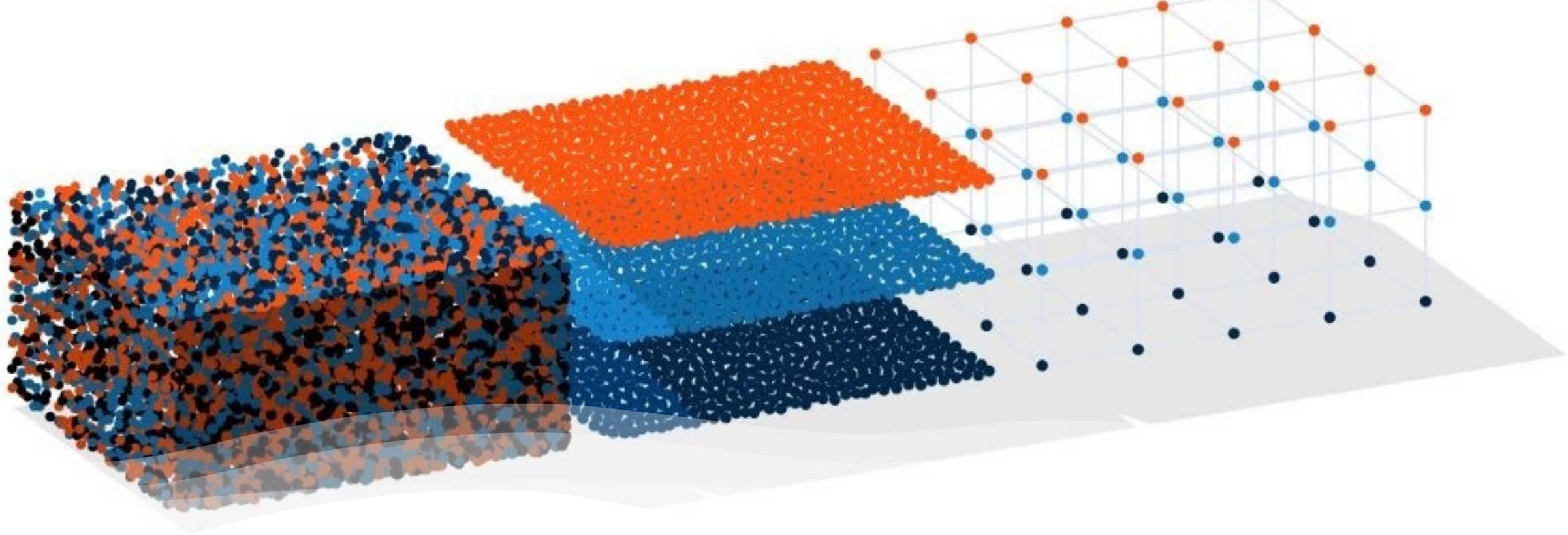


Preng Biba

Feature Engineering – Parte 1

Feature Engineering

- Se refiere al proceso de utilizar el conocimiento de la información/data que tenemos disponible para generar y modificar características nuevas.

- Muchos algoritmos de Machine Learning, funcionan con modelos base que requieren una estructuras específicas.

Algunas de la Tareas

- Imputación de datos faltantes,
- Codificación de Variables Categóricas,
- Tansformación de Variables,
- Generación de nuevas características (Mutación de datos).

Datos Faltantes

Faltantes

- Como sabemos al momento de utilizar un dataset, puede suceder que algunos de los valores no estén disponibles por diversas razones.

- La falta de datos, suele ser uno de los restos más importantes y comunes al momento de generar un modelo de ML.

- La falta de información puede generar efectos negativos en la capacidad predictiva de un modelo.

Faltantes

Perdida	• Un dato puede faltar debido a factores externos como: olvidar almacenarlo, al momento de almacenarlo, la información de guardo corrupta.
Inexistencia	• Suele suceder cuando se generan campos a partir de dos o más variables. Por ejemplo: si una persona no tiene ingresos, no es posible calcular el ratio de consumo/ingreso.
No hay disponibilidad	• El dominio de la variable no está correctamente especificado.

Faltantes

- Muchas librerías no pueden manejar datos ausentes para generar al momento de generar modelos.

- Puede existir un sesgo al momento de generar un modelo con datos faltantes.

- La falta de información puede producir variaciones en las distribuciones de los datos muestreados respecto a los datos poblacionales.

Faltantes

- Existen tres mecanismos principales para realizar imputación de datos, los:
  - MCAR (Missing Completely at Random).
  - MAR (Missing at Random).
  - MNAR (Missing not at Random).

MCAR

- La probabilidad de la falta de valores es la misma para cualquier observación.

- No existe ninguna relación entre las demás observaciones, tanto faltantes como existentes.

MAR – Missing at Random

- Se genera cuando existe una relación entre los faltantes y algunas variables disponibles en el dataset.

Tipo de Sangre	Peso
A	120lbs
A	NA
A	NA
A	144lbs
A	160lbs
A	125lbs
B	NA
B	NA
B	180lbs
B	150lbs
B	NA
B	145lbs

MNAR – Missing not ar Random

- Se presenta cuando existe un mecanismo que genere intrínsecamente la falta de los valores.

Obesidad	Dias de Ejercicio	Depresión
SI	NA	SI
SI	NA	NA
SI	NA	NO
SI	0	NA
SI	1	NA
SI	2	NO
NO	2	NO
NO	4	NO
NO	3	SI
NO	NA	NO
NO	2	SI
NO	0	NA

Data Imputation

Data Imputation

- La imputación de datos se refiera a la acción de reemplazar los valores faltantes de un conjunto de datos, con una estimación del posible valor real.

- La idea principal es proveerle un dataset con la mayor cantidad de información posible a un algoritmo de ML.

Data Imputation

Datos Numéricas	Variables Categóricas	Mixto
• Imputación de media y mediana. • Imputación de valores arbitrarios. • Imputación probabilística.	• Imputación por frecuencia. • Agregar categoría de faltante.	• Complete Case Analysis (CCA) • Indicador de Faltantes. • Imputación por muestra aleatoria.

CCA

- Se refiere a remover todas las observaciones que posean faltantes en cualquier variable del dataset.
- De este modo se utiliza la información que se considera “completa” dentro del dataste.
- Aplicable a datos numéricos, categóricos y mixtos.
- Se recomienda usar cuando la cantidad de datos faltantes es menor al 5%.

Ventajas

- Es un enfoque simple,
- No requiere ninguna manipulación interna de los datos,
- Prevalecen las propiedades probabilísticas de los datos, no modificamos la distribución de los datos.

Desventajas

- Puede excluir a una gran cantidad de datos del dataset original,
- Omite observaciones que podrían ser poarticularmente importantes para la construcción del modelo.
- Podría generarse un dataset sesgado debido a que se pueden omitir observaciones que contengan categorías específicas.
- En producción el modelo podría producir errores ya que si aparece una observación con alguna categoría eliminada, el predictor no sabrá como tratarla.


Jupyter DataFrames
File Edit View Insert Cell Kernel Widgets Help
#DataFrames in Python
This is pandas DataFrame
In [ ]:
In [6]:
import pandas as pd
import numpy as np
In [7]:
df = pd.DataFrame(data*np.array([[1,2,3],[4,5,6]], dtype=int), columns=['A', 'B', 'C'])
df
Jupyter
DataCamp

