2024/2025 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 2024

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.

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 de aprendizaje:

1. Conocer el análisis de problemas usando máquinas de soporte vectorial, árboles 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

Al resolver el problema explicad bien los que hacéis, no hacer ningún comentario o hacer comentarios superficiales tendrán una nota más baja.

Tenéis que mostrar que habéis entendido los métodos que estáis aplicando, así que un corta y pega de problemas similares no es suficiente.

Para obtener los datos para algunos de estos problemas necesitaréis instalaros la última versión de la librería `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_fraude
```

La función os retornará 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. **Fraude detectado**

   Una aplicación compleja en el área de comercio electrónico es la detección de usos fraudulentos de tarjetas robadas. El volumen de transacciones fraudulentas suele ser pequeño comparado con el volumen total de transacciones y es obviamente más importante que se escape el menor número de ellas. Vamos a trabajar con una parte de conjunto de datos Credit Card Fraud data[1] Este conjunto de datos tiene diferentes atributos correspondientes a la transacción, la persona que ha hecho la compra y el lugar de compra.

   Podéis obtener estos datos mediante la función `load_fraude` de la librería `apafib`. Resolved los siguientes apartados ilustrando los resultados de la manera que os parezca adecuada.

[1] Podéis encontrar los datos originales aquí https://www.kaggle.com/datasets/neharoychoudhury/credit-card-fraud-data.

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, pero eliminad solo las más obvias, si quitáis alguna más deberéis justificar porque lo hacéis. El conjunto de datos tiene una mezcla de atributos fecha, categóricos y numéricos, transformad los categóricos a numéricos. Podéis calcular algunas variables derivadas de las variables fecha que puedan tener sentido para los datos. Partid los datos en conjunto de entrenamiento y test (70 %/30 %).

b) Comenzaremos con un modelo de predicción base ajustando una regresión logística. Fijaos que los que nos interesa es que la mejor predicción sea para la clase *fraude*. Evaluad el modelo. Este modelo puede penalizar los errores en las clases minoritarias usando el parámetro `class_weight`. Usad el valor `balanced` y comprobad si eso mejora el modelo. Comprobad las curvas ROC.

c) Nos interesará también entender qué delata una transacción como fraudulenta. Podemos ajustar un árbol de decisión para obtener un modelo interpretable. Ajustad este modelo explorando adecuadamente sus hiperparámetros. Evaluad la calidad del modelo y comparadla con el anterior. Representad el árbol de decisión, ¿es suficientemente simple para saber cuándo una transacción es fraudulenta?

d) Buscamos la máxima precisión en la clasificación de transacciones fraudulentas. Ajustad un random forest y un gradient boosting explorando adecuadamente sus hiperparámetros. Comparad los resultados con los modelos anteriores.

e) Las SVM se benefician de datos con alta dimensionalidad, ajustad una SVM lineal, una con kernel cuadrático y otra con kernel RBF explorando adecuadamente sus hiperparámetros. Tendréis que hacer que los modelos predigan probabilidades para poder ver la curva ROC. Comparad los resultados con los modelos anteriores.

f) Como hemos visto en teoría, cuando tenemos varios modelos podemos combinarlos usando diferentes estrategias. Entrenad un `StackedRegressor` y un `VotingRegressor` combinando los tres mejores modelos que habéis encontrado en los apartados anteriores con sus mejores hiperparámetros. ¿Es mejor alguno de estos modelos combinados?

g) Calculad la *permutation importance* de los atributos sobre el test para el mejor modelo que habéis encontrado excepto el árbol de decisión ¿coincide la importancia de los atributos con los que considera el árbol en sus decisiones?

2. **Diagnostico del Alzheimer**

Con el envejecimiento de la población, la enfermedad de Alzheimer es una de las enfermedades del envejecimiento que ha cobrado mayor relevancia. Es por eso que un diagnóstico a partir de las características del paciente, pruebas diagnósticas y test de evaluación es una manera simple de obtener una aproximación del diagnóstico. Vamos a usar una parte del conjunto de datos Alzheimer's Disease Dataset[2].

Podéis obtener estos datos mediante la función `load_alzheimer` de la librería `apafib`. Resolved los siguientes apartados ilustrando los resultados de la manera que os parezca adecuada.

a) Los datos están ya preprocesados, veréis que hay variables binarias/categóricas y variables continuas, podéis eliminar las variables identificadoras y las que no tienen información útil. Estamos interesados en la capacidad para diagnosticar de los diferentes tipos de variables, así que analizaremos tres conjuntos de datos, uno solo con las variables binarias/categóricas, otro con las continuas y otro con todas las variables. Partid los tres

[2]Podéis encontrar los datos originales aquí https://www.kaggle.com/datasets/rabieelkharoua/alzheimers-disease-dataset.

conjuntos de datos en entrenamiento y test (70 %/30 %) de manera que cada uno tenga los mismos ejemplos.

