ntic master School

UNIVERSIDAD COMPLUTENSE DE MADRID

Data Understanding & Preparation

Minería de Datos y Modelización Predictiva

Máster en Big Data, Data Science & Inteligencia Artificial

Universidad Complutense de Madrid

Las fases de "Data understanding & preparation" de la metodología CRISP-DM comprenden lo que se conoce como *Depuración de datos*.

Depuración de datos

La depuración de datos es el proceso mediante el cuál se **adecúa** el conjunto de datos para la fase de modelización posterior. Consiste en "limpiar" el conjunto de datos original de manera que no haya observaciones **incoherentes**, las variables se ajusten a las **especificaciones** del modelo, etc.

Fases de la depuración de datos

- Comprobación de la correcta tipología y rol de las variables.
- Análisis exploratorio de los datos y corrección de errores detectados.
- Búsqueda de datos atípicos.
- Tratamiento de datos faltantes.
- Relaciones entre variables.
- Transformación de variables.

El primer paso que se ha de llevar a cabo es la **exploración** de los datos, lo que nos indicará si hay algún problema con los mismos, para lo que es imprescindible poseer cierto **conocimiento** sobre el significado de los datos. Lo realizaremos a través de gráficos y tablas.

Aspectos a evaluar

- Comprobación de la correcta tipología de las variables. Si las variables cualitativas están representadas con caracteres numéricos, Python suele considerarlas como numéricas. Además, es recomendable tratar como variables cualitativas aquellas variables numéricas con menos de 10 valores diferentes.
  - `DataFrame.dtypes`
  - `cuentaDistintos(dataFrame)`
- Análisis del número de datos faltantes (missings).
  - `DataFrame['variable'].isna().sum()`
- Comprobación de las categorías de las variables cualitativas.
  - `DataFrame['variable'].unique()`
- Comprobación de los límites de las variables cuantitativas.
  - `DataFrame.describe()`

Aspectos a evaluar

- Análisis de la coexistencia de datos faltantes en varias variables.
  - `patron_perdidos(DataFrame)`
- Tabla de frecuencias para variables cualitativas.
  - `analizar_variables_categoricas(DataFrame)`

Corrección de los errores detectados

- Variables cualitativas mal consideradas como cuantitativas.
  - `datos['VarCat'] = datos['VarCat'].astype(str)`
- Datos faltantes codificados ("-1", "9999") no declarados.
  - `datos['Var'] = datos['Var'].replace(9999, np.nan)`
- Valores de variables cuantitativas fuera de rango.
  - ```
    datos['VarCuant'] = [x if LimInf <= x <= LimSup
     else np.nan for x in datos['VarCuant']]
    ```
    ```

Corrección de los errores detectados

- Categorías de variables cualitativas erróneas o con poca representatividad.
  - `datos['VarCual'] = datos['VarCual'].replace(`
    `{'oldValue1': 'NewValue1','oldValue2': 'NewValue2'})`
- Unión de categorías de variables cualitativas con poca representatividad.
  - `datos['VarCat'] = datos['VarCat'].map(`
  - `{'oldValue1': 'NewValue1','oldValue2': 'NewValue2'})`

Datos atípicos o outliers

Un dato atípico es una observación que es **numéricamente distante** del resto de los datos.

El problema de los datos atípicos es que tienen una **gran influencia** en los resultados, si los modelos no son **robustos**.

Algunos métodos de detección de outliers son:

- **Desviación típica**: Se considerán atípicos aquellos datos que disten más de un número k (habitualmente entre 3 y 6) de desviaciones típicas de la media. Este método sólo es válido si las distribuciones son aproximadamente **simétricas** (coeficiente de asimetría entre -1 y 1).
- **MAD**: Se considerán atípicos aquellos datos que disten más de un número k (habitualmente entre 8 y 15) de MADs de la mediana. *Median Absolute Deviation* (MAD) es la mediana de las distancias absolutas a la mediana. Este método es más adecuado para distribuciones **asimétricas**, pero no es válido cuando la mediana es igual a 0.
- **Rango intercuartílico**: Se consideran atípicas aquellas observaciones que se alejen más de $\mathbf{1.5}$ o $\mathbf{3}$ veces el rango intercuartílico del primer y el tercer **cuartil**. Este método está asociado a los gráficos de cajas.

Es importante destacar que, por definición, el número de datos atípicos ha de ser pequeño pues, de lo contrario, no se podrán considerar atípicos.

Dado que ninguno de los métodos de detección de atípicos es *perfecto*, lo recomendable es **utilizar varios** de ellos y sólo considerar como atípicas aquellas observaciones que sean consideradas como tal por **más de un método**.

Para conocer la **proporción de datos atípicos** encontrados en cada variable usamos el siguiente código:

