ANÁLISIS DE DATOS

Clase 4. Taller de preparación de datos (cont)

Temario

1. Desbalance de datos
2. Discretización.
3. Tratamiento de valores extremos (outliers).
4. Feature scaling.
5. Transformación de variables.

The Problem

- ✓ Explosive availability of raw data
- ✓ Well-developed algorithms for data analysis

Requirement?

- ***Balanced*** *distribution* of data
- ***Equal*** *costs* of misclassification

What about data in reality?

Imbalance is Everywhere

SOURCE: H. HE AND E. A. GARCIA, “LEARNING FROM IMBALANCED DATA,” IEEE TRANS. KNOWLEDGE AND DATA ENGINEERING, VOL. 21, ISSUE 9, PP. 1263-1284, 2009

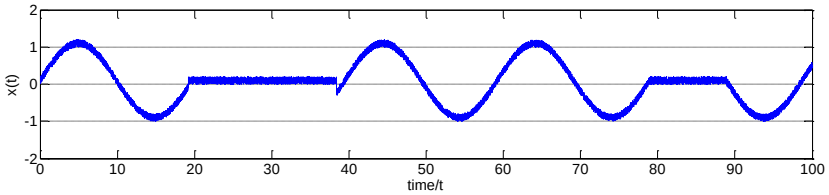
The Nature of Imbalance Learning

Mammography Data Set:
An example of ***between-class*** *imbalance*

	Negative/healthy	Positive/cancerous
Number of cases	10,923	260
Category	Majority	Minority
Imbalanced accuracy	≈ 100%	0-10 %

Imbalance can be on the order of 100 : 1 up to 10,000 : 1!

Intrinsic and ***extrinsic*** imbalance



- **Intrinsic:**
  - Imbalance due to the nature of the dataspace
- **Extrinsic:**
  - Imbalance due to time, storage, and other factors
  - **Example:**
    - **Data transmission over a specific interval of time with interruption**

SOURCE: H. HE AND E. A. GARCIA, "LEARNING FROM IMBALANCED DATA," IEEE TRANS. KNOWLEDGE AND DATA ENGINEERING, VOL. 21, ISSUE 9, PP. 1263-1284, 2009

The Nature of Imbalance Learning

Data Complexity

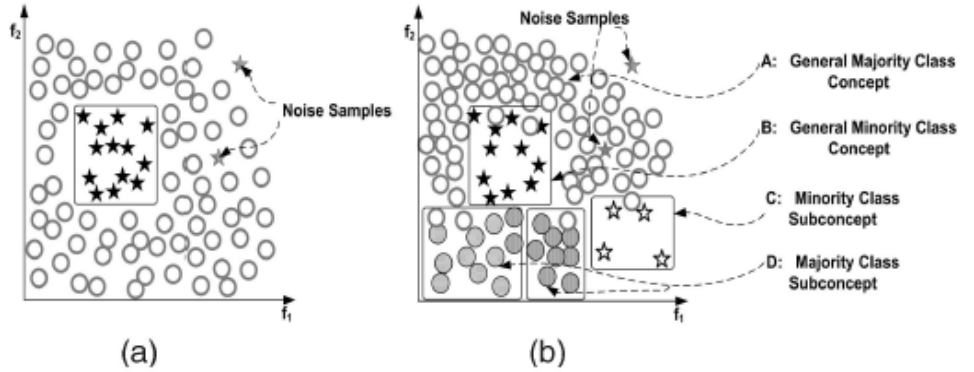


Fig. 2. (a) A data set with a between-class imbalance. (b) A high-complexity data set with both between-class and within-class imbalances, multiple concepts, overlapping, noise, and lack of representative data.

Relative imbalance and ***absolute rarity***

$$\boldsymbol{Q}\textbf{:}\ 1{,}000{,}000 : 1{,}000 = 1{,}000 : 1$$?

- The minority class may be outnumbered, but not necessarily rare
- Therefore they can be accurately learned with little disturbance

Imbalanced data with *small sample size*

Data with high dimensionality and small sample size

Face recognition, gene expression

Challenges with small sample size:

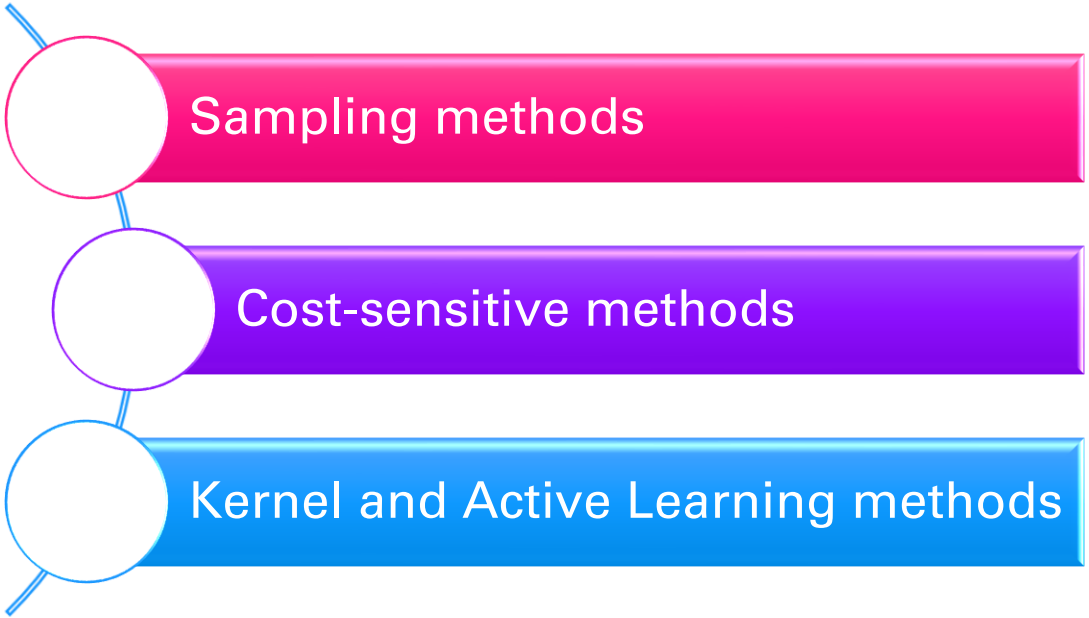
Embedded absolute rarity and within-class imbalances

Failure of generalizing inductive rules by learning algorithms

- Difficulty in forming good classification decision boundary over *more* features but *less* samples
- Risk of overfitting

THE SOLUTIONS TO IMBALANCED LEARNING PROBLEM

Solutions to imbalanced learning



Sampling methods

If data is Imbalanced...

