2024/2025 1Q

Lista de Problemas 2

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 individual 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.

Aseguraos de que los notebooks mantienen la solución que habéis obtenido, no los entreguéis sin ejecutar.

Objetivos de aprendizaje:

1. Identificar y aplicar el preproceso más adecuado a un conjunto de datos
2. Saber aplicar técnicas de reducción de dimensionalidad para la visualización de datos
3. Saber crear y aplicar clasificadores generativos y discriminativos
4. Interpretar los resultados de un problema de clasificación

CAPÍTULO 1

Problemas de Grupo

Al resolver el problema explicad bien lo 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.

No os aconsejo que uséis Colab para resolver los ejercicios, algunos experimentos os pueden tardar mucho. Es importante que en los métodos de `scikit learn` que tienen un parámetro `n_jobs` uséis como valor `-1` para aprovechar la ejecución multi núcleo.

Para obtener los datos de 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_stroke
```

La función por lo general os retornará un `DataFrame` de Pandas con los datos. Si no es así el enunciado explicará que retorna.

1. **¿Quién dijo pío?**

   El determinar la fuente de una noticia empieza a ser un problema importante en internet. En las diferentes redes sociales aparecen informaciones que a veces tienen atribuciones de fuentes falsas o se han tomado de algún sitio sin citar el origen. El conjunto de datos *Health News in Twitter*[1] contiene tweets sobre salud de diferentes fuentes periodísticas. Vamos a trabajar con un

[1] https://archive.ics.uci.edu/dataset/438/health+news+in+twitter

subconjunto de esos datos para determinar la fuente original de la información que se publicó en Twitter (ahora X).

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

a) El trabajar con texto es algo diferente de los datos tabulares habituales. En este tipo de datos se obtiene la matriz de datos a partir de extraer un subconjunto de las palabras de los textos y hacer un cálculo sobre ellas como por ejemplo determinar si están o no en un documento o el número de veces que aparecen. Esto lo podemos hacer con la función `CountVectorizer` de `scikit-learn` que es precisamente para este tipo de datos. Generaremos matrices de datos variando el tamaño del vocabulario, usaremos 50, 100 y 250 palabras[2]. Podemos generar la matriz de datos para que contenga solo si las palabras aparecen o no en cada ejemplo, comprobaremos si con eso es suficiente para resolver el problema. Utilizad el parámetro `stop_words` que tiene esta función indicando que el idioma del texto es el inglés.

   Esto nos dará tres conjuntos de datos diferentes. Fijaos que esto **es un preproceso** como otros que hemos ido empleando, así que aplicadlo correctamente a los datos. Generad una partición de entrenamiento y una de test (80 %/20 %) que sea estratificada (fijad también el estado del generador de números aleatorios para la reproducibilidad).

b) Haced una visualización de los datos mediante PCA y t-SNE. ¿Se puede ver alguna separabildad entre los datos en la proyección en 2D? Comentad los resultados.

c) Tenemos datos que son binarios (Bernoulli). Si asumimos que las palabras de un texto son independientes (no lo son realmente) podemos usar Naïve Bayes para clasificarlos. Tenéis en `scikit-learn` modelos de Naïve Bayes que usan esta distribución. Aplicad este método a los conjuntos de datos que habéis generado y determinad la calidad de los modelos (la calidad no se limita solo al acierto). Comentad los resultados.

d) Ajustad un modelo de regresión logística a los conjuntos de datos que habéis generado explorando los hiperparámetros del modelo y determinad la calidad de los mejores modelos. Comentad los resultados.

