ANÁLISIS DE DATOS

Clase 4. Taller de preparación de datos (cont)

Temario

1. Imputación de datos faltantes. (cont)
2. Codificación de variables categóricas.
3. Discretización.
4. Tratamiento de valores extremos (outliers).
5. Feature scaling.
6. Cadenas de procesamiento.
7. Transformación de variables.

1. IMPUTACIÓN DE DATOS FALTANTES.

Continuamos con lo de la clase pasada

Imputación multivariada.

- Las técnicas anteriores reemplazan los valores faltantes de una observación sin utilizar información del resto de las variables.
- Dos técnicas de imputación multivariada:
  - MICE (Multiple Imputation by Chained Equations)
  - Imputación por vecinos cercanos (KNN)

Multiple imputation by chained equations

- Es una de las técnicas de imputación multivariada más utilizada.
- Consiste en calcular valores faltantes para cada columna como una regresión lineal de las restantes (si bien también se pueden usar otros modelos).
- Es iterativo porque repite este proceso hasta que la diferencia entre las dos últimas iteraciones sea cercana a cero (varianza estable).
- Referencias:
  - Publicación original (2011): https://www.jstatsoft.org/article/view/v045i03
  - Multiple imputation by chained equations: what is it and how does it work? https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3074241/

MICE. Algoritmo. Ejemplo

- Un banco tiene una base de datos de clientes que compraron una tarjeta de crédito.
- Quiere desarrollar un modelo para saber si un cliente es un potencial comprador o no.
- Cada observación representa un cliente, y las variables observadas son:
  - Edad
  - Experiencia
  - Salario
- La variable a predecir para cada cliente es si decidió comprar la tarjeta (0=No,1=Sí).

	age	experience	salary	purchased
0	25.0	NaN	50.0	0
1	27.0	3.0	NaN	1
2	29.0	5.0	110.0	1
3	31.0	7.0	140.0	0
4	33.0	9.0	170.0	1
5	NaN	11.0	200.0	0

MICE. Algoritmo. Ejemplo

- Supondremos, por un momento que, disponemos de los datos originales.
- ¿Si imputáramos por promedio, esa imputación sería representativa?

	age	experience	salary	purchased
0	25	1	50	0
1	27	3	80	1
2	29	5	110	1
3	31	7	140	0
4	33	9	170	1
5	35	11	200	0

Datos originales.

	age	experience	salary
0	25.0	7.0	50.0
1	27.0	3.0	134.0
2	29.0	5.0	110.0
3	31.0	7.0	140.0
4	33.0	9.0	170.0
5	29.0	11.0	200.0

Imputación por media.

MICE. Algoritmo. Ejemplo

- Paso 1. Construir un dataset inicial, que llamaremos “0”, realizando una imputación estadística (por promedio o mediana).

	age	experience	salary
0	25.0	7.0	50.0
1	27.0	3.0	134.0
2	29.0	5.0	110.0
3	31.0	7.0	140.0
4	33.0	9.0	170.0
5	29.0	11.0	200.0

Dataset 0

MICE. Algoritmo. Ejemplo

- Paso 2. Crear un Dataset “1” completando los datos faltantes de cada columna a partir de un modelo regresivo sobre las columnas restantes.

	age	experience	salary
0	25.0	7.0	50.0
1	27.0	3.0	134.0
2	29.0	5.0	110.0
3	31.0	7.0	140.0
4	33.0	9.0	170.0
5	?	11.0	200.0

Dataset 1

```
def impute_column(df, col_to_predict, feature_columns):
        """ Imputar valores faltantes de una columna a partir de un
            modelo LR sobre columnas restantes
        """
        # Establecer en NaN los valores a predecir
        nan_rows = np.where(np.isnan(df[col_to_predict]))
        all_rows = np.arange(0,len(df))

        # Separar datos de entrenamiento de datos para predicciones
        train_rows_idx = np.argwhere(~np.isin(all_rows,nan_rows)).ravel()
        pred_rows_idx =  np.argwhere(np.isin(all_rows,nan_rows)).ravel()
         X_train,y_train = df[feature_columns].iloc[train_rows_idx],
       df[col_to_predict].iloc[train_rows_idx]

        # Predecir
        X_pred = df[feature_columns].iloc[pred_rows_idx]
        model = LinearRegression()
        model.fit(X_train,y_train)
        df[col_to_predict].iloc[pred_rows_idx] = model.predict(X_pred.values.reshape(1,-1))
    return df
```

MICE. Algoritmo. Ejemplo

- Paso 3. Calcular la diferencia entre los dos datasets anteriores.
- Continuar repitiendo pasos anteriores hasta que la diferencia sea próxima a cero, o haber alcanzado una determinada cantidad de iteraciones.

	age	experience	salary
0	25.000000	1.853863	50.00000
1	27.000000	3.000000	72.77481
2	29.000000	5.000000	110.00000
3	31.000000	7.000000	140.00000
4	33.000000	9.000000	170.00000
5	36.253165	11.000000	200.00000

Dataset 1

-

	age	experience	salary
0	25.0	7.0	50.0
1	27.0	3.0	134.0
2	29.0	5.0	110.0
3	31.0	7.0	140.0
4	33.0	9.0	170.0
5	29.0	11.0	200.0

Dataset 0

=

	age	experience	salary
0	0.000000	-5.146137	0.00000
1	0.000000	0.000000	-61.22519
2	0.000000	0.000000	0.00000
3	0.000000	0.000000	0.00000
4	0.000000	0.000000	0.00000
5	7.253165	0.000000	0.00000