b) Empezaremos usando un modelo lineal. Ajustad una regresión logística a cada conjunto de datos explorando adecuadamente sus hiperparámetros. Comparad los modelos. Mirad el peso que les asigna cada modelo a los atributos y comprobad que el orden de importancia que tienen las variables en los modelos con los subconjuntos de variables corresponde con el orden que tienen en el modelo con todos los datos.

c) Para poder entender bien la relación entre las variables y cómo permiten el diagnóstico podemos usar un árbol de decisión. Ajustad este modelo a los tres subconjuntos de datos y visualizad los árboles resultantes. Comparad los atributos que aparecen en los primeros niveles en los tres árboles.

d) La combinación de clasificadores permite tener una mejor precisión. Ajustad un random forest y un gradient boosting a los tres conjuntos de datos explorando adecuadamente sus hiperparámetros. Comparad sus resultados.

e) Calculad la *permutation importance* de los atributos sobre el test para el mejor modelo de combinación de clasificadores para todos los datos que habéis encontrado. Seleccionad un 25 %, 50 % y 75 % de los mejores atributos (siguiendo del orden de importancia) y ajustad al subconjunto de datos un árbol de decisión. Analizad qué atributos son los que usan los árboles en sus primeros niveles.

f) Como hemos visto en teoría, cuando tenemos varios modelos podemos combinarlos usando diferentes estrategias. Entrenad un `StackedRegressor` y un `VotingRegressor` usando los tres árboles de decisión que habéis ajustado en el apartado anterior. Comparad este modelo con el árbol de decisión con todos los datos.

3. **Bicicletas de nuevo**

Este problema continúa la exploración de los datos de bicicletas compartidas del repositorio de UCI que está en la primera lista de problems, si hicisteis ese problema igual tenéis curiosidad sobre si modelos más complejos pueden obtener un mejor resultado. Este conjunto de datos recopila estadísticas agregadas de uso de bicicletas junto con otra información adicional relevante. El objetivo es obtener predicciones del uso del servicio de bicicletas compartidas en una ciudad. Vamos a trabajar con el conjunto de datos Se pueden 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 al día siguiente a partir de los datos de días anteriores (usaremos 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 y explicad por qué las eliminái s.

Necesitaremos obtener variables que nos permitan predecir a partir de la historia del sistema. Tal como están los datos no podemos hacer eso, por lo que necesitaremos un poco de preproceso. La librería `pandas` permite generar una copia de una tabla de datos desplazada una serie de instantes temporales usando el método `shift`. Generad una copia de los datos desplazada un día, dos y tres días y añadidlas como nuevas columnas (eliminad los datos perdidos que se generan). Partid la tabla de datos en las columnas que usaremos para predecir el número de bicicletas que se usarán en un día concreto y añadid las variables del día actual que sabremos

Dado que tenemos que predecir el futuro vamos a seleccionar los primeros 500 ejemplos para entrenamiento y el resto para test. Estandarizad los datos antes de generar los modelos. Eliminad las ventanas del conjunto de test que comparten datos con el conjunto de entrenamiento.

b) Para tener un modelo base usaremos una regresion LASSO. Ajustad este modelo adecuadamente y calculad la *calidad* del modelo empleando validación cruzada y con los datos de test. Para hacer la validación cruzada tenéis que usar el método `TimeSeriesSplit` con 5 particiones en el parámetro `cv` de la validación. Calculad el error de validación cruzada y el del test con el mean absolute error. Representad los valores reales del test contra las predicciones.

c) Empezaremos por el modelo no lineal más sencillo. Ajustad un árbol de regresión a los datos explorando sus hiperparámetros de manera adecuada. Compara los resultados con los del modelo anterior. ¿Son las variables con mayor importancia en el modelo LASSO las que el árbol de decisión usa en sus primeros niveles?

d) La combinación de clasificadores puede permitir obtener un mejor ajuste. Entrenad un random forest y un gradient boosting explorando adecuadamente sus parámetros. Comparad la calidad de estos modelos con los anteriores.

e) Las máquinas de soporte vectorial de regresión permiten trabajar en un espacio de alta dimensionalidad que tiene en cuenta interacciones entre variables. Ajustad primero una SVM lineal y comparad sus pesos con los de la regresión LASSO ¿es el orden de la importancia asignada por los pesos similar? Ajustad SVM con kernel polinómico y kernel RBF explorando adecuadamente los parámetros. Comparad la calidad de estos modelos con los otros.

f) Como hemos visto en teoría, cuando tenemos varios modelos podemos combinarlos usando diferentes estrategias. Entrenad un `StackedRegressor` y un `VotingRegressor` usando el mejor modelo de cada tipo que habéis ajustado en los apartados anteriores (LASSO, combinación de clasificadores, SVM). Comparad los resultados.

CAPÍTULO 2

Problemas Individuales

Al resolver el problema explicad bien los que hacéis, no hacer ningún comentario o hacer comentarios superficiales tendrán una nota más baja.

Tenéis que mostrar que habéis entendido los métodos que estáis aplicando, así que un corta y pega de problemas similares no es suficiente.

Para obtener los datos para estos problemas necesitaréis instalaros la última versión de la librería `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_vuelos
```

