

A collection of approximately 15 squares in various shades of blue and grey, scattered across the top half of the slide.

# Pre-procesado de datos

**Data Mining**

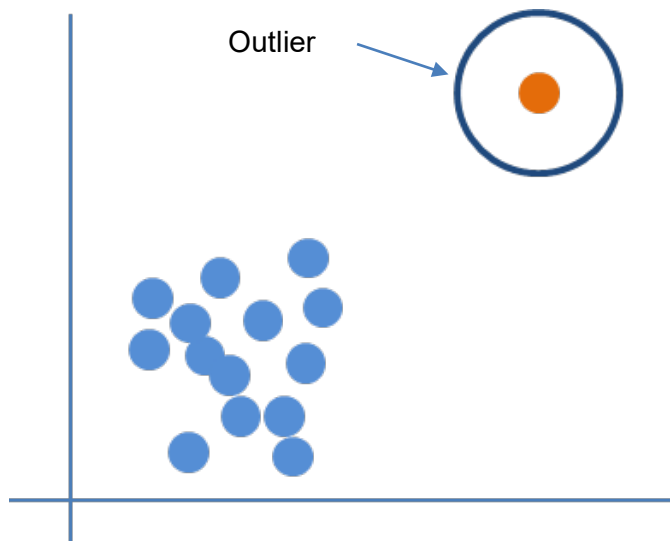
Ester Vidaña Vila

# Pre-procesado de datos

- El pre-procesamiento de datos es una etapa **esencial** en el proceso de descubrir información relevante de nuestros datos.
- Normalmente, el uso de datos de *baja calidad* implica la obtención de resultados pobres tras el proceso de minería de datos.
- Por lo tanto, el uso de técnicas de pre-procesamiento de datos y *feature engineering* es esencial para maximizar los resultados de nuestro proyecto de minería de datos.

# Detección de outliers - ¿Por qué?

- Los outliers pueden tener un gran impacto en los datos:
  - Pueden variar significativamente la media y desviación estándar de los datos.
  - Si están distribuidos de forma no-normal, pueden decrementar la normalidad en la distribución de los datos.



■ Ejemplo:

■ Sin outliers:

■ Datos = [1, 1, 2, 2, 2, 2, 3, 3, 3, 4, 4]

Mean = 2.45

Median = 2

Mode = 2

Standard Deviation = 0.98

■ Con un outlier:

■ Datos = [1, 1, 2, 2, 2, 2, 3, 3, 3, 4, 4, **400**]