e) Probablemente, la frontera entre las fuentes de datos sea bastante compleja. Dado que tenemos bastantes datos, esto es algo que podría aprovechar k-vecinos cercanos. Ajustad adecuadamente los hiperparámetros de este modelo a los datos y determina la calidad de los mejores modelos. Comentad los resultados.

f) Al calcular el vocabulario a partir de la frecuencia de las palabras, probablemente habrá algunas muy frecuentes que no aporten demasiado. Tomad el conjunto de datos con el vocabulario que haya funcionado mejor. El objeto que hace la transformación tiene un atributo `vocabulary_` que ordena el vocabulario que se ha usado según la frecuencia de las palabras. Eliminad las palabras que supongan el primer 10 % y 20 % más frecuentes del conjunto de datos. Reajustad los tres modelos que habéis utilizado con este nuevo conjunto de datos y comparad los resultados.

2. **Desbalance**

   En los dominios médicos, suele haber un desequilibrio entre las clases a predecir dado que muchas veces hay más datos para la condición normal que para las otras condiciones. Esto dificulta

[2] Fijate que este modelo tiene un parámetro `max_features` que te permite escoger el tamaño del vocabulario.

la obtención de un modelo con buena calidad que realmente discrimine las clases que nos interesan. El conjunto de datos *MIT-BIH Arrhythmia Database*[3] es un conjunto de datos que recopila ECGs de pacientes y tiene anotaciones médicas indicando latidos que corresponden a diferentes problemas de corazón, pero especialmente arritmias. Vamos a trabajar con un conjunto de datos reducido generado a partir del original. En el conjunto de datos que trataréis hay cuatro arritmias diferentes, más latidos normales. El objetivo es obtener un clasificador que sea capaz de distinguir entre las diferentes clases.

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

a) Dividid los datos en un conjunto de entrenamiento y otro de test (70 %/30 %), la partición debe estratificarse (fijad también el estado del generador de números aleatorios para la reproducibilidad). El conjunto de datos ya está preprocesado de manera que todas las variables estén en el rango 0-1. Haced una visualización de los datos mediante PCA y t-SNE. ¿Se puede ver alguna separabildad entre los datos en la proyección en 2D? Comentad los resultados.

b) Asumir que los atributos del conjunto de datos son independientes no es una buena suposición (¿por qué?), pero aun así podemos ajustar un modelo naive Bayes y tomarlo como resultado base. Comentad los resultados.

c) Ajustad un modelo discriminante lineal y una regresión logística con penalización `L2` usando como optimizador `lbfgs`. Ajustad adecuadamente los hiperparámetros de estos modelos a los datos y determinad la calidad de los mejores modelos. Comentad los resultados.

d) La regresión logística tiene un parámetro `class_weight` que permite cambiar el peso del error que tienen las diferentes clases. En este caso nos importa menos el error que se comete en la clase normal. Tenemos la opción de usar el valor `balanced` que calcula el peso de las clases a partir del inverso de su frecuencia. Ajustad de nuevo este modelo de esta manera. Comentad los resultados y comparad el modelo con los otros. Se puede asignar también un peso específico por clase. Explorad un poco los pesos de las clases para ver si obtenéis un mejor resultado, considerando que es mejor clasificar más ejemplos de arritmias.

e) Ajustad un modelo k-vecinos cercanos a los datos explorando adecuadamente sus parámetros. Evaluad la calidad del modelo y comparadlo con los otros.