```
numAtipicos = {x: atipicosAmissing(datos[x])[1] / len(datos)
               for x in VarNumericas}
```

Una vez detectados los datos atípicos, los ponemos como **ausentes** usando el código:

```
for x in VarNumericas:
    datos[x] = atipicosAmissing(datos[x])[0]
```

La presencia de datos faltantes en los conjuntos de datos ha de ser **analizada** pues da lugar a una reducción del número de **observaciones válidas** para los procedimientos estadísticos.

Adicionalmente, si la presencia de missings no es **aleatoria,** esta puede venir acompañada de **sesgo** en las respuestas y, por tanto, en los modelos (por ejemplo, encuestados que se nieguen a responder preguntas delicadas debido a su respuesta).

Los paquetes estadísticos más frecuentes **ignoran** la información procedente de observaciones con algún dato faltante (case deletion), por lo que la información que contienen **no se tiene en cuenta** en el modelo.

No obstante, algunas técnicas estadísticas (como los árboles o las técnicas basadas en árboles) modelizan los missings como **una categoría más**, incluyendo así sus posibles efectos predictivos y **reduciendo** considerablemente el sesgo.

Por tanto, habrá que **analizar detalladamente** los datos faltantes y aplicar una o más de las estrategias siguientes:

- **Eliminación**: tanto de variables como de observaciones.
- **Recategorización**: de los valores missing como una categoría válida. Para variables contínuas, esto implica una discretización de la misma.
- **Imputación**: es el proceso que consiste en sustituir los missing por valores válidos.

Eliminación

La eliminación de variables y/o observaciones sólo debe llevarse a cabo cuando la proporción de missings sea muy **elevada** (superior al 50 %) y, por tanto, la **pérdida de información** no lo sea tanto.

Por ese motivo, primero se lleva a cabo es un análisis de la proporción de missings:

- Por variables: Para conocer la proporción de datos perdidos, usamos el siguiente código:
  `prop_missingsVars = datos.isna().sum()/len(datos)`
- Por observación: En ocasiones existen **observaciones** con un gran número de missings (por ejemplo, por pereza del encuestado) que aportan **poca información** y que, por tanto, pueden ser eliminados.
  Para saber el número de **missings por observación**, debemos crear una variable que los cuente para, a continuación, utilizar dicha información para eliminar las observaciones "conflictivas":
  `datos['prop_missings'] = datos.isna().mean(axis = 1)`

Eliminación

El nuevo conjunto de datos se obtiene a partir de:

- Elimina las variables que tienen más del 50 % de las observaciones perdidas:

```
eliminar = [prop_missingsVars.index[x] for x in
        range(len(prop_missingsVars)) if prop_missingsVars[x] > 0.5]
datos_nuevo = datos[~eliminar]
```

- Elimina las observaciones con más del 50 % de variables con información perdida:

```
eliminar = datos_nuevo['prop_missings'] > 0.5
datos_nuevo = datos_input[~eliminar]
```

Recategorización

Cuando la presencia de missing sea **importante**, puede ser interesante definir una **nueva categoría** "Missing" de la variable (sobretodo para variables categóricas):