La función os retornará 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. **Tráfico aéreo en Barcelona**

   Uno de los datos que recolecta la web de *la ciutat al día* del ayuntamiento de Barcelona es el número de vuelos que llegan y parten del aeropuerto del Prat. Una cosa que nos podemos preguntar es si ese número de vuelos se puede predecir a partir del comportamiento de variables de la ciudad. Es lógico pensar que hay una relación por ejemplo con el tráfico de la ciudad, el número de visitantes de ciudades cercanas o incluso con la temperatura. 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 podemos predecir cuanto tráfico aéreo tiene el aeropuerto en un día específico.

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

a) Se nos ocurre que variables relacionada con la fecha podrían tener bastante importancia en esta predicción. El índice del conjunto de datos es la fecha, extrae el día, el día de la semana, la semana del año y el mes, y añádelos al conjunto de datos.

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. Aplica reducción de dimensionalidad con PCA y t-SNE y representa la variable respuesta sobre la transformación, ¿parece haber alguna relación entre las variables con la variable respuesta? Transforma las variables adecuadamente para poder ajustar modelos de regresión, tanto el conjunto de entrenamiento como el de test. Aplicaremos el logaritmo a la variable a predecir de manera que nos fijaremos en la magnitud del número de vuelos en lugar del número específico de vuelos.

b) Nos podríamos preguntar si la relación entre las variables es simplemente lineal. Ajusta una regresión LASSO a los datos y usa el MSE como medida base para comparar con el resto de modelos. Fíjate también en el valor de los pesos que asigna la regresión a cada variable.

c) Ajusta una SVM de regresión con kernel lineal y con kernel RBF explorando los hiperparámetros adecuadamente. Usa el método de *permutation importance* sobre el test para determinar qué atributos son más importantes para la SVM con kernel RBF. Compara los resultados de estos modelos con la regresión lineal.

d) 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.

e) Como hemos visto en teoría, cuando tenemos varios modelos podemos combinarlos usando diferentes estrategias. Entrena un `StackedRegressor` y un `VotingRegressor` con los tres 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?