Diferencia

MICE. Algoritmo. Convergencia

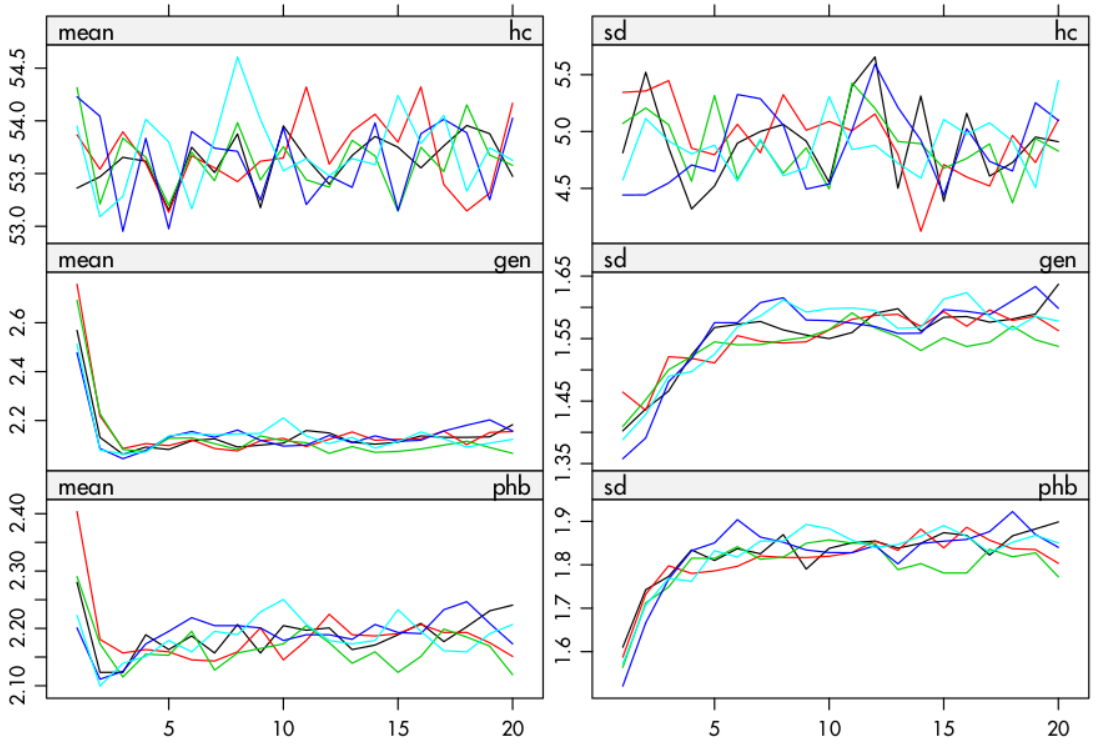
4.3. Assessing convergence

There is no clear-cut method for determining whether the MICE algorithm has converged. What is often done is to plot one or more parameters against the iteration number. The `mice()` function produces m parallel imputation streams. The `mids` object contains components `chainMean` and `chainVar` with the mean and variance of the imputations per stream, respectively. These can be plotted by the `plot.mids` object. On convergence, the different streams should be freely intermingled with each other, without showing any definite trends. Convergence is diagnosed when the variance between different sequences is no larger than the variance with each individual sequence.

Inspection of the stream may reveal particular problems of the imputation model. Section 3.4 shows that passive imputation should carefully set the predictor matrix. The code below is a pathological example where the MICE algorithm is stuck at the initial imputation.

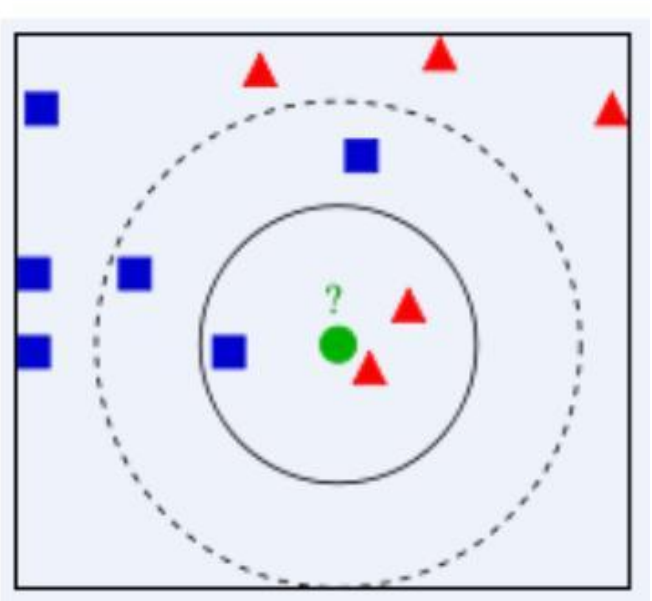
MICE. Algoritmo. Convergencia

Figure 9: Healthy convergence of the MICE algorithm for `hgt`, `wgt` and `bmi`, where feedback loop of `bmi` into `hgt` and `wgt` is broken (solution `imp.idx`).



Imputación multivariada. K-Nearest Neighbours

- Consiste en utilizar el algoritmo KNN para estimar los valores faltantes por semejanza a los más próximos.
- Se suele utilizar con una función de distancia que contemple la presencia de NaNs, como por ejemplo ***nan_euclidean_distance***.
- Al igual que en KNN, el número de vecinos suele ser un número impar.



Imputación multivariada. K-Ne o

```
from sklearn.metrics.pairwise import nan_euclidean_distances
import numpy as np
#dist(x,y) = sqrt(weight * sq. distance from present coordinates)
#where, weight = Total # of coordinates / # of present coordinates

row_to_impute = df.iloc[2]
feature_cols = ['Metallica', 'Luis Miguel', 'Astor Piazzolla', 'Jimi Hendrix']
music_df['nan_euc_dist'] = music_df[feature_cols].apply(
    lambda row: nan_euclidean_distances( row_to_impute.values.reshape(1,-1),
            row.values.reshape(1,-1))[0][0],axis=1)
music_df
```

	Metallica	Luis Miguel	Astor Piazzolla	Jimi Hendrix	nan_euc_dist
0	80.0	30.0	7	27.0	120.461612
1	44.0	NaN	10	29.0	121.155547
2	NaN	85.0	25	88.0	168.103143
3	50.0	70.0	74	49.0	90.917545
4	29.0	54.0	49	NaN	90.347477

39.5

```
(50+29)/2
```

```
39.5
```

(ejemplo para dos vecinos cercanos)

Imputación multivariada. K-Nearest Neighbours

- Demo interactiva: http://vision.stanford.edu/teaching/cs231n-demos/knn/
- nan_euclidean_distance: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.nan_euclidean_distances.html

Ejemplos en jupyter

Clase 4.3 - Preparación de datos - Imputación de datos faltantes.ipynb

2. CODIFICACIÓN DE VARIABLES CATEGÓRICAS

Codificación de variables categóricas.

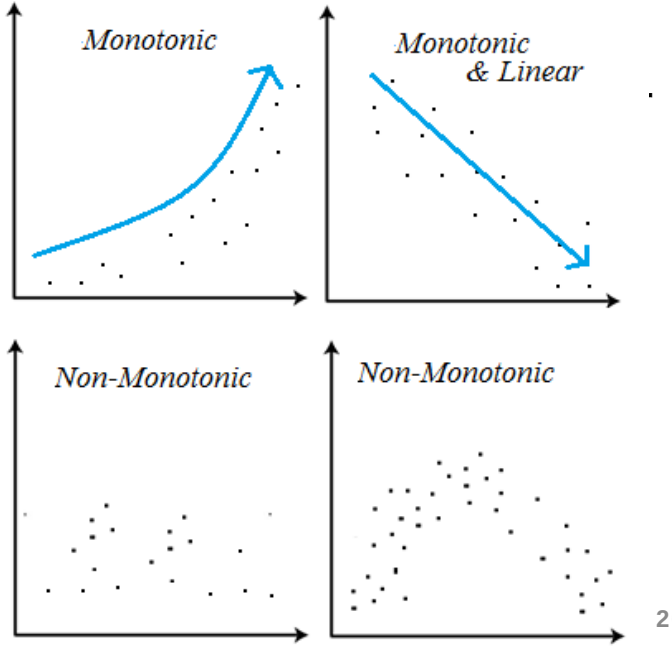
- Consiste en reemplazar una categoría (típicamente representada en forma de texto) por una representación numérica.
- El objetivo es disponer de variables que puedan ser utilizadas en los modelos de AA.

