRegressor)
  - Regresor lineal mediante descenso por el gradiente (SGDRegressor)
- **Métricas** para evaluar el rendimiento del meta-modelo:
  - MAE: mean_absolute_error
  - RMSE: mean_squared_error
    Aviso: Según los parámetros de entrada, este método puede calcular el MSE y el RMSE.
- **Conjuntos de datos:**
  - **Pendientes de publicación.**

2.1. Objetivos específicos

Generación de conjuntos de entrenamiento: Implementar un generador de conjuntos de datos, que a partir de una tabla de datos de entrada, devuelva un conjunto de datos similar, según las técnicas descritas en la introducción (bootstrapping y random subspaces).

Entrenamiento del meta-modelo: Implementar una función de entrenamiento para el meta-modelo. Esta función recibirá el conjunto de datos de entrenamiento, el tipo de algoritmo a usar y el número de modelos que se desean entrenar, además de otros parámetros de entrada que se especificarán en en la siguiente sección. Por cada modelo a entrenar se generará un conjunto de datos (usando el generador anterior) y se usará el algoritmo proporcionado para entrenar un modelo sobre dicho conjunto de datos. Esta función devolverá una lista de modelos entrenados.

Predicciones del meta-modelo: Implementar una función de predicción para el meta-modelo. Esta función recibirá el conjunto de modelos entrenados (obtenido mediante la función de entrenamiento del meta-modelo) y un conjunto de evaluación. Obtendrá la predicción promedio o mayoritaria (según estemos ante un problema de regresión o clasificación) a partir de las predicciones de cada uno de los modelos sobre el conjunto test.

Evaluación del rendimiento: Realizar la evaluación de las predicciones del meta-modelo obtenidas sobre el conjunto test, mediante diferentes métricas de error.

Experimentación: Experimentar con los conjuntos de datos proporcionados y los diferentes parámetros de entrada del meta-algoritmo (número de modelos y proporción de columnas, principalmente). Se seleccionarán y documentarán aquellos experimentos que proporcionen buen rendimiento (según las métricas de rendimiento indicadas) sobre el conjunto de evaluación.

Generalización de las funciones desarrolladas: Todo el desarrollo realizado debe ser generalizable, es decir, no debe estar vinculado a un conjunto de datos, algoritmo o parámetro concreto. En concreto debe ser capaz de abordar diferentes problemas (diferentes conjuntos de datos) de aprendizaje supervisado. Para esto, el código debe estar parametrizado según sea necesario.

3. Descripción del trabajo

3.1. Implementación del algoritmo

El uso de clases para construir el meta-algoritmo facilitará su diseño y usabilidad sin embargo, esto no es obligatorio.

Entrenamiento: El algoritmo de entrenamiento tendrá como entrada, al menos, los siguientes parámetros:

- Conjunto de entrenamiento: es el conjunto de entrenamiento original. Supongamos que tiene N filas y M columnas (excluyendo la variable objetivo del aprendizaje).
- Número de estimadores: Número de modelos que entrenaremos para construir nuestro ensemble.
- Algoritmo de entrenamiento a usar para entrenar los modelos: DecisionTreeClassifier, SGDClassifier, etc.
- Proporción de columnas: Es un número en el rango [0, 1] que indica la proporción de columnas (excluyendo la variable respuesta) o variables predictoras a considerar cuando se entrene cada modelo. Es decir, si tenemos 20 variables predictoras y fijamos una proporción de 0.75, cada modelo se entrenará usando 15 variables seleccionadas aleatoriamente.

El parámetro *Número de modelos* indica el número de modelos a entrenar. Por cada modelo a entrenar, el algoritmo de entrenamiento:

1. Generará un nuevo conjunto de datos con N filas mediante muestreo con reemplazamiento sobre las filas del *conjunto de entrenamiento*. Además, las columnas de este conjunto de datos serán un subconjunto (sin reemplazamiento) aleatorio de las M columnas del conjunto original.
2. Entrenará un modelo sobre ese conjunto de datos.

El resultado será el conjunto de modelos entrenados.

Predicción: La función de predicción recibirá un conjunto de datos de evaluación (no usado en el entrenamiento) y proporcionará una predicción. Para esto, realizará la predicción de cada modelo entrenado en la fase anterior y luego obtendrá una predicción promedio o mayoritaria (según estemos resolviendo un problema de regresión o de clasificación). Además, si se ha proporcionado un conjunto de evaluación para el que se conoce la variable respuesta, se calcularán y proporcionarán las métricas mencionadas en el apartado 2.