Modify data distribution

Create balanced dataset

Create balance though sampling

Sampling methods

Random Sampling

S: training data set; S_{min}: set of minority class samples, S_{maj}: set of majority class samples; ***E***: generated samples

Random oversampling

- Expand the minority
- $|S'_{min}| \leftarrow |S_{min}| + |E|$
- $|S'| \leftarrow |S_{min}| + |S_{maj}| + |E|$
- Overfitting due to multiple "tied" instances

Random undersampling

- Shrink the majority
- $|S'_{maj}| \leftarrow |S_{maj}| - |E|$
- $|S'| \leftarrow |S_{min}| + |S_{maj}| - |E|$
- Loss of important concepts

SOURCE: H. HE AND E. A. GARCIA, "LEARNING FROM IMBALANCED DATA," IEEE TRANS. KNOWLEDGE AND DATA ENGINEERING, VOL. 21, ISSUE 9, PP. 1263-1284, 2009

Sampling methods

Informed Undersampling

- *EasyEnsemble*
  - ***Unsupervised**: use random subsets of the majority class to create balance and form multiple classifiers*
- *BalanceCascade*
  - ***Supervised**: iteratively create balance and pull out redundant samples in majority class to form a final classifier*

1. Generate $E \subset S_{maj}$ $(s.t. |E| = |S_{\min}|)$, and $N = \{E \cup S_{\min}\}$
2. Induce $H(n)$
3. Identify N^*_{maj} as samples from N that are correctly classified
4. Remove N^*_{maj} from S_{maj}
5. Repeat (1) and induce $H(n+1)$ until stopping criteria is met

Sampling methods

Informed Undersampling

- ***Undersampling using K-nearest neighbor (KNN) classifier***
  - NearMiss-1, NearMiss-2, NearMiss-3, and the “most distant” method
  - NearMiss-2 provides competitive results for imbalanced learning
- ***One-sided selection (OSS)***
  - Selects representative subset E from the majority class
  - Combine with the minority class $N = \{E \cup S_{\min}\}$
  - Refine N with data cleaning techniques

Sampling methods

Synthetic Sampling with Data Generation

- Synthetic minority oversampling technique (SMOTE)
  - Creates artificial minority class data using feature space similarities
  - For $\forall x_i \in S_{\min}$
    1. Randomly choose one of the k nearest neighbor $\hat{x}_i$;
    2. Create a new sample $x_{new} = x_i + (\hat{x}_i - x_i) \times \delta$, where δ is a uniformly distributed random variable.

Sampling methods

Synthetic Sampling with Data Generation

- Synthetic minority oversampling technique (SMOTE)

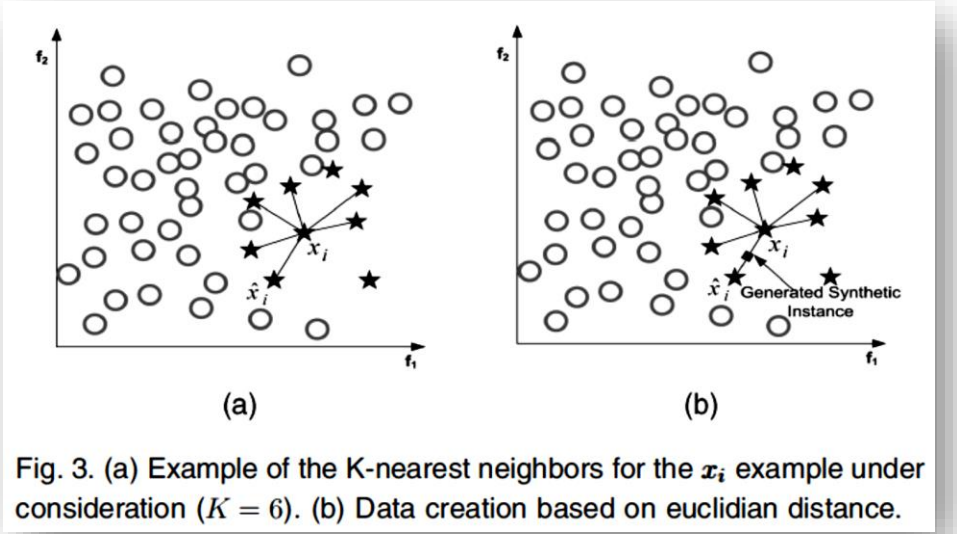


Fig. 3. (a) Example of the K-nearest neighbors for the x_i example under consideration ($K = 6$). (b) Data creation based on euclidian distance.

Sampling methods

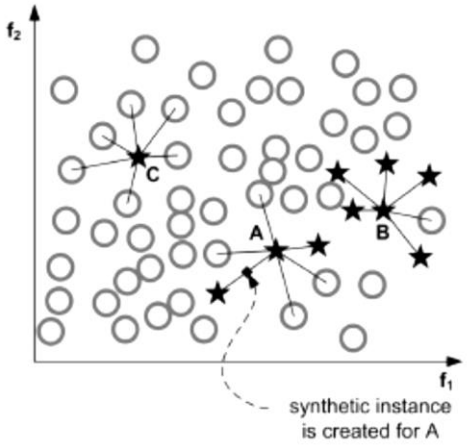
Adaptive Synthetic Sampling

- Overcomes over generalization in SMOTE algorithm
  - Border-line-SMOTE:
    1. Determine the set of m-nearest neighbors for each $x_i \in S_{min}$, call it $S_{i:m-NN}$
    2. Identify the number of nearest neighbors in majority class, i.e., $|S_{i:m-NN} \cap S_{maj}|$
    3. Select x_i that satisfies: $\frac{m}{2} \leq |S_{i:m-NN} \cap S_{maj}| < m$

Sampling methods

Adaptive Synthetic Sampling

- Overcomes over generalization in SMOTE algorithm
  - Border-line-SMOTE



Consider 6-nearest neighbor: m=6

For A: number of minority instance: 2
number of majority instance: 4 → "DANGER" instance

For B: number of minority instance: 5
number of majority instance: 1 → "SAFE" instance

For C: number of minority instance: 6
number of majority instance: 0 → "NOISE" instance

Fig. 4. Data creation based on Borderline instance.