Codificación de variables categóricas. Técnicas.

Tradicionales	• One Hot Encoding • Count/frequency encoding • Ordinal/label encoding
Relación monotónica	• Label encoding ordenado. • Encoding por promedio • Peso de la evidencia (WoE)
Alternativas	• Binary encoding • Feature hashing • Otros

Relación monotónica

- Decimos que existe una relación monotónica entre una variable independiente X e una variable objetivo Y cuando:
  - Si se incrementa el valor de X ta de Y ó
  - Si se incrementa el valor de X, Y



Relación monotónica. Importancia.

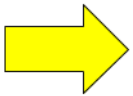
- Las relaciones monotónicas entre variables de entrada y de salida pueden mejorar el desempeño de los modelos lineales.
- En los modelos de árboles, puede traducirse en menor profundidad (sin comprometer su desempeño).
- A menudo, estas relaciones aparecen de manera natural en los datos. Por ejemplo: las primas de los seguros suelen disminuir a medida que aumenta la edad de los asegurados.
- Algunas de las formas de codificación que se mostrarán, estarán orientadas intentar obtener esta relación entre la variable transformada y la variable objetivo.

One Hot Encoding

- Consiste en codificar cada valor de una variable categórica con un conjunto de variables booleanas que pueden tomar 0 o 1, indicando si esa categoría está o no presente en cada observación.
- Si la variable tiene k-valores posibles, puede codificarse utilizando k o k-1 nuevas variables (en el último caso se suele llamar dummy encoding).

Color
Red
Red
Yellow
Green
Yellow

OHE



Red	Yellow	Green
1	0	0
1	0	0
0	1	0
0	0	1

DE

	Travel_Class_2	Travel_Class_3
Passenger 1	0	0
Passenger 2	1	0
Passenger 3	0	1
Passenger 4	0	0

One Hot Encoding. k o k-1?

- Codificar utilizando k-1 variables disminuye el costo adicional de creación de variables (tener presente que es común realizar entrenamientos sobre el dataset completo).
- En algunos modelos que tienen un término de sesgo (bias), como por ejemplo regresión lineal, la presencia de una matriz OHE con k variables haría que la matriz sea singular y por lo tanto no invertible, imposibilitando la solución por fórmula cerrada.
- No obstante, para los siguientes escenarios se aconseja utilizar k variables:
  - Algoritmos basados en árboles.
  - Feature selection por algoritmos recursivos.
  - Para determinar la importancia de cada categoría.

One Hot Encoding. Ventajas.

- No realiza ningún supuesto sobre la distribución de las categorías de la variable.
- Mantiene toda la información de la variable categórica.
- Es apta para modelos lineales.

One Hot Encoding. Limitaciones.

- Aumenta la dimensión del espacio de variables de entrada.
- No agrega información.
- Introduce muchas variables con información redundante.

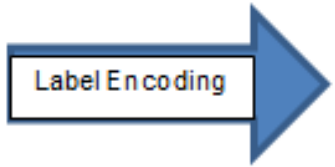
One Hot Encoding. Variante para top-n categorías.

- Una variante de OHE es utilizarla sólo para las top n categorías más frecuentes.
- Esta técnica fue la solución ganadora en KDD 2009: http://www.mtome.com/Publications/CiML/CiML-v3-book.pdf

Label/integer encoding. Definición.

- Consiste en reemplazar las categorías por valores numéricos de 0 a N-1.
- Los números se asignan de manera arbitraria.

	occupation
0	programmer
1	data scientist
2	engineer
3	manager
4	ceo



	occupation
0	4
1	1
2	2
3	3
4	0

Label/integer encoding. Ventajas.

- Fácil de implementar, no expande la dimensión del espacio de variables de entrada.
- Puede funcionar bien con algoritmos basados en árboles.

Label/integer encoding. Limitaciones.

- La codificación no aporta información.
- No es apropiado para modelos lineales.
- En producción, no maneja automáticamente nuevas categorías.

Count/Frequency encoding. Definición.

- Cada categoría se reemplaza por el valor o porcentaje de observaciones en que aparece en el dataset.
- Se captura la representación de cada categoría.
- Muy utilizado en competencias de Kaggle.
- Se hace un supuesto importante: la cantidad de observaciones para cada categoría está relacionada con la variable a predecir.

Count/Frequency encoding. Ventajas.

- Fácil de implementar.
- No expande las dimensiones del espacio de variables de entrada.
- Puede tener un desempeño aceptable con modelos basados en árboles.

Count/Frequency encoding. Limitaciones.

- No apropiado para modelos lineales.
- No maneja automáticamente nuevas categorías en test set/producción.
- Si aparecen dos categorías la misma cantidad de veces, pueden ser reemplazadas por el mismo número, con la consecuente pérdida de información.

Ordinal encoding c/ orden. Definición

- Consiste en reemplazar las categorías por valores numéricos de 0 a N-1, pero en este caso el ordenamiento no es arbitrario.
- La asignación de cada entero asignado corresponde con el promedio de la variable objetivo de cada categoría.

Ordinal encoding c/ orden. Ventajas

- Fácil de implementar.
- No expande las dimensiones del espacio de variables de entrada.
- Crea una relación monotónica entre las categorías y la variable objetivo.

Ordinal encoding c/ orden. Limitaciones

- Puede introducir overfitting.
- Las librerías estándar como SkLearn no lo soportan directamente, por lo que no es directo su uso en un esquema de cross-validation o k-folds validation.

Mean encoding. Definición

- Consiste en reemplazar cada categoría por el promedio de la variable objetivo para esa categoría.

Mean encoding. Ventajas

- Fácil de implementar.
- No expande las dimensiones del espacio de características.
- Crea una relación monotónica entre las categorías y la variable objetivo.

Mean encoding. Limitaciones

- Puede introducir overfitting.
- Las librerías estándar como SkLearn no lo soportan directamente, por lo que no es directo su uso en un esquema de cross-validation o k-folds validation.
- Puede ocurrir un escenario en el que dos categorías tengan un promedio muy similar, con la consecuente pérdida de información.

Peso de Evidencia (*Weight of Evidence*).

Definicion

- Esta técnica se originó para modelos financieros y riesgo crediticio.
- Consiste en reemplazar una variable binaria por el logaritmo natural del ratio del valor positivo respecto del valor negativo o por defecto (el orden puede invertirse dependiendo del caso, por ejemplo, en modelos financieros se suele usar p(0)/p(1)).

$$WoE = ln\frac{P(X=1)}{P(X=0)}$$

Peso de Evidencia (*Weight of Evidence*).