f) Una posibilidad que nos podemos plantear es obtener un modelo que sea capaz de distinguir entre latidos normales y arritmias y luego tener un clasificador solo de arritmias. Escoged el modelo que haya funcionado mejor y ajustadlo para tener un modelo para clasificar normal/arritmia y para clasificar arritmias entre si. Usad la combinación del resultado de los dos modelos para clasificar el conjunto de datos y comparad los resultados con los modelos anteriores.

3. **¿Sabes qué comes?**

El comer adecuadamente es importante para mantener la salud. Necesitamos una dieta equilibrada con los suficientes micronutrientes para mantenernos en forma. Vamos a utilizar un conjunto de datos que describe los nutrientes de un conjunto de alimentos y platos[4] Este conjunto de datos se ha clasificado en ocho grupos diferentes de alimentos a partir de sus nutrientes, vamos a explorar diferentes clasificadores lineales para ver si podemos encontrar un modelo que sea capaz de predecir el grupo al que pertenece un alimento.

[3] La fuente original se encuentra en https://physionet.org/content/mitdb/1.0.0/.

[4] Podéis encontrar una descripción de los datos en https://www.kaggle.com/datasets/utsavdey1410/food-nutrition-dataset/data.

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

a) Dividid los datos en un conjunto de entrenamiento y otro de test (70 %/30 %), la partición debe estratificarse (fijad también el estado del generador de números aleatorios para la reproducibilidad). Normalizad los datos. Haced una visualización mínima de las variables para haceros una idea de los datos y calculad sus correlaciones. Haced una visualización de los datos mediante PCA y t-SNE. ¿Se puede ver alguna separabildad entre los datos en la proyección en 2D? Comentad los resultados.

b) Podemos asumir cierta independencia entre los nutrientes (o quizás no) así que podemos ajustar un modelo naive Bayes y tomarlo como resultado base. Comentad los resultados.

c) Ajustad un modelo discriminante lineal y una regresión logística con penalización `L1` usando como optimizador `saga`. Ajustad adecuadamente los hiperparámetros de estos modelos a los datos y determinad su calidad. Comentad los resultados.

d) La regresión logística es un modelo que podemos interpretar a partir de sus pesos. Dado que hemos usado `L1` como regularización es posible que algunos atributos no sean importantes para algunas clases o directamente no intervengan en la separación de las clases. Analizad los pesos y comentad lo que observéis. Un algoritmo eficiente de reducción de atributos es la *eliminación recursiva de características* (Recursive Feature Elimination). Tenéis una implementación en `scikit-learn` que utiliza validación cruzada para la evaluación (`RFECV`). Para aplicarlo hay que pasarle un modelo ya ajustado. Usaremos el modelo de regresión logística. Obtened con este método un conjunto de atributos reducido y volved a ajustar los hiperparámetros de este modelo con estos datos. Comparad los resultados con los modelos anteriores. ¿Han cambiado los pesos de los atributos más importantes?

e) Ajustad un modelo k-vecinos cercanos a los datos explorando adecuadamente sus parámetros. Evaluad la calidad del modelo y comparadlo con los otros.

f) Dado que hemos reducido el conjunto de datos a partir de los resultados de la regresión logística podríamos plantearnos si los otros modelos podrían beneficiarse de ello. Ajustad el resto de modelos con el conjunto de datos reducido y evaluad los resultados ¿Qué modelo es el mejor de todos?

CAPÍTULO 2

Problemas Individuales

Al resolver el problema explicad bien lo 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.

No os aconsejo que uséis Colab para resolver los ejercicios, algunos experimentos os pueden tardar mucho. Es importante que en los métodos de `scikit learn` que tienen un parámetro `n_jobs` uséis como valor `-1` para aprovechar la ejecución multi núcleo.

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_literature
```

