Preprocessing

Preprocessing

Encodage

$$\begin{bmatrix} chien \\ chat \\ chien \\ oiseau \end{bmatrix} \rightarrow \begin{bmatrix} 0 \\ 1 \\ 0 \\ 2 \end{bmatrix}$$

Preprocessing

Encodage

$$\begin{bmatrix} chien \\ chat \\ chien \\ oiseau \end{bmatrix} \rightarrow \begin{bmatrix} 0 \\ 1 \\ 0 \\ 2 \end{bmatrix}$$

Normalisation

$$\begin{bmatrix} 2 \\ 10 \\ 4 \\ 6 \end{bmatrix} \rightarrow \begin{bmatrix} 0 \\ 1 \\ 0.25 \\ 0.5 \end{bmatrix}$$

Preprocessing

Encodage

$$\begin{bmatrix} chien \\ chat \\ chien \\ oiseau \end{bmatrix} \rightarrow \begin{bmatrix} 0 \\ 1 \\ 0 \\ 2 \end{bmatrix}$$

Normalisation

$$\begin{bmatrix} 2 \\ 10 \\ 4 \\ 6 \end{bmatrix} \rightarrow \begin{bmatrix} 0 \\ 1 \\ 0.25 \\ 0.5 \end{bmatrix}$$

Imputation

$$\begin{bmatrix} 2 \\ 1 \\ 3 \\ 'nan' \end{bmatrix} \rightarrow \begin{bmatrix} 2 \\ 1 \\ 3 \\ 2 \end{bmatrix}$$

Preprocessing

Encodage

$$\begin{bmatrix} chien \\ chat \\ chien \\ oiseau \end{bmatrix} \rightarrow \begin{bmatrix} 0 \\ 1 \\ 0 \\ 2 \end{bmatrix}$$

Normalisation

$$\begin{bmatrix} 2 \\ 10 \\ 4 \\ 6 \end{bmatrix} \rightarrow \begin{bmatrix} 0 \\ 1 \\ 0.25 \\ 0.5 \end{bmatrix}$$

Imputation

$$\begin{bmatrix} 2 \\ 1 \\ 3 \\ 'nan' \end{bmatrix} \rightarrow \begin{bmatrix} 2 \\ 1 \\ 3 \\ 2 \end{bmatrix}$$

Sélection

$$\begin{bmatrix} \boxed{1} & \cancel{0} \\ \boxed{2} & \cancel{1} \\ \boxed{3} & \cancel{0} \\ \boxed{4} & \cancel{0} \end{bmatrix} \rightarrow \begin{bmatrix} 1 \\ 1 \\ 3 \\ 4 \end{bmatrix}$$

Preprocessing

Encodage

$$\begin{bmatrix} chien \\ chat \\ chien \\ oiseau \end{bmatrix} \rightarrow \begin{bmatrix} 0 \\ 1 \\ 0 \\ 2 \end{bmatrix}$$

Normalisation

$$\begin{bmatrix} 2 \\ 10 \\ 4 \\ 6 \end{bmatrix} \rightarrow \begin{bmatrix} 0 \\ 1 \\ 0.25 \\ 0.5 \end{bmatrix}$$

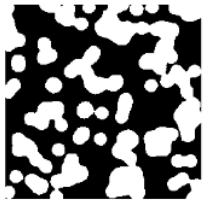
Imputation

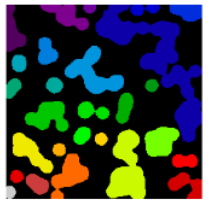
$$\begin{bmatrix} 2 \\ 1 \\ 3 \\ 'nan' \end{bmatrix} \rightarrow \begin{bmatrix} 2 \\ 1 \\ 3 \\ 2 \end{bmatrix}$$

Sélection

$$\begin{bmatrix} 1 & \cancel{0} \\ 2 & \cancel{1} \\ 3 & \cancel{0} \\ 4 & \cancel{0} \end{bmatrix} \rightarrow \begin{bmatrix} 1 \\ 1 \\ 3 \\ 4 \end{bmatrix}$$

Extraction





Preprocessing

Encodage

$$\begin{bmatrix} chien \\ chat \\ chien \\ oiseau \end{bmatrix} \rightarrow \begin{bmatrix} 0 \\ 1 \\ 0 \\ 2 \end{bmatrix}$$

Normalisation

$$\begin{bmatrix} 2 \\ 10 \\ 4 \\ 6 \end{bmatrix} \rightarrow \begin{bmatrix} 0 \\ 1 \\ 0.25 \\ 0.5 \end{bmatrix}$$

sklearn.**preprocessing**

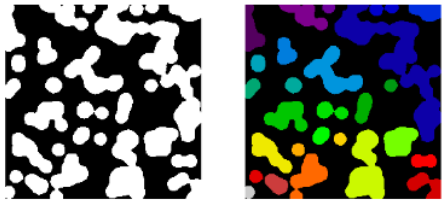
Imputation

$$\begin{bmatrix} 2 \\ 1 \\ 3 \\ 'nan' \end{bmatrix} \rightarrow \begin{bmatrix} 2 \\ 1 \\ 3 \\ 2 \end{bmatrix}$$

Sélection

$$\begin{bmatrix} \boxed{\begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix}} & \begin{matrix} 0 \\ 1 \\ 0 \\ 0 \end{matrix} \end{bmatrix} \rightarrow \begin{bmatrix} 1 \\ 1 \\ 3 \\ 4 \end{bmatrix}$$

Extraction



Preprocessing

Encodage

$$\begin{bmatrix} chien \\ chat \\ chien \\ oiseau \end{bmatrix} \rightarrow \begin{bmatrix} 0 \\ 1 \\ 0 \\ 2 \end{bmatrix}$$

Normalisation

$$\begin{bmatrix} 2 \\ 10 \\ 4 \\ 6 \end{bmatrix} \rightarrow \begin{bmatrix} 0 \\ 1 \\ 0.25 \\ 0.5 \end{bmatrix}$$

sklearn.**preprocessing**

Imputation

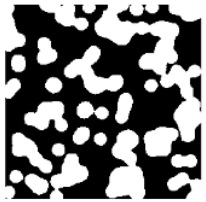
$$\begin{bmatrix} 2 \\ 1 \\ 3 \\ 'nan' \end{bmatrix} \rightarrow \begin{bmatrix} 2 \\ 1 \\ 3 \\ 2 \end{bmatrix}$$

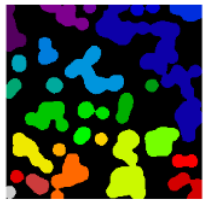
sklearn.**impute**

Sélection

$$\begin{bmatrix} \boxed{1} & \cancel{0} \\ \boxed{2} & \cancel{1} \\ \boxed{3} & \cancel{0} \\ \boxed{4} & \cancel{0} \end{bmatrix} \rightarrow \begin{bmatrix} 1 \\ 1 \\ 3 \\ 4 \end{bmatrix}$$

Extraction





Preprocessing

Encodage

$$\begin{bmatrix} chien \\ chat \\ chien \\ oiseau \end{bmatrix} \rightarrow \begin{bmatrix} 0 \\ 1 \\ 0 \\ 2 \end{bmatrix}$$

Normalisation

$$\begin{bmatrix} 2 \\ 10 \\ 4 \\ 6 \end{bmatrix} \rightarrow \begin{bmatrix} 0 \\ 1 \\ 0.25 \\ 0.5 \end{bmatrix}$$

sklearn.**preprocessing**

Imputation

$$\begin{bmatrix} 2 \\ 1 \\ 3 \\ 'nan' \end{bmatrix} \rightarrow \begin{bmatrix} 2 \\ 1 \\ 3 \\ 2 \end{bmatrix}$$

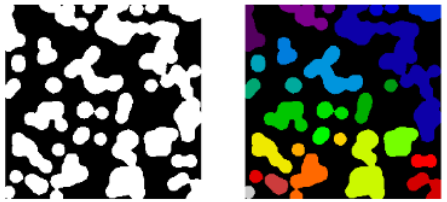
sklearn.**impute**

Sélection

$$\begin{bmatrix} \boxed{1} & \cancel{0} \\ 2 & \cancel{1} \\ 3 & \cancel{0} \\ 4 & \cancel{0} \end{bmatrix} \rightarrow \begin{bmatrix} 1 \\ 1 \\ 3 \\ 4 \end{bmatrix}$$

