ANÁLISIS DE DATOS

Clase 4. Taller de preparación de datos (cont)

Temario

1. Desbalance de datos (Cont)
2. Imputación de datos faltantes. (cont)
3. Codificación de variables categóricas.
4. Discretización.
5. Tratamiento de valores extremos (outliers).
6. Feature scaling.
7. Cadenas de procesamiento.
8. 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

Growing interest

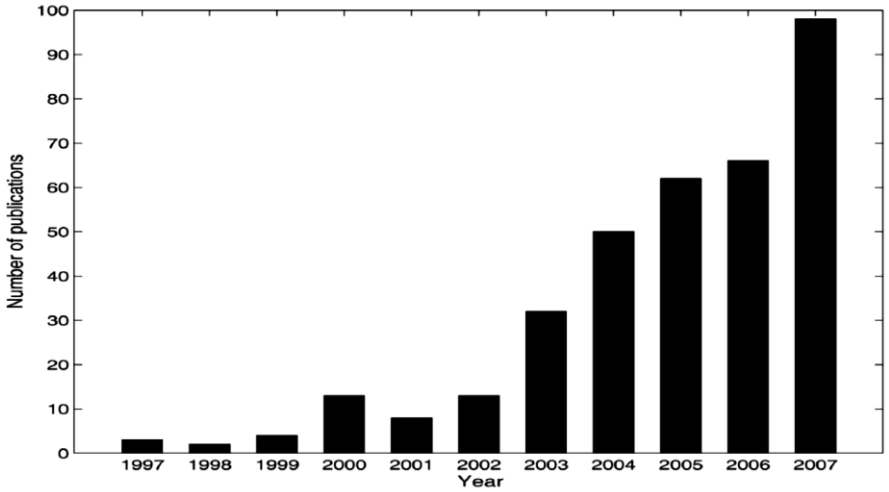


Fig. 1. Number of publications on imbalanced learning.

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

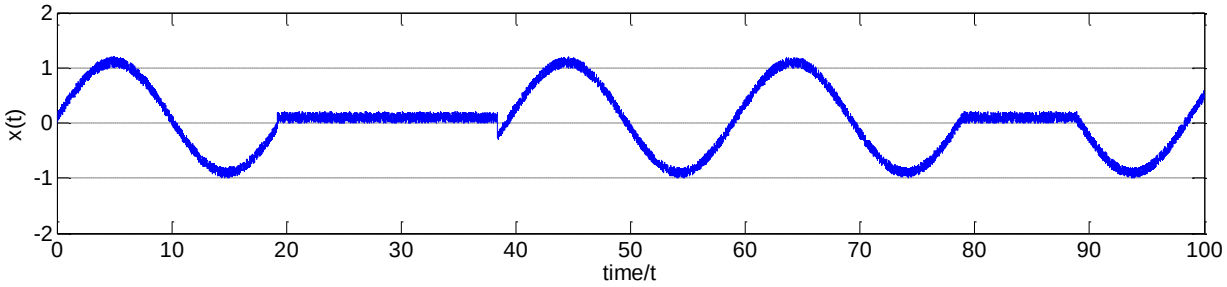
Intrinsic:

- 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

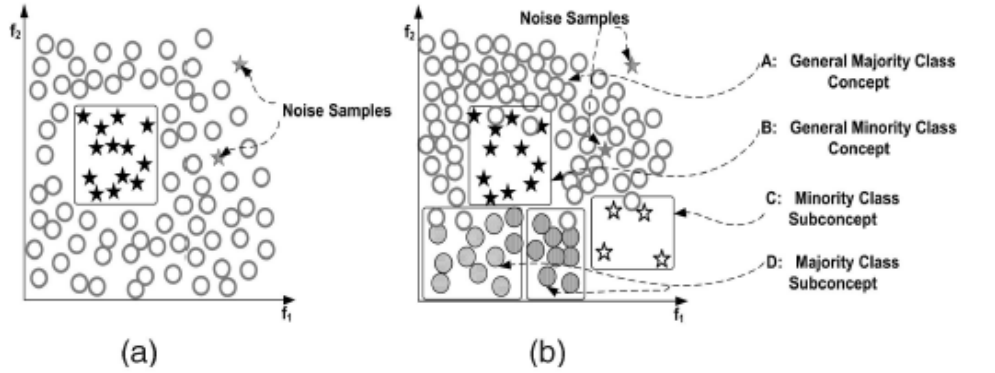


Fig. 2. (a) A data set with a between-class imbalance. (b) A high-complexity data set with both between-class and within-class imbalances, multiple concepts, overlapping, noise, and lack of representative data.

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

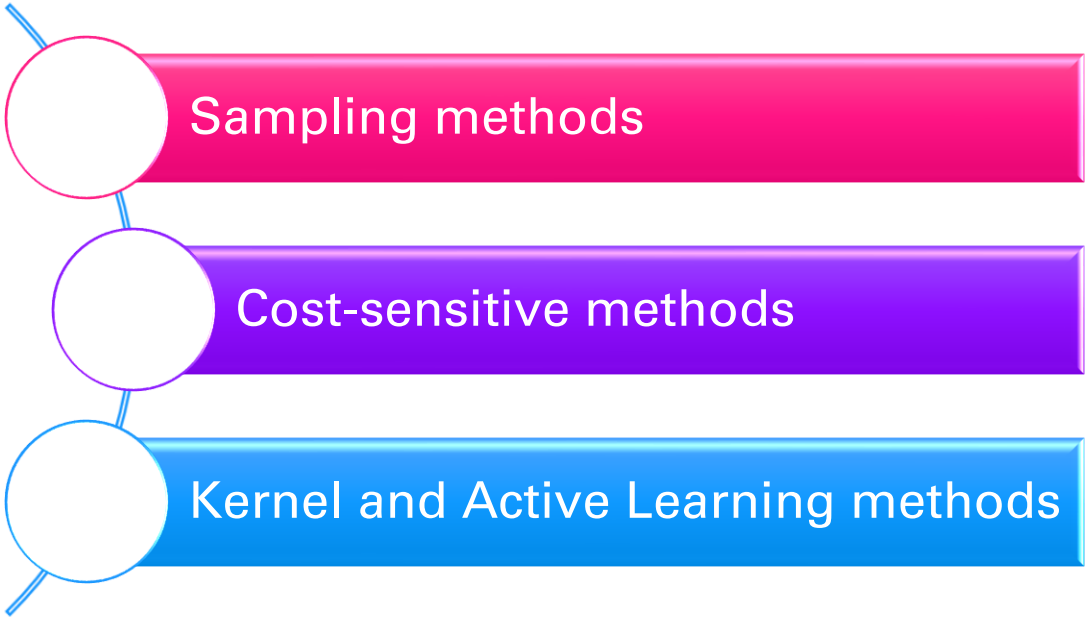
Imbalanced data with ***small sample size***

- Data with high dimensionality and small sample size
  - Face recognition, gene expression

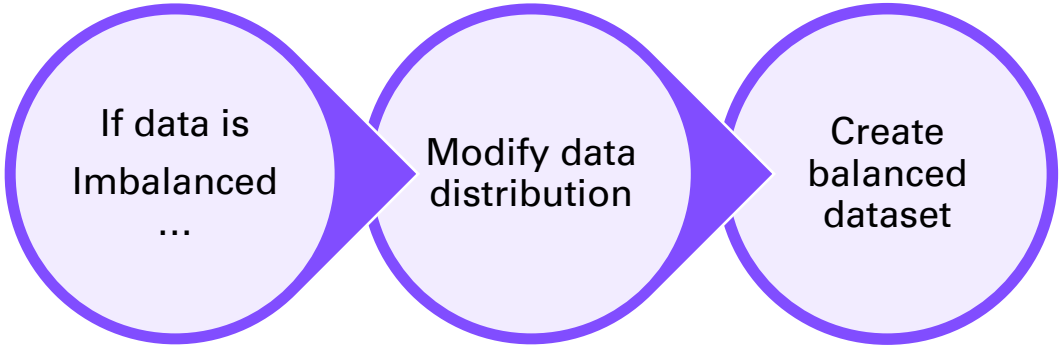
- Challenges with small sample size:
  1. Embedded absolute rarity and within-class imbalances
  2. Failure of generalizing inductive rules by learning algorithms
     - Difficulty in forming good classification decision boundary over *more* features but *less* samples
     - Risk of overfitting