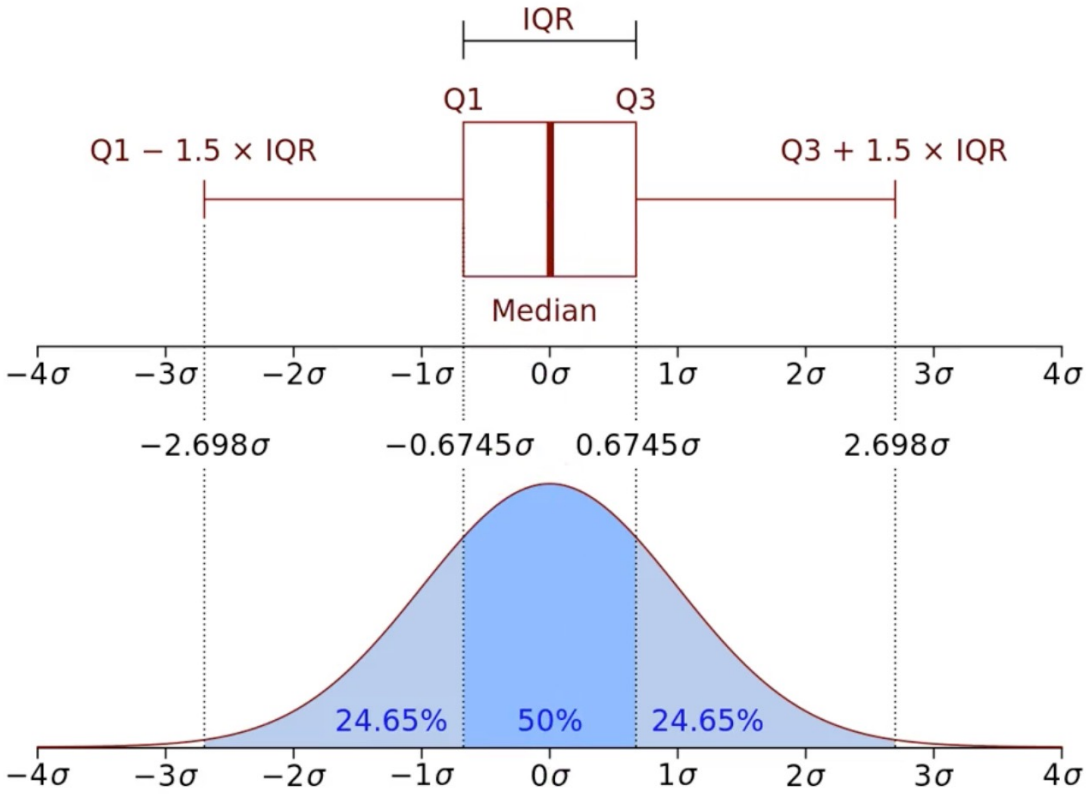
Imputación de Media y Mediana

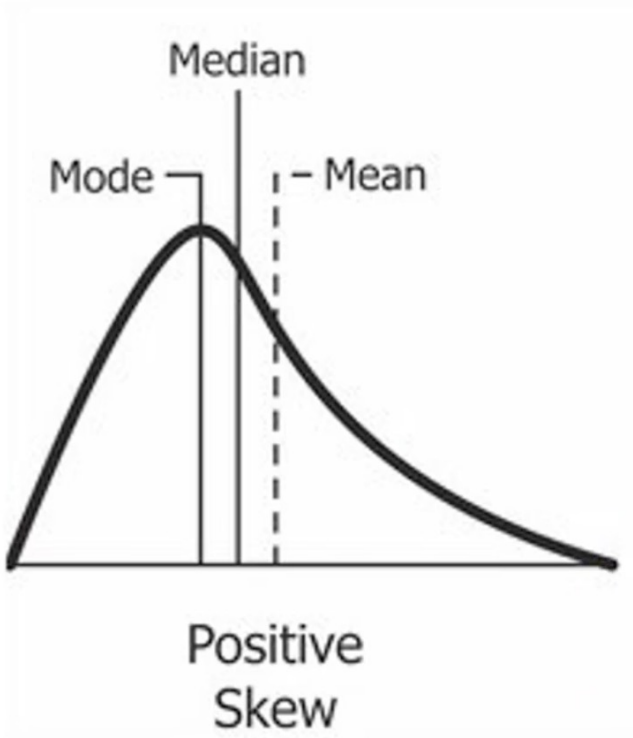
Imputación de Media y Mediana

- Consiste en imputar la media o la mediana en una distribución de datos donde existen faltantes, por su naturaleza numérica no es posible aplicar este tipo de imputación a variables categóricas.

- Es importante saber que tipo de imputación utilizar, es decir imputación por media o imputación por varianza.

Imputación de Media y Varianza





Imputación de Media y Mediana

- Es un enfoque fácil y rápido de implementar.
- Permite obtener datasets completos de forma eficiente.
- Puede integrarse en producción, si un valor es faltante al momento de realizar una predicción.

Imputación de Media y Mediana

- Distorsiona la distribución de la variable original.
- Distorsiona la varianza de la variable original.
- Distorsiona la correlación y covarianza de la variable imputada respecto a las demás en el dataset.
- Entre más NAs, mayor será la distorsión.

Imputación de Media y Mediana

- Se recomienda utilizar cuando los faltantes son generados por MCAR.

- Cuando los faltantes no son más del 5% del total de la variable.

Imputación de Media y Mediana

- Una buena práctica para tratar datasets en general es utilizar la combinación de la imputación de media y mediana junto con la combinación de un indicador de faltantes (aka “missing indicator”)


Jupyter DataFrames
File Edit View Insert Cell Kernel Widgets Help
#DataFrames in Python
This is pandas DataFrame
In [ ]:
In [6]:
import pandas as pd
import numpy as np
In [7]:
df = pd.DataFrame(data*np.array([[1,2,3],[4,5,6]], dtype=int), columns=['A', 'B', 'C'])
df
Jupyter
DataCamp

Arbitrary Imputation

Arbitrary Imputation

- Consiste en reemplazar los faltantes utilizando un valor arbitrario.
- Los valores típicos utilizados en este tipo de imputación son 0, -999, 999 o -1 si la distribución es completamente positiva.
- Generalmente se busca que el valor imputado, este fuera del rango de la distribución de la variable.
- En el caso cualitativo, se utiliza la categoría “faltante”.

Arbitrary Imputation

- Fácil de implementar,
- Permite obtener datasets completos eficientemente.
- Puede integrarse en producción.
- Captura la importancia de valores faltantes.

Arbitrary Imputation

- Distorsiona la distribución original de la variable,
- Distorsiona la varianza de la variable original,
- Distorsiona la covarianza de la variable,
- Genera outliers,
- Se debe selecionar un valor que no sea común o descriptivo de la distribución (media, mediana, moda).
- A mayor NAs, mayor distorsión.


Jupyter DataFrames
File Edit View Insert Cell Kernel Widgets Help
#DataFrames in Python
This is pandas DataFrame
In [ ]:
In [6]:
import pandas as pd
import numpy as np
In [7]:
df = pd.DataFrame(data*np.array([[1,2,3],[4,5,6]], dtype=int), columns=['A', 'B', 'C'])
df
Jupyter
DataCamp

End of Tail Distribution Imputation

End of Distribution

- Similar al caso de Arbitrary Imputation, con la diferencia fundamental que el valor a imputar debe ser un valor que esté contenido dentro de la distribución de la variable.

- Si la distribución de la variable es normal, podemos usar como valor a imputar el valor $x_{imp} = \mu + 3\sigma$

- Si existe un sesgo en la distribución de la variable, podemos usar la información del IQR para generar el valor.

- Por su naturaleza numérica, solo se puede aplicar a variables numéricas.

End of Tail Distribution: IQR

- $IQR = 75^{th} Quantile\ - 25^{th} Quantile$
- $LS = 75^{th} Quantile + 1.75 * IQR$
- $LI = 25^{th} Quantile - 1.75 * IQR$


Jupyter DataFrames
File Edit View Insert Cell Kernel Widgets Help
#DataFrames in Python
This is pandas DataFrame
In [ ]:
In [6]:
import pandas as pd
import numpy as np
In [7]:
df = pd.DataFrame(data*np.array([[1,2,3],[4,5,6]], dtype=int), columns=['A', 'B', 'C'])
df
Jupyter
DataCamp

Imputación para Variables Categóricas

Imputación de Moda

- Para datos categóricos, se puede utilizar la moda como mecanismo de imputación de faltantes,

Imputación de Media y Mediana

- Es un enfoque fácil y rápido de implementar.
- Permite obtener datasets completos de forma eficiente.
- Puede integrarse en producción, si un valor es faltante al momento de realizar una predicción.

Imputación de Media y Mediana

- Distorsiona la relación entre las demás variables y la variable imputada.
- Puede generar demasiado ajuste sobre la variable categórica imputada.
- Entre más NAs, mayor será la distorsión.


Jupyter DataFrames
File Edit View Insert Cell Kernel Widgets Help
#DataFrames in Python
This is pandas DataFrame
In [ ]:
In [6]:
import pandas as pd
import numpy as np
In [7]:
df = pd.DataFrame(data*np.array([[1,2,3],[4,5,6]], dtype=int), columns=['A', 'B', 'C'])
df
Jupyter
DataCamp

Missing Category Imputation

Missing Category Imputation

- Este método consiste en agregar una nueva clase llamada “faltante” buscando que el algoritmo de ML sea capaz de predecir esta categoría como parte de la estructura del algoritmo.

- Es el método más utilizado para realizar imputación en variables categóricas.

