Sistema Experto

Un Sistema Experto (SE) es un programa informático que tiene como solucionar problemas concretos utilizando la Inteligencia Artificial (IA). De esta forma, se busca que el SE se comporte como un profesional en la materia.

En este caso, el SE debe ser capaz de detectar, prevenir y monitorizar conductas autolesivas en adolescentes.

Hay 4 grupos de SEs:

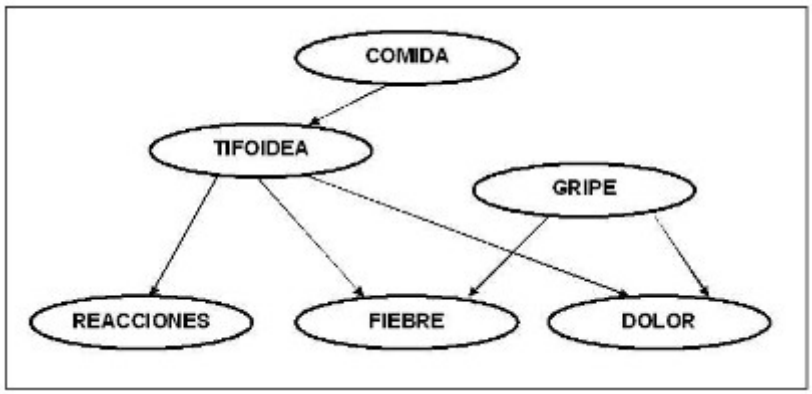
- SE basados en reglas
- SE basados en casos
- **SE basados en redes Bayesianas**
- SE basados en lógica difusa o borrosa

Red bayesiana

Una red de Bayes o red bayesiana es un modelo gráfico que se muestran variables aleatorias (**nodos**) en un conjunto de datos y las dependencias que hay entre ellas (**arcos**), formando de esta manera, un grafo acíclico dirigido (DAG).

Cabe destacar que no todas las relaciones son directas entre las variables, sino que también pueden haber relaciones indirectas independientes, es decir, variables que están relacionados con otras a través de otros nodos.

Por ejemplo:



Este grafo muestra los síntomas, causas y factores que pueden venir de una enfermedad.

reacciones es condicional independiente de C, G, F, D dado *tifoidea* (donde: C es comida, T es tifoidea, G es gripe, R es reacciones, F es fiebre y D es Dolor). Esto es:

$$P(R|C, T, G, F, D) = P(R|T)$$

Esto se representa gráficamente por el nodo T separando al nodo R del resto de las variables.

Por ejemplo, una red bayesiana puede representar las relaciones probabilísticas entre enfermedades y síntomas. Dados los síntomas, la red puede ser usada para computar la probabilidad de la presencia de varias enfermedades.

Una red bayesiana es un modelo gráfico que muestra variables (que se suelen denominar **nodos**) en un conjunto de datos y las independencias probabilísticas o condicionales entre ellas. Las relaciones causales entre los nodos se pueden representar por una red bayesiana; sin embargo, los enlaces en la red (también denominados **arcos**) no representan necesariamente una relación directa de causa y efecto. Por ejemplo, una red bayesiana se puede utilizar para calcular la probabilidad de un paciente con una enfermedad concreta, con la presencia o no de algunos síntomas y otros datos relevantes, si las independencias probabilísticas entre síntomas y enfermedad son verdaderas, tal y como se muestra en el gráfico. Las redes son muy robustas en los puntos en los que falta información y realizan las mejores predicciones posibles utilizando la información disponible.