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. **De vacaciones**

   Un problema frecuente en el análisis de las web de opiniones es el obtener perfiles de las personas para luego poder hacer predicciones y asociarlo con recomendaciones. El conjunto de datos *Travel Reviews*[1] recolecta la media de las opiniones de un conjunto de usuarios sobre diferentes

[1] Los datos provienen de https://archive.ics.uci.edu/dataset/484/travel+reviews y se han añadido un conjunto de perfiles sobre estos.

categorías de atracciones/lugares/locales turísticos. Todas las opiniones de cada usuario de una categoría se asociaron a un valor entre 0 y 4, y luego se hizo una media de las opiniones. El objetivo de este problema es experimentar con diferentes clasificadores lineales para evaluar la posibilidad de usarlos para hacer predicciones del perfil de los usuarios (atributo `class`).

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

a) Divide los datos en un conjunto de entrenamiento y otro de test (70 %/30 %), la partición debe estratificarse (fija también el estado del generador de números aleatorios para la reproducibilidad). Haz una visualización mínima de las variables para hacerte una idea de los datos y calcula sus correlaciones. Verás que las variables son todas continuas. Algunas variables no parecen muy gausianas. Haremos dos conjuntos de datos, uno que tenga las variables originales y otro en el que las variables que se desvíen bastante de la gausianidad las transformaremos a algo más cercano a una gausiana usando el método `QuantileTransformer` de `scikit-learn`. Este usa los cuantiles de los datos para que coincidan con los cuantiles de la gausiana, básicamente hace corresponder la CDF empírica de los datos y la CDF teórica de la gausiana. Normaliza adecuadamente los datos.

b) Empezaremos por Naïve Bayes, ajusta un modelo para distribuciones gausianas a los dos conjuntos de datos y evalúa adecuadamente la calidad de los modelos.

c) Al hacer la exploración de los datos probablemente viste que hay cierta correlación entre variables, por lo que es posible que los datos se desvíen de lo que asume Naïve Bayes. Ajusta un modelo discriminante lineal (LDA) a los dos conjuntos de datos y evalúa adecuadamente la calidad de los modelos.

d) Ajusta un modelo de regresión logística a los dos conjuntos de datos explorando los hiperparámetros de este modelo adecuadamente (el conjunto de datos es pequeño, así que el ajuste debería ser rápido). Evalúa la calidad de los modelos.

Dado que las clases que tiene el conjunto de datos no están completamente balanceadas podríamos usar el parámetro `class_weight` de este modelo para que se tenga en cuenta esta circunstancia usando del valor `balanced`. Haz un nuevo ajuste de la regresión logística con los dos conjuntos de datos utilizando este hiperparámetro. ¿Qué modelo elegirías de entre todos los que has obtenido? ¿Por qué?

Analiza los pesos del mejor modelo de regresión logística e intenta interpretar que atributos permiten separar las clases.

2. **Spam vs ham**

El spam es un problema generalizado en las redes sociales y las aplicaciones de mensajería. El interés para las compañías que gestionan estas redes y aplicaciones es el poder detectar lo antes posible cuando se empiezan a enviar estos mensajes para evitar el coste de su procesado y el que lleguen a los usuarios. El conjunto de datos *SMS Spam Collection*[2] es un corpus de diferentes colecciones de mensajes SMS legítimos y no deseados.

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

a) El trabajar con texto es algo diferente de los datos tabulares habituales. En este tipo de datos se obtiene la matriz de datos a partir de extraer un subconjunto de las palabras de los textos y hacer un cálculo sobre ellas como por ejemplo determinar si están o no en

[2]Podéis consultar su documentación en https://archive.ics.uci.edu/dataset/228/sms+spam+collection

un documento o el número de veces que aparece. Esto lo podemos hacer con la función `CountVectorizer` de `scikit-learn` que es precisamente para este tipo de datos. Generaremos matrices de datos variando el tamaño del vocabulario, usaremos 100 y 250 palabras[3]. Podemos generar la matriz de datos para que contenga solo si las palabras aparecen o no en cada ejemplo o el número de veces que aparecen, usaremos las dos opciones. Utiliza el parámetro `stop_words` que tiene esta función indicando que el idioma del texto es el inglés. Esto nos dará cuatro conjuntos de datos diferentes. Fíjate que esto **es un preproceso** como otros que hemos ido empleando, así que aplícalo correctamente a los datos. Genera una partición de entrenamiento y una de test (80 %/20 %) que sea estratificada (fija también el estado del generador de números aleatorios para la reproducibilidad).