2. **Noches de ruido**

Todas las ciudades grandes son ruidosas, y Barcelona no es una excepción. El nivel de ruido de madrugada puede ser algo molesto, y nos gustaría poder predecir cuanto ruido tendremos a partir de un conjunto de variables que se capturan de la ciudad. Esto nos permitirá ver qué factores influyen para ver como mitigar el problema.

El ayuntamiento de Barcelona recolecta diversos datos sobre la ciudad en su portal de datos abiertos[1]. Vamos a trabajar con un extracto de esos datos para los años 2022-2023, eligiendo un subconjunto de variables que pueden tener alguna relación con el ruido, como por ejemplo el nivel de tránsito, variables meteorológicas, el número de aviones que salen y llegan al aeropuerto, el uso diario de electricidad y el movimiento de personas desde ciudades cercanas.

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

a) Se nos ocurre que variables relacionadas con la fecha podrían tener bastante importancia en esta predicción. El índice del conjunto de datos es la fecha, extrae el día, el día de la semana, la semana del año y el mes, y añádelos al conjunto de datos.

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. Aplica reducción de dimensionalidad con PCA y t-SNE y representa la variable respuesta sobre la transformación, ¿parece haber alguna relación entre las variables con la variable respuesta? Transforma las variables adecuadamente para poder ajustar modelos de regresión, tanto el conjunto de entrenamiento como el de test.

b) Nos podríamos preguntar si la relación entre las variables es simplemente lineal. Ajusta una SVM de regresión con kernel lineal a los datos y usa el MSE como medida base para comparar con el resto de modelos. Fíjate también en el valor de los pesos que asigna la SVM a cada variable.

c) Ajusta una SVM de regresión con kernel polinómico y RBF explorando los hiperparámetros adecuadamente. Usa el método de *permutation importance* sobre el test para determinar qué atributos son más importantes. Compara los resultados de estos modelos con la SVM lineal.

d) 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.

e) Como hemos visto en teoría, cuando tenemos varios modelos podemos combinarlos usando diferentes estrategias. Entrena un `StackedRegressor` y un `VotingRegressor` usando los tres mejores 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?

3. **De visita**

A diario, mucha gente que está durante el día en Barcelona trabaja en la ciudad, pero vive en una localidad cercana. Los datos que tiene el ayuntamiento de Barcelona sobre su número proviene de información que obtiene de los operadores móviles[1]. Esta información obviamente tiene un coste, y nos gustaría saber si la podemos aproximar con variables que toma directamente la ciudad. Para ello vamos a trabajar con un extracto de los datos que recogió el ayuntamiento de Barcelona durante los años 2022-2023, eligiendo variables como la temperatura, la contaminación acústica, el tráfico, los accidentes en la ciudad, las multas de tráfico diarias, o el consumo de electricidad de la ciudad.

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

a) Se nos ocurre que variables relacionadas con la fecha podrían tener bastante importancia en esta predicción. El índice del conjunto de datos es la fecha, extrae el día, el día de la semana, la semana del año y el mes, y añádelos al conjunto de datos.

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. Aplica reducción de dimensionalidad con PCA y t-SNE y representa la variable respuesta sobre la transformación, ¿parece haber alguna relación entre las variables con la variable respuesta? Transforma las variables adecuadamente para poder ajustar

[1] Lo saben todo de nosotros, es el precio que pagamos por estar siempre conectados.

modelos de regresión, tanto el conjunto de entrenamiento como el de test. Aplicaremos el logaritmo a la variable a predecir de manera que nos fijaremos en la magnitud del número de personas en lugar de su número específico.

b) Asumimos que días parecidos tendrán un número de visitantes parecido, así que podríamos ajustar un K vecinos cercanos para tener un resultado base con el que comparar. Ajusta este modelo explorando adecuadamente sus hiperparámetros, Usa *permutation importance* sobre el test para determinar qué atributos son más importantes en el modelo para predecir.

c) El usar K vecinos cercanos nos obliga guardar todos los datos para predecir. Con una SVM solo tenemos que guardar los que son necesarios para calcular los vectores de soporte. Ajusta una SVM de regresión con kernel polinómico y RBF explorando los hiperparámetros adecuadamente. Usa el método de *permutation importance* sobre el test para determinar qué atributos son más importantes. Compara los resultados de estos modelos con los del modelo anterior.

d) 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.

e) Como hemos visto en teoría, cuando tenemos varios modelos podemos combinarlos usando diferentes estrategias. Entrena un `StackedRegressor` y un `VotingRegressor` usando la combinación del mejor modelo que has encontrado en los apartados anteriores para cada tipo de clasificador (Knn, Árboles de decisión, SVM) 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?

