Preprocesado de Datos

Juan Carlos Pérez González - IES de Teis

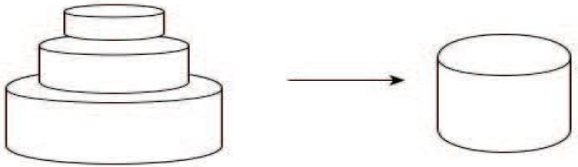
Preprocesamiento

- Después de la carga de datos hay que realizar la búsqueda de información necesaria y de **valor** que muchas veces se basa en la **reducción de la "masa" de datos,** previo a la **minería de datos**
- Necesitamos *preprocesar* dichos datos según su destino y he aquí algunas tareas:
  - En **machine learning** necesita ***datos categóricos(altura, sexo, fecha nacimiento, color de pelo…)***, los algoritmos trabajan con números
  - Analizar los registros incompletos para su transformación o eliminación
  - Revisar la calidad de la calidad de los datos (no todo vale)
  - Eliminar la información redundante e innecesaria, así como **registros repetidos**, es lo que se conoce como **limpieza de ruido**
  - Detectar y corregir **registros corruptos**, y si no su eliminación
  - En fechas y tiempos, extraer características relevantes o crear nuevas a partir de ellas.
  - Determinar la posible existencia de correlación entre columnas (**Análisis Exploratorio de Datos**)
  - Estandarizar el nombre de columnas de acuerdo a su contenido
  - **Normalización:** Escalar características para que tengan una escala similar (p.e., escalar valores entre 0 y 1).

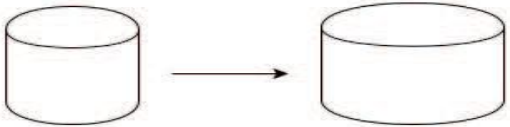
Preprocesamiento

Limpieza de datos

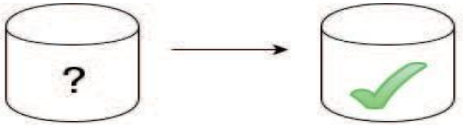
Normalización de datos



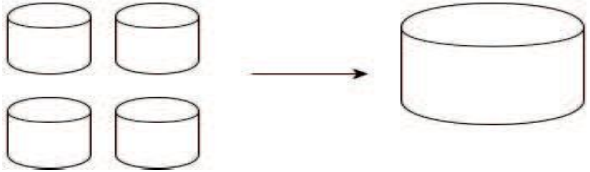
Transformación de datos



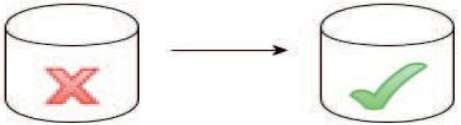
Imputación de valores perdidos



Integración de datos



Identificación de ruido



Preprocesamiento

Una vez procesado se pueden usar para múltiples fines:

- **Análisis de negocios y comercialización:** segmentar clientes, optimizar precios, tendencia de mercado
- **Ciencia e investigación**: modelos climáticos, análisis de pandemias, orígenes de enfermedades
- **Industria y fabricación**: control de calidad, optimización de suministro y logísticas
- **Gobierno y Sector Público**: seguridad nacional y prevenir crimen, toma de decisiones políticas
- **Redes Sociales**: publicidad dirigida, análisis del comportamiento del usuario, recomendaciones
- **Salud y atención médica:** análisis de la salud de la población, diagnósticos mejorados con IA
- **Educación**: evaluación de las actividades enseñanza-aprendizaje, mejora de gestión escolar
- **IoT (Internet de las Cosas):** monitorizar dispositivos, análisis de sensores para toma de deciciones
- **Finanzas**: análisis de riesgos, modelos de seguros de vida, coche… predicción financiera y bolsa
- **Transporte y movilidad**: optimización de rutas, ciudades inteligentes.

Preprocesamiento. Herramientas

Pandas: es una de las herramientas más potente para el **análisis y procesamiento** de datos, entre otros:

- Métodos para trabajar con **valores no disponibles** (NA)
- Métodos para **detectar duplicados:** no debemos tener información repetida
- **Transformaciones** usando funciones
- **Reemplazo** de valores
- **Modificación** de índices
- **Sustitución** de valores fuera de rango
- **Sampleado** o generación de columnas de datos aleatorias
- **Eliminación** de columnas
- **Tratamiento** de datos categóricos
- **Normalización** de valores
- …

Preprocesamiento. Herramientas

Pandas: he aquí algunas herramientas para el análisis y procesamiento de datos veremos:

- ***.isnull() o .isna*** → Devuelve un booleano si tiene valor o no (NaN)
- ***.notnull() \| .notna()*** → opuesto al anterior, devuelve filas con valores
- ***.dropna()*** → devuelve un dataframe sin filas o columnas con valores nulos, con ***inplace=True*** es permanente
- ***.fillna()*** → rellena los valores nulos con el valor que le indiquemos
- ***.duplicated()*** → para cada fila indica si está duplicada o no. Con ***.drop_duplicates()*** las elimina
- ***.map()*** → permite aplicar funciones a una dataframe, p.e., *.map(lambda elemento: num:_a_car(elemento))*
- ***.replace()*** → permite reemplazar en paralelo varios valores por otros
- ***.index*** ó ***.columns*** → permite acceder a índices o columnas, respectivamente
- ***.rename(I)*** → permite renombrar índices y nombres de columnas
- ***.nanpercent()*** → permite sustituir valores fuera de rango
- ***.sample()*** → permite obtener un dataframe con un número aleatorio de filas o columnas
- ***.corr()*** → permite detectar relaciones entre columnas
- ***.drop()*** → permite eliminar una columna

Actividad. **3.1.preprocesado.ipynb**

scikit-learn

- Dependiendo de lo que vayamos a hacer con los datos, puede ser necesario preprocesar los **datos categóricos (datos discretos y no numéricos.** Así, si vamos a utilizar los datos para **Machine Learning**, los algoritmos **trabajan con números**.
- Puede haber otras situaciones en las que podamos seguir trabajando con los datos categóricos. Por ejemplo: **BI para el análisis y visualización de datos** previo a toma de decisiones.
- Existe un paquete que emplean los científicos de datos: ***scikit-learn***
- Se trata de un paquete muy potente, utilizado para el **ML**. Entre otras tareas, permite realizar las operaciones que veremos a continuación: ***label encoding y one hot encoding.***
- Sin embargo, utilizaremos métodos proporcionados por **Pandas**, que son más sencillos.

Recordatorio. El aprendizaje automático, o **"machine learning"** en inglés, es una rama (IA) que se centra en el **desarrollo de algoritmos y modelos** que permiten a las computadoras **aprender patrones** y r**ealizar tareas sin ser programadas explícitamente**. Es decir en vea de ejecutar instrucciones, los **sistemas de aprendizaje automático utilizan datos para aprender** y mejorar su rendimiento con el tiempo.

Hay tres tipos principales de aprendizaje automático:

1. **Aprendizaje Supervisado:** En este enfoque, el modelo **se entrena utilizando un conjunto de datos etiquetado**, donde **cada entrada** del conjunto de datos *tiene una etiqueta que indica la salida deseada*. El modelo aprende a mapear las entradas a las salidas correctas, y luego se puede utilizar para predecir la salida de nuevas entradas no etiquetadas.
   Ejemplos de tareas de aprendizaje supervisado incluyen la clasificación (por ejemplo, predecir si un correo electrónico es spam o no) y la regresión (por ejemplo, predecir el precio de una casa en función de sus características).
2. **Aprendizaje No Supervisado:** En este caso, el modelo se entrena utilizando un **conjunto de datos no etiquetado**, y la tarea principal es descubrir **patrones y estructuras en los datos**. El modelo no recibe información explícita sobre la salida esperada y debe encontrar relaciones o agrupaciones por sí mismo.
   Ejemplos de tareas de aprendizaje no supervisado incluyen el agrupamiento (clusterización) de datos similares y la reducción de dimensionalidad para simplificar conjuntos de datos complejos.
3. **Aprendizaje Reforzado:** En el aprendizaje reforzado, un **agente aprende a tomar decisiones** en un entorno para maximizar una recompensa acumulativa. El agente recibe **retroalimentación en forma de recompensas o penalizaciones** según las acciones que toma en el entorno.
   Ejemplos de tareas de aprendizaje reforzado incluye juegos, robótica y sistemas de control automático.

El aprendizaje automático utiliza una variedad de algoritmos, incluyendo árboles de decisión, máquinas de soporte vectorial (SVM), redes neuronales, algoritmos de clustering, entre otros. Estos algoritmos pueden adaptarse a diferentes tipos de datos y problemas, y el rendimiento de los modelos **mejora a medida que se les proporciona más datos de entrenamiento**. El aprendizaje automático tiene aplicaciones en una amplia gama de campos, como la medicina, la finanzas, la investigación científica, la detección de fraudes, la visión por computadora y muchos más.

scikit-learn - label encoding

En el ejemplo de la actividad ***Actividad. 3.1.preprocesado.ipynb*** hay datos como:

- importe, que es numérico
- sexo, de dos valores
- país, de tres valores

Label encoding consiste en **transformar cada categoría en un valor numérico** (0,1, 2, ...).

Según esta técnica:

- Sexo: tomaría los valores 0 y 1
- País: tomaría valores 0, 1 y 2 (por ejemplo: Alemania, España, Francia)

Hay otros datos en los cuales esto no es posible, los **datos categóricos nominales**, por ejemplo, fecha nacimiento, color de pelo, código postal ya que **aunque sean numéricos no tienen valor** y utilizarlos podría dar a errores (tener un código postal mayor o menor no significa nada en el ML)

scikit-learn - label encoding

Por otro lado, ***label encoding*** es correcta con datos categóricos ordinales (hay un orden).