Ventajas.

- Crea una relación monotónica entre la variable objetivo y las variables de entrada.
- Funciona muy bien con regresión logística, por la forma en que quedan ordenadas las categorías.
- Las variables transformadas pueden ser comparadas porque están en la misma escala, por lo tanto, es casi inmediato determinar cuál es más predictiva.

Peso de Evidencia (Weight of Evidence). Limitaciones.

- Puede llevar a overfitting.
- Indefinida cuando el denominador es cero.

Codificación de etiquetas poco frecuentes.

- Las etiquetas poco frecuentes aparecen en una proporción pequeña de las observaciones del dataset.
- Suelen presentarse en los siguientes escenarios:
    - Variables con una categoría predominante.
    - Variables con pocas categorías.
    - Variables con alta cardinalidad.
- La recomendación es agrupar todas las categorías poco frecuentes en una nueva categoría “raras”.

Binary encoding / feature hashing. Definición

- Binary encoding es una combinación de OHE y ordinal encoding. Se codifica cada variable de entrada como una composición de variables binarias.
- Feature hashing, aplica una función de hash a cada valor de entrada para obtener un entero.
- En ambos casos:
  - Ventaja → eficiencia en la representación
  - Desventaja → pérdida de interpretabilidad.

			Binary Encoded			
Categorical Feature		=	x1	x2	x4	x8
Louise	=>	1	1	0	0	0
Gabriel	=>	2	0	1	0	0
Emma	=>	3	1	1	0	0
Adam	=>	4	0	0	1	0
Alice	=>	5	1	0	1	0
Raphael	=>	6	0	1	1	0
Chloe	=>	7	1	1	1	0
Louis	=>	8	0	0	0	1
Jeanne	=>	9	1	0	0	1
Arthur	=>	10	0	1	0	1

Ejemplos en jupyter

Clase 4.4 - Preparación de datos - Codificación de variables categóricas.ipynb

3. DISCRETIZACIÓN

Definición

- Es el proceso de transformar variables continuas en variables con valores discretos mediante la creación de intervalos contiguos que cubren el espectro de valores que toma la variable.
- También se le llama **binning**, donde “bin” (en inglés “cesto”) es un nombre alternativo para un intervalo.

Motivación

- En preparación de datos para modelos de ML, es de interés discretizar por una o más de las siguientes razones:
  - algunas técnicas para discretizar permiten convertir distribuciones con oblicuidad en distribuciones normales o uniformes.
  - manejo de outliers: se pueden agrupar valores extremos en el primer y último intervalo.
  - al obtener una cantidad finita de valores posibles para una variable, puede combinarse con otros métodos, por ejemplo, codificación de variables categóricas.

Métodos de discretización

- No supervisados:
  - Ancho fijo
  - Frecuencia fija
  - K-means
- Supervisados:
  - Árboles de decisión

Discretización por intervalos de ancho fijo.
Definición y características.

- Divide el espectro de valores posible en N bins del mismo ancho.
- El tamaño de cada intervalo está dado por:
  - ancho = (max-min)/N
- Características:
  - No mejora la distribución.
  - Permite manejar valores extremos (dejan de ser extremos, y quedan asignados a los intervalos de de los extremos).
  - Crea una variable discreta.
  - Puede combinarse con codificaciones categóricas.

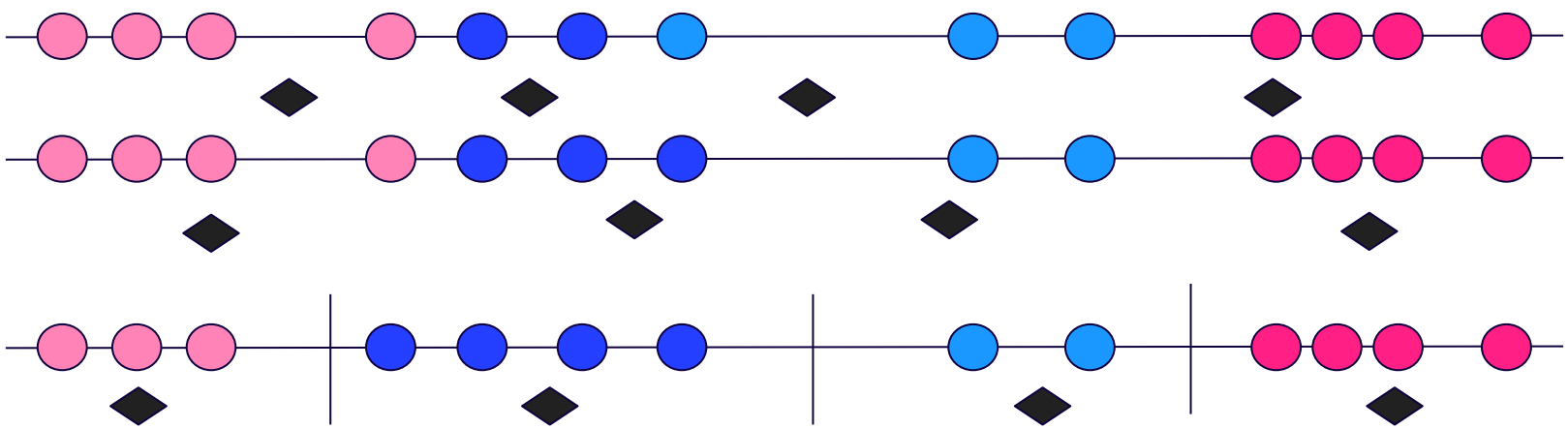
Discretización por intervalos de frecuencia fija.

Definición y características.

- Divide el espectro de valores posible en N bins, donde cada bin tiene aproximadamente la misma cantidad de observaciones.
- La definición de cada intervalo se realiza a partir de los cuantiles.
- Características:
  - Mejora la distribución.
  - Permite manejar valores extremos.
  - Crea una variable discreta.
  - Puede combinarse con codificaciones categóricas.

Discretización por K-means. Definición.

- Consiste en aplicar k-means para hallar n intervalos.



Discretización por K-means. Características.

- No mejora la distribución.
- Permite manejar valores extremos, si bien pueden influenciar la ubicación de los centroides.
- Crea una variable discreta.
- Puede combinarse con codificaciones categóricas.