Mean = 35.38

Median = 2.5

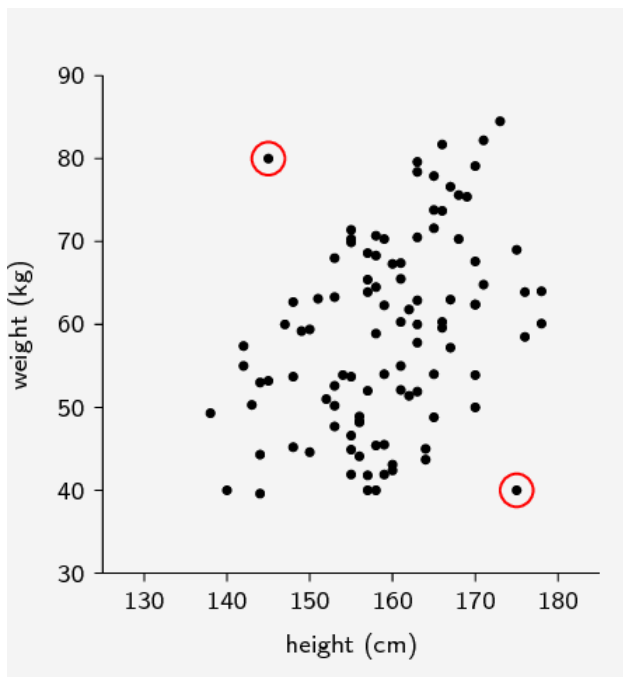
Mode = 2

Standard Deviation = 114.74

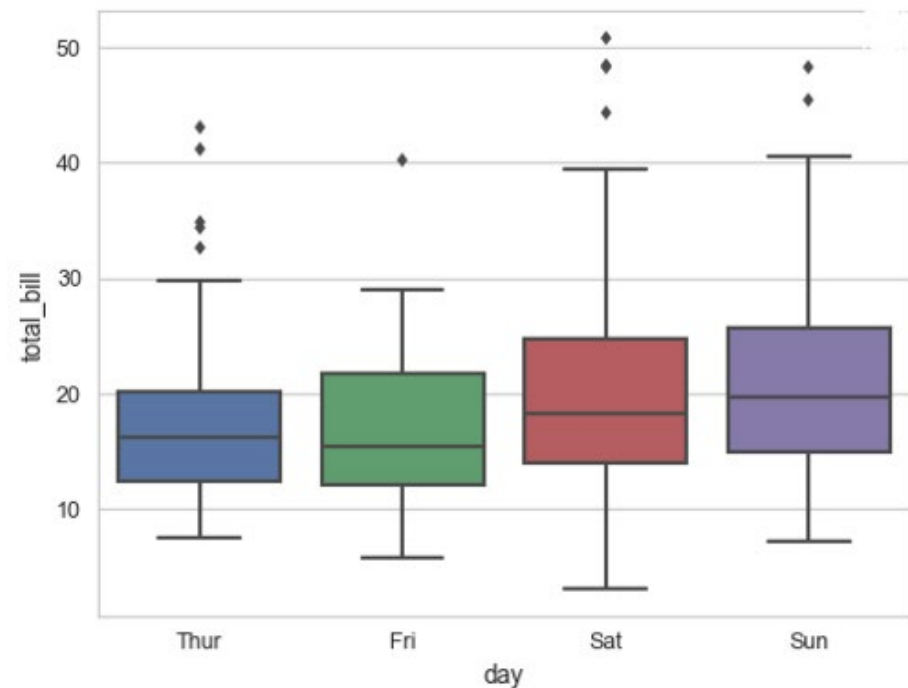
# Detección de outliers - ¿Cómo los identificamos?

■ Visualmente:

■ Con un scatter plot:

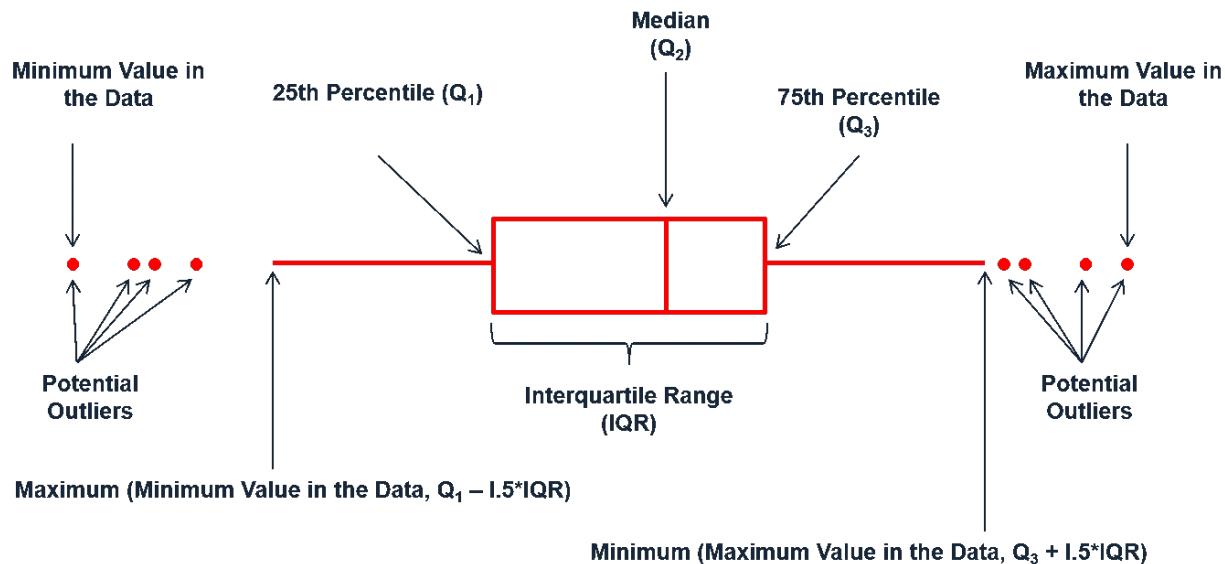


■ Con un boxplot:



# Detección de outliers - ¿Cómo los identificamos?

## ■ Recordemos...



corte inferior = primer cuartil - (rango intercuartil \* 1.5)

corte superior = tercer cuartil + (rango intercuartil \* 1.5)

# Detección de outliers - ¿Y en Python?

■ Pandas nos ofrece la función **.quantile()**, que es útil para cortar los outliers:

```
Q1 = dataset["col1"].quantile(0.25)
```

```
Q3 = dataset["col1"].quantile(0.75)
```

Por lo tanto, el rango intercuartil se calcula como:

```
IQR = Q3 - Q1
```

Así que:

```
corte_inferior = Q1 - (IQR * 1.5)
```

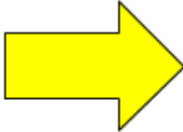
```
corte_superior = Q3 + (IQR * 1.5)
```

Ahora, simplemente tenemos que quedarnos con los datos que estén dentro de este rango del dataset:

```
dataset_filtrado = dataset[(dataset["col1"] >= corte_inferior) & (dataset["col1"] <= corte_superior )]
```

# Dummies

- Las variables dummy son variables numéricas que representan subgrupos de clases de un estudio.
- Son variables que únicamente adquieren el valor de 0 o 1, indicando la presencia o ausencia de una clase en una instancia concreta.
- Ejemplo:



Color
Red
Red
Yellow
Green
Yellow

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

# Escalado y normalización de datos

- Con el KNN vimos la importancia de normalizar y escalar los datos.
- ¿Cuál es la diferencia?
- Escalar los datos:
  - Datos entre 0 y 1.
- Normalizar los datos:
  - Datos de media 0 y desviación estándar 1.

$$X_{sc} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

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

```
from sklearn.preprocessing import MinMaxScaler  
X = MinMaxScaler().fit_transform(X)
```

```
from sklearn.preprocessing import StandardScaler  
X = StandardScaler().fit_transform(X)
```



# Escalado y normalización de datos

- En sklearn, también tenemos el módulo RobustScaler.
- Este módulo escala sin tener en cuenta los outliers:

```
from sklearn.preprocessing import RobustScaler  
X = RobustScaler().fit_transform(X)
```

# Reducción de características

- A veces, nos es útil ver nuestros datos representados.
- Hasta ahora hemos visto cómo hacer gráficos en 2D o en 3D, pero...
- ¿Qué pasa si nuestros datos tienen más de 2 dimensiones?
- PCA (Principal Component Analysis)
  - Va a ser útil para visualizar los datos, pero también para incrementar la velocidad de nuestros algoritmos si nuestro dataset tiene muchas features.
  - Es un método NO supervisado (no hay variable target).
  - La idea matemática es conseguir  $p$  variables ortogonales nuevas (**componentes principales**).
  - Cada una de ellas es una combinación lineal de las  $m$  variables originales, de forma que expliquen la **mayor cantidad de variabilidad** del conjunto de datos original posible.

# Reducción de características

- PCA busca maximizar la varianza.
- Valor y vector propio:
  - Los vectores propios **son conjuntos de elementos (vectores)** que mediante la multiplicación de una **constante** cualquiera, son equivalentes con la multiplicación de la matriz original y los conjuntos de elementos.

$$vA = vc,$$

- A es una matriz de mxm dimensiones.
- v es el vector propio.
- c es el valor propio.

# PCA

- Paso 1: **Estandarización.**
- Paso 2: Cálculo de la matriz de covarianzas.
- Paso 3: Cálculo de valores propios y vectores propios de la matriz de covarianza para identificar los componentes principales.
- Paso 4: Obtener el vector de características.
- Paso 5: Convertir los datos a sus componentes principales.