4. **Love is in the air**

Uno de los múltiples negocios que nos trajo la era de internet es el de las aplicaciones de citas. Lo habitual es que se construyan perfiles a partir de las respuestas de las personas que permitan después hacer recomendaciones o estudiar las tendencias que siguen los usuarios. Vamos a trabajar con un subconjunto de datos que corresponden a perfiles de OK Cupid[2]. Se han construido ocho perfiles a partir de los datos que corresponden a un análisis de los expertos de OKCupid. El objetivo es obtener un clasificador que permita identificar el perfil de nuevos usuarios a partir de sus respuestas.

Puedes obtener estos datos mediante la función `load_cupid` 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 %). 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. Aplica reducción de dimensionalidad con PCA y t-SNE y representa la variable respuesta sobre la transformación, ¿parece haber alguna relación entre las variables con la variable respuesta?

El conjunto de datos tiene mayoritariamente variables categóricas y alguna continua. Transforma primero las variables categóricas a one hot encoding[3]. Normaliza los datos.

[2] La fuente original la podéis encontrar aquí https://www.kaggle.com/datasets/andrewmvd/okcupid-profiles

[3] Mejor que hagas la transformación antes de partir los datos para evitar que haya categorías que no aparezcan en una de las particiones, no es como deberíamos hacerlo, pero es más sencillo.

b) Partiremos de un resultado base con el que comparar. Ajusta una regresión logística a los datos explorando adecuadamente sus hiperparámetros. Evalúa adecuadamente el resultado del modelo.

c) Los perfiles probablemente correspondan a particiones no lineales de los datos. Ajusta una SVM con kernel cuadrático y con kernel RBF. Configúralos para que predigan la probabilidad de las clases. Explorando adecuadamente sus hiperparámetros. Evalúa el resultado de los modelos.

d) El espacio que tenemos para los datos es mayoritariamente discreto, y probablemente sea sencillo hacer particiones que permitan separar los perfiles. Ajusta un modelo Random Forest y un modelo Extreme Randomized Trees a los datos explorando adecuadamente sus hiperparámetros. Evalúa el resultado de los modelos y compáralos con los modelos anteriores.

e) Como hemos visto en teoría, cuando tenemos varios modelos podemos combinarlos usando diferentes estrategias. Entrena un `StackedClassifer` y un `VotingClassifier` usando la combinación de los tres mejores modelos que has encontrado en los apartados anteriores usando sus mejores hiperparámetros ¿Es mejor alguno de estos modelos combinados?

5. **Conoce a tus clientes**

La segmentación de clientes es una de las tareas más frecuentes en el análisis de datos. Un problema de los algoritmos de segmentación es que no permiten saber cuales son los criterios que definen la partición de los datos. Para ello hay que recurrir a métodos supervisados e interpretar que atributos definen las clases de clientes. Vamos a trabajar con un conjunto de datos de clientes[4] que ya ha sido segmentado, y queremos obtener clasificadores de los que podamos extraer los atributos más importantes o que podamos interpretar.