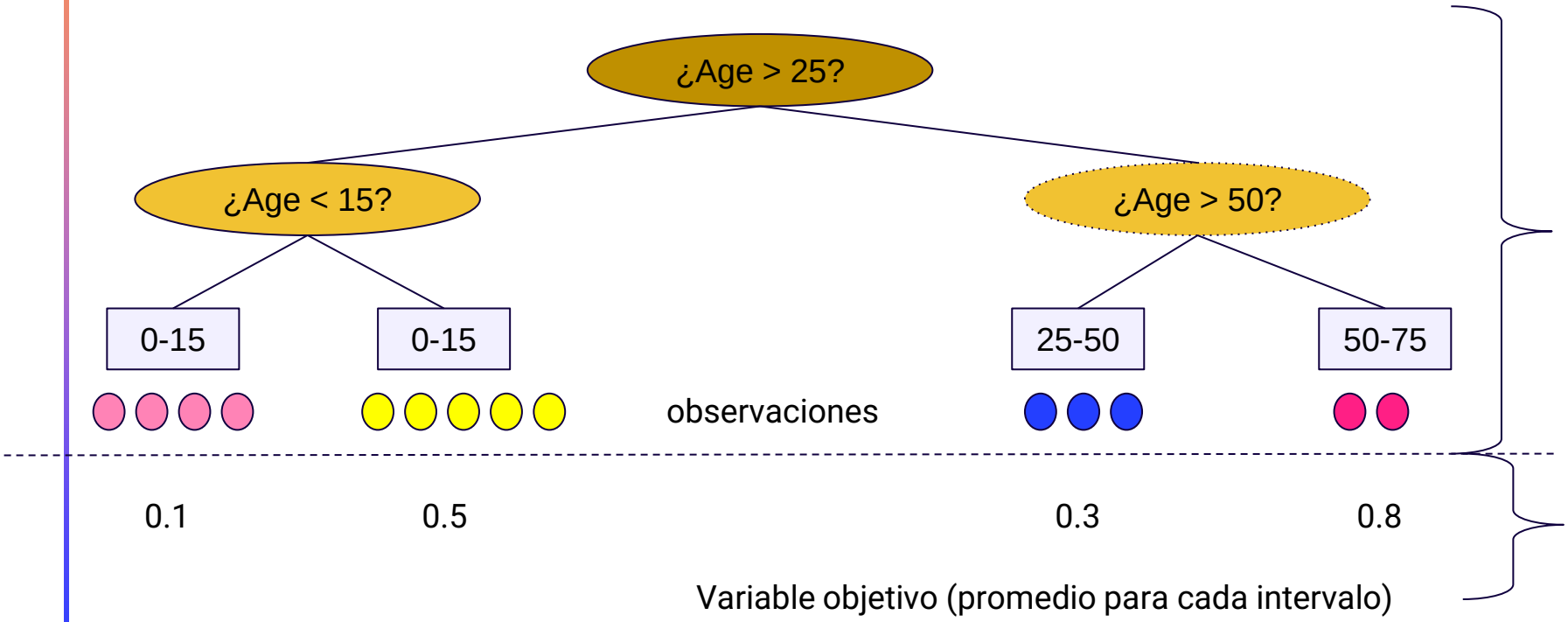
Discretización + codificación

- Mencionamos que la discretización es el proceso de representar un intervalo continuo como una variable de k intervalos (de 0 a k-1).
- Si se está construyendo un modelo no-lineal, entonces puede utilizarse esta nueva variable directamente.
- Si en cambio, se está trabajando con un modelo lineal se pueden interpretar los bins como si fueran categorías.
- En este último caso es posible aplicar las mismas técnicas vistas en codificación categórica para obtener una relación monotónica entre la variable independiente y la variable objetivo.

Discretización con árboles de decisión. Definición.

- Consiste en utilizar un árbol de decisión para hallar los bins óptimos.
- Una decisión de un árbol asigna una observación a una de sus N hojas.
- Los árboles generan una salida discreta, cuyos valores son las predicciones para cada una de sus N hojas.

Discretización con árboles de decisión. Ejemplo.



Tratamiento de valores extremos

- No mejora la distribución.
- Maneja outliers.
- Crea una variable discreta.
- Crea una relación monotónica.

Ejemplos en jupyter

Clase 4.6 - Preparación de datos - Discretización.ipynb

4. TRATAMIENTO DE VALORES EXTREMOS

Tratamiento de valores extremos. Motivación.

- En la sección 2 se introdujo el concepto de “outlier” y cómo identificarlo para distribuciones con y sin oblicuidad.
- Dado que los valores extremos pueden tener un efecto negativo en la capacidad de generalizar de algunos algoritmos, se expondrán algunas técnicas para su tratamiento y sus ventajas y desventajas.

Tratamiento de valores extremos. Métodos.

Poda

- Remover los valores extremos del dataset.

Datos faltantes

- Tratar los valores extremos como datos faltantes y aplicar métodos de imputación.

Discretización

- Ubicar los valores extremos en primer y último bin.

Censura

- Limitar valor (capping)
- Codificación extremo inferior/superior.
- Winsorization

Vistos en secciones 3 y 6.

Poda (trimming). Definición y características.

- Consiste en eliminar los outliers del dataset.
- Ventajas:
  - Facilidad de implementación, rapidez.
- Desventajas:
  - Pérdida de información.
  - En la mayoría de los casos se debe eliminar una observación completa aún cuando sólo falten valores en alguna de las variables.

Censura/limitar valor (capping). Definición y características.

- Consiste en reemplazar los outliers por el valor límite más cercano.
- Ventajas:
  - Facilidad de implementación, rapidez.
  - No es necesario eliminar una observación completa.
- Desventajas:
  - Distorsiona la distribución original.

Detección de valores extremos.

- Distribución normal → Media y desvío estándar.
- IQR, regla de proximidad.
- Cuantiles.

Ejemplos en jupyter

Clase 4.7 - Preparación de datos - Valores extremos.ipynb

5. FEATURE SCALING

Motivación. Cómo influye la magnitud de las variables de entrada (1 / 2)

- El coeficiente de regresión está directamente influenciado por la escala de la variable.
- Las variables con mayor rango de magnitud/valor dominan sobre las que tienen un menor rango de magnitud/valor.
- Los algoritmos de la familia de Gradient Descent convergen más rápidamente cuando las variables de entrada están en la misma escala.
- Las distancias euclídeas son sensibles a la magnitud de las variables.

Motivación Cómo influye la magnitud de las variables de entrada (2 / 2)

- Algoritmos afectados
  - Regresión lineal y logística.
  - Redes neuronales
  - Support Vector Machines
  - KNN
  - K-means
  - Principal Component Analysis (PCA)
- No afectados (por ejemplo, los basados en árboles)
  - Árboles de clasificación y regresión.
  - Random Forest
  - Gradient Boosted Trees.

Feature scaling (escalado de variables)

- En ML, se refiere a los métodos utilizados para normalizar el rango de valores que puede tomar una variable independiente.
- Suele ser el último paso en una cadena de procesamiento de datos (realizado justo antes del entrenamiento de algoritmos).

Feature scaling. Métodos

- Los más utilizados:
  - Estandarización
  - Escalado a mínimo-máximo.
- Otros métodos:
  - Normalización de media.
  - Escalado a valor absoluto.
  - Escalado a mediana y cuantiles.
  - Escalado a norma unitaria.