Haz una visualización de los datos mediante PCA. ¿Se puede ver alguna separabildad entre los datos en la proyección en 2D? Comenta los resultados.

b) Tenemos datos que o son binarios (Bernoulli) o corresponden a distribuciones multinomiales respectivamente. Si asumimos que las palabras de un texto son independientes (no lo son realmente) podemos usar Naïve Bayes para clasificarlos. Tienes en `scikit-learn` modelos de Naïve Bayes que usan estas dos distribuciones. Aplica estos métodos a los conjuntos de datos que has generado y determina la calidad de los modelos (la calidad no se limita solo al acierto). Comenta los resultados.

c) Los modelos generativos funcionan bien si la suposición que hacemos sobre la distribución de los datos es correcta. Los modelos discriminativos como Regresión Logística pueden ser adecuados cuando no esta clara la distribución de los atributos. Ajusta un modelo de regresión logística a los conjuntos de datos que has generado explorando los hiperparámetros del modelo y determina la calidad de los mejores modelos. Comenta los resultados.

d) Probablemente, la frontera entre los SMS legítimos y los no deseados sea bastante compleja. Dado que tenemos bastantes datos, esto es algo que podría aprovechar k-vecinos cercanos. Ajusta adecuadamente los hiperparámetros de este modelo a los datos y determina la calidad de los mejores modelos. Comenta los resultados.

¿Qué modelo elegirías entre todos? Si nuestro objetivo fuera el poder determinar la clase de un SMS de la manera más eficiente posible ¿qué modelo te parecería el mejor? No te limites a escoger uno, piensa en todas las cosas que influyen en el coste de clasificar un ejemplo.

3. **Funciona hasta que se rompe**

En la gestión de los centros de datos es importante el anticipar cualquier fallo de hardware para mantener la calidad de servicio. El conjunto de datos *tokyo1* tiene las estadísticas de varios meses de la carga de un conjunto de servidores clasificados según si el servidor tuvo un fallo durante el siguiente mes (0) o no (1). El objetivo es obtener un modelo lineal que nos pueda servir para poder prever futuros fallos.

Este conjunto de datos está incluido en los Penn Machine Learning Benchmarks. Para descargarlo tendrás que seguir las instrucciones de su página web (https://epistasislab.github.io/pmlb/index.html).

a) Vamos a comenzar con el conjunto de datos original. Divide los datos en entrenamiento y test (70 %/30 %), la partición debe estratificarse (fija también el estado del generador de números aleatorios para la reproducibilidad). Haz una visualización mínima de las variables para hacerte una idea de los datos y calcula sus correlaciones. Aplica al menos dos métodos de reducción de dimensionalidad para explorar la separabilidad de los datos según sus clases.

[3] Fijate que este modelo tiene un parámetro `max_features` que te permite escoger el tamaño del vocabulario.

b) Entrena un naive bayes, un discriminante lineal y una regresión logística explorando sus hiperparámetros adecuadamente. Preprocesa primero los datos para estos modelos. Obtén el acierto de validación cruzada, el acierto en test, la matriz de confusión y el informe de clasificación. Analiza también las curvas ROC.

c) Muchas de las variables que se están utilizando en el conjunto de datos son redundantes. Podemos reducir los atributos necesarios explorando la relevancia de los atributos. El discriminante lineal y la regresión logística permiten determinar la relevancia a partir de los coeficientes del modelo. Un algoritmo eficiente de reducción de atributos es la *eliminación recursiva de características* (Recursive Feature Elimination). Tienes una implementación en `scikit-learn` que utiliza validación cruzada para la evaluación (`RFECV`). Para aplicarlo tendrás que pasarle un modelo ajustado con los mejores parámetros que has encontrado en el apartado anterior. Obtén con este método un conjunto de atributos reducido para los dos modelos y vuelve a ajustar sus hiperparámetros con estos datos. Compara los resultados con los modelos anteriores. ¿Han cambiado los pesos/orden de los atributos más importantes?

d) Usa PCA para obtener un conjunto de datos que mantenga al menos un 95 % de la variancia y ajusta de nuevo los dos modelos con estos datos. Compara los resultados con el resto de los modelos. ¿Qué modelo te parece mejor?