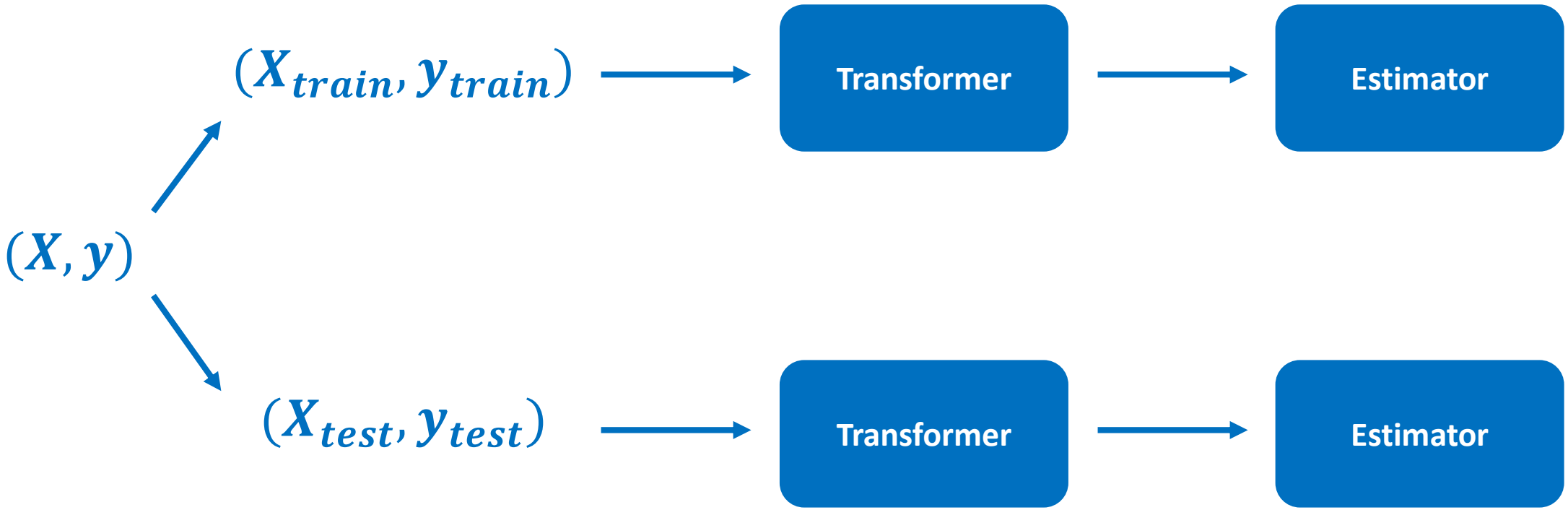
sklearn
feature_selection

Extraction



sklearn
feature_extraction

Transformer et Estimator

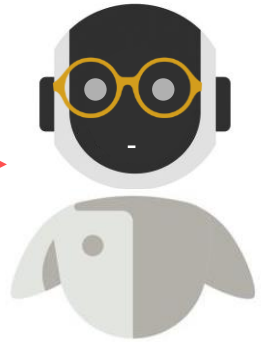


Transformer

fit(X_{train}) : développe une fonction de transformation à partir de X_{train}

Chat : 0
Chien : 1
Oiseau : 2

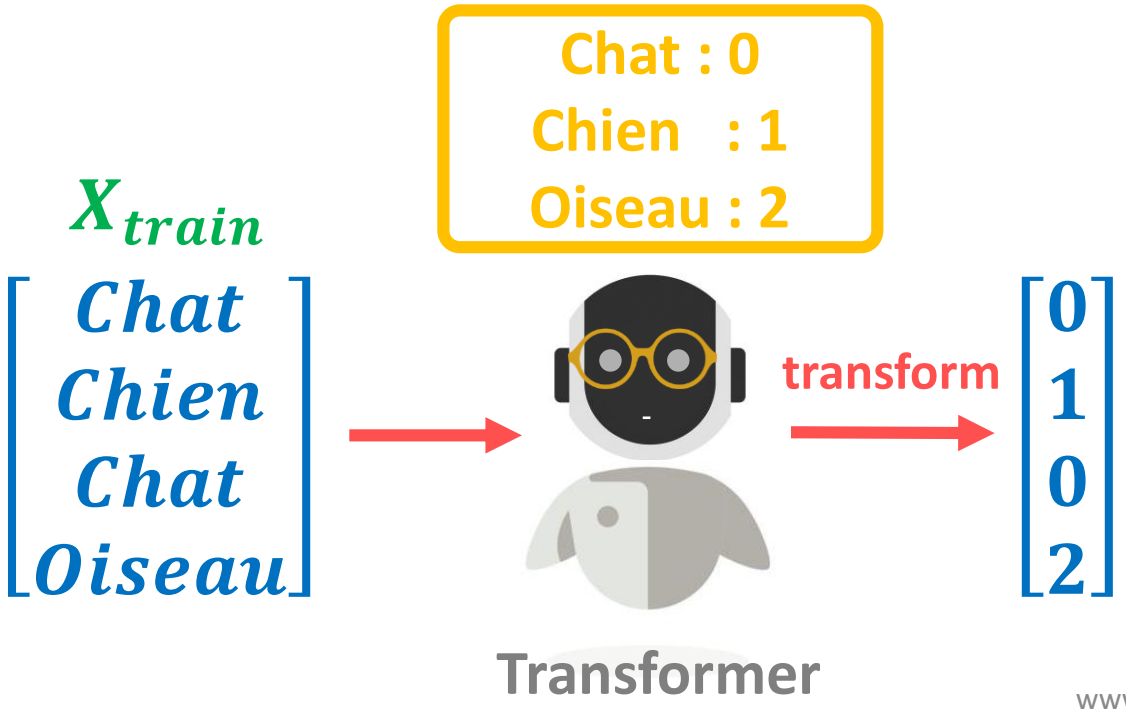
$$X_{train} \begin{bmatrix} Chat \\ Chien \\ Chat \\ Oiseau \end{bmatrix} \xrightarrow{\text{fit}}$$



Transformer

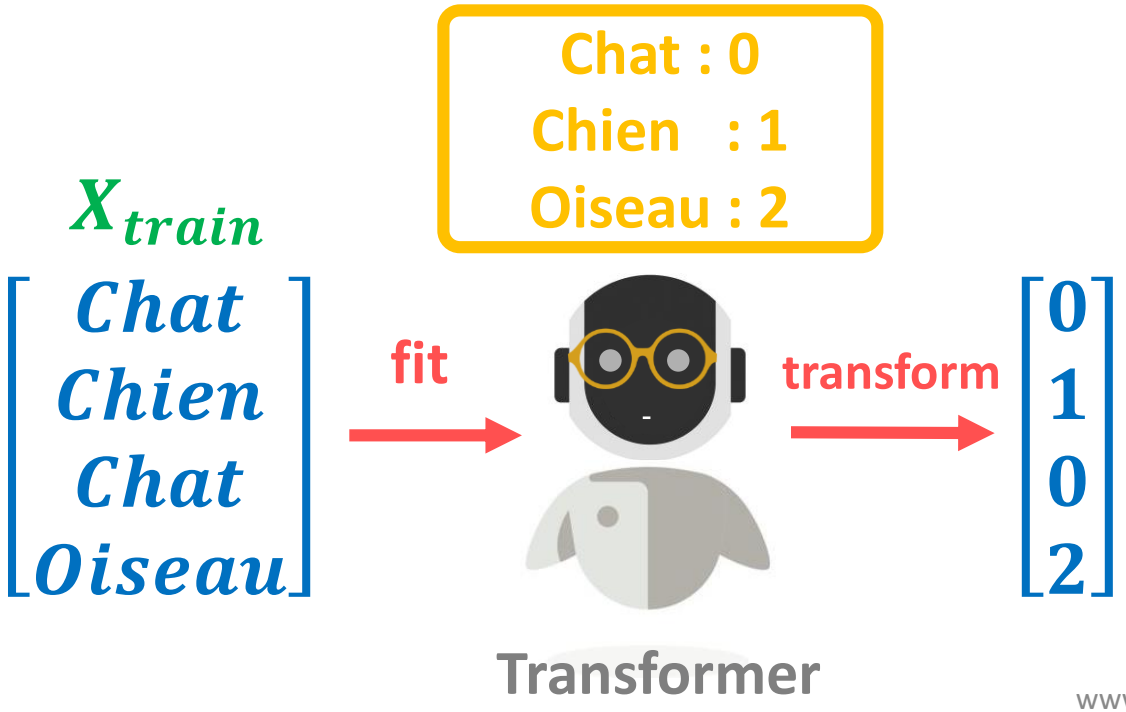
Transformer

transform(X) : applique la **transformation** sur les données X_{train}, X_{test} et toutes autres données futures.



Transformer

fit_transform(X_{train}) : **développe** la fonction de **transformation** puis **l'utilise** pour transformer X_{train}



Encodage Ordinal

Associe chaque catégorie ou classe d'une variable à une valeur **décimale unique**.