Puedes obtener estos datos mediante la función `load_clientes` 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 %). 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. Aplica reducción de dimensionalidad con PCA y t-SNE y representa la variable respuesta sobre la transformación, ¿parece haber alguna relación entre las variables con la variable respuesta? Normaliza los datos.

b) Un modelo lineal es directamente interpretable a partir de sus pesos. Ajusta una regresión logística y una SVM lineal explorando adecuadamente sus hiperparámetros. Evalúa la calidad de los modelos y compara la importancia que asignan los modelos a los atributos para cada clase.

c) Otro modelo directamente interpretable es el árbol de decisión. Ajusta un árbol de decisión a los datos explorando adecuadamente sus hiperparámetros. Representa el árbol de decisión resultante y mira cuáles son los atributos que corresponden a los primeros niveles del árbol, compáralos con los atributos más importantes de los modelos lineales del apartado anterior.

d) Es posible que haya interacciones no lineales entre los atributos que expliquen mejor las clases. Ajusta una SVM con kernel RBF explorando adecuadamente sus hiperparámetros. Compara la calidad de estos modelos con los previos. Calcula la *permutation importance*

[4] Puedes acceder a la documentación de los datos aquí https://www.kaggle.com/datasets/mbsoroush/customer-data/code.

sobre el test para determinar qué atributos son más importantes en el modelo para predecir. Compara los atributos más importantes con los de los apartados anteriores.

e) Los conjuntos de clasificadores permiten adaptarse mejor a la complejidad de los datos. Ajusta un random forest a los datos explorando adecuadamente sus hiperparámetros. Compara la calidad del modelo con los anteriores. Calcula la *permutation importance* sobre el test para determinar qué atributos son más importantes en el modelo para predecir. Compara los atributos más importantes con los de los apartados anteriores. ¿Hay cierto acuerdo entre los modelos respecto a cuáles son los atributos más importantes?

6. **Duerme bien**

Los problemas del sueño son algo común en las sociedades industrializadas que pueden derivar en problemas graves de salud. Por eso es importante detectarlos a tiempo. Vamos a trabajar con una versión de un conjunto de datos que recoge diferentes variables sobre pacientes que tienen problemas de sueño (insomnio/apnea) y sujetos controles que no tienen ningún problema[5]. El objetivo es tener un modelo que pueda clasificar nuevas personas y poder obtener unos criterios que puedan permitir entender el diagnóstico.

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

a) Primero elimina las variables identificadoras y transforma la variable *Blood Pressure* de manera que sean dos variables enteras (alta y baja presión). Transforma las variables categóricas con *one hot encoding*. Divide el conjunto de datos en entrenamiento y test (70 %/30 %). 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. Aplica reducción de dimensionalidad con PCA y t-SNE y representa la variable respuesta sobre la transformación, ¿parece haber alguna relación entre las variables con la variable respuesta? Normaliza los datos.

b) Un modelo lineal es directamente interpretable a partir de sus pesos. Ajusta una regresión logística y una SVM lineal explorando adecuadamente sus hiperparámetros. Evalúa la calidad de los modelos y compara la importancia que asignan los modelos a los atributos para cada clase.

c) Otro modelo directamente interpretable es el árbol de decisión. Ajusta un árbol de decisión a los datos explorando adecuadamente sus hiperparámetros. Representa el árbol de decisión resultante y mira cuáles son los atributos que corresponden a los primeros niveles del árbol, compáralos con los atributos más importantes de los modelos lineales del apartado anterior.

d) Los conjuntos de clasificadores permiten adaptarse mejor a la complejidad de los datos. Ajusta un random forest a los datos explorando adecuadamente sus hiperparámetros. Compara la calidad del modelo con los anteriores. Calcula la *permutation importance* sobre el test para determinar qué atributos son más importantes en el modelo para predecir. Compara los atributos más importantes con los de los apartados anteriores. ¿Hay cierto acuerdo entre los modelos respecto a cuáles son los atributos más importantes?

e) Como hemos visto en teoría, cuando tenemos varios modelos podemos combinarlos usando diferentes estrategias. Entrena un `StackedClassifer` y un `VotingClassifier` usando los tres mejores modelos ente los dos modelos lineales y los dos de combinación de

[5]Puedes encontrar la descripción de las variables aquí https://www.kaggle.com/datasets/uom190346a/sleep-health-and-lifestyle-dataset.

árboles de decisión usando sus mejores hiperparámetros ¿Es mejor alguno de estos modelos combinados? Calcula la *permutation importance* sobre el test para determinar qué atributos son más importantes en la mejor combinación. Compara los atributos más importantes con los de los apartados anteriores.