## Más información:

<https://towardsdatascience.com/a-one-stop-shop-for-principal-component-analysis-5582fb7e0a9c>

# T-SNE

Artículo original: Van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of machine learning research*, 9(11).

- Es otro método de reducción de dimensionalidad del dataset.
- También es no supervisado.
- Proceso matemático complejo, pero el resultado nos da una orientación de cómo están distribuidos los datos en un espacio de mayores dimensiones.
- Mientras que hemos dicho que PCA preserva las distancias grandes para maximizar la varianza, t-SNE preserva distancias pequeñas o locales:

**t-SNE reduce las dimensiones intentando mantener cerca las instancias similares y lejos las instancias más diferentes.**

Mayor coste computacional cuando se usan muchas variables de entrada.

- Solamente sirve para visualización o exploración de datos, pero no sirve para hacer reducción de dimensionalidad para clasificación.

<https://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf>

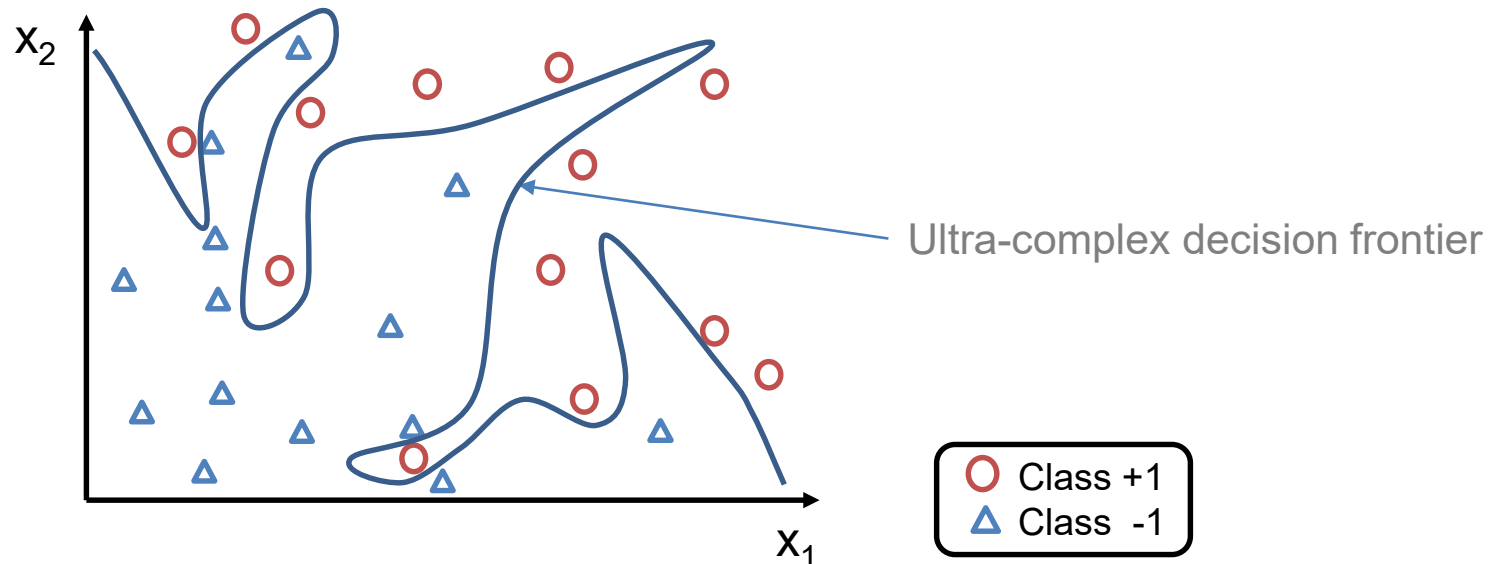
# Diferencias entre PCA y T-SNE

	PCA	t-SNE
1.	It is a linear Dimensionality reduction technique.	It is a non-linear Dimensionality reduction technique.
2.	It tries to preserve the global structure of the data.	It tries to preserve the local structure (cluster) of data.
3.	It does not work well as compared to t-SNE.	It is one of the best dimensionality reduction technique.
4.	It does not involve Hyperparameters.	It involves Hyperparameters such as perplexity, learning rate and number of steps.
5.	It gets highly affected by outliers.	It can handle outliers.
6.	PCA is a deterministic algorithm.	It is a non-deterministic or randomised algorithm.