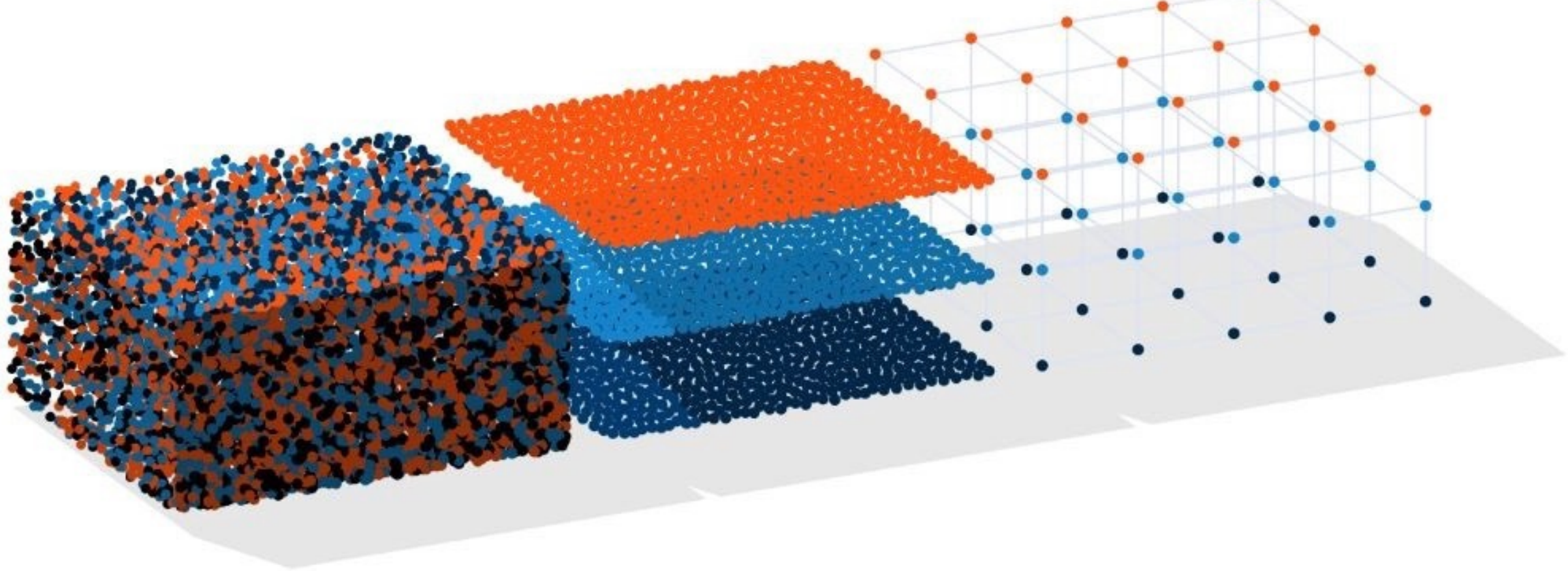
Missing Category Imputation

- Fácil de implementar.
- Permite obtener datasets completos de forma eficiente.
- Puede integrarse en producción.
- Muestra la importancia de datos faltantes para el modelo.

Missing Category Imputation

- Si la frecuencia de NAs es baja, puede resultar en que el dataset tenga una nueva clase de baja frecuencia.


Jupyter DataFrames
File Edit View Insert Cell Kernel Widgets Help
#DataFrames in Python
This is pandas DataFrame
In [ ]:
In [6]:
import pandas as pd
import numpy as np
In [7]:
df = pd.DataFrame(data*np.array([[1,2,3],[4,5,6]], dtype=int), columns=['A', 'B', 'C'])
df
Jupyter
DataCamp



Preng Biba

Feature Engineering – Parte 2

Codificación de Variables Categóricas.

Categorical Encoding

- Se refiere al proceso de producir valores numéricos a partir de una variable categórica, esto con la finalidad de:

  - Producir variables numéricas que pueden ser utilizadas por el algoritmo de ML.
  - Producir nuevas características a partir de las categorías disponibles en el dataset.

Categorical Encoding

Técnicas Tradicionales

- One hot Encoding.
- Frecuency Ecoiding.
- Ordinal Encoding.

Relaciones Monotonicas

- Ordered Label Encoding.
- Mean Encoding.
- Weight of Evidence.

Técnicas Modernas

- Binary Encoding.
- Feature Hashing.

One Hot Encodign

One Hot Encoding

- Consiste en codificar los valores de una variable categórica con un conjunto de valores booleanos (0 o 1), esto nos permite indicarle al algoritmo de ML si el valor de la categoría esta presente o no en la observación.

Valor
Texas
Florida
California
Texas
Florida
California

Texas	California	Florida
1	0	0
0	0	1
0	1	0
1	0	0
0	0	1
0	1	0

One Hot Encoding k - 1

- Consiste en utilizar one hot encoding para codificar una variable categórica, pero utilizando únicamente k – 1 valores para la variable original.

- Esto permite ”ahorrar” una columna al hacer la codificación, es muy útil para ahorrar recurso computacionales y evitar redundancias.

Valor
Texas
Florida
California
Texas
Florida
California

Texas	California
1	0
0	0
0	0
1	0
0	0
0	1

Cuando usar k y no k - 1

- Existen algunas ocasiones donde es adecuado utilizar todo los valores presentes en la variable categórica:
  - Cuando estamos usando arboles de decisión como algoritmo de ML.
  - Cuando se está realizando una selección de características recursiva (AdaBoost).
  - Si necesitamos evaluar la relevancia de cada valor de la categoría en nuestro algoritmo de ML.

One Hot Encoding

- Este enfoque no asume ningún tipo de distribución para las variables que se está aplicando.
- Mantiene la información de todas los valores en la variable categórica.
- Ideal para trabajar con modelos lineales.

One Hot Encoding

- Expande el espacio de características de un modelo.
- No introduce más información que la ya conocida al modelo.
- En algunos casos, en una cantidad muy grande de categorías puede llevar a generar redundancias.
- Es una vez implemnetado, puede costoso para el entrenamiento de algoritmos de ML en datasets muy grandes


Jupyter DataFrames
File Edit View Insert Cell Kernel Widgets Help
#DataFrames in Python
This is pandas DataFrame
In [ ]:
In [6]:
import pandas as pd
import numpy as np
In [7]:
df = pd.DataFrame(data*np.array([[1,2,3],[4,5,6]], dtype=int), columns=['A', 'B', 'C'])
df
Jupyter
DataCamp

OHE para Categorías de Alta Frecuencia.

- Se refiera a utilizar el ohe en una variable para categorías dentro de variables que tiene alta frecuencia, y no para todas las categorías.

- Texas => 1000
- California => 800
- Florida => 250
- Michigan, Arizona, Pensilvania => 300

Texas	California	Florida
1	0	0
0	0	1
0	1	0
1	0	0
0	0	1
0	1	0

OHE para Categorías de Alta Frecuencia

- Muy fácil de implementar.
- No requiere exploración compleja de variables.
- No expande enormemente el espacio de características del dataset.
- Permite manejar nuevas categorías en el test set.
- Reduce la complejidad de entrenamiento al usar OHE.

OHE para Categorías de Alta Frecuencia

- No agrega nueva información sobre las categorías.
- No guarda ningún tipo de información sobre las categorías que se ignoraron.


Jupyter DataFrames
File Edit View Insert Cell Kernel Widgets Help
#DataFrames in Python
This is pandas DataFrame
In [ ]:
In [6]:
import pandas as pd
import numpy as np
In [7]:
df = pd.DataFrame(data*np.array([[1,2,3],[4,5,6]], dtype=int), columns=['A', 'B', 'C'])
df
Jupyter
DataCamp

Label Encoding

Label Encoding

- Este enfoque consiste en codificar el valor de las categorías utilizando un valor numérico arbitrario que va desde 1 a n (o de 0 a n – 1). La idea es que cada categoría tenga un valor numérico distinto.

- Se puede utilizar para codificar variables categóricas con escala nominal, es decir, no es necesario que exista un nivel de importancia intrínseco en las categorías de la variable.

Label Encodign

Valor
Texas
Florida
California
Texas
Florida
California

Valor
1
2
3
1
2
3

Label Encoding

- Es muy simple de implementar.
- No expande el espacio de características del dataset.
- Funciona muy bien con algoritmos basados en arboles.

Label Encodign

- No agrega información adicional sobre las características.
- No es recomendable para trabajar modelos lineales.
- No maneja categorías que se presenten en el dataset de prueba.


Jupyter DataFrames
File Edit View Insert Cell Kernel Widgets Help
#DataFrames in Python
This is pandas DataFrame
In [ ]:
In [6]:
import pandas as pd
import numpy as np
In [7]:
df = pd.DataFrame(data*np.array([[1,2,3],[4,5,6]], dtype=int), columns=['A', 'B', 'C'])
df
Jupyter
DataCamp

Frequency Encodign

Frequency Encoding

- Este enfoque consiste en sustituir las categorías de una variable en función de la frecuencia o densidad de cada categoría en las variables el dataset.
- Este enfoque la ”fuerza” que cada categoría tiene dentro del dataset.
- Es un enfoque muy popular en las competencias de Kaggle.
- Para que funcione adecuadamente, se asume que las categorías presentes en una variable tiene de alguna forma una relación con la variable a predecir.

Frecuency Encoding

Valor
Texas
Texas
California
Texas
Florida
California

Valor
1
1
3
1
2
3

Valor
0.5
0.5
0.333
0.5
0.166
0.333

Frequency Encoding

- Fácil de implementar.
- No expande el espacio de características del dataset.
- Funciona muy bien con algoritmos basados en arboles.