Sampling methods

Adaptive Synthetic Sampling

- Overcomes over generalization in SMOTE algorithm
  - ADASYN
    1. Calculate number of synthetic samples $G = \left(\left|S_{maj}\right| - |S_{\min}|\right) \times \beta$
    2. for each $x_i \in S_{\min}$, find k-nearest neighbors and calculate ratio $\Gamma_i = \frac{\Delta_i / K}{Z}, i = 1, \ldots, |S_{min}|$ as a distribution function;
    3. Identify the number of synthetic samples to be generated for x_i by $g_i = \Gamma_i \times G$
    4. Generate x_{new} using SMOTE algorithm: $x_{new} = x_i + (\hat{x}_i - x_i) \times \delta$

Sampling with Data Cleaning

- Tomek links
  1. Given a pair (x_i, x_j) where $x_i \in S_{\min}$, $x_j \in S_{maj}$, and the distance between them as $d(x_i, x_j)$
  2. If there is no instance $x_k \; s.t. \; d(x_i, x_k) <. d(x_i, x_j)$ or $d(x_j, x_k) < d(x_i, x_j)$, then (x_i, x_j) is called a Tomek link
- Clean up unwanted inter-class overlapping after synthetic sampling
- Examples:
  - OSS, condensed nearest neighbor and Tomek links (CNN + Tomek links), neighborhood cleaning rule (NCL) based on edited nearest neighbor (ENN), SMOTE+ENN, and SMOTE+Tomek

Sampling methods

Sampling with Data Cleaning

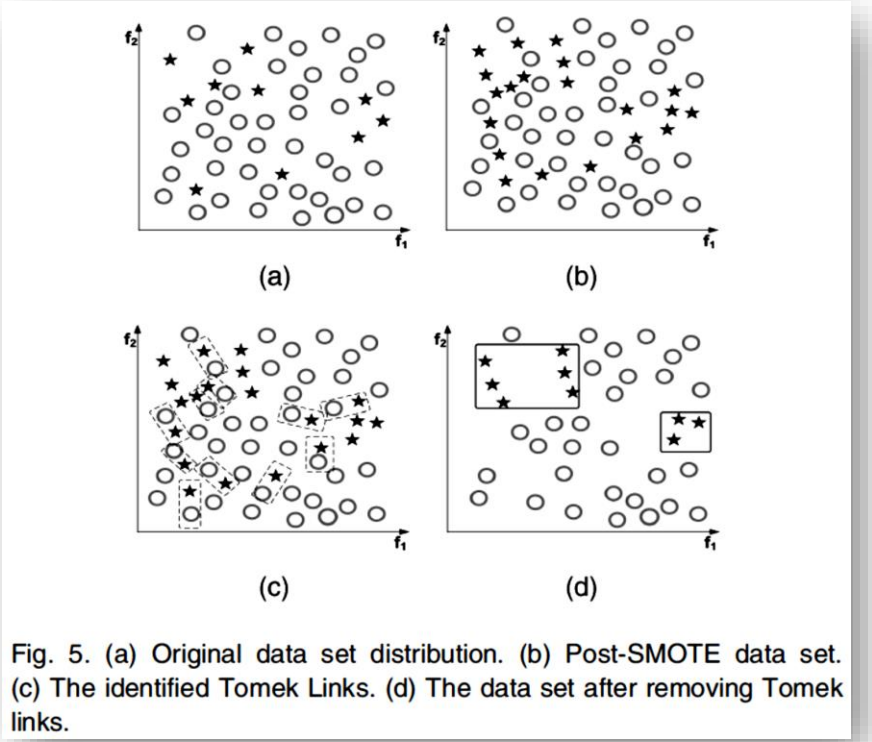


Fig. 5. (a) Original data set distribution. (b) Post-SMOTE data set. (c) The identified Tomek Links. (d) The data set after removing Tomek links.

Sampling methods

Cluster-based oversampling (CBO) method

1. For the majority class S_{maj} with m_{maj} clusters
   I. Oversample each cluster $C_{maj:j} \subset S_{maj}, j = 1, \dots, m_{maj}$ except the largest $C_{maj:\max}$, so that for $\forall j, |C_{maj:j}| = |C_{maj:\max}|$
   II. Calculate the number of majority class examples after oversampling as N_{CBO}
2. For the minority class S_{min} with m_{min} clusters
   I. Oversample each cluster $C_{min:i} \subset S_{min}, i = 1, \dots, m_{\min}$ to be of the same size $N_{CBO}/m_{\min}$, so that for $\forall i, |C_{min:i}| = N_{CBO}/m_{\min}$