THE SOLUTIONS TO IMBALANCED LEARNING PROBLEM

Solutions to imbalanced learning



Sampling methods



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

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)

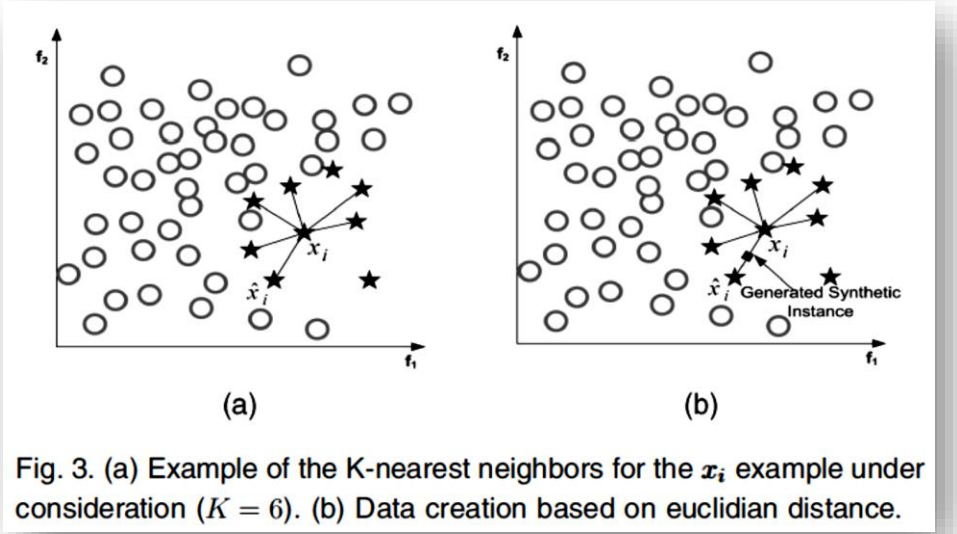


Fig. 3. (a) Example of the K-nearest neighbors for the x_i example under consideration ($K = 6$). (b) Data creation based on euclidian distance.

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

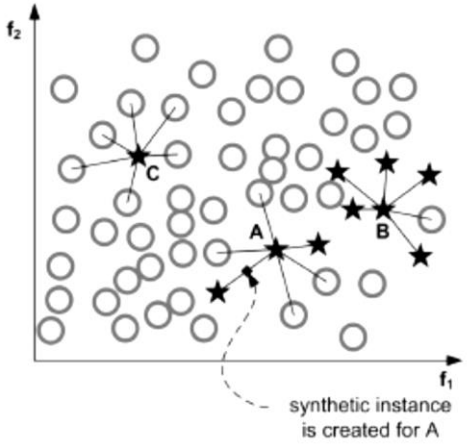


Fig. 4. Data creation based on Borderline instance.