```
datos['varCual'] = datos['varCual'].fillna('Desconocido')
```

Para las variables de intervalo, habría que **discretizarlas** antes y crear una categoría "Missing" para la nueva variable. Hay que tener cuidado con esta alternativa pues no siempre es buena idea categorizar variables numéricas.

Una **alternativa** a caballo entre la recategorización y la eliminación consiste en **imputar** los valores faltantes y **crear una variable** nueva que cuente el número de variables imputadas para cada observación (alternativamente, se puede crear una por variable imputada). De esta forma, se puede trabajar con **todas** las observaciones (pues ya son "válidas") y se mantiene la información asociada a la **falta de respuesta** (es una opción especialmente interesante para variables de intervalo con una proporción de missings elevada). La variable "prop_missings" ya creada puede servir para esta función.

Imputación

La imputación consiste en sustituir los datos faltantes por **valores válidos**.

Existen varios tipos de imputación:

- Imputación simple:
  a) Se sustituyen los missings por algún **estadístico de localización**, como la media o la mediana para variables de intervalo, o la moda para variables categóricas.
  b) Se imputan los datos faltantes **aleatoriamente** teniendo en cuenta la **distribución** de la variable.
- Imputación por modelos: se sustituyen los missings por una predicción basada en **otras variables** del conjunto de datos.

Existen algunos inconvenientes a la hora de aplicar la imputación por modelos, como el sobreajuste, por lo que nos centraremos en la imputación simple:
`ImputacionCuant(varCuant, 'tipo')`, donde tipo toma los valores: "media", "mediana" y "aleatorio".
`ImputacionCuali(varCual, 'tipo')`, donde tipo toma los valores: "moda" y "aleatorio".

Relaciones bivariables

Una de las claves para la creación de un buen **modelo predictivo**, además de la calidad de los datos, son las variables input que se usan en la predicción.

Como ya hemos comentado previamente, en **minería de datos** generalmente se trabaja con un gran número de variables. Por ello, en ocasiones se hace necesario realizar una **preselección** de las que serán útiles en la fase de modelización.

Así mismo, es posible que la relación existente entre las variables input y la objetivo no sea lineal y sea recomendable aplicar algún tipo de **transformación** a las primeras, de manera que los modelos de predicción puedan recoger de una manera más eficaz la relación entre ellas.

Para poder determinar qué variables van a ser útiles para la predicción, así como las mejores transformaciones, debemos definir una serie de medidas que permitan cuantificar la información que poseen las variables input sobre la variable objetivo, es decir, debemos evaluar si las variables están relacionadas.

Así mismo, siempre es recomendable llevar a cabo un **análisis gráfico bivariable** que nos permita detectar dichas relaciones.

Análisis gráfico de relaciones entre variables

Algunos gráficos útiles para determinar visualmente la presencia de relación entre las variables son los que siguen:

- Histograma: `hist_targetbinaria(var, target, nombre_eje)`
- Diagrama de Cajas y Bigotes: `boxplot_targetbinaria(var, target, nombre_eje)`
- Diagrama de Mosaico: `mosaico_targetbinaria(var, target, nombre_eje)`

Cuantificación de la relación entre variables

Las siguientes medidas permiten observar el valor de dos medidas habitualmente utilizadas para la medición de relaciones entre variables: coeficiente de correlación (que evalúa la relación lineal entre dos variables cuantitativas) y el estadístico V de Cramer (evalúa la relación no lineal entre cualesquiera dos variables).

- Correlaciones:
```
matriz_corr = pd.concat([varObjCont,
                         datos_input[numericas]],
                         axis= 1).corr(method='pearson')
```
- V de Cramer: `Vcramer(datos['Variable'], VarObj)`