Encodage Ordinal

Associe chaque catégorie ou classe d'une variable à une valeur **décimale unique**.

$$\begin{bmatrix} \textit{Chat} \\ \textit{Chat} \\ \textit{Chien} \\ \textit{Oiseau} \\ \textit{Chien} \end{bmatrix} \begin{matrix} \text{----------} \\ \text{----------} \\ \text{----------} \\ \text{----------} \\ \text{----------} \end{matrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 2 \\ 1 \end{bmatrix}$$

Encodage Ordinal

Associe chaque catégorie ou classe d'une variable à une valeur **décimale unique**.

$$\begin{bmatrix} Chat \\ Chat \\ Chien \\ Oiseau \\ Chien \end{bmatrix} \begin{matrix} \text{----------} \\ \text{----------} \\ \text{----------} \\ \text{----------} \\ \text{----------} \end{matrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 2 \\ 1 \end{bmatrix}$$

→ LabelEncoder()

→ OrdinalEncoder()

Encodage Ordinal

$$\begin{bmatrix} Chat \\ Chat \\ Chien \\ Oiseau \\ Chien \end{bmatrix} \begin{matrix} \text{----------} \\ \text{----------} \\ \text{----------} \\ \text{----------} \\ \text{----------} \end{matrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 2 \\ 1 \end{bmatrix}$$

$$0 < 1 < 2$$

$$\Rightarrow Chat < Chien < Oiseau$$

??? Ca n'a pas de sens...

Encodage One Hot

$$\begin{bmatrix} Chat \\ Chat \\ Chien \\ Oiseau \\ Chien \end{bmatrix} \begin{array}{c} \\ \text{----------} \\ \text{----------} \\ \text{----------} \\ \text{----------} \\ \text{----------} \end{array} \begin{array}{c} \begin{array}{ccc} Chat & Chien & Oiseau \end{array} \\ \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \end{array}$$

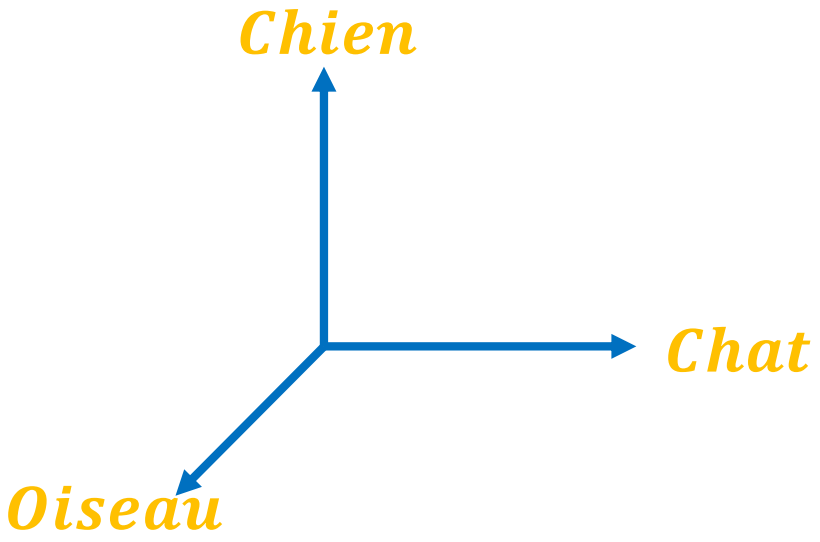
→ LabelBinarizer()

→ MultiLabelBinarizer()

→ OneHotEncoder()

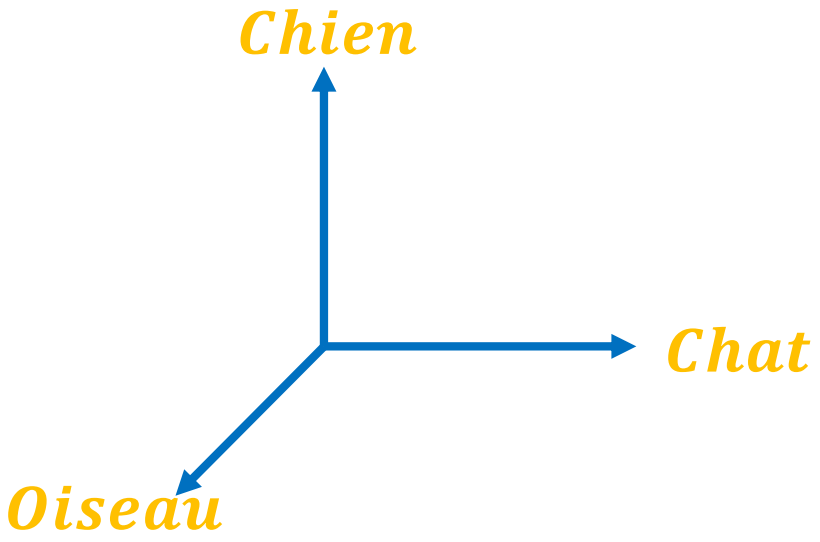
Encodage One Hot

$$\begin{bmatrix} Chat \\ Chat \\ Chien \\ Oiseau \\ Chien \end{bmatrix} \begin{array}{c} \\ \\ \\ \\ \\ \end{array} \begin{array}{c} \begin{array}{ccc} Chat & Chien & Oiseau \end{array} \\ \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \end{array}$$



Encodage One Hot

$$\begin{bmatrix} Chat \\ Chat \\ Chien \\ Oiseau \\ Chien \end{bmatrix} \begin{array}{c} \\ \\ \\ \\ \\ \end{array} \begin{array}{c} \begin{array}{ccc} Chat & Chien & Oiseau \end{array} \\ \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \end{array}$$



→ LabelBinarizer()

→ MultiLabelBinarizer()

→ OneHotEncoder()

Encodage One Hot

$$
\begin{bmatrix} \textit{Paris} \\ \textit{Marseille} \\ \textit{Lyon} \\ \textit{Toulouse} \\ \textit{Paris} \\ \textit{Paris} \\ \textit{Lille} \\ \textit{Toulouse} \\ \textit{Bordeaux} \\ \dots \end{bmatrix}
\longrightarrow
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots\ \dots\ \dots & 1 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & & 0
\end{bmatrix}
$$

Encodage One Hot

$$
\begin{bmatrix} \textit{Paris} \\ \textit{Marseille} \\ \textit{Lyon} \\ \textit{Toulouse} \\ \textit{Paris} \\ \textit{Paris} \\ \textit{Lille} \\ \textit{Toulouse} \\ \textit{Bordeaux} \\ \dots \end{bmatrix}
\longrightarrow
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots\ \dots\ \dots & 1 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & & 0
\end{bmatrix}
$$

Sparse Matrix

$$\begin{bmatrix} a & 0 & 0 & 0 & 0 \\ 0 & 0 & b & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & c \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Sparse Matrix

$$\begin{bmatrix} a & 0 & 0 & 0 & 0 \\ 0 & 0 & b & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & c \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{aligned} values &= [\quad\quad] \\ rows &= [\quad\quad] \\ cols &= [\quad\quad] \end{aligned}$$

Sparse Matrix

$$\begin{bmatrix} a & 0 & 0 & 0 & 0 \\ 0 & 0 & b & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & c \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{aligned} values &= [a, b, c] \\ rows &= [\quad\quad] \\ cols &= [\quad\quad] \end{aligned}$$

Sparse Matrix

$$
\begin{array}{cc}
 & \begin{array}{ccccc} 0 & & & & \end{array} \\
\begin{array}{c} 0 \\ \\ \\ \\ \\ \end{array} & \begin{bmatrix} \boxed{a} & 0 & 0 & 0 & 0 \\ 0 & 0 & b & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & c \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}
\end{array}
$$

$$
\begin{aligned}
values &= [a, b, c] \\
rows &= [0 \quad\quad\quad] \\
cols &= [0 \quad\quad\quad]
\end{aligned}
$$

Sparse Matrix

$$
\begin{array}{cc}
 & \begin{array}{ccccc} & & 2 & & \end{array} \\
1 & \begin{bmatrix} a & 0 & 0 & 0 & 0 \\ 0 & 0 & \boxed{b} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & c \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}
\end{array}
$$

$$
\begin{aligned}
values &= [a, b, c] \\
rows &= [0, 1 \quad] \\
cols &= [0, 2 \quad]
\end{aligned}
$$

Sparse Matrix

$$
\begin{array}{cc}
 & \begin{array}{ccccc} & & & & 4 \end{array} \\
3 & \begin{bmatrix} a & 0 & 0 & 0 & 0 \\ 0 & 0 & b & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \boxed{c} \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}
\end{array}
$$

$$
\begin{aligned}
values &= [a, b, c] \\
rows &= [0, 1, 3] \\
cols &= [0, 2, 4]
\end{aligned}
$$

Encoding

```
preprocessing.Binarizer([threshold, copy])
preprocessing.FunctionTransformer([func, ...])
preprocessing.KBinsDiscretizer([n_bins, ...])
preprocessing.KernelCenterer()
preprocessing.LabelBinarizer([neg_label, ...])
preprocessing.LabelEncoder
preprocessing.MultiLabelBinarizer([classes, ...])
preprocessing.MaxAbsScaler([copy])
preprocessing.MinMaxScaler([feature_range, copy])
preprocessing.Normalizer([norm, copy])
preprocessing.OneHotEncoder([categories, ...])
preprocessing.OrdinalEncoder([categories, dtype])
preprocessing.PolynomialFeatures([degree, ...])
preprocessing.PowerTransformer([method, ...])
preprocessing.QuantileTransformer([...])
preprocessing.RobustScaler([with_centering, ...])
preprocessing.StandardScaler([copy, ...])
```