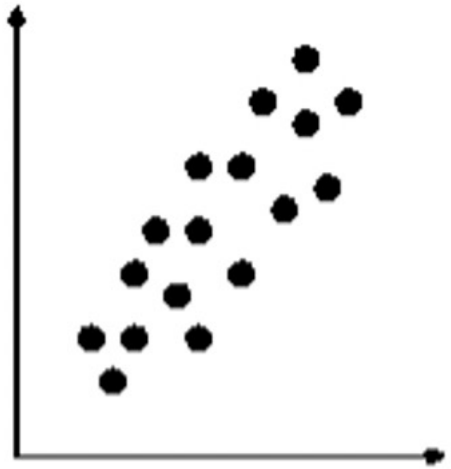
Frequency Encoding

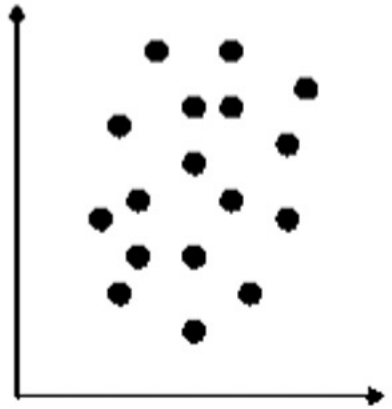
- No es muy bueno con modelos lineales.
- No maneja categorías nuevas en el test set.
- Si dos categorías tiene el mismo peso, se les asignará el mismo valor, por tanto serán una misma categoría y puede llevar a la perdida de información.

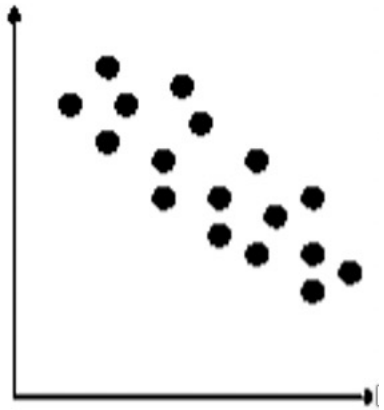

Jupyter DataFrames
File Edit View Insert Cell Kernel Widgets Help
#DataFrames in Python
This is pandas DataFrame
In [ ]:
In [6]:
import pandas as pd
import numpy as np
In [7]:
df = pd.DataFrame(data*np.array([[1,2,3],[4,5,6]], dtype=int), columns=['A', 'B', 'C'])
df
Jupyter
DataCamp

Codificación con Relaciones Monotónicas

Codificación con Relaciones Monotónicas







Codificación con Relaciones Monotónicas

Texas
Texas
Texas
California
Texas
Florida
California

Target
1
0
0
1
1
0

Target Guide Ordinal Encoding:

Mean Encoding:

Codificación con Relaciones Monotónicas

- Fácil de implementar.
- No modifica el espacio de características.
- Crea una relación monotonica entre los valores de la variable categórica y el target.

Codificación con Relaciones Monotónicas

- Puede genrar overfitting.
- Difícil de implementar con validación cruzada.
- Si dos categorías tiene la misma media en el target, se les asignará el mismo valor, por tanto serán una misma categoría y puede llevar a la perdida de información.


Jupyter DataFrames
File Edit View Insert Cell Kernel Widgets Help
#DataFrames in Python
This is pandas DataFrame
In [ ]:
In [6]:
import pandas as pd
import numpy as np
In [7]:
df = pd.DataFrame(data*np.array([[1,2,3],[4,5,6]], dtype=int), columns=['A', 'B', 'C'])
df
Jupyter
DataCamp

Feature Scaling

Feature Scaling

- Los algoritmos que están basados en linealidad son sensibles a la escala de las variables.
- Las variables con rangos de magnitudes más grandes suelen dominar sobre las que tienen rangos de magnitudes más pequeñas.
- Gradient Decent converge mejor si las variables tiene la misma escala.
- En el caso de los SVM, es más fácil trabajar con variables de escala similar.
- En algoritmos basados en distancia (KNN, Kmeans, PCA) la magnitud de las variables puede generar diferencias muy grandes difíciles de manipular e interpretar

Feature Scaling

- Algoritmos Sensibles a la Escala:
  - Regresiones Lineal y Logística,
  - Redes Neurales,
  - SVM,
  - KNN,
  - LDA,
  - QDA,
  - PCA,
  - K-means.
- Algoritmos Insensibles a la Escala:
  - Todos los algoritmos basados en arboles: Arboles de Decisión, Random Forest, AdaBoost, XGBoost, etc…

Feature Scaling

- Se refiere a un mecanismo el cual consiste en normalizar los valores del conjunto de variables de un dataset.

- La idea es igualar la escala de todas las variables descritas en el dataset.

- Típicamente, el procedimiento de FS es el ultimo paso antes de entrenar el modelo de ML.

Feature Scaling

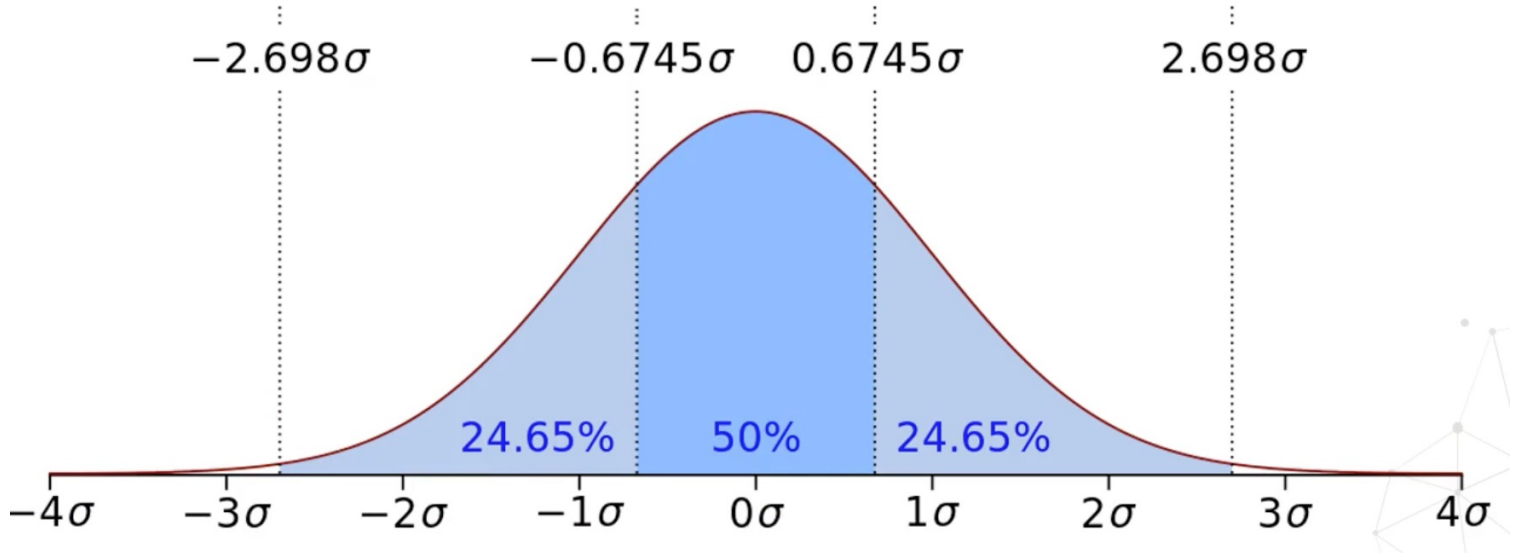
- Standardization (*).
- Normalización de Media.
- MinMax Scaling(*)
- Max Absolute Scaling
- Robust Scaling.

Standaridzation

Standaridzation

- Consiste en realizar la escala de los valores utilizando una conversión al valor Z de la variable, basada en la media y desviación estándar.

$$Z = \frac{x_i - \bar{\mu}}{\sigma}$$



Standaridzation

- Centra la media en 0.
- Escala la varianza a 1.
- Conserva la forma de la disribución original.
- Conserva los valores máximos y mínimos.
- Conserva los outliers.


Jupyter DataFrames
File Edit View Insert Cell Kernel Widgets Help
#DataFrames in Python
This is pandas DataFrame
In [ ]:
In [6]:
import pandas as pd
import numpy as np
In [7]:
df = pd.DataFrame(data*np.array([[1,2,3],[4,5,6]], dtype=int), columns=['A', 'B', 'C'])
df
Jupyter
DataCamp

Mean Noramlization

Mean Noramalization

- Consiste en utilizar información sobre los valores extremos para realizar la nomralización, según se muestra en la siguiente ecuación:

$$X_{scaled} = \frac{x_i - \bar{\mu}}{\max(x) - \min(x)}$$

Mean Noramlization

- Centra la distribución en 0,
- Cambia la varianza de la distribución si la varaible tiene mucho sesgo.
- Podría modificar la forma de la distribución.
- Los valores varían entre -1 y 1.
- Conserva los outliers.


Jupyter DataFrames
File Edit View Insert Cell Kernel Widgets Help
#DataFrames in Python
This is pandas DataFrame
In [ ]:
In [6]:
import pandas as pd
import numpy as np
In [7]:
df = pd.DataFrame(data*np.array([[1,2,3],[4,5,6]], dtype=int), columns=['A', 'B', 'C'])
df
Jupyter
DataCamp

MinMax Scaling

MinMax Scaling

- Consiste en utilizar información sobre los valores extremos para realizar la estandarización, según se muestra en la siguiente ecuación:

$$X_{scaled} = \frac{x_i - \min(x)}{\max(x) - \min(x)}$$

MinMax Scaling

- Devuelve valores entre 0 y 1,
- Modifica la media y la varianza.
- Podría modificar la forma de la distribución.
- Los valores varían entre 0 y 1.
- Conserva los outliers.