4. **Adaptando los datos**

Es usual que haya conjuntos de datos que mezclen valores discretos y continuos, especialmente en dominios médicos. Lo habitual es intentar adaptar los datos para que todos estén representados de la misma manera. El conjunto de datos *Early Stage Diabetes Risk Prediction*[4] es un cuestionario recogido para determinar el riesgo de diabetes temprana de un conjuto de personas. Este conjunto de datos tiene la particularidad de que se solamente uno de los atributos es continuo, así que experimentaremos con diferentes maneras de adaptar este atributo al resto.

Los datos se pueden cargar usando la librería `ucimlrepo` mediante el método `fetch_ucirepo` usando 529 como valor del parámetro `id`. Esto retorna una estructura desde la que se puede acceder a los datos y sus características.

a) Divide los datos en entrenamiento y test (70 %/30 %), la partición debe estratificarse (fija también el estado del generador de números aleatorios para la reproducibilidad). Haz una visualización mínima de las variables para hacerte una idea de los datos y calcula sus correlaciones. Transformaremos el atributo edad a discreto utilizando el método `KBinsDiscretizer`. Obtendremos una matriz de datos binaria usando como parametro `encoder-onehot-dense` aplicando dos formas alternativas de obtener los intervalos `uniform` y `quantile`. Generaremos conjuntos de datos con 2, 3 y 5 particiones. Aplica correctamente estos preprocesos a los datos.

b) Empezaremos por Naïve Bayes y regresión logística. Ajusta un modelo para los conjuntos de datos que has generado y evalúa adecuadamente la calidad de los modelos. Determina cuál es el mejor preproceso para la variable `edad`.

c) Una ventaja que tiene el usar un modelo interpretable como la regresión logística es que podemos saber la importancia de los atributos. Mira los pesos del mejor modelo y determina qué es lo más importante para el diagnóstico. ¿Es la edad importante?

d) A veces es mejor tener una mayor precisión en el diagnóstico que el poder interpretar los resultados. Ajusta un modelo k-vecinos cercanos al conjunto de datos que obtiene el mejor

[4] Podéis consultar su documentación en https://archive.ics.uci.edu/dataset/529/early+stage+diabetes+risk+prediction+dataset

resultado en los modelos anteriores explorando adecuadamente sus parámetros. Compara los resultados con los otros modelos. Todos los modelos que has usado permiten una predicción probabilística y, por lo tanto, podemos decidir en que punto podemos poner el límite para asegurar que todos los casos positivos se acierten a pesar de introducir falsos positivos. ¿Qué modelo sería mejor para hacer eso?

5. **Un problema de peso**

   La prevención es importante en medicina y la evaluación de manera automática de diferentes condiciones a partir de cuestionarios permite agilizar el proceso. El conjunto de datos *Estimation of Obesity Levels*[5] recoge la información de una población sobre sus hábitos y los clasifica según una escala discreta. El objetivo del problema es probar diferentes modelos que puedan dar de manera automática esa escala.

   Los datos se pueden cargar usando la librería `ucimlrepo` mediante el método `fetch_ucirepo` usando 544 como valor del parámetro `id`. Esto retorna una estructura desde la que se puede acceder a los datos y sus características.

   a) Divide los datos en entrenamiento y test (70 %/30 %), la partición debe estratificarse (fija también el estado del generador de números aleatorios para la reproducibilidad). Tienes un conjunto de variables que son discretas, transfórmalas para generar con ellas variables binarias (one hot encoding). Normaliza adecuadamente el conjunto de datos. Haz una visualización mínima de las variables para hacerte una idea de los datos y calcula sus correlaciones. Aplica un PCA y un t-SNE y evalúa la separación de los datos en clases según lo que se observa en la representación en dos dimensiones.

   b) Empezaremos por Naïve Bayes y un discriminante lineal, dado que tenemos una mezcla de variables binarias y continuas asumiremos que todos los datos son continuos. Elige adecuadamente los modelos y evalúa su calidad.