Encoding

```
preprocessing.Binarizer([threshold, copy])
preprocessing.FunctionTransformer([func, ...])
preprocessing.KBinsDiscretizer([n_bins, ...])
preprocessing.KernelCenterer()
preprocessing.LabelBinarizer([neg_label, ...])
preprocessing.LabelEncoder
preprocessing.MultiLabelBinarizer([classes, ...])
preprocessing.MaxAbsScaler([copy])
preprocessing.MinMaxScaler([feature_range, copy])
preprocessing.Normalizer([norm, copy])
preprocessing.OneHotEncoder([categories, ...])
preprocessing.OrdinalEncoder([categories, dtype])
preprocessing.PolynomialFeatures([degree, ...])
preprocessing.PowerTransformer([method, ...])
preprocessing.QuantileTransformer([...])
preprocessing.RobustScaler([with_centering, ...])
preprocessing.StandardScaler([copy, ...])
```

	Encodage Ordinal	Encodage One-Hot
y	**LabelEncoder**	**LabelBinarizer** (et MultiLabelBinarizer)
X	**OrdinalEncoder**	**OneHotEncoder**

Label Encoder

Encode **chaque classe** de la ***variable y*** en une **valeur numérique** *(0, n_classe-1)*

$$y$$

$$\begin{bmatrix} Chat \\ Chien \\ Chat \\ Oiseau \end{bmatrix} \rightarrow \begin{bmatrix} 0 \\ 1 \\ 0 \\ 2 \end{bmatrix}$$

*Note : la méthode **inverse_transform** permet de **<u>décoder</u>** les données*

Ordinal Encoder

Encode **les catégories des** $\boldsymbol{variables\ X}$ en **valeurs numériques** *(0, n_classe-1)*

$$\begin{array}{cc} X_1 & X_2 \\ \end{array}$$
$$\begin{bmatrix} Chat & Poils \\ Chien & Poils \\ Chat & Poils \\ Oiseau & Plumes \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 1 \\ 1 & 1 \\ 0 & 1 \\ 2 & 0 \end{bmatrix}$$

Note : C'est l'équivalent de ***LabelEncoder****, mais pour les* $\boldsymbol{features\ X}$

Label Binarizer

Encode **chaque classe** de la $\boldsymbol{variable\ y}$ en **One-Hot**

$$\begin{array}{c} \boldsymbol{y} \\ \begin{bmatrix} \boldsymbol{Chat} \\ \boldsymbol{Chien} \\ \boldsymbol{Chat} \\ \boldsymbol{Oiseau} \end{bmatrix} \end{array} \rightarrow \begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}$$

*Note : la méthode **inverse_transform** permet de **<u>décoder</u>** les données*

OneHot Encoder

Encode **les catégories des** $variables\ X$ en **One-Hot**

$$\begin{array}{cc} X_1 & X_2 \\ \end{array}$$

$$\begin{bmatrix} Chat & Poils \\ Chien & Poils \\ Chat & Poils \\ Oiseau & Plumes \end{bmatrix} \rightarrow \begin{array}{c} \begin{array}{ccccc} Chat & Chien & Oiseau & Poils & Plumes \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix} \end{array}$$

Note : C'est l'équivalent de ***LabelBinarizer***, mais pour les $features\ X$

Encoding

	Encodage Ordinal	Encodage One-Hot
y	**LabelEncoder**	**LabelBinarizer** (et MultiLabelBinarizer)
X	**OrdinalEncoder**	**OneHotEncoder**

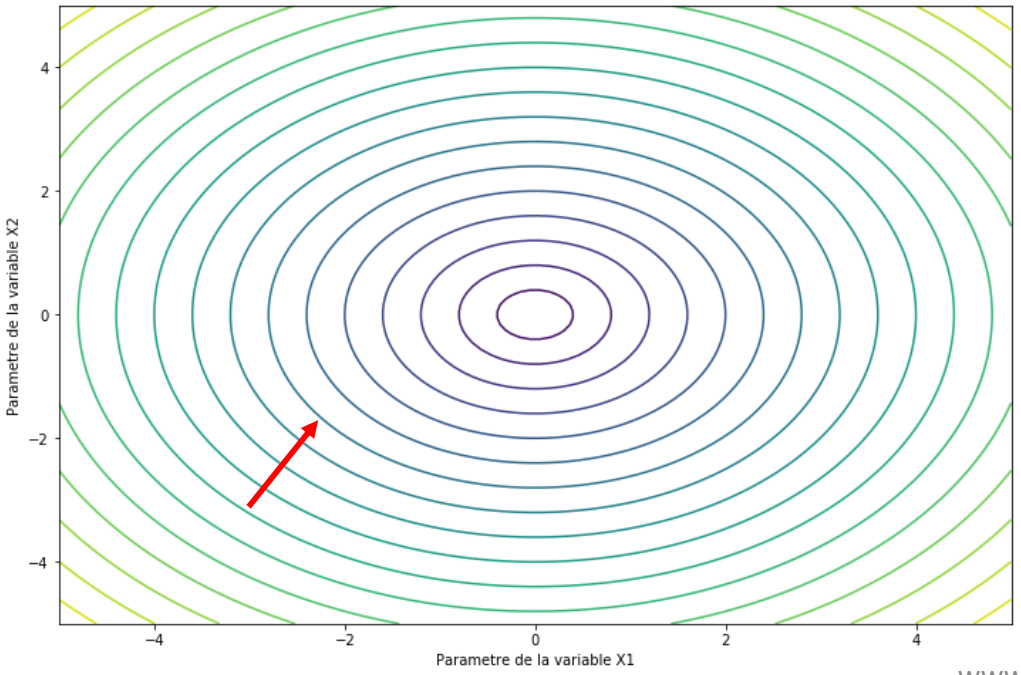
Notes *:*

- ***Inverse_transform*** *pour vos prédictions*
- ***sparse=True*** *pour OneHotEncoder*
- *Attention : Sklearn v22.0 gère mal les catégories inconnues.*