Jupyter DataFrames
File Edit View Insert Cell Kernel Widgets Help
#DataFrames in Python
This is pandas DataFrame
In [ ]:
In [6]:
import pandas as pd
import numpy as np
In [7]:
df = pd.DataFrame(data*np.array([[1,2,3],[4,5,6]], dtype=int), columns=['A', 'B', 'C'])
df
Jupyter
DataCamp

Robust Scaling

Robust Scaling

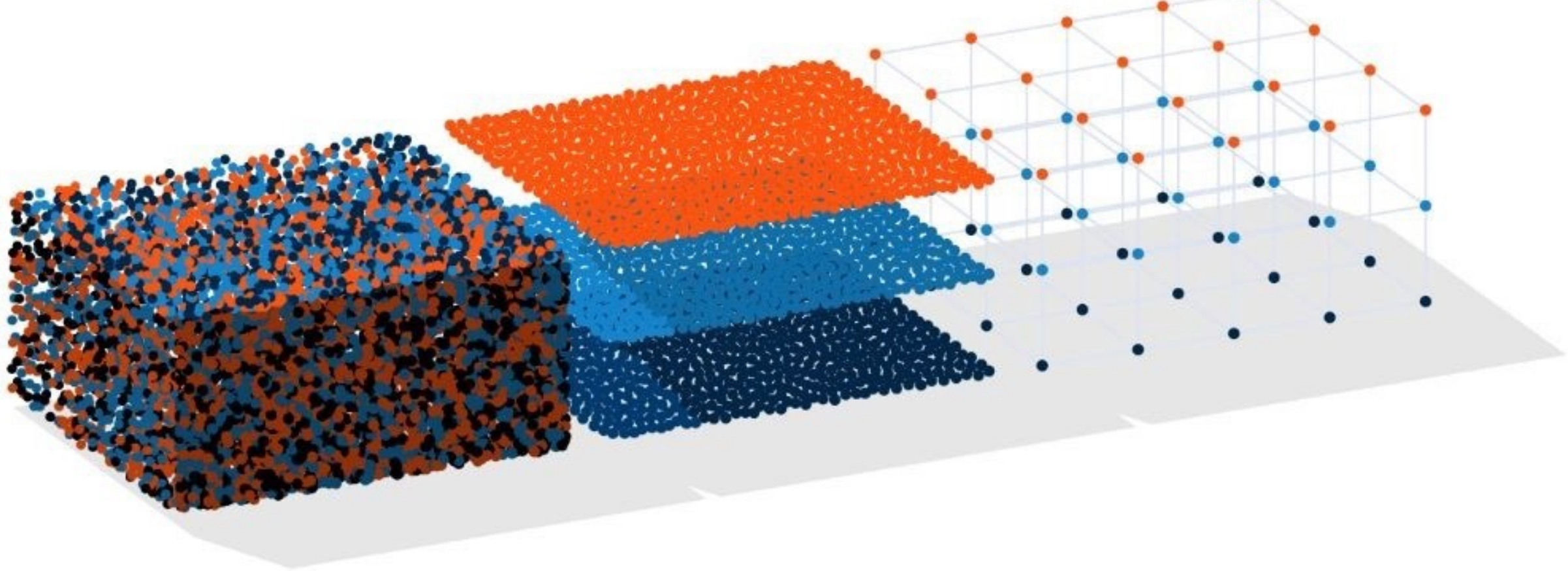
- Consiste en utilizar información sobre los valores extremos para realizar la estandarización, según se muestra en la siguiente ecuación:

$$X_{scaled} = \frac{x_i - \mathrm{m}edian(x)}{3Q(x) - 1Q(x)}$$

Robust Scaling

- Centra la mediana en 0,
- Modifica la media y la varianza.
- Podría modificar la forma de la distribución.
- Maneja Outliers.


Jupyter DataFrames
File Edit View Insert Cell Kernel Widgets Help
#DataFrames in Python
This is pandas DataFrame
In [ ]:
In [6]:
import pandas as pd
import numpy as np
In [7]:
df = pd.DataFrame(data*np.array([[1,2,3],[4,5,6]], dtype=int), columns=['A', 'B', 'C'])
df
Jupyter
DataCamp



Preng Biba

Feature Engineering – Parte 3

Outline

Transformación de Variables.

Discretización de Caracteristicas.

Tratamiento de Outliers.

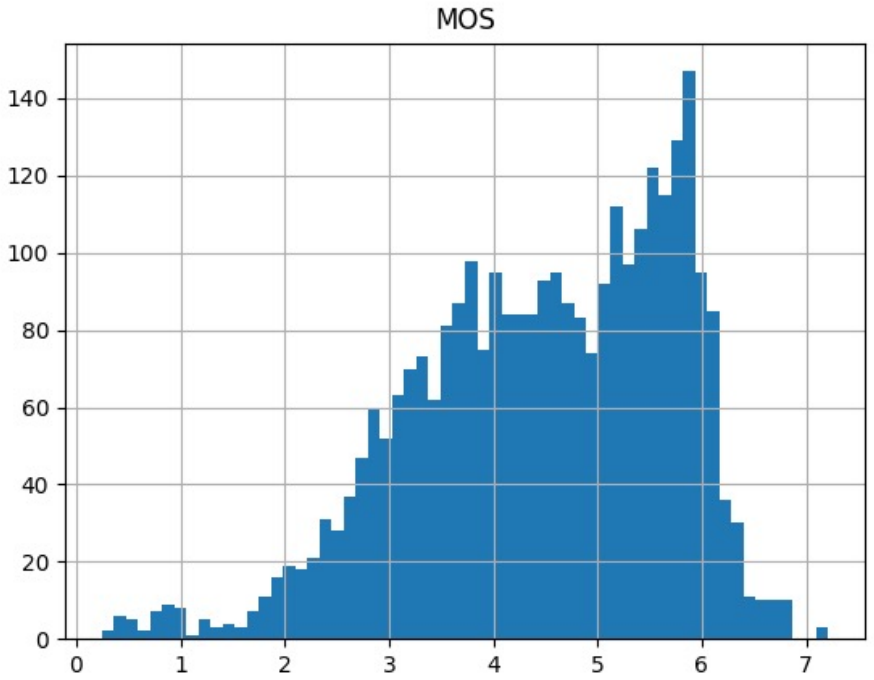
Piplines Feature Engineering Design.

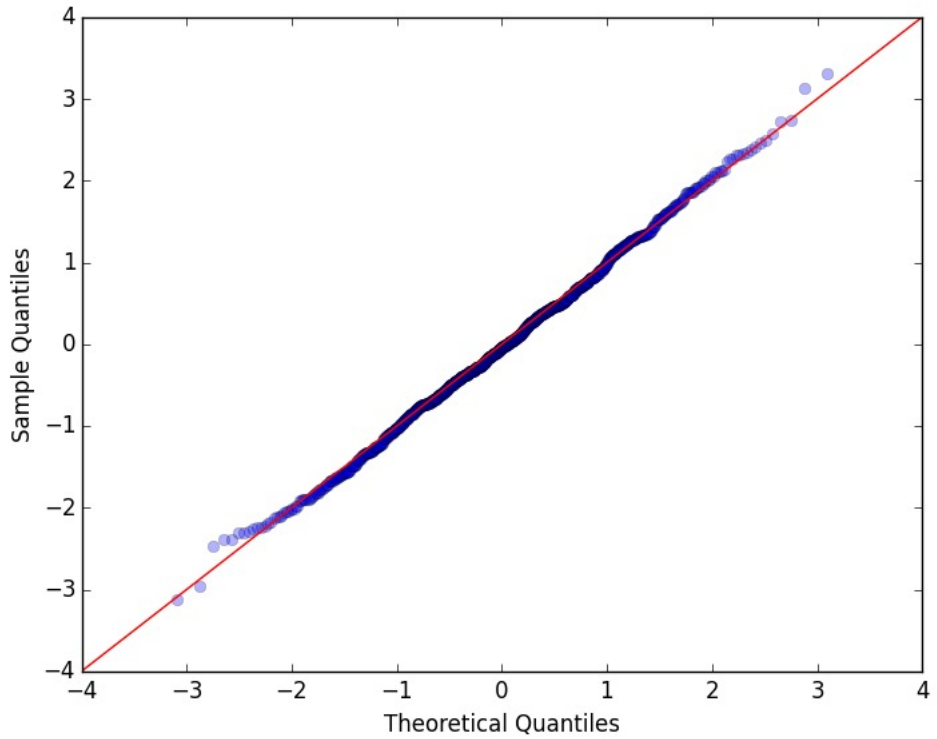
Transformación de Variables

Debido a que muchos algoritmos de ML asumen que el comportamiento de todas las variables numéricas que lo conforman.