Lauritzen y Spiegelhalter crearon un ejemplo común y básico de una red bayesiana en 1988. También se conoce como modelo "Asia" y es una versión simplificada de una red que se puede utilizar para diagnosticar a los nuevos pacientes de un médico; la dirección de los enlaces corresponde por lo general a la causalidad. Cada nodo representa una faceta que se puede relacionar con el estado de un paciente; por ejemplo, "fumador" indica que se trata de un fumador habitual y "VisitaAsia" muestra que recientemente ha viajado a Asia. Los enlaces entre los nodos muestran las relaciones probabilísticas; por ejemplo, fumar aumenta las posibilidades de que el paciente padezca bronquitis y cáncer de pulmón, mientras que la edad parece estar relacionada únicamente con la posibilidad de desarrollar cáncer de pulmón. De la misma forma, las anomalías detectadas en una radiografía de los pulmones pueden estar causadas por tuberculosis o cáncer de pulmón, mientras que las posibilidades de que un paciente tenga dificultades respiratorias (disnea) aumentan si también padece bronquitis o cáncer de pulmón.

Desarrollo del SE en Python

Scikit-learn y Pandas

De acuerdo a la documentación enviada, necesitamos que el sistema experto reciba, como input, un conjunto de puntuaciones procedentes de los formularios.

Se debe tener en cuenta que esta propuesta se realizará sin conocer todavía la estructura ni el contenido exacto de la base de datos, ni de la estructura exacta de los datos que se quiere enviar al sistema experto. Por lo que considerenlo como una propuesta inicial.

La idea del sistema experto será recibir un conjunto de puntuaciones y ser capaz de predecir síntomas autolesivos en una persona.

Para ello, el sistema experto creará un dataframe en base a los datos recibidos del exterior. Esto se realizará con la librería Pandas.

Una vez almacenado los datos en un dataframe, se procederá a dividir el conjunto entre los datos que se utilizarán de entrenamiento (TRAIN) y los que serán para testear (TEST). En este caso, se propone hacer una división 75% (TRAIN) – 25% (TEST).

Posteriormente, se usará la librería **SKLearn** de Python para generar una matriz de confusión. Para esto, se propone utilizar inicialmente el **clasificador Naive-Bayes** que ya está integrado dentro de propia librería. De todas formas, después de realizar la clasificación, se puede comprobar si el clasificador que hemos escogido es bueno o no calculando su **p-valor**. Si el p-valor es pequeño, quiere decir que la exactitud (ACC) del clasificador es significativamente alta, por lo que el clasificador es válido.

Para calcular el p-valor del clasificador para este problema, se tendrá que generar nuevos conjuntos de datos "aleatorios" a partir del conjunto general que se haya recibido. Estos serán las permutaciones. Se creará un script de Python que valide el clasificador mediante **bootstraping**.

El bootstraping es un técnica que permite estimar el ratio de error de un modelo mediante la generación N muestras equiprobables con reemplazamiento del conjunto original.

La validación se hará de la siguiente manera:

1) Realizamos un entrenamiento inicial al modelo para calcular la exactitud inicial (ACC_ini).
2) Establecemos un número de veces que se va a repetir el proceso de bootstraping, N.
3) Establecemos el número de permutaciones (muestras) que se van a recoger en cada repetición, M. En este caso, se generarán las muestras a partir del conjunto de entrenamiento, por lo tanto:

   M = Tamaño del conjunto de datos de entrenamiento.

4) Iniciamos el proceso de revisión del modelo. Comenzamos con el bucle. Generamos las M permutaciones aleatorias en cada iteración y se calcula la exactitud de esa iteración (ACC-tmp). Si esta exactitud es mejor al inicial (ACC_tmp > ACC_ini), entonces se aumenta un contador que lleva la cantidad de veces que ACC_tmp > ACC_ini.
5) Al final del proceso, calculamos el p-valor con esta fórmula

$$p\_valor = \frac{Cont + 1}{M + 1}$$

   Donde

   - *Cont* es el número de veces donde ACC_tmp > ACC_ini
   - *M* es el número de muestras.

Generación de la matriz de confusión

Para generar la matriz, habrá que utilizar la librería Pandas para guardar todos los datos de la base de datos que nos proporcionen en un dataframe, y SKLearn, para realizar la matriz de confusión, y calcular la precisión de la clasificación.