7.	It works by rotating the vectors for preserving variance.	It works by minimising the distance between the point in a gaussian.
8.	We can find decide on how much variance to preserve using eigen values.	We cannot preserve variance instead we can preserve distance using hyperparameters.

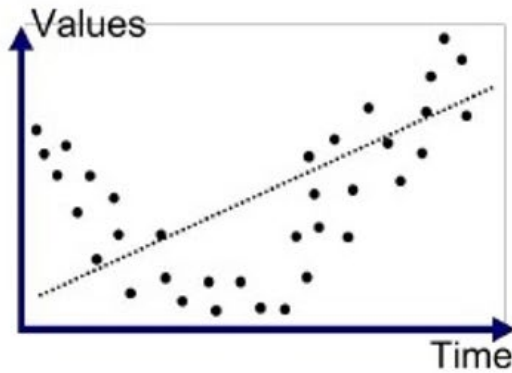
<https://www.geeksforgeeks.org/difference-between-pca-vs-t-sne/>

# Overfitting o sobreajuste

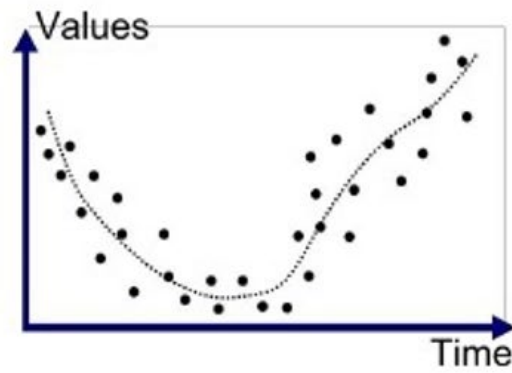
- El sobreajuste es la producción de un análisis que se corresponde demasiado o exactamente con un conjunto particular de datos.
- Por lo tanto, puede no ajustarse a datos adicionales o predecir futuras observaciones de manera confiable.
- Si tenemos **overfitting**, el éxito al responder las muestras de entrenamiento es muy alto, mientras que la actuación del modelo con muestras es mucho peor.



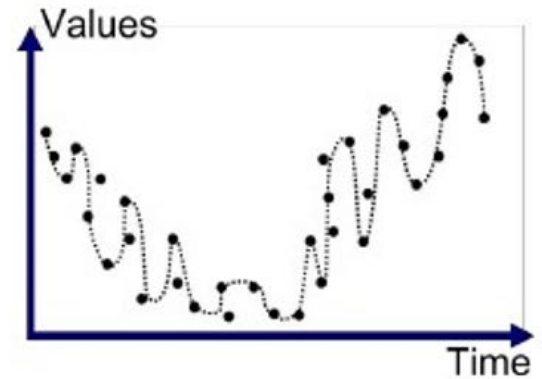
# Overfitting



Underfitted



Good Fit/Robust



Overfitted



## ¿Por qué ocurre?

- El overfitting puede tener muchas causas.
- Normalmente, es una combinación de las siguientes:
  - Modelo demasiado potente para el problema en cuestión: por ejemplo, permite polinomios hasta el grado 100. Tal vez, con polinomios más pequeños sería menos propenso al sobreajuste...
  - No hay suficientes datos: el modelo no ha podido entrenarse correctamente porque no había suficientes datos de entrenamiento.
  - Para asegurarnos de no tener overfitting, siempre vamos a cross-validar nuestros resultados.
  - Otra manera de no tener overfitting será usar **ensemble methods**:
    - Bagging: (e.g. Random Forest)
    - Boosting: (e.g. Gradient Boosted Trees)