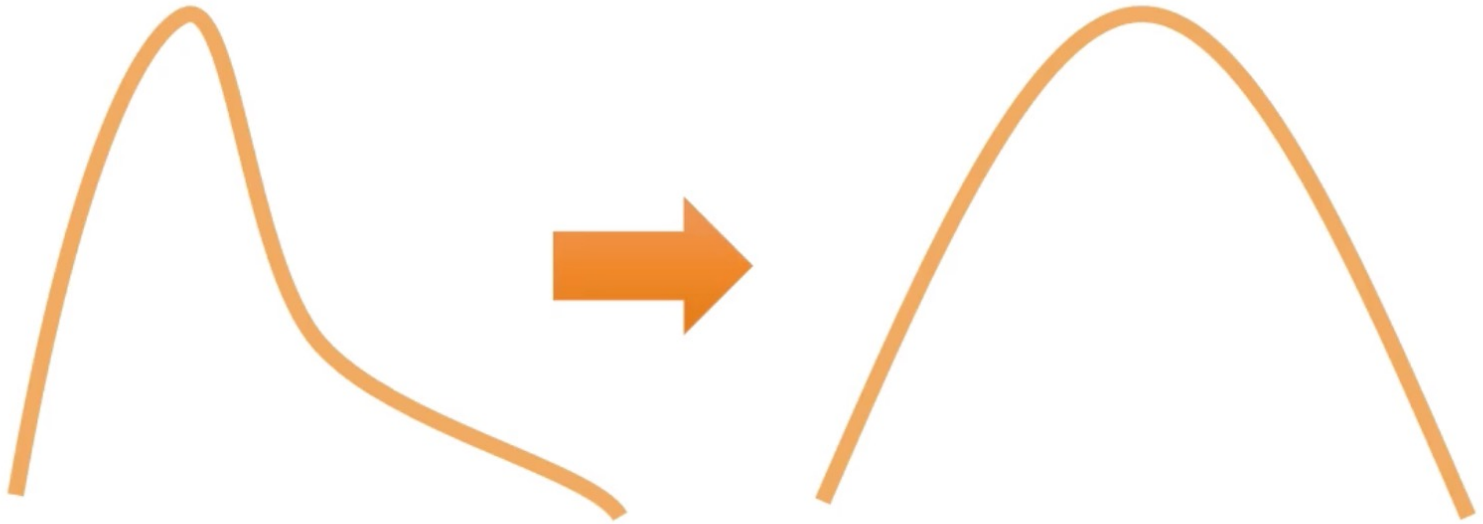
Debido a esto, en ocasiones resulta útil transformar las variables que no tiene una distribución de probabilidad aproximadamente normal.

Verificación de la Normalidad





Transformación de Variables



Tipos de Transformaciones

- Logarítmica.
- Exponencial.
- Potencia.
- Reciproca.
- Box-Cox.
- Yeo-Johnson

Algunas Consideraciones

Logarítmica	Reciproca	Exponencial y Potencia	Box-Cox
• X > 0	• x != 0	• Restricciones de Dominio	• X > 0

Box-Cox

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

Yeo-Johnson

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


Jupyter DataFrames
File Edit View Insert Cell Kernel Widgets Help
#DataFrames in Python
This is pandas DataFrame
In [ ]:
In [6]:
import pandas as pd
import numpy as np
In [7]:
df = pd.DataFrame(data*np.array([[1,2,3],[4,5,6]], dtype=int), columns=['A', 'B', 'C'])
df
Jupyter
DataCamp

Discretización

Discretización

- Transformar una variable continua a una variable discreta por medio de un conjunto de intervalos continuos los cuales incluyen todos los valores de la variable original.

- La discretización permite manejar outlier y reducir el sesgo que pueda tener una variable por su naturaleza numérica.