Estandarización

- Centra la variable en cero y establece su varianza a 1.

$$Z = \frac{x - \mu}{\sigma}$$

Estandarización. Efecto

- Centra la media en 0.
- Escala la varianza a 1.
- Preserva la forma de la distribución original.
- Los mínimos y máximos pueden variar.
- Preserva los valores extremos (outliers).

Escalado a mínimo-máximo.

- Esta transformación restringe el valor de la variable a un rango (típicamente x_min=0,x_max=1).

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

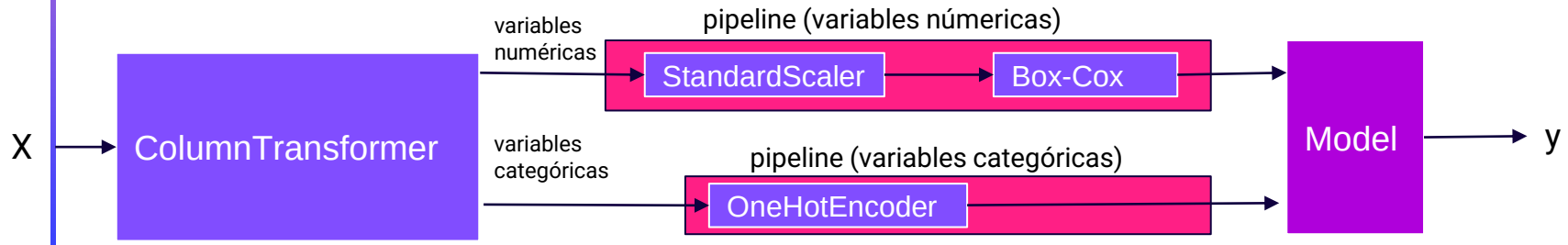
Escalado a mínimo-máximo. Efecto.

- La media puede variar.
- La varianza puede variar.
- Puede modificar la forma de la distribución original.
- Los mínimos y máximos quedan restringidos.
- Preserva los valores extremos (outliers).

6. CADENAS DE PROCESAMIENTO

Implementación de cadenas de procesamiento con SKLearn

- Arquitectura de SKlearn:
  - **Transformer**: clase para transformar datos. Debe implementar fit() y transform().
  - **Predictor:** clase para realizar predicciones. Debe implementar fit() y predict().
  - **Pipeline:** clase que ejecuta secuencia de transformers y/o predictors.
    - Cada elemento de la lista es un paso (step).
    - El único elemento de la lista que puede ser predictor es el último (los predecesores deben ser transformers).
  - **Columntransformer**: clase que permite aplicar transformaciones específicas para cada columna de un dataset.



Implementación de cadenas de procesamiento con SKLearn

- Ejemplos de uso de clases **Pipeline** y **ColumnTransformer** para problemas de clasificación y regresión.
- Referencias:
  - https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html
  - https://scikit-learn.org/stable/modules/compose.html#columntransformer-for-heterogeneous-data
  - Sección de trransformaciones de datos de documentación de SKLearn: https://scikit-learn.org/stable/data_transforms.html

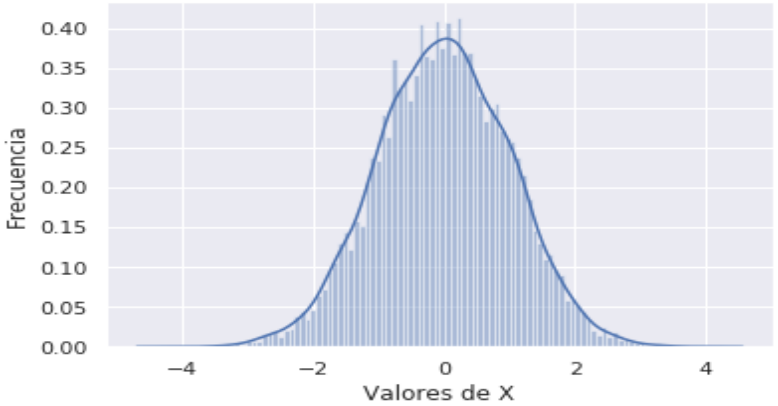
Ejemplos en jupyter

Clase 4.8 - Preparación de datos - Cadenas de procesamiento.ipynb

7. TRANSFORMACIÓN DE VARIABLES

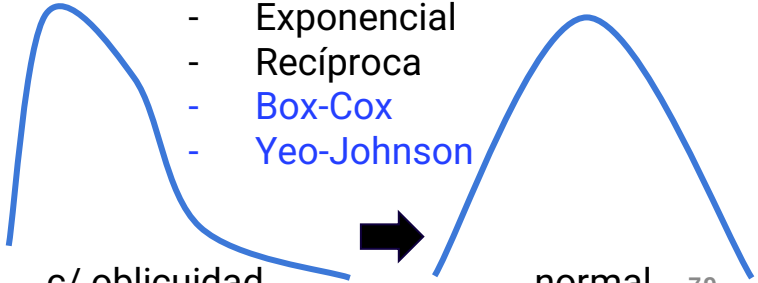
Distribución normal en modelos lineales

- Es deseable que los valores de cada variable independiente (X) tengan una distribución normal.



- Es muy común que esto no se cumpla en los datos originales. En este caso, puede intentar obtenerse una distribución más semejante a una normal luego de aplicar una transformación.

- Logarítmica
- Exponencial
- Recíproca
- Box-Cox
- Yeo-Johnson



Transformaciones matemáticas

- Logarítmica: $\log(x)\,, x > 0$
- Recíproca: $\frac{1}{x}, \forall x \in \mathbb{R} - \{0\}$
- Potencia/exponencial
  - $X\ e^{\lambda}$
  - $X^{1/2}/X^{3}$
  - No definido para todo X.
- Exponencial (casos especiales):
  - *Box-Cox*, X>0
  - *Yeo-Johnson*

Transformación de Box Cox

- La transformación de Box-Cox estima un valor de lambda que minimiza la desviación estándar de una variable transformada estandarizada.
- El método de Box-Cox busca entre muchos tipos de transformaciones.

$$y_i^{(\lambda)} = \begin{cases} \dfrac{y_i^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0, \\ \ln(y_i) & \text{if } \lambda = 0, \end{cases}$$

L	Y'
-2	$Y^{-2} = 1/Y^2$
-1	$Y^{-1} = 1/Y^1$
-0.5	$Y^{-0.5} = 1/(\text{Sqrt}(Y))$
0	log(Y)
0.5	$Y^{0.5} = \text{Sqrt}(Y)$
1	$Y^1 = Y$
2	Y^2