Por ejemplo, si tenemos tallas: S/M/L, o también en los nominales como el sexo, si podemos usar también ***label encoding.***

Primero modificamos el tipo de la columna a category:

```
df_pruebas['Sexo'] = df_pruebas['Sexo'].astype('category')
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   Importe  5 non-null      float64
 1   Sexo     5 non-null      category
 2   País     5 non-null      object
dtypes: category(1), float64(1), object(1)
```

	Importe	Sexo	País
0	567.87	Hombre	España
1	12.00	Mujer	Francia
2	29.56	Mujer	Alemania
3	1001.54	Mujer	Alemania
4	1.00	Hombre	España

scikit-learn - label encoding

Ahora al ser tipo categoría, accedemos a la propiedad ***.cat.codes,*** sustituyendo el valor textual por el **identificador numérico asignado por Pandas**, con lo que ya podemos trabajar con ellos

```
df_pruebas['Sexo'] = df_pruebas['Sexo'].cat.codes
```

	Importe	Sexo	País
0	567.87	0	España
1	12.00	1	Francia
2	29.56	1	Alemania
3	1001.54	1	Alemania
4	1.00	0	España

Ya tenemos valores numéricos con los que trabajar en caso necesario. También puedes usar esta técnica con datos **categóricos ordinales.**

Un analista de datos puede realizar esta operación con la clase *LabelEncoder()* de sklean.preprocessing (documentado en Notebook).

scikit-learn - one hot encoding

Cuando hay datos categóricos **de más de dos valores** se genera una nueva columna en las cuales se almacena 0 o 1 según la fila sea de esa categoría o no. **Estas columnas se denominan *dummies*.** En el caso de los países:

	Importe	Sexo	País
0	567.87	0	España
1	12.00	1	Francia
2	29.56	1	Alemania
3	1001.54	1	Alemania
4	1.00	0	España

	Importe	Sexo	País_Alemania	País_España	País_Francia
0	567.87	0	0	1	0
1	12.00	1	0	0	1
2	29.56	1	1	0	0
3	1001.54	1	1	0	0
4	1.00	0	0	1	0

scikit-learn - one hot encoding

Ahora con el método ***.get_dummies()***

pd.get_dummies(df_pruebas, columns=['País'], prefix="País")

Indicamos la columna y podemos añadir un prefijo que se añade antes del valor de cada categoría en las columnas generadas. (***País***_*Alemania*,...)

Si nos fijamos existe una correlación entre una de las nuevas columnas y las otras dos, cuando Alemania y España = 0, Francia = 1.

Como los algoritmos de ML no deben recibir información redundante podemos eliminar una de ellas, la que queramos

	Importe	Sexo	País_Alemania	País_España	País_Francia
0	567.87	0	0	1	0
1	12.00	1	0	0	1
2	29.56	1	1	0	0
3	1001.54	1	1	0	0
4	1.00	0	0	1	0

scikit-learn - one hot encoding

df_pruebas.drop(['País_Francia'], axis=1, inplace=True)

	Importe	Sexo	País_Alemania	País_España
0	567.87	0	0	1
1	12.00	1	0	0
2	29.56	1	1	0
3	1001.54	1	1	0
4	1.00	0	0	1

Esta misma operación se puede hacer con la clase

OneHotEncoder.

Otra cosa es que no siempre se realiza esta conversión de categorías, siempre depende de lo que hagas con los datos.

Como en el caso de las categorías, **no siempre** se debe hacer.

La NV es necesaria, por ejemplo, si vamos a emplear los datos para alimentar algoritmos de ML, que utilizan los valores para realizar predicciones o clasificaciones.

Normalización de valores

Existen diferentes tipos de NV: respecto de la **media**, de un **percentil**, **mediana**, ...

- Uno de los más usados es el ***Min-Max*.** Genera un conjunto equivalente de valores en el que el menor valor se transforma en 0 y el mayor en 1.
- Se calcula **tomando el valor, restando el mínimo de la columna** y **dividiendo todo entre la diferencia del máximo y mínimo de la columna.**

```
df_normalizado = (df_pruebas-df_pruebas.min())/(df_pruebas.max()-df_pruebas.min())
```

	Importe	Sexo	País_Alemania	País_España
0	567.87	0	0	1
1	12.00	1	0	0
2	29.56	1	1	0
3	1001.54	1	1	0
4	1.00	0	0	1

df_normalizado

	Importe	Sexo	País_Alemania	País_España
0	0.566564	0.0	0.0	1.0
1	0.010994	1.0	0.0	0.0
2	0.028545	1.0	1.0	0.0
3	1.000000	1.0	1.0	0.0
4	0.000000	0.0	0.0	1.0

Normalización de valores

Cada vez que decidas incorporar los datos de una nueva fuente debes analizarla en detalle para entender su contenido y decidir qué transformaciones hacer.

Vamos a trabajar con el fichero *RegistroComprasOnline.ods*. Realizaremos operaciones descritas anteriormente. En concreto:

- ¿Existen filas repetidas?
- ¿Tenemos algún valor no informado? ¿Qué haremos con ellos?
- ¿Hay columnas que no aporten nueva información?
- Si hay categóricos, ¿debemos hacer algo con ellos?
- ¿Es necesario normalizar datos?

Actividad: **A.3.1.preprocesado.ipynb**