Enfoques

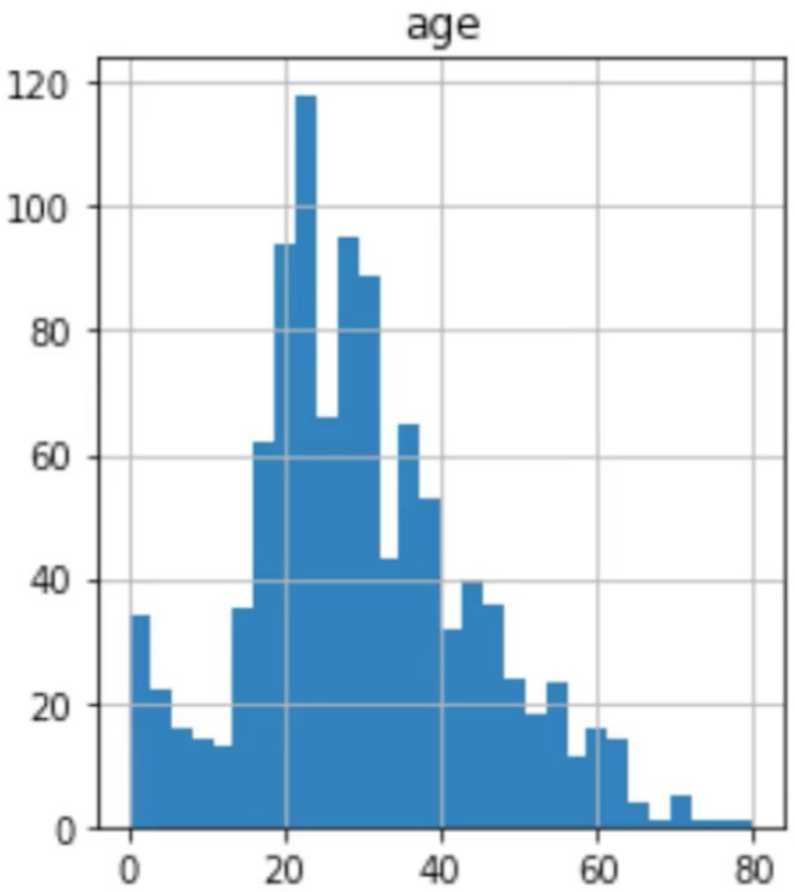
Unsupervised

- Equal With
- Equal Frecuency
- K-Means

Superviced

- Decision Trees

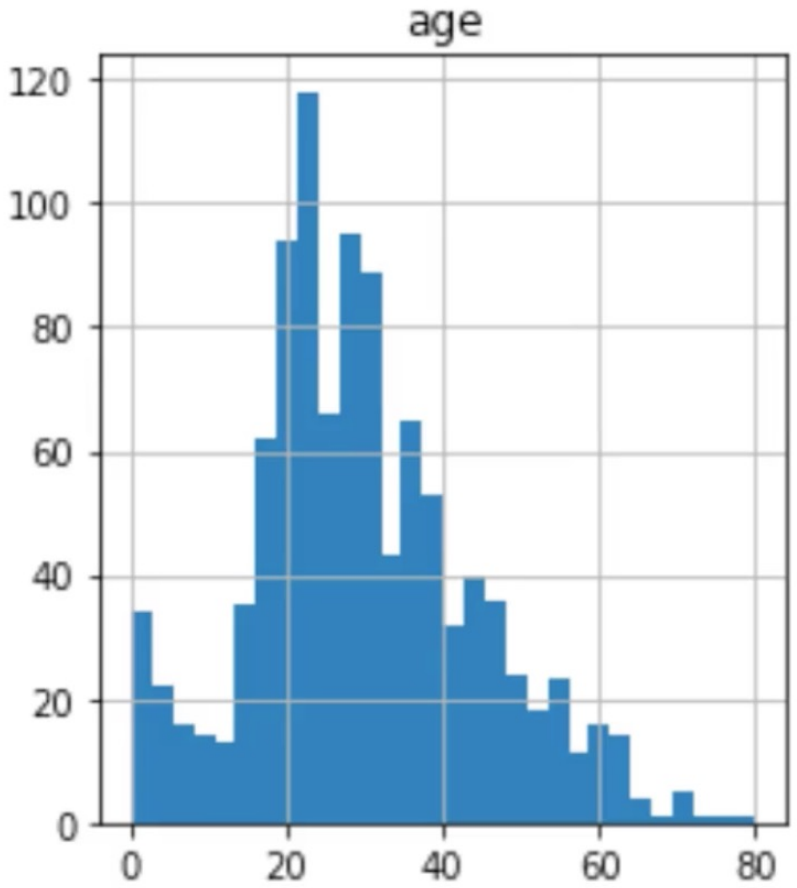
K-means: Discretization

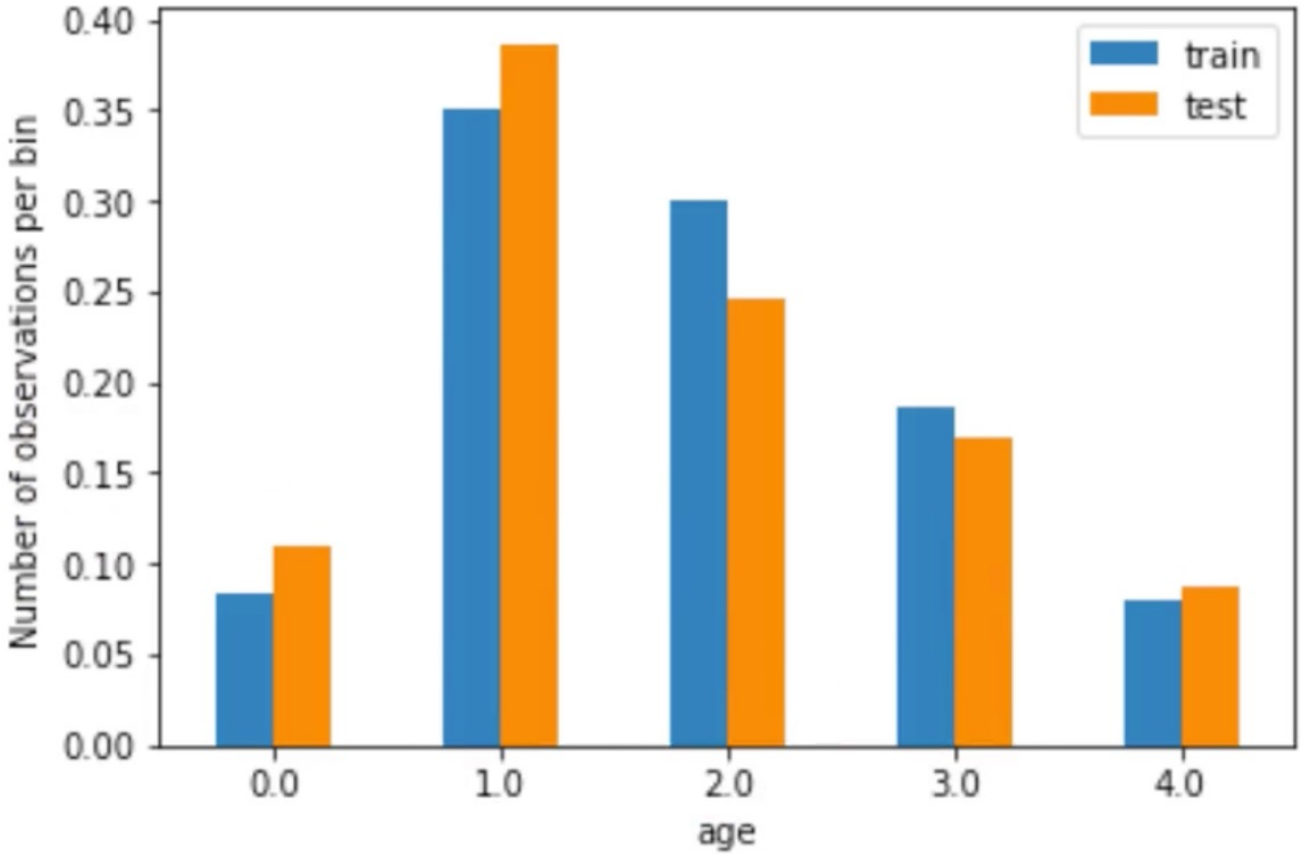


Discretizar en 5 grupos

K-Means: Discretización

K-means: Discretización





K-means: Discretization

- No permite que los valores se salgan del grueso de la muestra.
- Maneja outliers, sin embargo hay que tomar en cuenta que los outliers podrían afectar los valores de los centroides.
- Genera la discretización de una variable.
- Este enfoque es muy bueno para combinarse con codificación de categorías.


Jupyter DataFrames
File Edit View Insert Cell Kernel Widgets Help
#DataFrames in Python
This is pandas DataFrame
In [ ]:
In [6]:
import pandas as pd
import numpy as np
In [7]:
df = pd.DataFrame(data*np.array([[1,2,3],[4,5,6]], dtype=int), columns=['A', 'B', 'C'])
df
Jupyter
DataCamp

Tratamiento de Outliers

Outliers

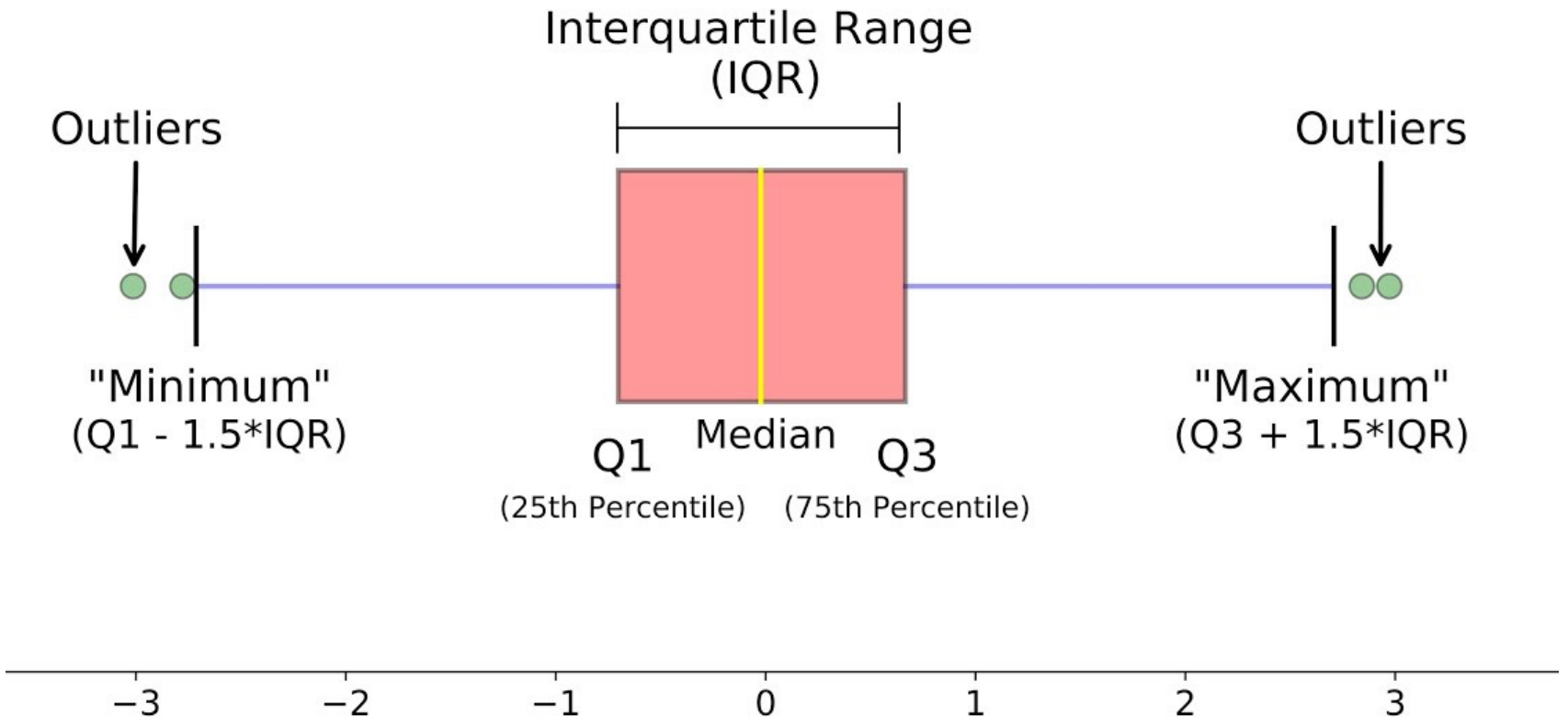
- Como sabemos un outlier se refiere a un valor que está muy desviado del grueso de la distribución de los datos de una variable.
- Debido a la naturaleza numérica de los algoritmos de ML, esto puede afectar el desempeño del algoritmo.
- Muchas veces los outliers son valores particulares que no reflejan el comportamiento general de la distribución de la variable.
- Los outliers son parte de la estructura de los datos iniciales, por tal motivo es necesario tratarlos ya que forman parte del dataset.

Algunos Enfoques

- Triming: Eliminar los valores que sean outliers.
- Data Faltante: Tratar los outliers como si fueran datos faltantes y utilizar métodos de imputación para sostituirlos.
- Discretización: Absorber el valor del outlier dentro de los bins de los extremos al discretizar.
- Censoring: Capping.