Sampling methods

CBO Method

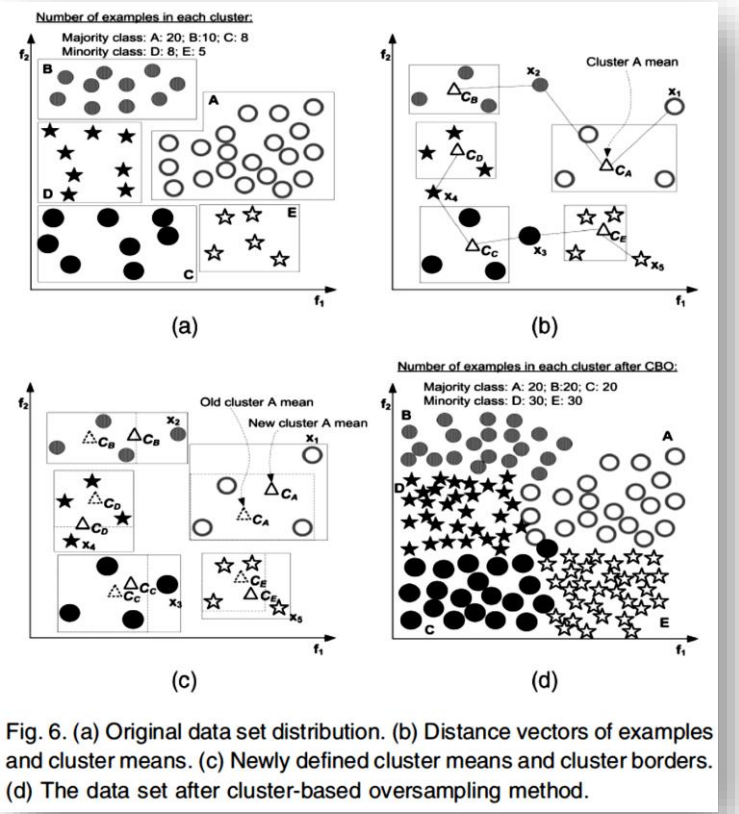


Fig. 6. (a) Original data set distribution. (b) Distance vectors of examples and cluster means. (c) Newly defined cluster means and cluster borders. (d) The data set after cluster-based oversampling method.

Sampling methods

Integration of Sampling and Boosting

1. SMOTEBoost
   - SMOTE + Adaboost.M2
   - Introduces synthetic sampling at each boosting iteration
2. DataBoost-IM
   - AdaBoost.M1
   - Generate synthetic date of hard-to-learn samples for both majority and minority classes (usually $|E_{maj}| < |E_{min}|$)
3. JOUS-Boost

Integration of Sampling and Boosting

DataBoost-IM

1. Collect the set E of top misclassified samples (hard-to-learn samples) for both classes with subsets $E_{maj} \subset E$ and $E_{min} \subset E$
2. Identify M_L seeds from E_{maj} and M_S seeds from E_{min}, where

   $M_L = \min(\frac{|S_{maj}|}{|S_{min}|}, |E_{maj}|)$ and $M_S = \min(\frac{|S_{maj}| \times M_L}{|S_{\min}|}, |E_{\min}|)$
3. Generate synthetic set E_{syn} with subsets for both classes:

   $E_{s\min} \subset E_{syn}$ and $E_{smaj} \subset E_{syn}$

   $s.t. |E_{smin}| = M_S \times |S_{\min}|$ and $|E_{smaj}| = M_L \times |S_{maj}|$

Cost-Sensitive Methods

SOURCE: H. HE AND E. A. GARCIA, "LEARNING FROM IMBALANCED DATA," IEEE TRANS. KNOWLEDGE AND DATA ENGINEERING, VOL. 21, ISSUE 9, PP. 1263-1284, 2009

Instead of modifying data…

Considering the cost of misclassification

Utilize cost-sensitive methods for imbalanced learning

Cost-Sensitive methods

Cost-Sensitive Learning Framework

- Define the cost of misclassifying a majority to a minority as $C(Min, Maj)$
- Typically $C(Maj, Min) > C(Min, Maj)$
- Minimize the overall cost - usually the *Bayes conditional risk* - on the training data set

$$R(i|x) = \sum_j P(j|x)C(i,j)$$

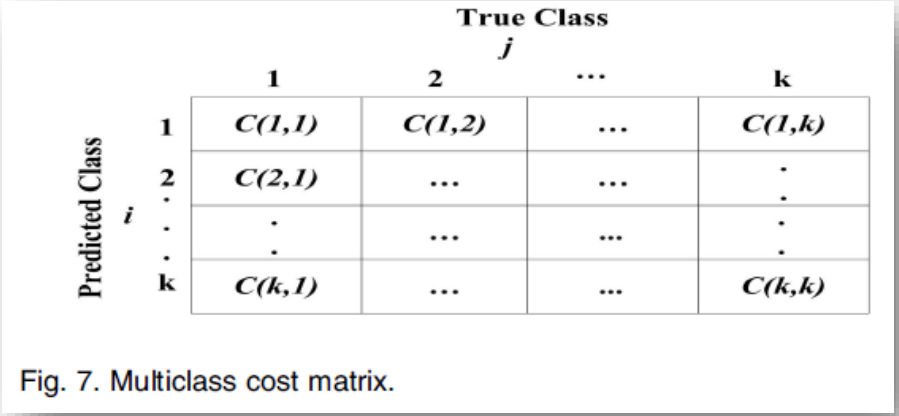


Fig. 7. Multiclass cost matrix.

Cost-Sensitive methods

Cost-Sensitive Dataspace Weighting with Adaptive Boosting

- Iteratively update the distribution function D_t of the training data according to error of current hypothesis h_t and cost factor C_i
  - Weight updating parameter $\alpha_t = \frac{1}{2}\ln(\frac{1-\varepsilon_t}{\varepsilon_t})$
  - Error of hypothesis h_t: $\varepsilon_t = \sum_{i:h_t(x_i)\neq y_i} D_t(i)$

Cost-Sensitive methods

Cost-Sensitive Dataspace Weighting with Adaptive Boosting

- Given $D_t, h_t, C_i, \alpha_t,$ and ε_t

1. AdaC1: $D_{t+1}(i) = \frac{D_t(i)\ exp(-\alpha_t C_i h_t(x_i) y_i)}{Z_t}$
2. AdaC2: $D_{t+1}(i) = \frac{C_i D_t(i)\ exp(-\alpha_t h_t(x_i) y_i)}{Z_t}$
3. AdaC3: $D_{t+1}(i) = \frac{C_i D_t(i)\ exp(-\alpha_t C_i h_t(x_i) y_i)}{Z_t}$
4. AdaCost: $D_{t+1}(i) = \frac{D_t(i)\ exp(-\alpha_t h_t(x_i) y_i \beta_i)}{Z_t}$,