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 forx_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

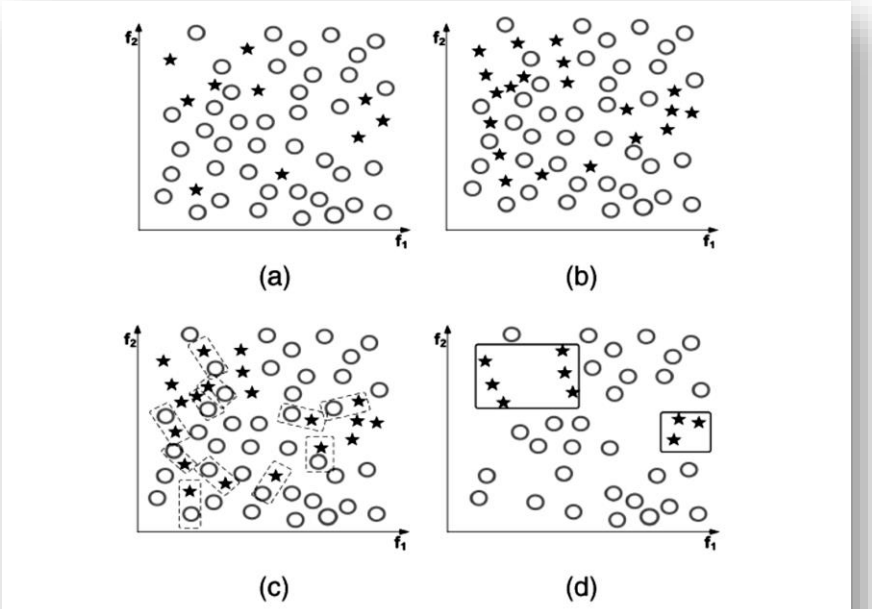


Fig. 5. (a) Original data set distribution. (b) Post-SMOTE data set. (c) The identified Tomek Links. (d) The data set after removing Tomek links.

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

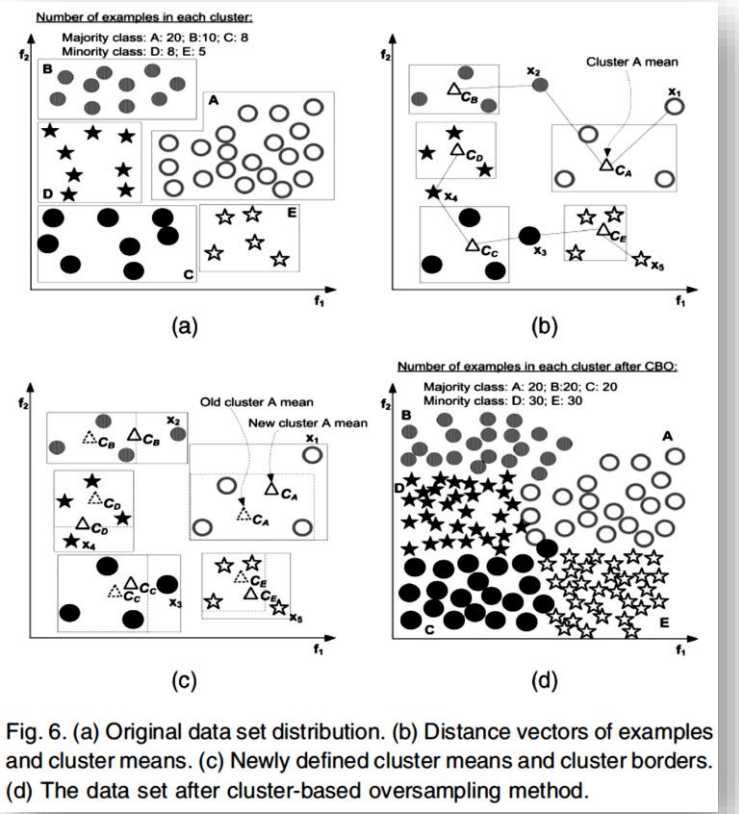


Fig. 6. (a) Original data set distribution. (b) Distance vectors of examples and cluster means. (c) Newly defined cluster means and cluster borders. (d) The data set after cluster-based oversampling method.