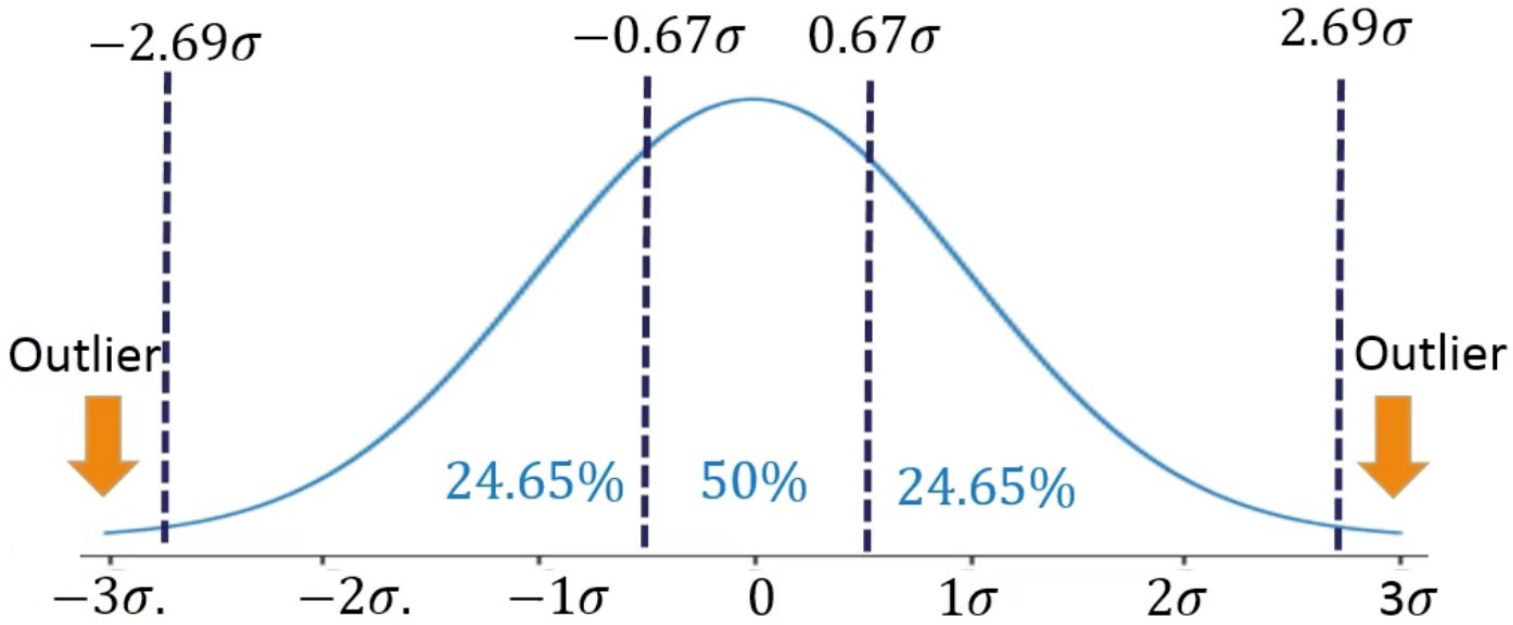
Detección

- Box Plots.



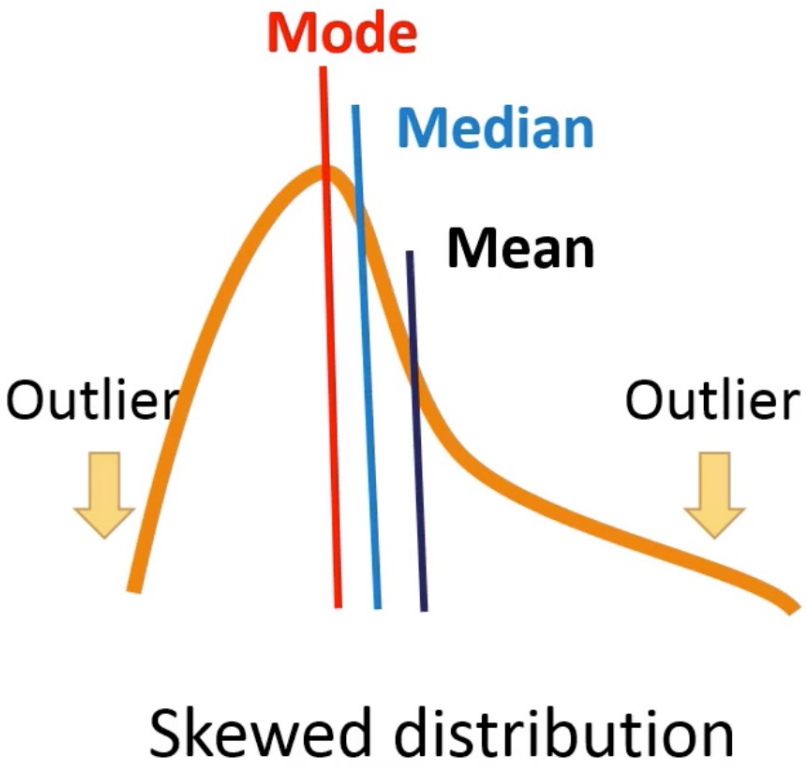
Detección

- Aproximación Gaussiana no paramétrica con KDE.
  - Si el valor se encuentra fuera de $\mu \pm 3\sigma$, el valor se considera un outlier.



Detección

- IQR:



- IQR = 75Pth – 25Pth.
- UL = 75Pth + 1.75*IQR.
- LL = 25Pth - 1.75*IQR.

Trimming

- Consiste en eliminar las observaciones que presentan outliers dentro del dataset.

- Debe tenerse en cuenta que si hay muchos outliers podría reducir significativamente el dataset original.

- Este enfoque es muy fácil de implementar, pero si la distribución de los datos está muy sesgada, existe la posibilidad de que siempre hayan outliers en la variable.

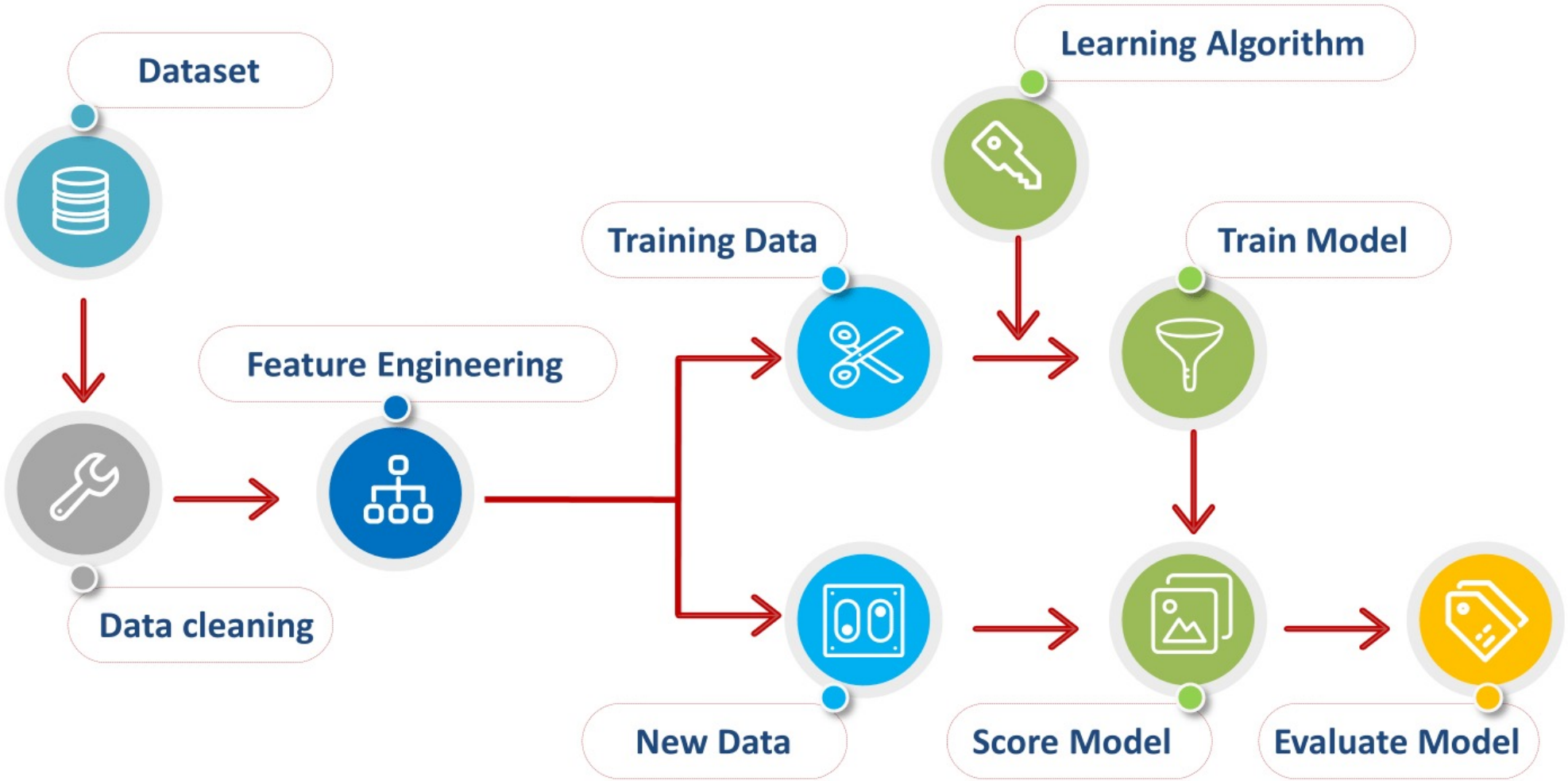
Censoring: (Capping)

- Consiste en convertir los outliers a los valores limites dentro de la distribución de la variable.

- Este método evita la reducción del dataset original.

Feature Engineering Pipeline

Feature Engineering Pipeline



Pipeline

- Hasta el momento hemos hablado de todas las técnicas utilzadas para hacer ingeniería de características:
  - Imputación de datos.
  - Codificación de varaibles categóricas.
  - Scaling.
  - Transformación de Variables.
  - Discretización.
  - Tratamiento de Outliers

Pipeline

- El pipeline facilita el desarrollo de los pasos necesarios en el proceso de FE, sin embargo es importante realizar un análisis adecuado para poder determinar cual es el orden y los pasos que se desarrollarán en el pipeline