Normalisation

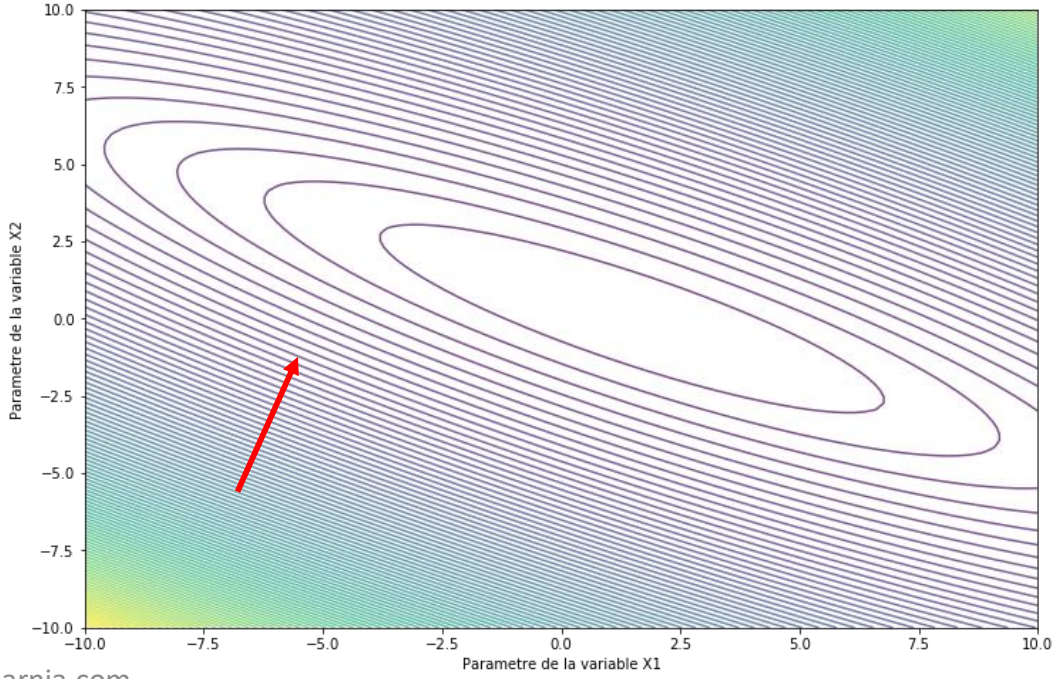
Avec Normalisation

Fonction Cout



Sans Normalisation

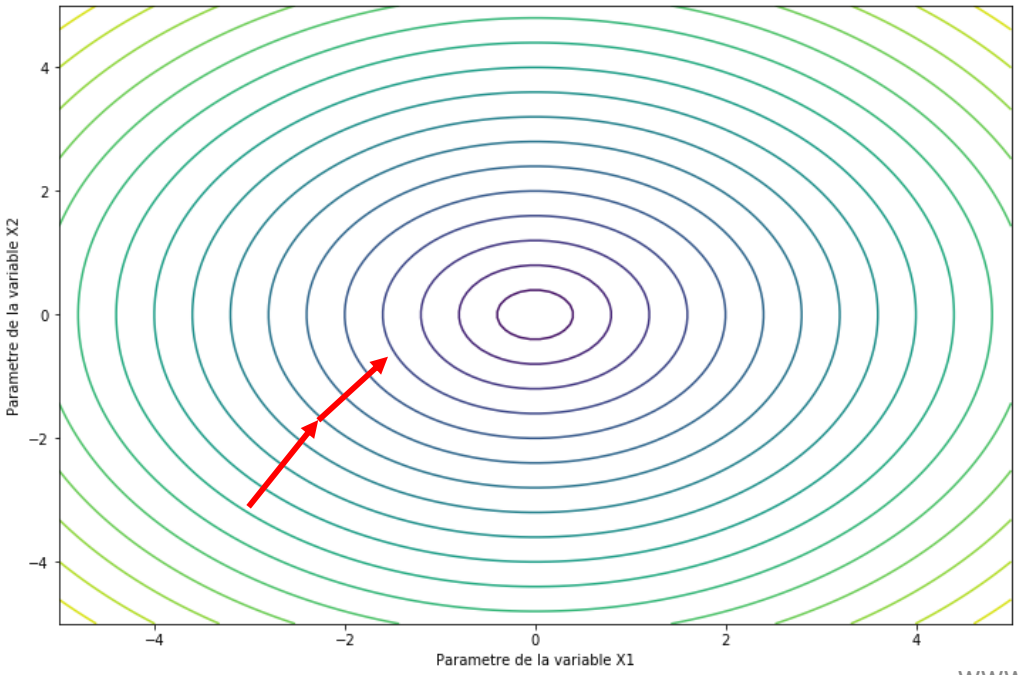
Fonction Cout



Normalisation

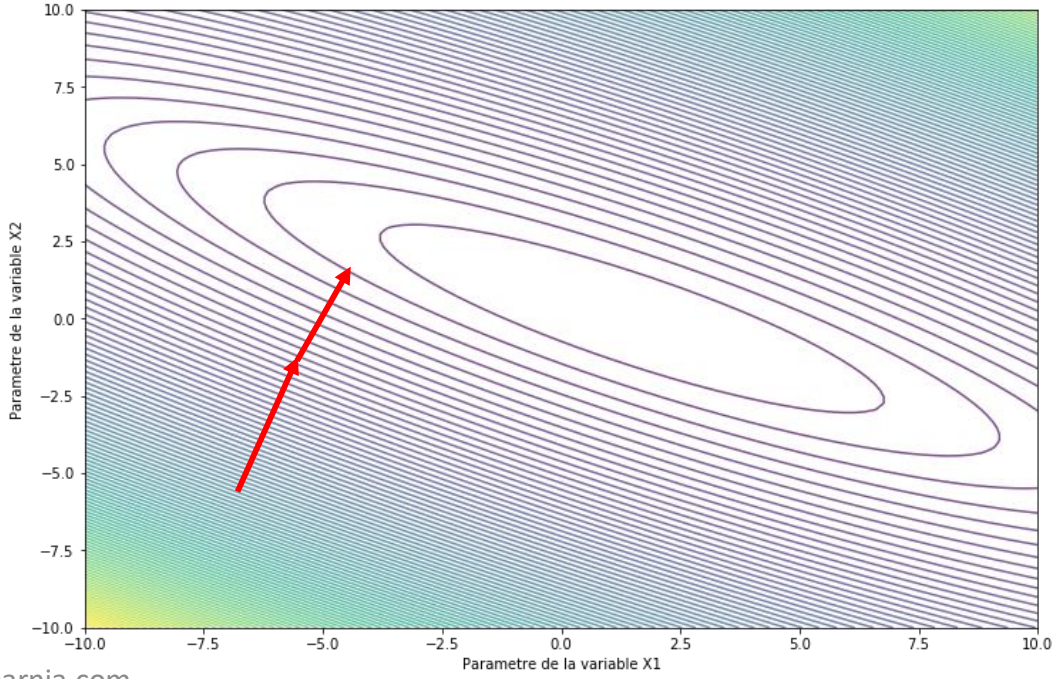
Avec Normalisation

Fonction Cout



Sans Normalisation

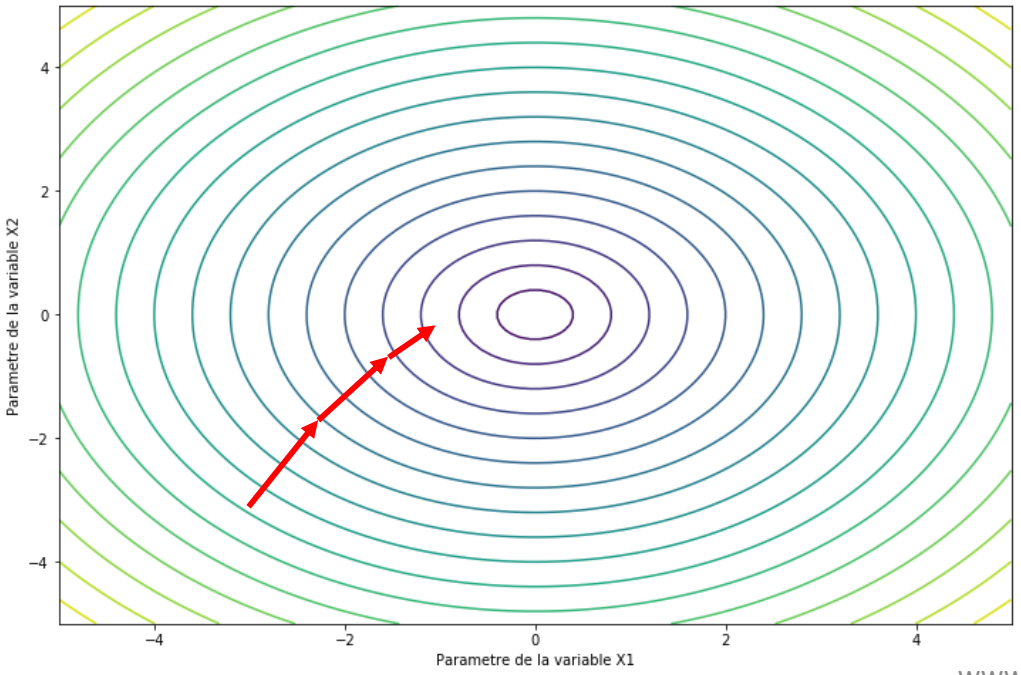
Fonction Cout



Normalisation

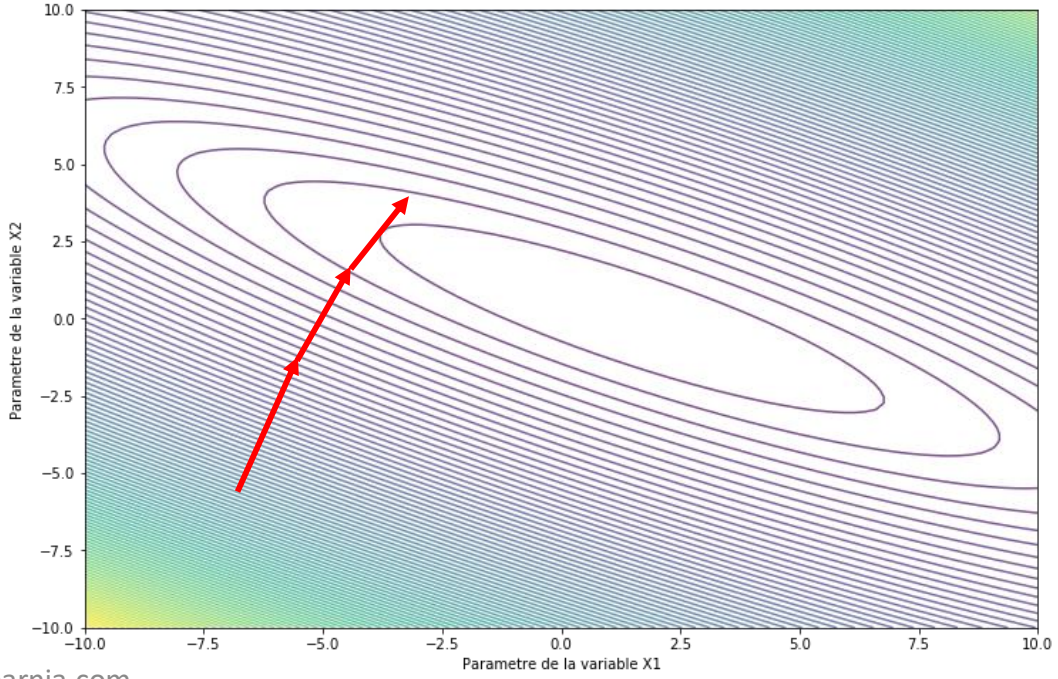
Avec Normalisation

Fonction Cout



Sans Normalisation

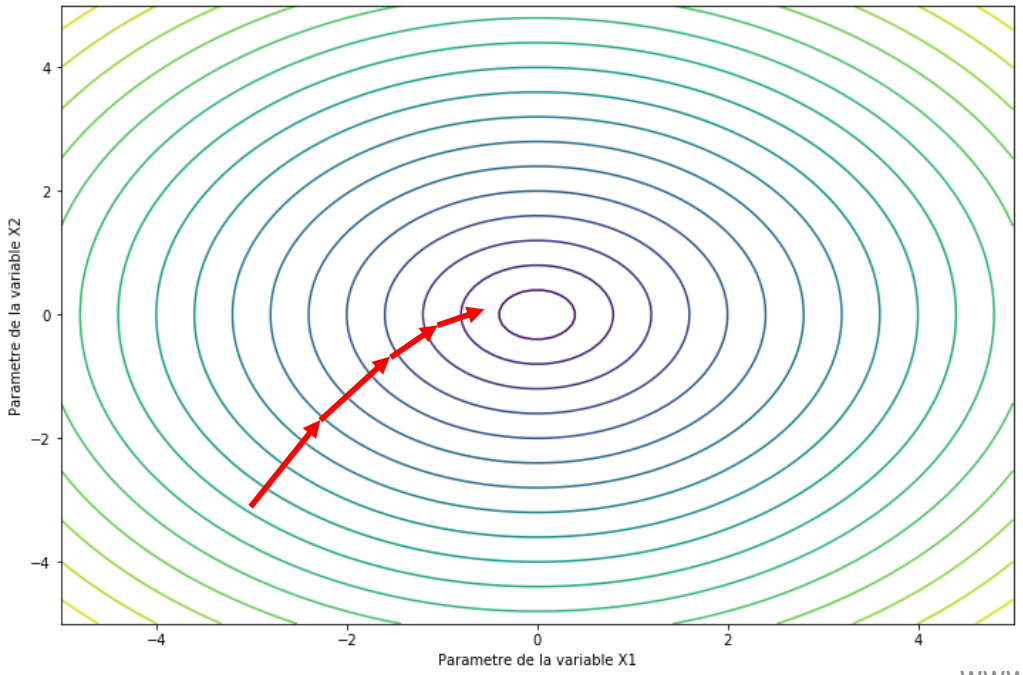
Fonction Cout



Normalisation

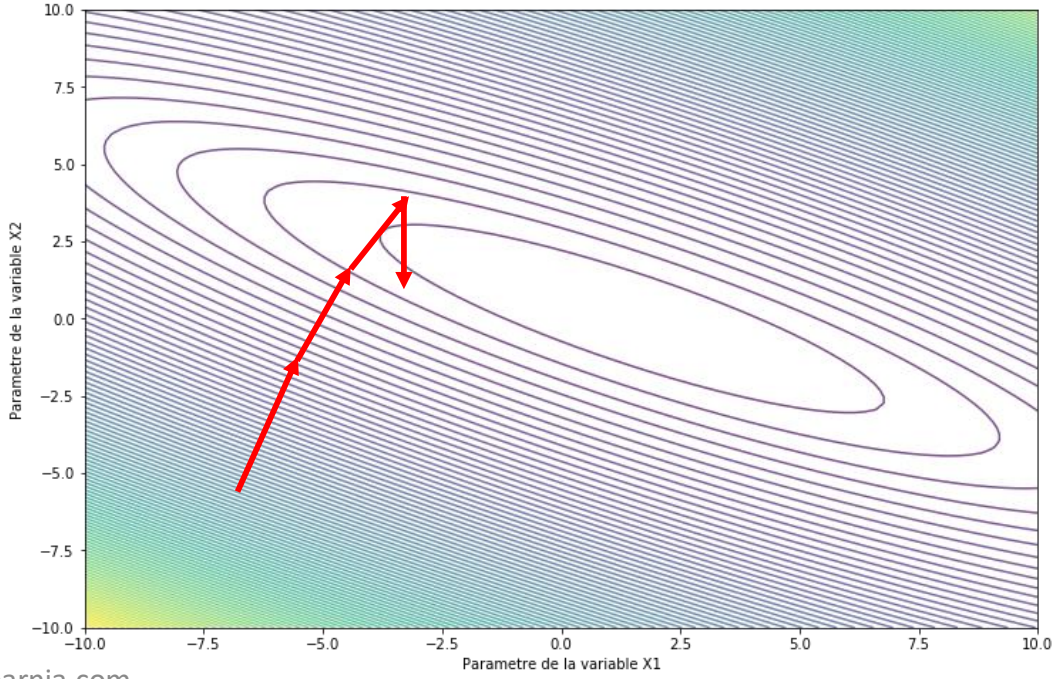
Avec Normalisation

Fonction Cout



Sans Normalisation

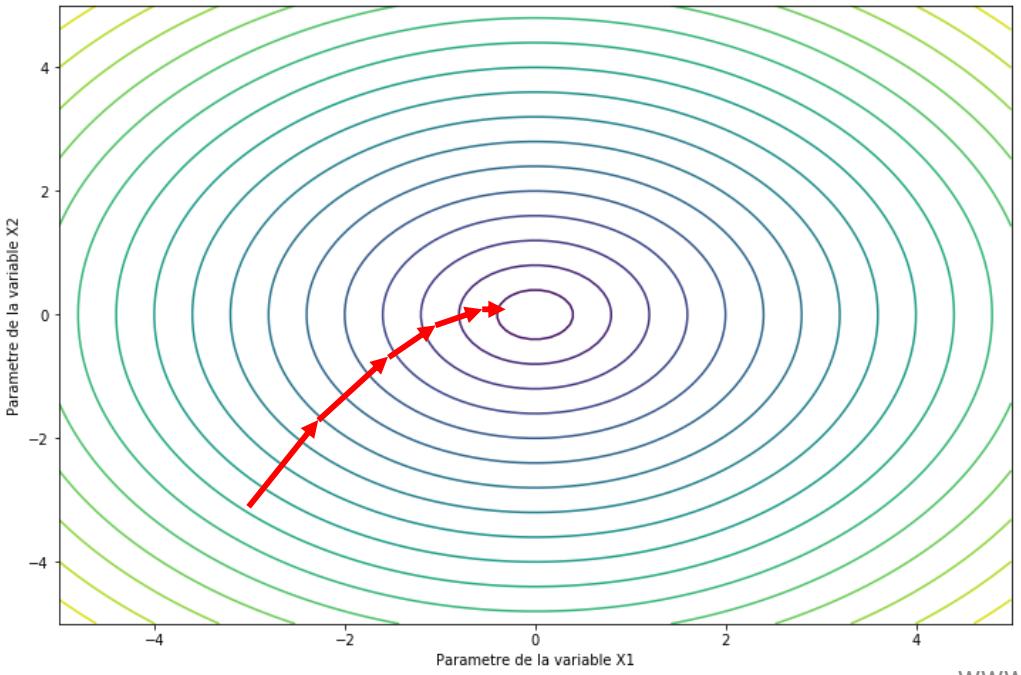
Fonction Cout



Normalisation

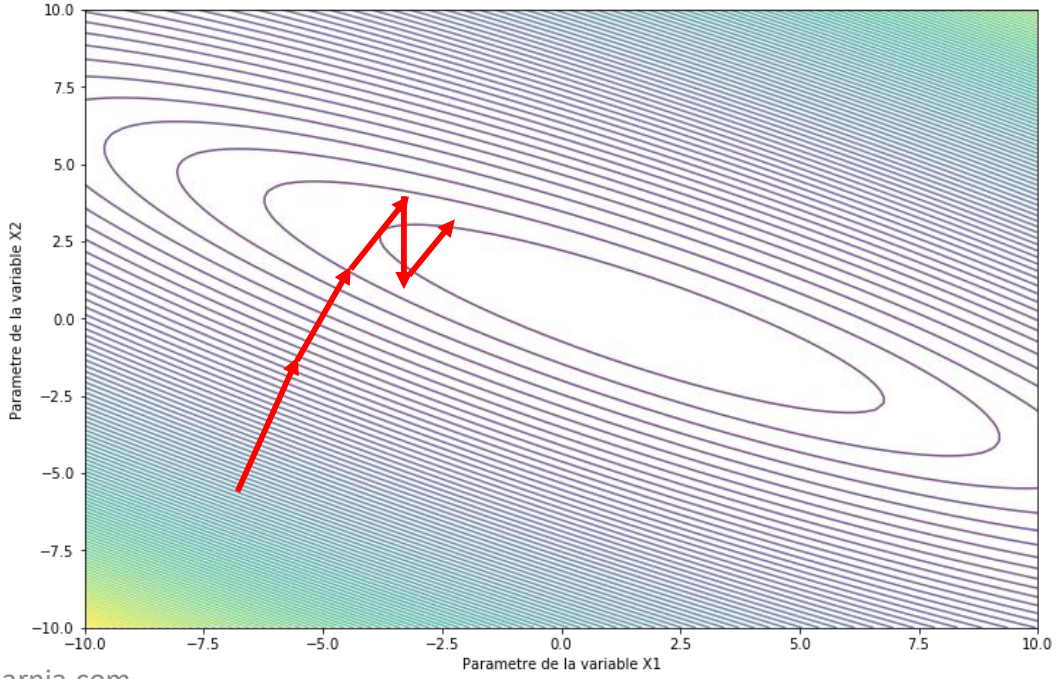
Avec Normalisation

Fonction Cout



Sans Normalisation

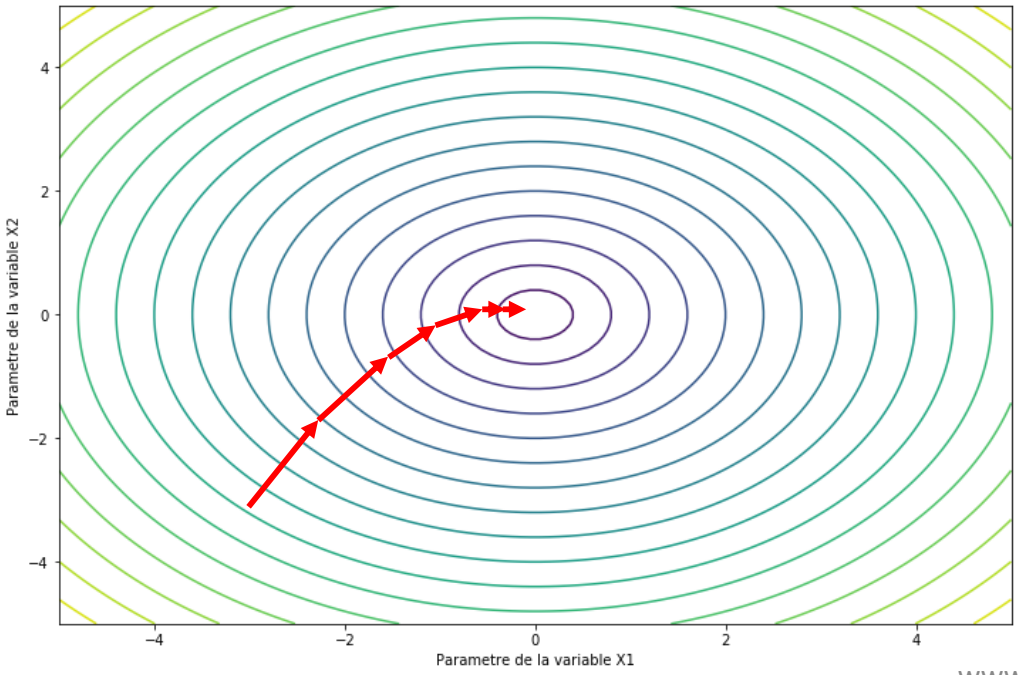
Fonction Cout



Normalisation

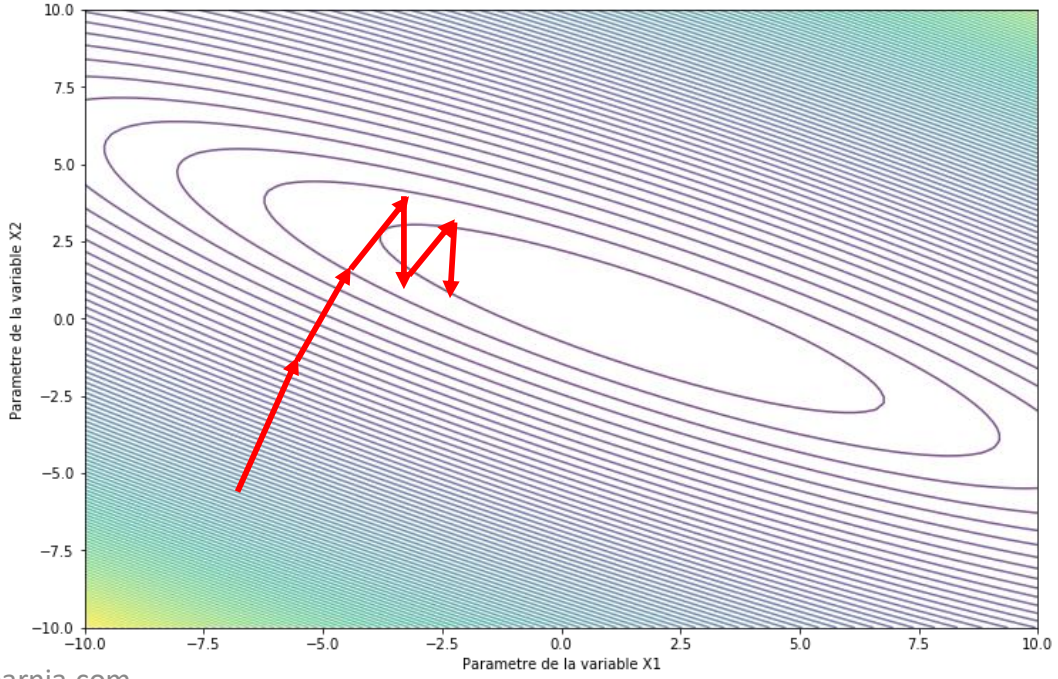
Avec Normalisation

Fonction Cout



Sans Normalisation

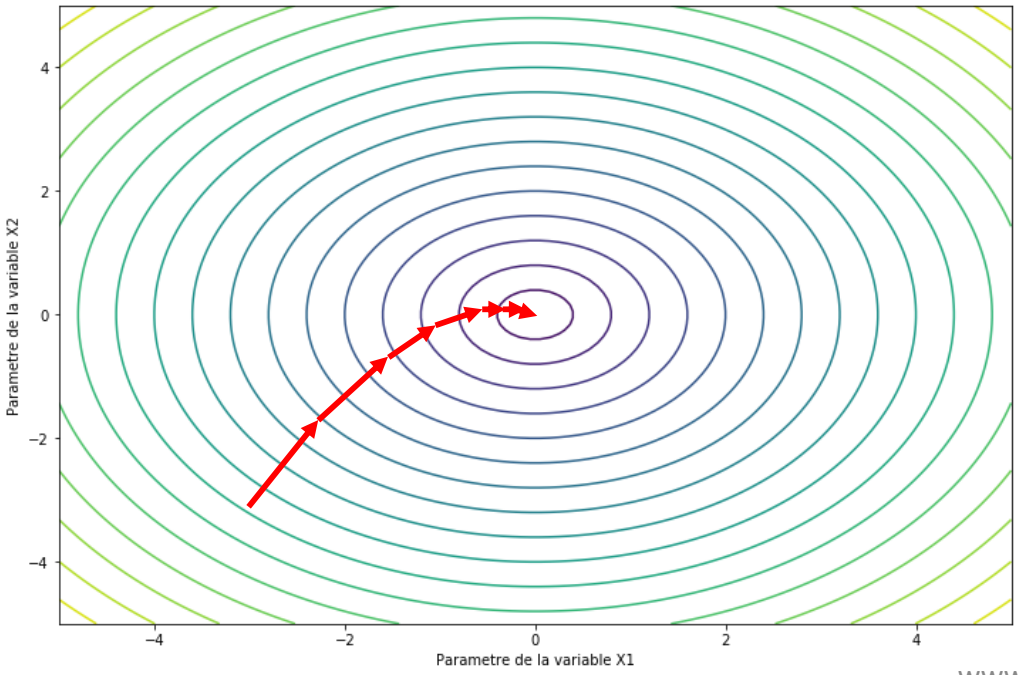
Fonction Cout



Normalisation

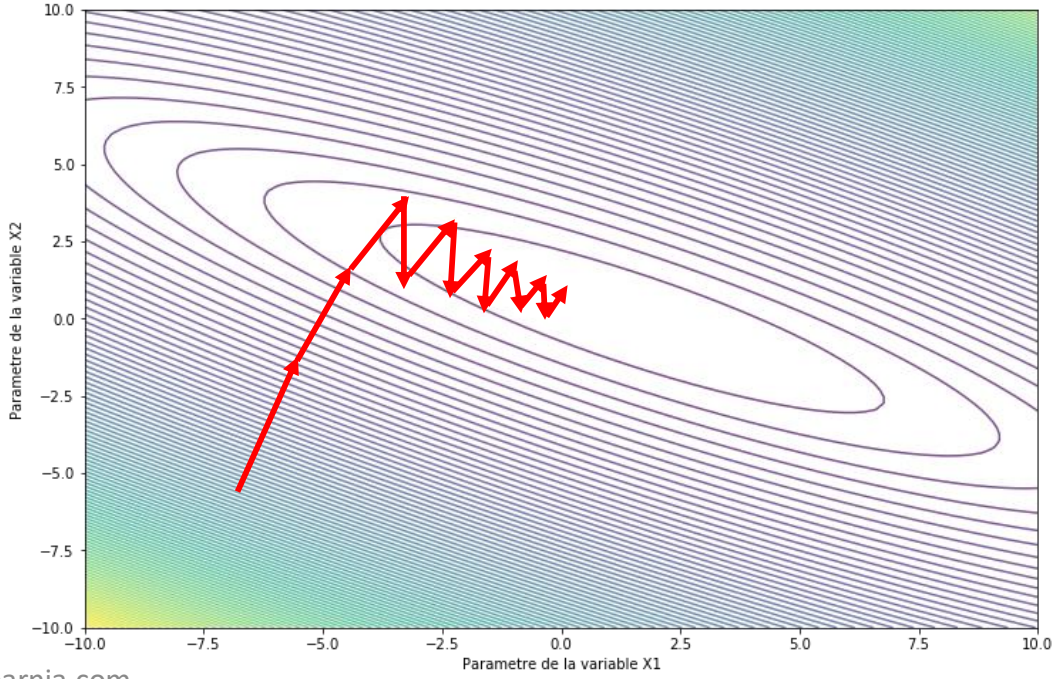
Avec Normalisation

Fonction Cout



Sans Normalisation

Fonction Cout



MinMaxScaler

Transforme **chaque variable X** de telle sorte à être comprise entre **0 et 1.**

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

Exemple:

$$\begin{bmatrix} 70 \\ 80 \\ 120 \end{bmatrix} \rightarrow \begin{bmatrix} 0 \\ 0.2 \\ 1 \end{bmatrix}$$

Note : *Pour remettre vos prédictions à leur véritable échelle :* ***inverse_transform***

MinMaxScaler

fit_transform(X_{train})