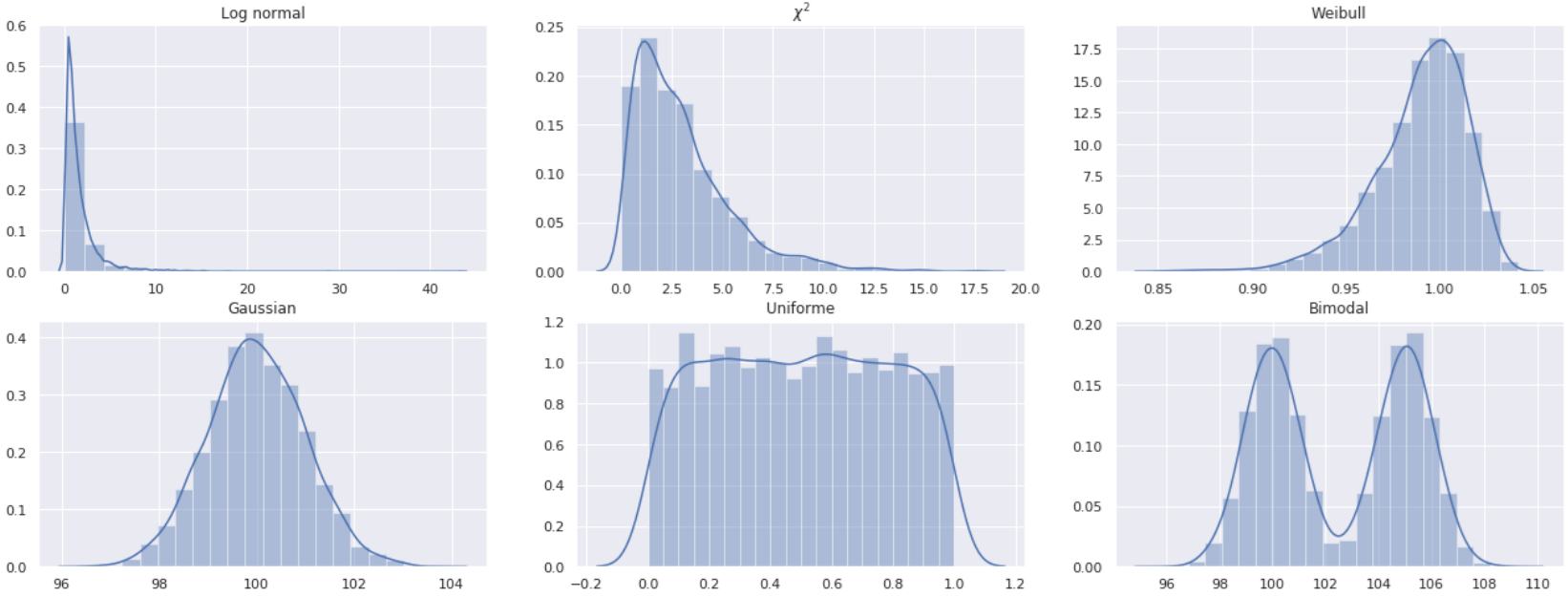
Source: Box and Cox (1964). Where, 'Y' is the transformation of t
Note that for Lambda = 0, the transformation is NOT Y^0 (because t
every

Transformación de Yeo-Johnson

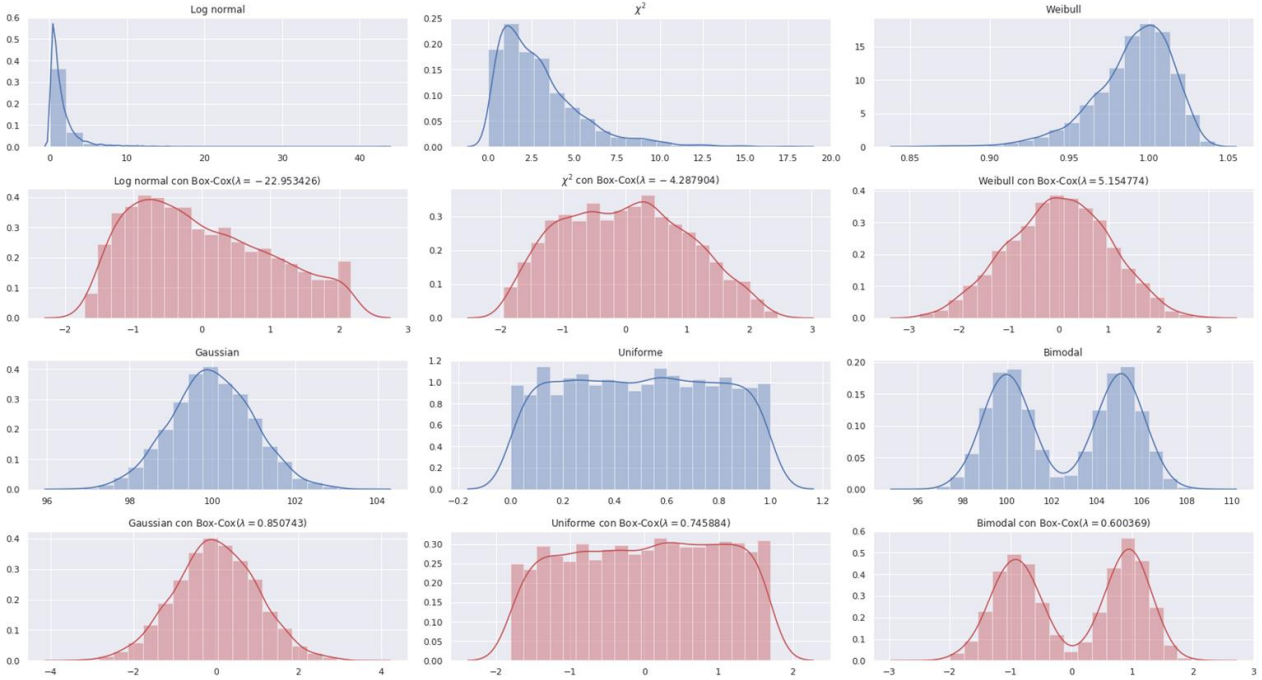
- Junto con Box-Cox, es otra de las transformaciones más utilizadas.
- Admite que X tome valores negativos.

$$y_i^{(\lambda)} = \begin{cases} ((y_i + 1)^{\lambda} - 1)/\lambda & \text{if } \lambda \neq 0, y \geq 0 \\ \log(y_i + 1) & \text{if } \lambda = 0, y \geq 0 \\ -[(-y_i + 1)^{(2-\lambda)} - 1]/(2 - \lambda) & \text{if } \lambda \neq 2, y < 0 \\ -\log(-y_i + 1) & \text{if } \lambda = 2, y < 0 \end{cases}$$