   $\beta_i = \beta(sign(y_i, h_t(x_i)), C_i)$

Cost-Sensitive methods

Cost-Sensitive Decision Trees

1. Cost-sensitive adjustments for the decision threshold
   - The final decision threshold shall yield the most dominant point on the ROC curve
2. Cost-sensitive considerations for split criteria
   - The impurity function shall be insensitive to unequal costs
3. Cost-sensitive pruning schemes
   - The probability estimate at each node needs improvement to reduce removal of leaves describing the minority concept
   - Laplace smoothing method and Laplace pruning techniques

Cost-Sensitive methods

Cost-Sensitive Neural Network

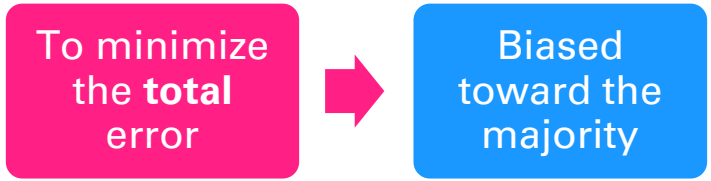
Four ways of applying cost sensitivity in neural networks

Modifying probability estimate of outputs	Altering outputs directly	Modify learning rate	Replacing error-minimizing function
• *Applied only at testing stage* • *Maintain original neural networks*	• *Bias neural networks during training to focus on expensive class*	• *Set η higher for costly examples and lower for low-cost examples*	• *Use expected cost minimization function instead*

Kernel-Based Methods

Kernel-based learning framework

- Based on statistical learning and Vapnik-Chervonenkis (VC) dimensions
- Problems with Kernel-based support vector machines (SVMs)
  1. Support vectors from the minority concept may contribute less to the final hypothesis
  2. Optimal hyperplane is also biased toward the majority class



Kernel-Based Methods

Integration of Kernel Methods with Sampling Methods

1. SMOTE with Different Costs (SDCs) method
2. Ensembles of over/under-sampled SVMs
3. SVM with asymmetric misclassification cost
4. Granular Support Vector Machines—Repetitive Undersampling (GSVM-RU) algorithm

Kernel-Based Methods

Kernel Modification Methods

1. Kernel classifier construction
   - Orthogonal forward selection (OFS) and Regularized orthogonal weighted least squares (ROWLSs) estimator
2. SVM class boundary adjustment
   - Boundary movement (BM), biased penalties (BP), class-boundary alignment(CBA), kernel-boundary alignment (KBA)
3. Integrated approach
   - Total margin-based adaptive fuzzy SVM (TAF-SVM)
4. K-category proximal SVM (PSVM) with Newton refinement
5. Support cluster machines (SCMs), Kernel neural gas (KNG), P2PKNNC algorithm, hybrid kernel machine ensemble (HKME) algorithm, Adaboost relevance vector machine (RVM), …

Active Learning Methods

Additional methods

- SVM-based active learning
- Active learning with sampling techniques
  - Undersampling and oversampling with active learning for the word sense disambiguation (WSD) imbalanced learning
  - New stopping mechanisms based on maximum confidence and minimal error
  - Simple active learning heuristic (SALH) approach

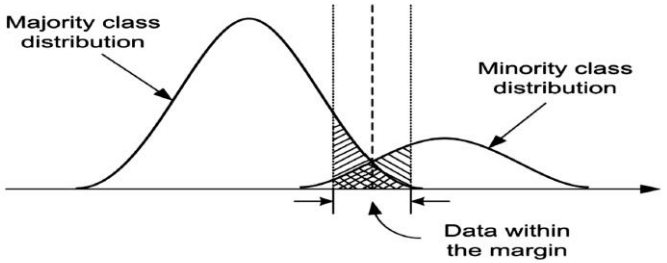


Fig. 8. Data imbalance ratio within and outside the margin [98].

Active Learning Methods

Additional methods

One-class learning/novelty detection methods

Mahalanobis-Taguchi System

Rank metrics and multitask learning

- Combination of imbalanced data and the small sample size problem

Multiclass imbalanced learning

- AdaC2.M1
- Rescaling approach for multiclass cost-sensitive neural networks
- the ensemble knowledge for imbalance sample sets (eKISS) method

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.

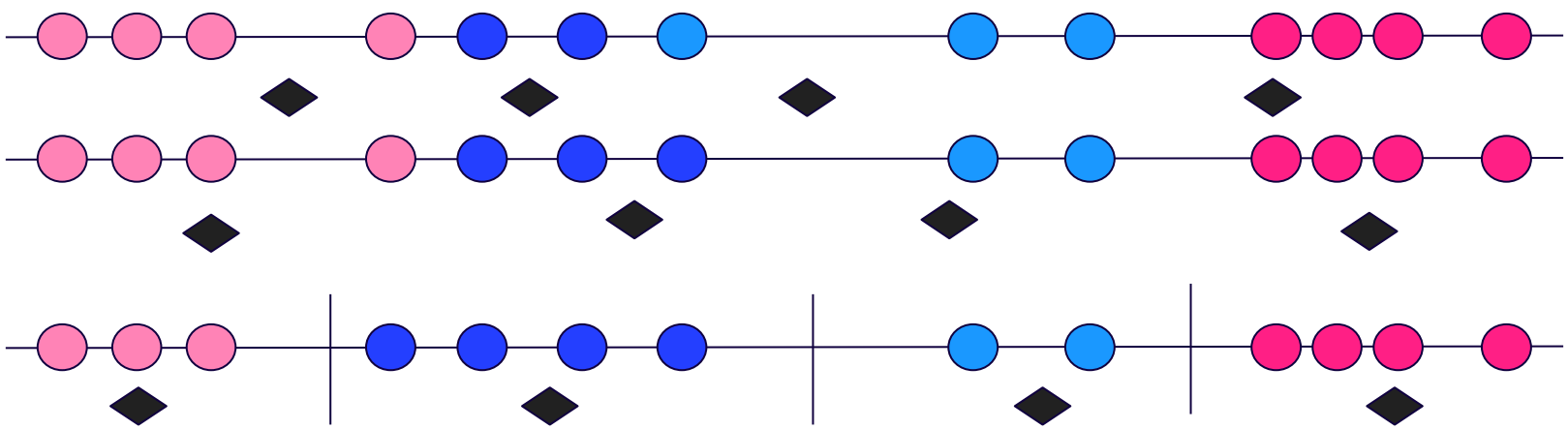
Discretización por intervalos de frecuencia fija.