$$X_{train_{scaled}} = \frac{X_{train} - X_{train_{min}}}{X_{train_{max}} - X_{train_{min}}}$$

transform(X_{test})

$$X_{test_{scaled}} = \frac{X_{test} - X_{train_{min}}}{X_{train_{max}} - X_{train_{min}}}$$

SandardScaler

Standardise chaque variable $\boldsymbol{X}$: La Moyenne est **nulle** et l'écart type égale **1**

$$X_{scaled} = \frac{X - \mu_X}{\sigma_X}$$

Exemple:

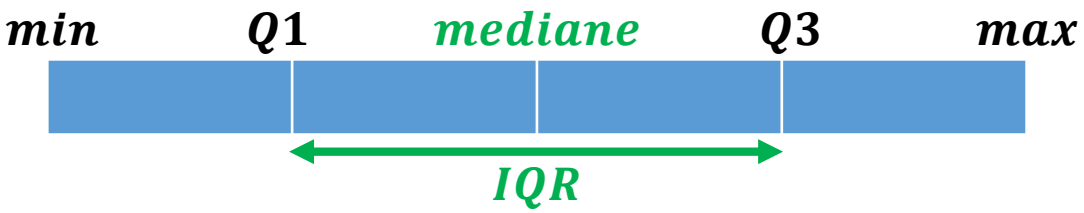
$$\begin{bmatrix} 70 \\ 80 \\ 120 \end{bmatrix} \rightarrow \begin{bmatrix} -0.92 \\ -0.46 \\ 1.38 \end{bmatrix}$$

Note : *Pour remettre vos prédictions à leur véritable échelle :* ***inverse_transform***

RobustScaler

Transforme **chaque variable X** en étant peu sensible aux ***outliers***

$$X_{scaled} = \frac{X - mediane}{IQR}$$



Exemple:

$$\begin{bmatrix} 70 \\ 80 \\ 120 \end{bmatrix} \rightarrow \begin{bmatrix} -0.4 \\ 0 \\ 1.6 \end{bmatrix}$$

PolyFeatures

Crée de nouvelles variables **polynômiales** à partir des variables existantes.

Exemple : 1 variable x → Polynôme 2

$$\begin{matrix} x & \rightarrow & 1 & x^1 & x^2 \end{matrix}$$

$$\begin{bmatrix} 1 \\ 2 \\ 0.5 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 0.5 & 0.25 \end{bmatrix}$$

La machine apprend alors un modèle :

$$f(x) = ax^2 + bx + c$$

PolyFeatures

Crée de nouvelles variables **polynômiales** à partir des variables existantes.

Exemple : 2 variables $x_1, x_2 \rightarrow$ Polynôme 2

$$x_1, x_2 \rightarrow 1, x_1, x_2, x_1^2, x_1 x_2, x_2^2$$

PolyFeatures

Crée de nouvelles variables **polynômiales** à partir des variables existantes.