   c) Ajusta una regresión logística explorando adecuadamente sus hiperparámetros. Evalúa adecuadamente la calidad de los modelos y compáralos entre sí. Una ventaja que tiene el usar un modelo interpretable como la regresión logística es que podemos saber la importancia de los atributos. Mira los pesos del modelo y determina que es lo más importante para el diagnóstico en cada clase. ¿Hay variables que tienen más peso que otras en las diferentes clases?

   d) A veces es mejor tener una mayor precisión en el diagnóstico que el poder interpretar los resultados. Ajusta un modelo k-vecinos cercanos explorando adecuadamente sus parámetros. Compara los resultados con los otros modelos. ¿Cuál es el que podría hacer mejor la labor de asignar un valor de la escala a nuevas personas?

6. **La dura vida del estudiante**

   Estudiar en la universidad es duro y el controlar el nivel de estrés es importante para mantener un buen nivel académico. Existen tests psicométricos que permiten evaluar diferentes problemas que pueden aparecer. Vamos a analizar los datos de estos tests recogidos a partir de una muestra de estudiantes universitarios[6]. Estudiaremos la relación entre las preguntas del cuestionario y otras variables demográficas y académicas en el nivel de estrés de los estudiantes.

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

[5] Podéis consultar su documentación en https://archive.ics.uci.edu/dataset/544/estimation+of+obesity+levels+based+on+eating+habits+and+physical+conditiont.

[6] Puedes acceder a la descripción de los datos originales en https://www.kaggle.com/datasets/mohsenzergani/bangladeshi-university-students-mental-health/data.

a) Divide los datos en entrenamiento y test (70 %/30 %), la partición debe estratificarse (fija también el estado del generador de números aleatorios para la reproducibilidad). Tienes un conjunto de variables que son discretas. Las del cuestionario son binarias, el resto son categóricas. Generaremos dos conjuntos de datos, uno en el que usemos la codificación numérica de las variables categóricas y otro en el que binarizemos las variables. Haz una visualización mínima de las variables para hacerte una idea de los datos y calcula sus correlaciones. Aplica un PCA y un t-SNE y evalua la separación de los datos en clases según lo que se observa en la representación en dos dimensiones.

b) Empezaremos por Naive Bayes y el discriminante lineal. Aplica el modelo a los dos conjuntos de datos. Elige adecuadamente los modelos y evalúa su calidad.

c) Ajusta una regresión logística usando regularización `L1` explorando adecuadamente sus hiperparámetros a los dos conjuntos de datos. Evalúa adecuadamente la calidad de los modelos y compáralos entre sí. Una ventaja que tiene el usar un modelo interpretable como la regresión logística es que podemos saber la importancia de los atributos. Mira los pesos del modelo y determina que es lo más importante para el diagnóstico en cada clase. ¿Hay variables que tienen más peso que otras en la clasificación?

d) Parece que puede haber variables que se están utilizando en el conjunto de datos son redundantes. Podemos reducir los atributos necesarios explorando la relevancia de los atributos. Un algoritmo eficiente de reducción de atributos es la *eliminación recursiva de caracterísiticas* (Recursive Feature Elimination). Tienes una implementación en `scikit-learn` que utiliza validación cruzada para la evaluación (`RFECV`). Vamos a usarlo con el conjunto de datos representados como categóricos.

Para aplicarlo tendrás que pasarle un modelo ajustado de regresión logística con los mejores parámetros que has encontrado en el apartado anterior para este conjunto de datos. Obtén con este método un conjunto de atributos reducido y vuelve a ajustar sus hiperparámetros con estos datos. Compara los resultados con los modelos anteriores. Fíjate en los pesos que has obtenido con el conjunto de datos simplificado ¿Podrías decir a partir de los pesos del modelo como se está evaluando el nivel de estrés en el estudio?