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

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

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

Predicted Class i \ True Class j	1	2	...	k
1	C(1,1)	C(1,2)	...	C(1,k)
2	C(2,1)	...	...	.
.	.	...	...	.
k	C(k,1)	...	...	C(k,k)

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 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

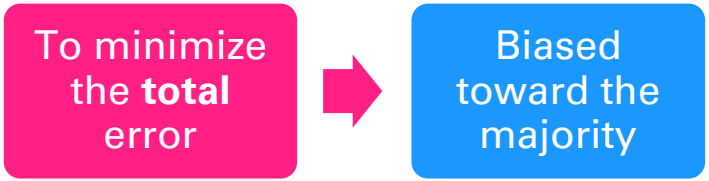
Cost-Sensitive Neural Network

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 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 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

- SVM-based active learning

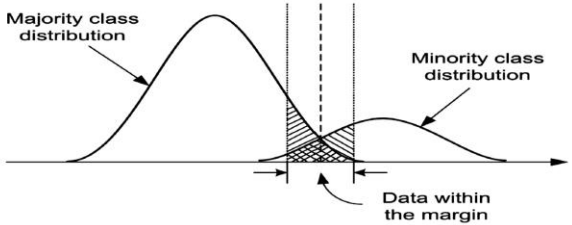


Fig. 8. Data imbalance ratio within and outside the margin [98].

- 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

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

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

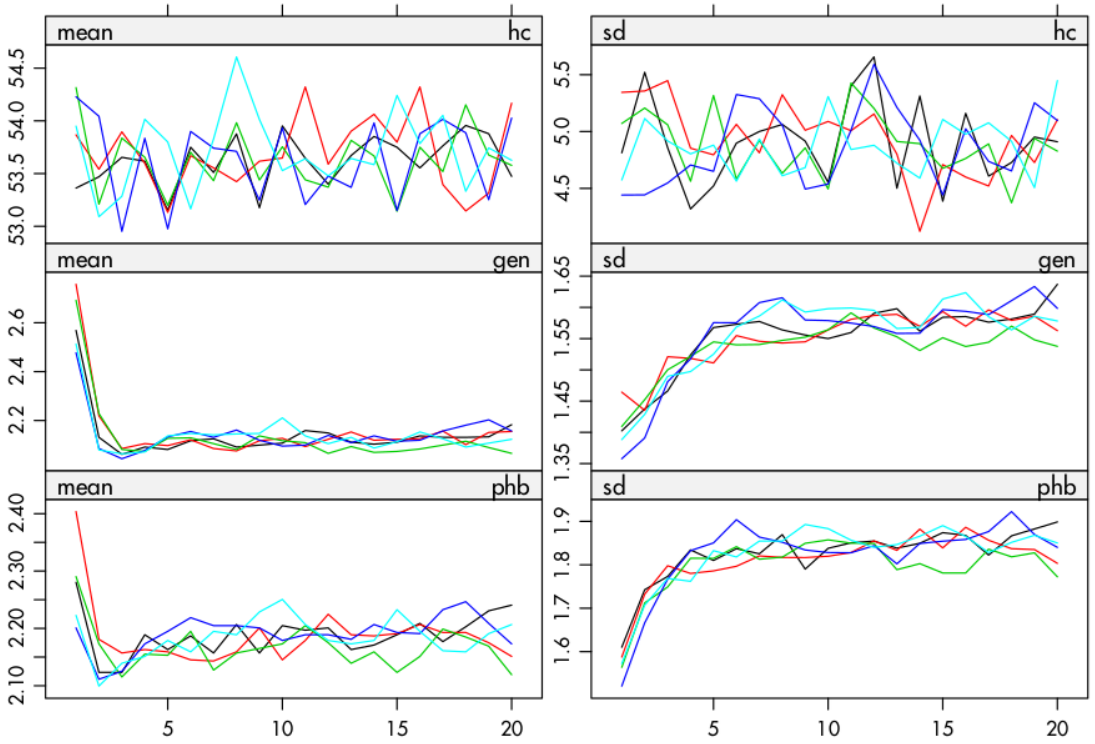
4.3. Assessing convergence

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.