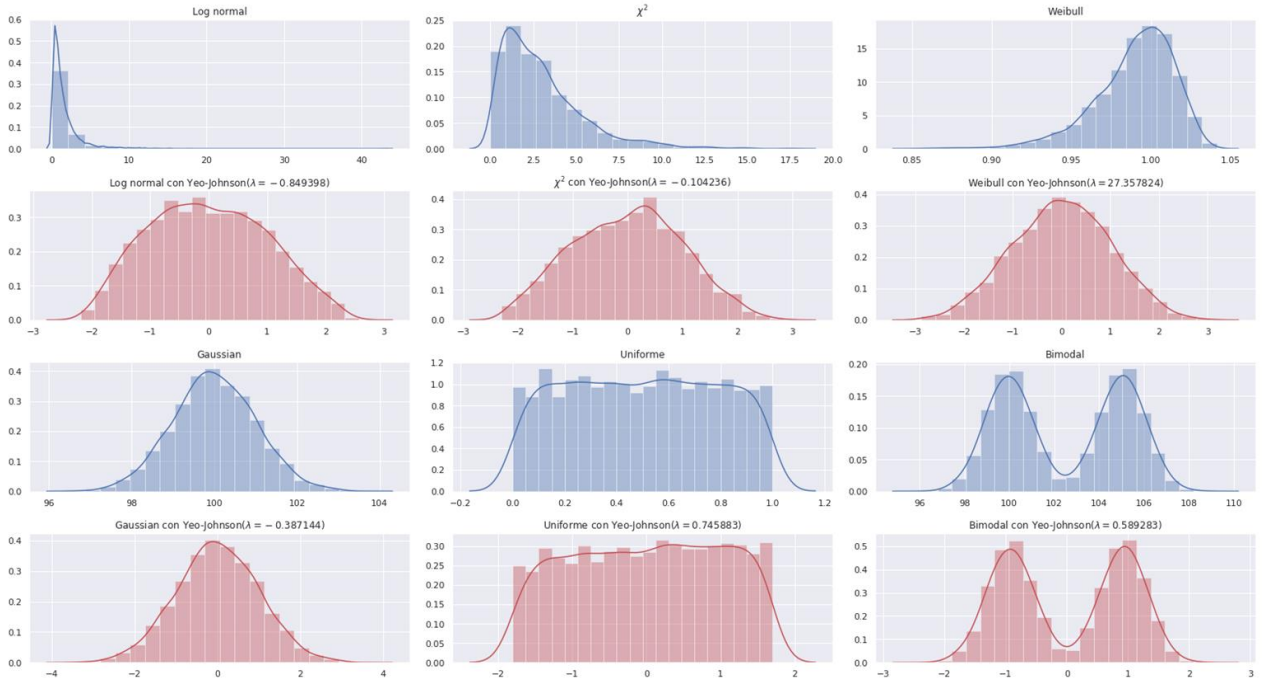
Ejemplos



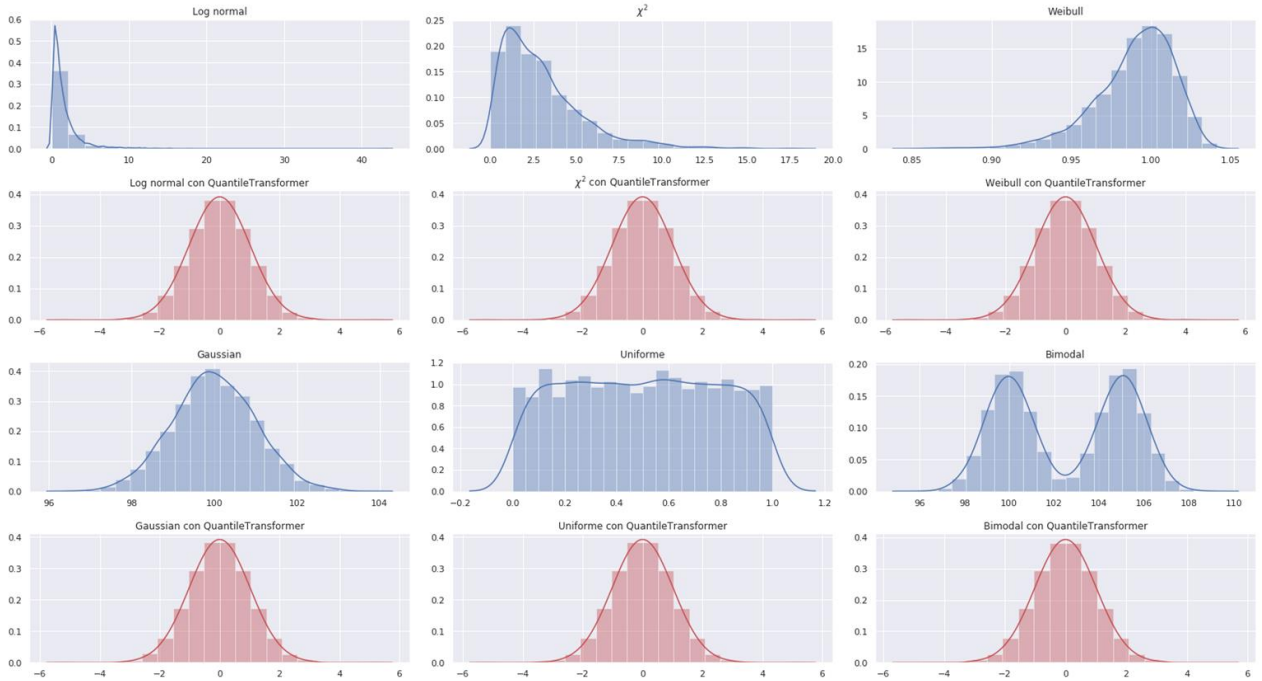
Ejemplos. Box-Cox



Ejemplos. Yeo-Johnson



Ejemplos. QuantileTransformer



Ejemplos en jupyter

Clase 4.5 - Preparación de datos - Transformación de variables.ipynb

Resumen

El objetivo de esta clase fue presentar algunas de las técnicas más utilizadas para la preparación de datasets y cómo combinarlas para construir cadenas de procesamiento en SKLearn.

En las siguientes clases, se estudiarán otros aspecto de la **preparación de datos**:

- La **selección de variables** de entrada: ¿Qué variables son más relevantes para el modelo de AA?
- **Ingeniería de variables** (como generar variables que aporten mayor información, a partir de las existentes).
- **Reducción de dimensiones**. Creación de un nuevo espacio de variables entrada, a partir de proyecciones compactas.

Bibliografía y referencias

- *"Python Feature Engineering Cookbook" .**Soledad Galli.** Packt (2020).*
- *"Applied Predictive Modeling". **Max Kuhn; Kjell Johnson**. Springer (2016).*
- *"Feature Engineering and Selection". **Max Kuhn; Kjell Johnson**. CRC Press (2016).*
- *"Feature Selection for Data and Pattern Recognition". **Urszula Stańczyk; Lakhmi C. Jain**. Springer (2014).*
- *"Feature Engineering for Machine Learning". **Alice Zheng; Amanda Casari**. O'Reilly (2018).*
- *"The Art of Feature Engineering". **Pablo Doboue**. Cambridge University Press (2020).*
- *"Feature Engineering IFT6758 - Data Science"* Université de Montréal.

DUDAS?

ENCUESTA