Patrón de fondo

Descripción generada automáticamente**MINERÍA PRÁCTICA 01**

Índice

PABLO SIMÓN SAINZ

IVÁN RUIZ GÁZQUEZ

[Conjunto de datos 3](#_Toc100180743)

[Primera Fase: Adaptación para distancias 3](#_Toc100180744)

[Segunda Fase: Normalización y ajuste de población 3](#_Toc100180745)

[Correlación 5](#_Toc100180746)

[Experimentos 6](#_Toc100180747)

[KNIME model 6](#_Toc100180748)

[Python model 6](#_Toc100180749)

[K neighbours comparación 7](#_Toc100180750)

[Diferencia entre funciones de distancia 9](#_Toc100180751)

[Conclusión 11](#_Toc100180752)

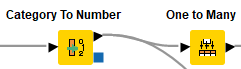
Introducción

En esta segunda práctica haremos un estudio del modelo KNN (k nearest neighbours).

El dataset sobre el que trabajaremos recoge varios datos médicos, y los emplearemos para hacer una predicción sobre si el paciente padece una enfermedad cardiovascular.

# Conjunto de datos

Dado que el algoritmo se basa su entrenamiento en las distancias sobre el campo de búsqueda, es importante modificar los datos para poder hacer este cálculo correctamente.



## Primera Fase: Adaptación para distancias

En este caso, hemos identificado tres escenarios en la primera fase de procesamiento de datos:

* **Valores numéricos**, para estos no tendremos que realizar cambios de primera mano.
* **Valores categóricos**:
  + **Con distancias internas**, en este caso nos encontramos valores como ‘AgeCategory’, que nos encontramos con grupos de edad los cuales cada categoría tiene una distancia diferente a otra (un rango de edad pequeño estará más cerca de uno mediano que de uno grande).

En este dataset los de este tipo son solamente lineales, por lo que bastará con usar el nodo ‘Category To Number’, para dar valores del 0 a n.



Un problema que nos hemos encontrado es que hemos tenido que gestionar a parte la asignación de números debido a que KNIME asigna el número en función del orden en el que se va encontrando una categoría nueva.

* + **Sin distancias internas**, valores como la Raza o los de “YES”/”NO”, en un principio no tienen variaciones de distancia entre ellas, por lo que aplicaremos ‘One shoot’ mediante el nodo ‘One to Many’.



## Segunda Fase: Normalización y ajuste de población

En este caso también aplicaremos normalización a los datos como hicimos en la práctica anterior.

En cuanto a la población, nos encontramos con que existen más pacientes sin problemas del corazón que con, por lo que esta está un tanto desbalanceada.

Esto provocará una mejor adaptación en una clase que en otra, por lo que nos interesa o bien duplicar elementos de la minoritaria, o bien eliminar elementos de la mayoritaria.

### Acc sin SMOTE

#### KNIME

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

#### Python

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

El problema del oversampling es que, aunque nos aumente el accuracy, no es un accuracy real, ya que estamos duplicando los datos de entrada (test y entrenamiento). Si incluyéramos un nuevo dataset de test, este tendría menor accuracy.

### Ejemplo de oversampling X2

Un ejemplo de oversampling de x2 nos otorga el siguiente accuracy con KNIME:

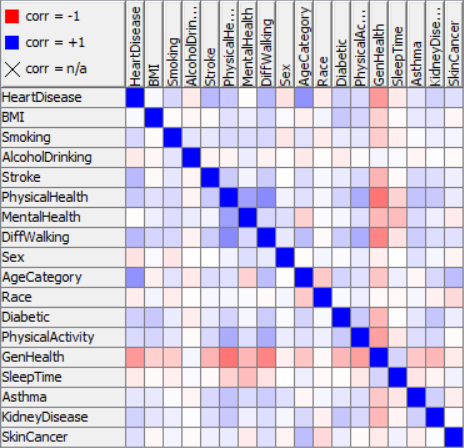
Interfaz de usuario gráfica, Tabla

Descripción generada automáticamente

El accuracy es de 98,77. Pero es un accuracy que se adapta mucho a los datos insertados, y que no se adaptaría correctamente a nuevos datos.

## Correlación

En comparación con la práctica anterior, encontramos correlaciones relativamente bajas, por lo que obtendremos la clasificación mediante varios parámetros con un coeficiente medio-bajo, con un rango de [0.012, 0.431].

Tabla

Descripción generada automáticamente

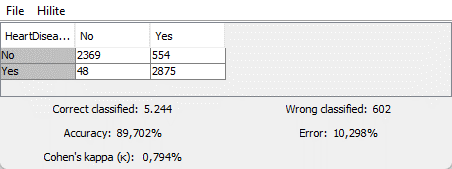
En este caso hemos obtenido peores resultados al realizar el filtro, por lo que en este caso no lo aplicaremos para los experimentos.

#### KNIME

Interfaz de usuario gráfica, Aplicación, Tabla

Descripción generada automáticamente

#### PYTHON



# Experimentos

A continuación, haremos comparaciones a distintas variaciones en el modelo para poder analizar las diferencias y ver cuál es el método más adecuado y por qué.

## KNIME model

Como modelo pivote utilizaremos el creado mediante funciones de KNIME, ya que nos ha resultado más sencillo de manipular.

Interfaz de usuario gráfica, Tabla

Descripción generada automáticamente

## Python model

Comparando este modelo respecto a KNIME, tenemos un resultado muy parecido, tanto en accuracy como en kappa. Solo que al parecer uno clasifica mejor los de una clase que el otro.

Interfaz de usuario gráfica, Tabla

Descripción generada automáticamente

## K neighbours comparación

### 5 vecinos

#### KNIME

Interfaz de usuario gráfica, Aplicación, Tabla

Descripción generada automáticamente

#### PYTHON

Interfaz de usuario gráfica, Tabla

Descripción generada automáticamente

### 7 vecinos

#### KNIME

Tabla

Descripción generada automáticamente

#### PYTHON

Tabla

Descripción generada automáticamente

## Diferencia entre funciones de distancia

### KNIME

#### Distancia euclídea:

Tabla

Descripción generada automáticamente

#### Distancia Manhattan:

Interfaz de usuario gráfica, Tabla

Descripción generada automáticamente

### PYTHON

#### Distancia euclídea:

Interfaz de usuario gráfica, Tabla

Descripción generada automáticamente

#### Distancia Manhattan:

Interfaz de usuario gráfica, Tabla

Descripción generada automáticamente

# Conclusión

Obviando las conclusiones de la práctica anterior.

Respecto al tratamiento de datos, se ha añadido dificultad con los datos nominales y sus distintos tratamientos para el cálculo de distancias. Y en cuanto a la cantidad de población hemos comprobado que para que no pierda peso y accuracy para la clase minoritaria, debemos ajustar esta cantidad para equilibrar e incluso mejorar el rendimiento general.

Por la otra parte, en el ajuste del modelo, hemos visto que en el KNN en este caso funciona mejor con distancia manhattan, lo más probable que sea debido a que da mayor peso a las distancias de los valores nominales.

Como dato interesante, hemos notado a la herramienta KNIME le toma más tiempo realizar operaciones con nodos de Python. Esto puede ser debido al paso de lenguajes.