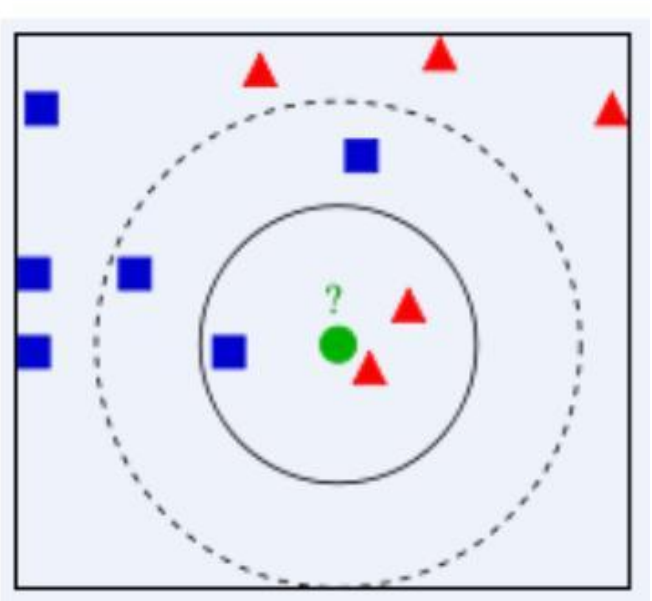
MICE. Algoritmo. Convergencia

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

```python
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

```python
(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

CODIFICACIÓN DE VARIABLES CATEGÓRICAS

Codificación de variables categóricas.

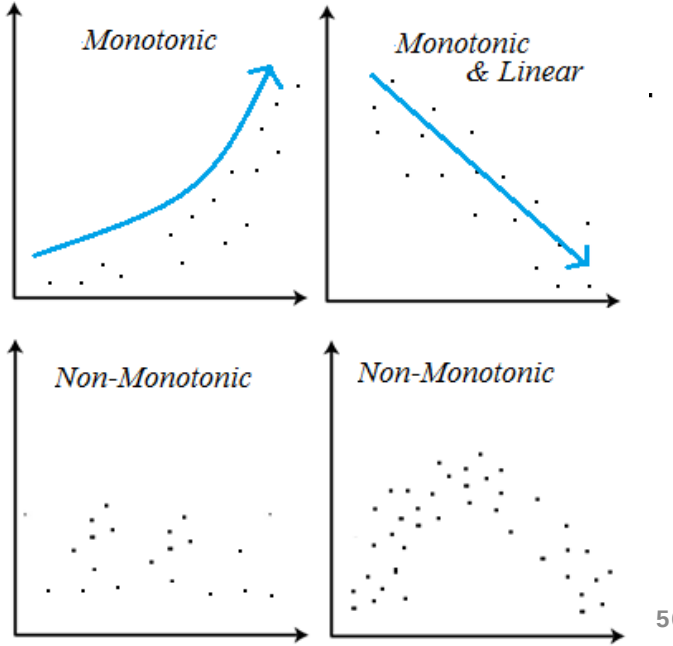
- Consiste en reemplazar una categoría (típicamente representada en forma de texto) por una representación numérica.
- El objetivo es disponer de variables que puedan ser utilizadas en los modelos de AA.

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.

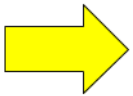
- Las relaciones monotónicas entre variables de entrada y de salida pueden mejorar el desempeño de los modelos lineales.
- En los modelos de árboles, puede traducirse en menor profundidad (sin comprometer su desempeño).
- A menudo, estas relaciones aparecen de manera natural en los datos. Por ejemplo: 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.

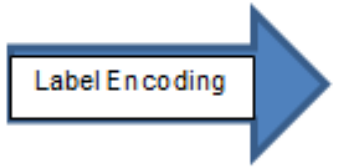
One Hot Encoding. Variante para top-n categorías.

- 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

DUDAS?

ENCUESTA