Importante: La función de predicción necesitará conocer, para cada modelo, los nombres o índices de las columnas con las que ha sido entrenado.

3.2. Experimentación

Conjuntos de datos: Para esta fase se proporcionarán dos conjuntos de datos (dos para junio y dos para julio) obligatorios y otros adicionales que los alumnos podrán usar **opcionalmente**. Deben usarse los archivos de datos proporcionados por el profesor y no otros disponibles en la web ya que se han realizado tareas de limpieza de datos e imputación de valores faltantes sobre los conjuntos originales.

Nota: los conjuntos de datos se proporcionarán en los días posteriores a la publicación de este documento.

Por cada conjunto de datos, se proporcionan dos archivos, "XXXX_train.csv" y "XXXX_test.csv". El primer fichero es el de entrenamiento y el segundo el de evaluación. Esto significa, que el segundo fichero debe usarse únicamente para evaluar el rendimiento de los modelos. Y el primero para entrenarlos.

Experimento de referencia: En la fase de experimentación, para tener un valor de referencia con el que comparar el rendimiento de nuestro meta-modelo, debemos entrenar y evaluar un modelo simple del mismo tipo que hemos usado para construir el meta-modelo. Si por ejemplo, vamos a evaluar un meta-modelo construido a partir de modelos del tipo SGDClassifier, entonces entrenaremos y evaluaremos un modelo de este tipo.

Mejores hiper-parámetros para el meta-algoritmo: Se explorarán diferentes parámetros de entrada de nuestro meta-algoritmo con el fin de obtener aquellos que mejor rendimiento proporcionen. Se espera que el rendimiento del meta-algoritmo supere ligeramente el del experimento de referencia. Los parámetros a explorar son:

- Número de modelos
- Proporción de columnas
- Hiper-parámetros de los modelos (**opcional**): Podemos variar los hiper-parámetros de los modelos usados para construir el ensamble, como por ejemplo, la profundidad máxima en los árboles de decisión o la tasa de aprendizaje en el algoritmo de descenso por el gradiente.

Repetir experimentos y promediar: Para obtener una estimación de rendimiento fiable (eliminando la variabilidad introducida por el muestro aleatorio) es importante medir el rendimiento para cada combinación de parámetros (es decir, repetir el experimento) mas de una vez y luego calcular el promedio.
Por último, si además de medir el rendimiento sobre el conjunto de evaluación, lo hacemos también sobre el conjunto de entrenamiento, podremos observar como el sobreajuste afecta de diferente manera a los modelos sencillos que al ensamble de modelos.

3.3. Otros detalles y recomendaciones

Librerías externas: Se pueden usar funciones ya implementadas para el cálculo de métricas, entrenamiento de modelos predictivos y obtención de predicciones. Las principales librerías a considerar para el trabajo serán, Pandas y/o Numpy, Scikit-learn y opcionalmente Matplotlib o Seaborn si se desea incluir algún gráfico. No se permite el uso de funciones de alto nivel que implementen el ensemble de modelos completo.

Documentación previa: Antes de comentar a implementar el trabajo, se recomienda que los alumnos se familiaricen con:

- La práctica de aprendizaje automático.
- Las librerías para manipulación de conjuntos de datos Pandas y Numpy.
- Operaciones de preprocesado de datos.
- El framework de aprendizaje automático Scikit-learn.
- El entrenamiento de modelos sobre un conjunto de datos y la evaluación de sus predicciones sobre el conjunto de prueba mediante diferentes métricas.
- Interpretación de las métricas mencionadas en este documento.
- Los conjuntos de datos proporcionados.
- Los ensambles de modelos en general.

Presentación del código: El trabajo debe presentarse en forma de notebook de jupyter que puede ser desarrollados tanto con jupyter-notebook como jupyter-lab, ambos disponibles en el paquete Anaconda. Se proporcionarán las implementaciones solicitadas, así como su aplicación sobre los conjuntos de datos proporcionados. El código y los experimentos realizados deben estar debidamente documentados, las decisiones tomadas debidamente justificadas y los resultados obtenidos deben ser interpretados.

3.4. Documentación y entrega

El trabajo deberá documentarse siguiendo un formato de artículo científico, con una extensión mínima de 6 páginas. En la página web de la asignatura se pueden encontrar plantillas donde se sugiere una estructura general. Estas plantillas siguen el formato de los IEEE conference proceedings, cuyo sitio web guía para autores[1] ofrece información más detallada. El documento entregado deberá estar en formato PDF. Se valorará el uso del sistema LaTeX[2].