Exemple : 2 variables $x_1, x_2 \rightarrow$ Polynôme 2

$$x_1, x_2 \rightarrow 1, x_1, x_2, x_1^2, x_1 x_2, x_2^2$$

Ces variables n'étant **pas sur la même échelle**, il faut les **normaliser** avant leur passage dans l'estimateur !

Preprocessing

preprocessing.Binarizer([threshold, copy])
preprocessing.FunctionTransformer([func, ...])
preprocessing.KBinsDiscretizer([n_bins, ...])
preprocessing.KernelCenterer()
preprocessing.LabelBinarizer([neg_label, ...])
preprocessing.LabelEncoder
preprocessing.MultiLabelBinarizer([classes, ...])
preprocessing.MaxAbsScaler([copy])
preprocessing.MinMaxScaler([feature_range, copy])
preprocessing.Normalizer([norm, copy])
preprocessing.OneHotEncoder([categories, ...])
preprocessing.OrdinalEncoder([categories, dtype])
preprocessing.PolynomialFeatures([degree, ...])
preprocessing.PowerTransformer([method, ...])
preprocessing.QuantileTransformer([...])
preprocessing.RobustScaler([with_centering, ...])
preprocessing.StandardScaler([copy, ...])

Encodage

Preprocessing

preprocessing.Binarizer([threshold, copy])
preprocessing.FunctionTransformer([func, ...])
preprocessing.KBinsDiscretizer([n_bins, ...])
preprocessing.KernelCenterer()
preprocessing.LabelBinarizer([neg_label, ...])
preprocessing.LabelEncoder
preprocessing.MultiLabelBinarizer([classes, ...])
preprocessing.MaxAbsScaler([copy])
preprocessing.MinMaxScaler([feature_range, copy])
preprocessing.Normalizer([norm, copy])
preprocessing.OneHotEncoder([categories, ...])
preprocessing.OrdinalEncoder([categories, dtype])