A parte de los gráficos que ya se han generado, una buena forma de evaluar el poder predictivo de las variables input es a través del estadístico "V de Cramer". Este estadístico se calcula a partir del estadístico χ^2 y tiene la ventaja de que es capaz de capturar las relaciones no lineales existentes entre las variables.

Estadístico χ^2

El estadístico χ^2 es un método muy común para detectar **relaciones** entre dos variables, siendo especialmente útil para relaciones **no lineales**.

Los datos de entrada requeridos para calcular este estadístico consisten en una **tabla de contingencia**, por lo que las variables de intervalo han de ser **discretizadas** previamente.

Sean C_i y D_j las clases en las que están divididas las variables de interes, entonces:

	C_1	C_2	$\dots$	C_l	Total
D_1	n_{11}	n_{12}	$\dots$	n_{1l}	$n_{1.}$
D_2	n_{21}	n_{22}	$\dots$	n_{2l}	$n_{2.}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
D_k	n_{k1}	n_{k2}	$\dots$	n_{kl}	$n_{k.}$
Total	$n_{.1}$	$n_{.2}$	$\dots$	$n_{.l}$	n

$$\chi^2 = \sum_{i=1}^{k}\sum_{j=1}^{l} \frac{\left(n_{ij} - \frac{n_{i.}n_{.j}}{n}\right)^2}{\frac{n_{i.}n_{.j}}{n}}$$

Estadístico χ^2

El estadístico χ^2 toma valor 0 cuando las variables son **independientes** (no hay relación entre ellas). Matemáticamente esto equivale a decir que las frecuencias relativas de los cruces de categorías **sólo dependen** de las frecuencias relativas de las categorías:

$$f_{ij} = f_{i.} f_{.j}, \text{ donde } f_{ij} = \frac{n_{ij}}{n} \quad f_{i.} = \frac{n_{i.}}{n} \quad f_{.j} = \frac{n_{.j}}{n}, \text{ es decir, } n_{ij} = \frac{n_{i.} n_{.j}}{n}$$

Por lo tanto, las variables que estén fuertemente relacionadas tendrán un estadístico χ^2 **alto**. Sin embargo, este estadístico no está limitado por lo que resulta complicado determinar cuando está tomando un valor significativamente alto.

Estadístico V de Cramer

El estadístico V de Cramer está basado en el **estadístico** χ^2 pero tiene la ventaja de que su valor está **acotado** entre 0 y 1. Viene dado por:

$$V = \sqrt{\frac{\chi^2}{n \times \min(l-1, k-1)}}$$

Para obtener y representar la V de Cramer para todas las variables input, usamos la función:

`graficoVcramer(dataFrame,varObj)`

Crear una variable aleatoria permite tener una referencia de la utilidad de las variables.

En ocasiones es necesario realizar alguna **transformación** en las variables para que el modelo de **predicción** funcione mejor o se pueda plasmar la verdadera **relación** con la variable objetivo.

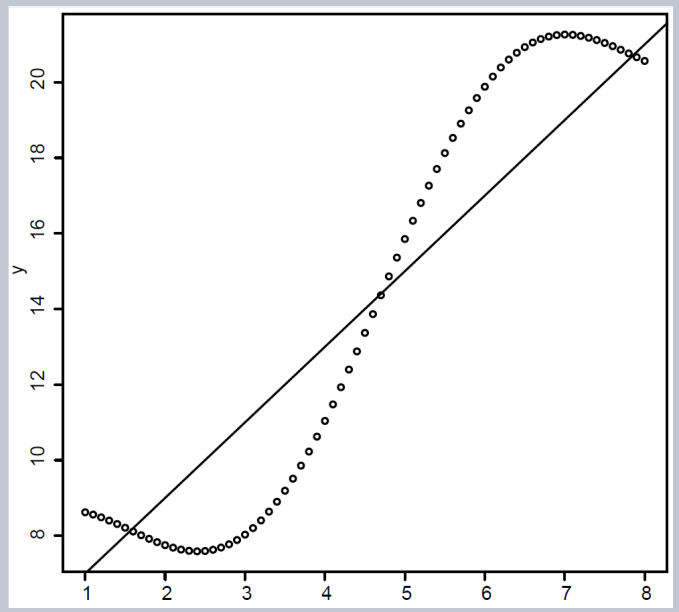
Discretización de variables continuas

La **discretización** (en inglés, binning) permite descubrir **relaciones complejas** (no lineales) entre las variables de entrada y la variable objetivo.

Consiste en obtener una variable categórica a partir de la **división** de una variable continua.

Se persigue obtener tramos lo más **heterogéneos** posible con respecto a la variable objetivo. De esa forma, pertenecer a uno u otro grupo ofrecerá información acerca de la variable a predecir.

Se puede obtener una variable en k partes iguales con la función `pd.cut(varCuant, k)`.



Mejorar la relación con la variable objetivo

En ocasiones es preferible *linealizar* la relación entre la variable objetivo y las input cuantitativas para que el modelo aproveche mejor el poder predictivo de éstas. Para ello, podemos transformar dichas variables eligiendo entre varias transformaciones aquella que maximice dicha relación, ya sea en términos del coeficiente de correlación (si la variable objetivo es cuantitativa) o de la V de Cramer (si la variable objetivo es cualitativa).

Las funciones `mejorTransfCorr(varCuant, varObjCont)` y `mejorTransfVcramer(varCuant,varObjBin)` permiten seleccionar entre las siguientes transformaciones: $X, log(X), e^X, X^2, \sqrt{X}, X^4, \sqrt[4]{X}$.

La función `Transf_Auto(dataFrame, varObj)` permite automatizar la transformación de todas las variables cuantitativas del *dataFrame*.

Una vez obtenidas las transformaciones, se puede elegir entre mantener las variables originales o rechazarlas.