La estructura general del documento debe ser como sigue: en primer lugar realizar una **introducción** al trabajo explicando el objetivo fundamental, incluyendo un breve repaso de antecedentes en relación con la temática del trabajo. A continuación, describir la **estructura** del trabajo, las **decisiones de diseño** que se hayan tomado a lo largo de la elaboración del mismo, y la **metodología** seguida al implementarlo (**nunca poner código**, pero sí pseudocódigo), y seguidamente detallar los **experimentos** llevados a cabo, **analizando los resultados obtenidos** en cada uno de ellos. Por último, el documento debe incluir una sección de **conclusiones**, y una **bibliografía** donde aparezcan no solo las referencias citadas en la sección de introducción, sino cualquier documento consultado durante la realización del trabajo (incluidas las referencias web a páginas o repositorios).

La entrega del trabajo consistirá de **un único fichero comprimido zip** conteniendo la memoria, el código implementado (algoritmo y experimentos) en forma de cuaderno de Jupyter y los conjuntos de datos necesarios para ejecutar los experimentos. Es importante usar **rutas relativas** para que el profesor pueda ejecutar el cuaderno en otro ordenador sin modificar el código. Todos los resultados obtenidos y mencionados en el documento deben ser reproducibles en el cuaderno de Jupyter.

1 https://www.ieee.org/conferences/publishing/templates.html
2 https://es.wikibooks.org/wiki/Manual_de_LaTeXa

3.5. Presentación y defensa

El día de la defensa se deberá realizar una pequeña presentación (PDF, PowerPoint o similar) de 10 minutos en la que participarán activamente todos los miembros del grupo que ha desarrollado el trabajo. Esta presentación seguirá, a grandes rasgos, la misma estructura que el documento, pero se deberá hacer especial mención a los resultados obtenidos y al análisis crítico de los mismos. Se podrá usar un portátil (personal del alumno), diapositivas y/o pizarra. En los siguientes 10 minutos de la defensa, el profesor procederá a realizar preguntas sobre el trabajo, que podrán ser tanto del documento como del código fuente. Adicionalmente, el profesor podrá proporcionar un nuevo conjunto de datos con el que probar el algoritmo implementado.

4. Criterios de evaluación

Para la evaluación del trabajo se tendrán en cuenta los siguientes criterios, considerando una nota total máxima de 4 puntos:

Código fuente, experimentación y resultados (2 puntos): se valorará la claridad y buen estilo de programación, corrección, eficiencia y usabilidad de la implementación, y calidad de los comentarios. La documentación del código y los experimentos en el cuaderno de Jupyter también se valorará. En ningún caso se evaluará un trabajo con código copiado directamente de Internet o de otros compañeros. Con respecto a la experimentación, se valorará la calidad y completitud de los experimentos realizados. Además se tendrá en cuenta el rendimiento del algoritmo y el conjunto de mejores parámetros proporcionado. Por último, no se tendrán en cuenta aquellos resultados experimentales que no sean reproducibles.

El documento – artículo científico (1 punto): Se valorará el uso adecuado del lenguaje y el estilo general del documento (por ejemplo, el uso de la plantilla sugerida). Se valorará en general la claridad de las explicaciones, el razonamiento de las decisiones, y especialmente el análisis y presentación de resultados en las secciones de experimentación y conclusiones.

La presentación y defensa (1 punto): se valorará la claridad de la presentación y la buena explicación de los contenidos del trabajo, así como, especialmente, las respuestas a las preguntas realizadas por el profesor.

IMPORTANTE: Cualquier **plagio**, **compartición de código** o uso de material que no sea original y del que no se cite convenientemente la fuente, significará automáticamente la **calificación de cero** en la asignatura para **todos los alumnos involucrados**. Por tanto, a estos alumnos **no se les conserva**, ni para la actual ni para futuras convocatorias, **ninguna nota** que hubiesen obtenido hasta el momento. Todo ello sin perjuicio de las correspondientes **medidas disciplinarias** que se pudieran tomar.

5. Referencias

[1] Métodos de ensamble. http://www.cs.us.es/~fsancho/?e=106
[2] Bootstrapping. https://es.wikipedia.org/wiki/Bootstrapping_(estadística)
[3] Random subspace method. https://en.wikipedia.org/wiki/Random_subspace_method
[4] Random Forests. https://es.wikipedia.org/wiki/Random_forest
[5] Plantilla de documento científico IEEE.