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.

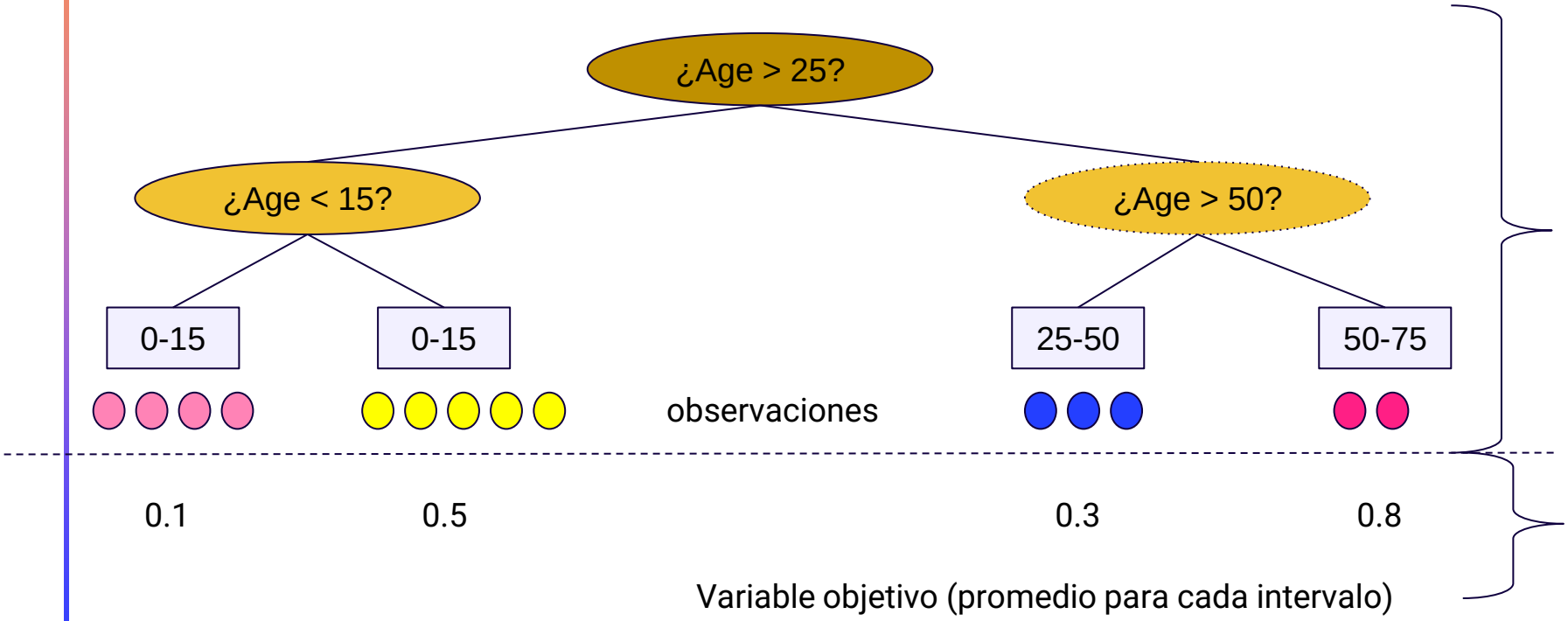
Discretización + codificación

- 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

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

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).

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

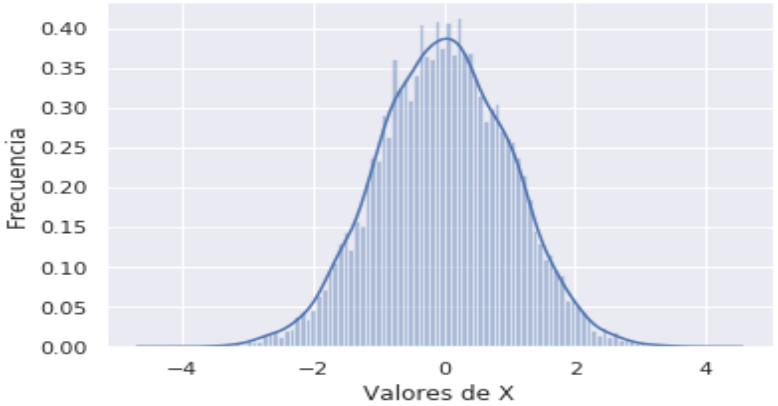
Escalado a mínimo-máximo. Efecto.

- 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).