Es necesario establecer las clases que va a tener el modelo. En este caso, se propone utilizar como clases los tipos de comportamientos suicidas, de acuerdo a la documentación entregada:

1. Ideación
2. Planificación
3. Gestos
4. Intento

De tal forma, sería una matriz 4x4 que sería de la siguiente manera:

		Predicción			
		Ideación	Planificación	Gestos	Intento
Resultado real	Ideación	Correcto	Incorrecto	Incorrecto	Incorrecto
	Planificación	Incorrecto	Correcto	Incorrecto	Incorrecto
	Gestos	Incorrecto	Incorrecto	Correcto	Incorrecto
	Intento	Incorrecto	Incorrecto	Incorrecto	Correcto

Como no es una clasificación binaria, es decir, una clasificación de Verdadero/Falso (matriz 2x2), entonces no existe los términos Verdaderos/Falsos Positivos o Negativos como tal.

En este caso, se debería interpretar la matriz de la siguiente manera.

Imaginemos que tenemos el siguiente ejemplo de matriz:

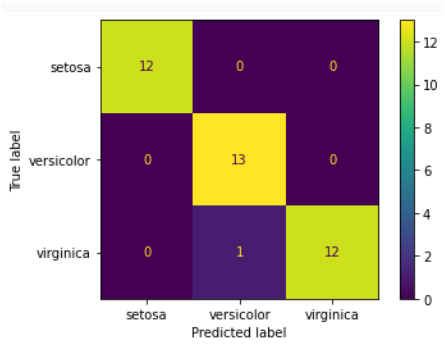
		Predicción			
		Ideación	Planificación	Gestos	Intento
Resultado real	Ideación	100	60	50	40
	Planificación	30	90	25	45
	Gestos	20	15	80	55
	Intento	10	5	35	70

Los valores de la diagonal son las predicciones correctas. En el caso de la fila de *Ideación*, podemos concluir viendo la tabla, que el modelo ha predicho correctamente 100 personas que tenían ideación de conductas autolesivas, mientras que ha predicho incorrectamente a 60 personas con ideación,

concluyendo que tenían planeado realizar conductas autolesivas; 50 con gestos y otras 40 con intento. Lo mismo se aplica para el resto de filas.

Mientras que mirando las columnas, se puede concluir que de las 160 personas (suma de todos los valores 100 + 30 + 20 + 10) predichas por el modelo que tenían ideación de conductas autolesivas, 100 tenían realmente esas intenciones de ideación, 30 realmente eran de Planificación, 20 de Gestos y 10 de Intento. Lo mismo se aplica para el resto de columnas.

Como en el siguiente ejemplo de los datos de un color de iris:



Con la matriz creada, se puede calcular la precisión de la clasificación realizada, calculando el error (ERR) y la exactitud (ACC):

ERR = (FP + FN)/(VP + FN + FP + VN)

ACC = (VP + VN)/(VP + FN + FP + VN) = 1 - ERR

Además, también se va a calcular la precisión (PRE), que es el ratio de verdaderos positivos frente al número total de positivos de la predicción.

PRE = (VP) / (VP + FP)

En un conjunto de datos de entrenamiento donde **las clases están equilibradas** son suficientes los anteriores ratios y también son utiles los ratios:

- **Sensibilidad - SEN, TPR** *(True Positive Rate o Razón de Verdaderos Positivos)* : Es la probabilidad de que un positivo sea realmente positivo o capacidad del estimador para dar casos positivos (“*enfermos*”). Es la tasa de verdaderos positivos frente a positivos. Este parámetro también se denomina recall o exhaustividad.
- **Especificidad - SPC, TNR** *(True Negative Rate o Razón de Verdaderos Negativos)* : Es la probabilidad de que un negativo sea realmente negativo o capacidad del estimador de dar casos negativos (“*sanos*”). Tasa de verdaderos negativos frente a negativos.
- **Razón de Falsas Alarmas - FPR** *(False Posive Rate o Ratio de Falsos Positivos)* : Es la tasa de falsos positivos entre los positivos reales. Es igual **1 - Especificidad**:

$$SEN = \frac{VP}{P} = \frac{VP}{VP + FN}$$

$$SPC = \frac{VN}{N} = \frac{VN}{VN + FP}$$

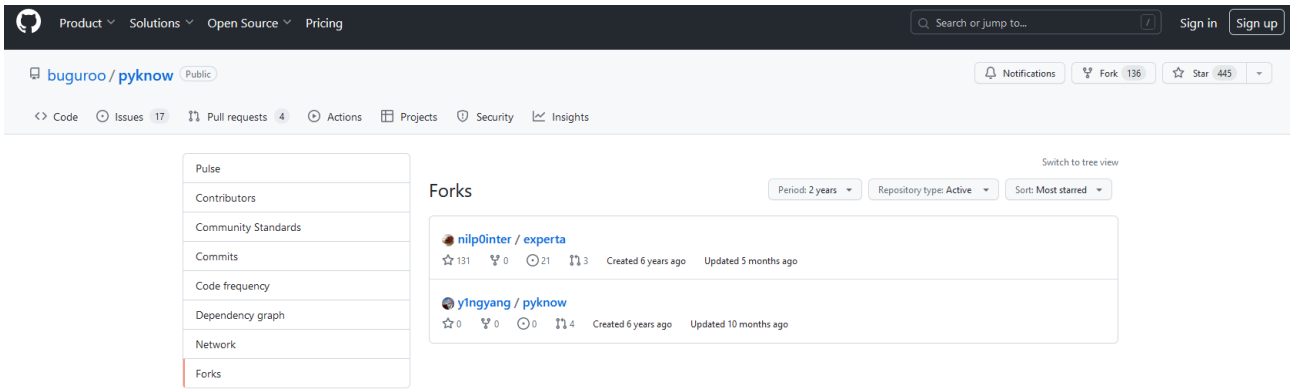
$$FPR = \frac{FP}{N} = \frac{FP}{VN + FP} = 1 - SPC$$

Experta (aún por decidir si se usa o no)

Para programarlo, vamos a usar una librería de Python llamado Experta, que es una librería basada en la librería PyKnow (de hecho, *experta* es una fork del repositorio de PyKnow).

NOTA: Todavía no estoy seguro de si emplear esta librería para desarrollar el SE.

Repositorio de PyKnow: https://github.com/buguroo/pyknow



Forks del repositorio de PyKnow

A su vez, PyKnow está inspirada en CLIPS, un lenguaje de programación basada en C para sistemas expertos (Repositorio de Github de Experta: https://github.com/nilp0inter/experta/tree/develop).

Básicamente, *experta* es una librería que ofrece una alternativa de CLIPS de C en Python. *Experta* es compatible con Python3.

Documentación de Experta:

- https://github.com/nilp0inter/experta/blob/develop/docs/talks/Sistemas%20Expertos%20en%20Python%20con%20PyKnow%20-%20PyConES%202017/slides.pdf
- https://experta.readthedocs.io/en/latest/
- https://readthedocs.org/projects/experta/downloads/pdf/stable/

Bibliografía:

- https://www.ibm.com/docs/es/spss-modeler/saas?topic=models-bayesian-network-node
- https://es.wikipedia.org/wiki/Red_bayesiana
- https://ccc.inaoep.mx/~esucar/Clases-mgp/caprb.pdf

- https://www.unir.net/ingenieria/revista/sistema-experto/

Librerías que descargar para ejecutar el SE:

Todos estos comandos deberían estar en un fichero que se pueda ejecutar previamente llamado *requirements*

- *Experta*: $ pip install experta