TRANSFORMACIÓN DE VARIABLES

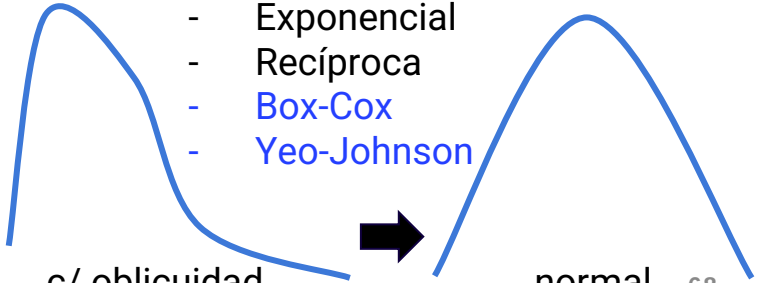
Distribución normal en modelos lineales

- 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

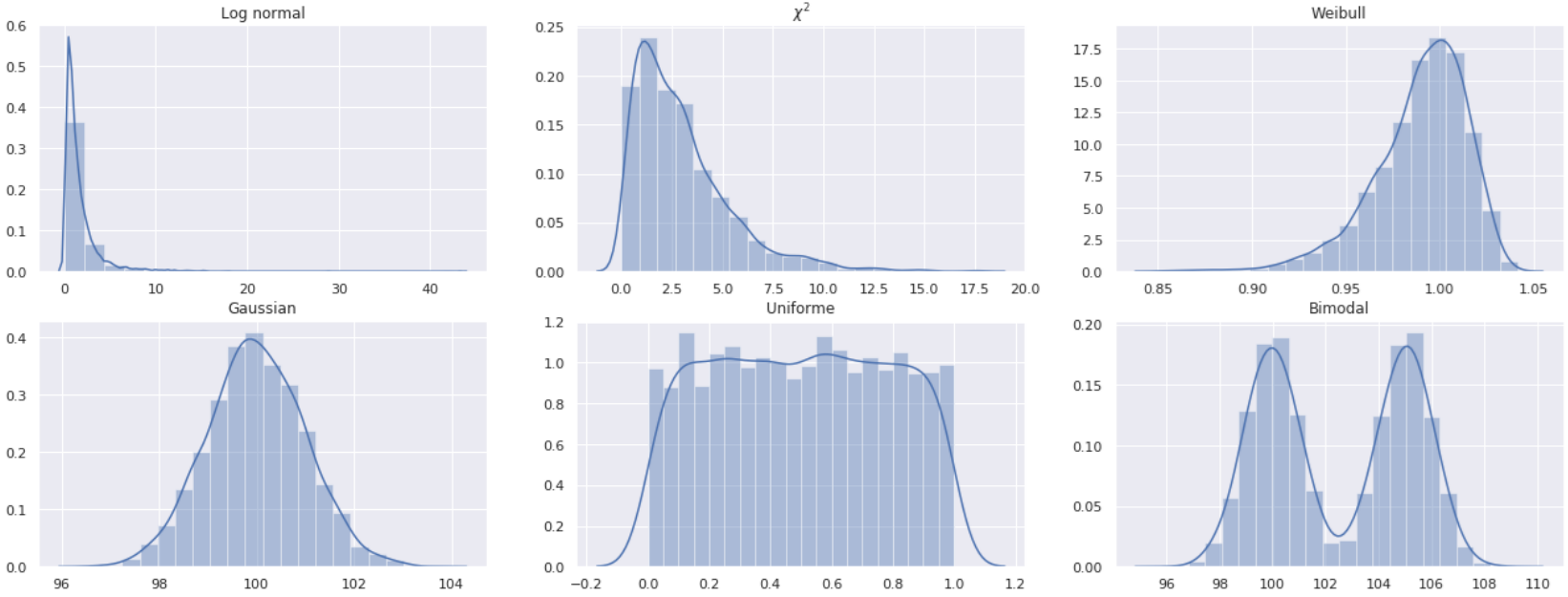
Source: Box and Cox (1964). Where, 'Y' is the transformation of t
Note that for Lambda = 0, the transformation is NOT Y^0 (because t
every

Transformación de Yeo-Johnson

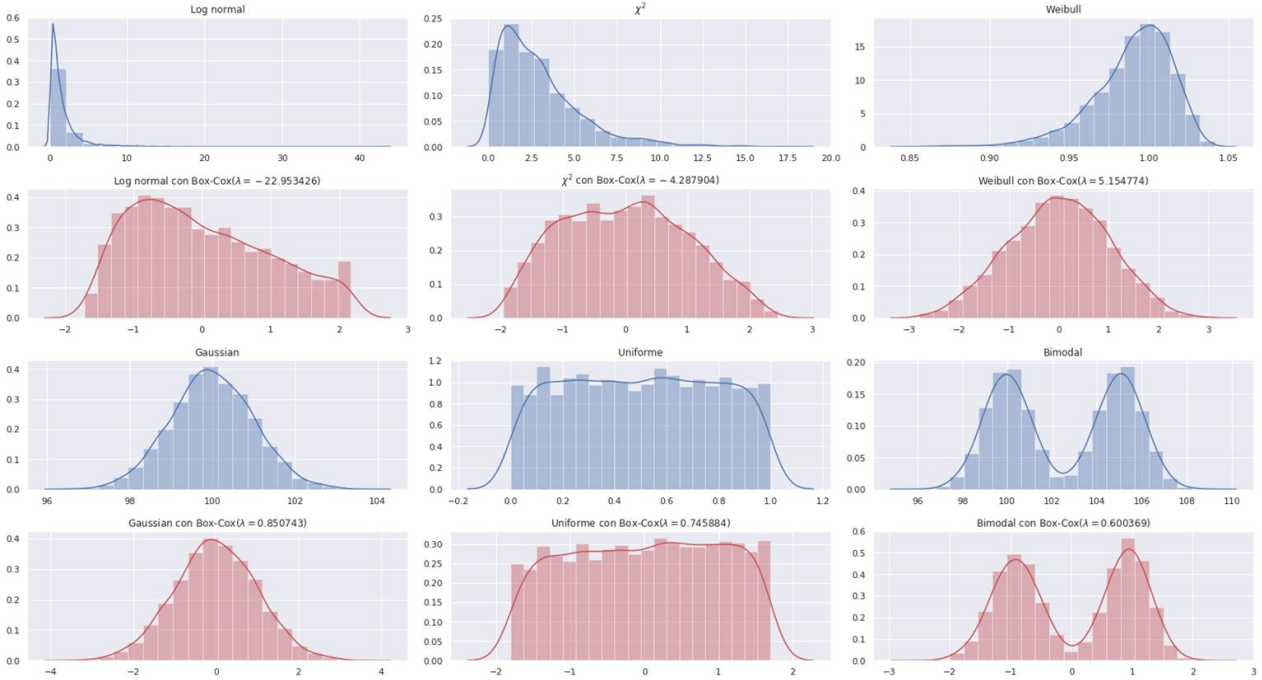
- Junto con Box-Cox, es otra de las transformaciones más utilizadas.
- Admite que X tome valores negativos.

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

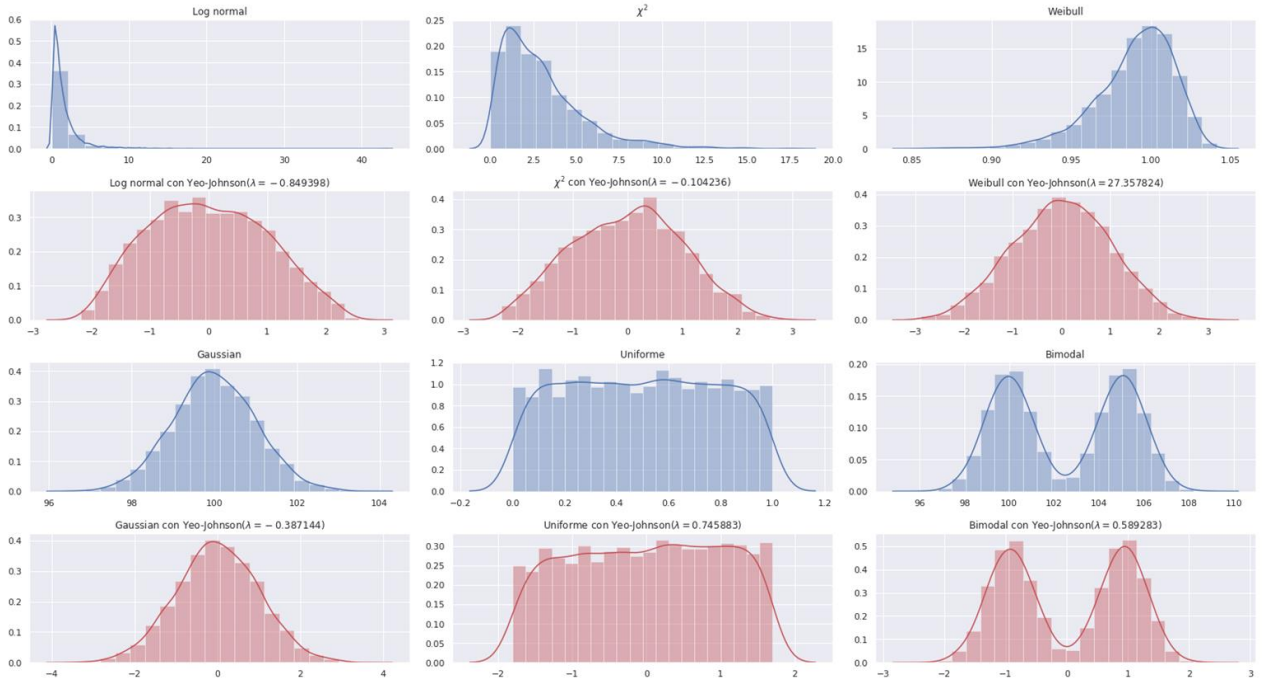
Ejemplos



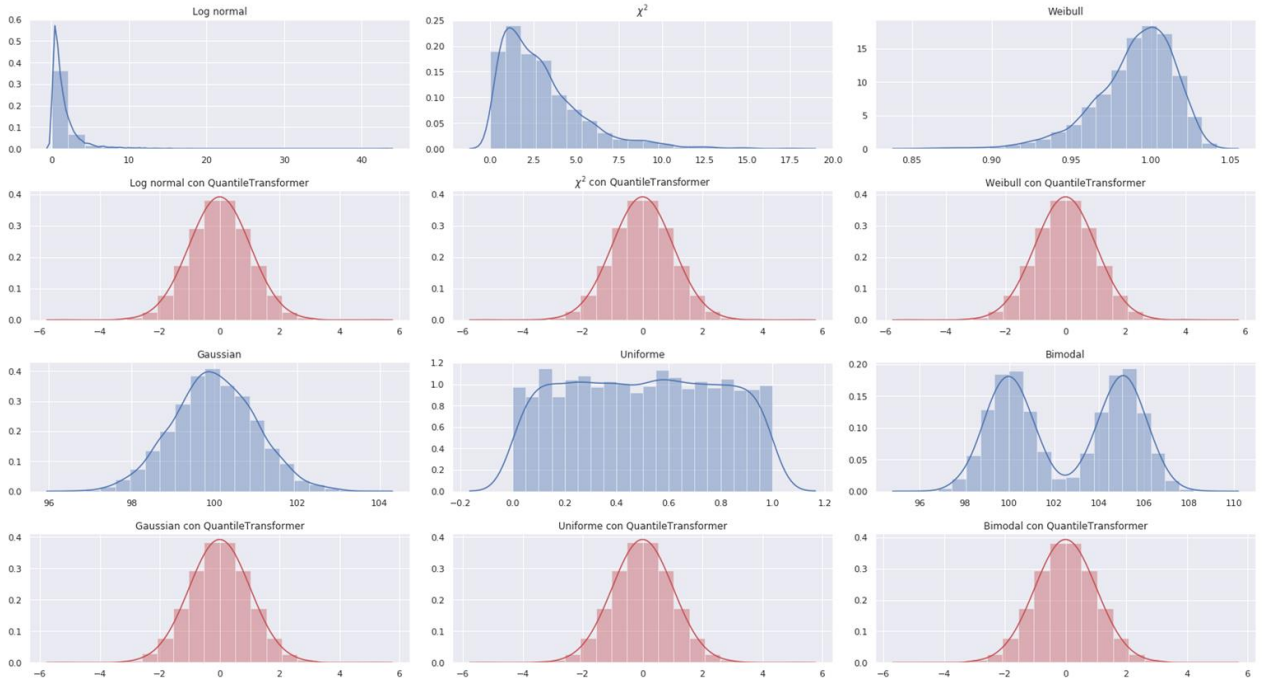
Ejemplos. Box-Cox



Ejemplos. Yeo-Johnson



Ejemplos. QuantileTransformer



Ejemplos en jupyter

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

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