preprocessing.PolynomialFeatures([degree, ...])
preprocessing.PowerTransformer([method, ...])
preprocessing.QuantileTransformer([...])
preprocessing.RobustScaler([with_centering, ...])
preprocessing.StandardScaler([copy, ...])

Encodage

Preprocessing

```
preprocessing.Binarizer([threshold, copy])
preprocessing.FunctionTransformer([func, ...])
preprocessing.KBinsDiscretizer([n_bins, ...])
preprocessing.KernelCenterer()
preprocessing.LabelBinarizer([neg_label, ...])
preprocessing.LabelEncoder
preprocessing.MultiLabelBinarizer([classes, ...])
preprocessing.MaxAbsScaler([copy])
preprocessing.MinMaxScaler([feature_range, copy])
preprocessing.Normalizer([norm, copy])
preprocessing.OneHotEncoder([categories, ...])
preprocessing.OrdinalEncoder([categories, dtype])
preprocessing.PolynomialFeatures([degree, ...])
preprocessing.PowerTransformer([method, ...])
preprocessing.QuantileTransformer([...])
preprocessing.RobustScaler([with_centering, ...])
preprocessing.StandardScaler([copy, ...])
```

Encodage

Normalisation

Preprocessing

```
preprocessing.Binarizer([threshold, copy])
preprocessing.FunctionTransformer([func, ...])
preprocessing.KBinsDiscretizer([n_bins, ...])
preprocessing.KernelCenterer()
preprocessing.LabelBinarizer([neg_label, ...])
preprocessing.LabelEncoder
preprocessing.MultiLabelBinarizer([classes, ...])
preprocessing.MaxAbsScaler([copy])
preprocessing.MinMaxScaler([feature_range, copy])
preprocessing.Normalizer([norm, copy])
preprocessing.OneHotEncoder([categories, ...])
preprocessing.OrdinalEncoder([categories, dtype])
preprocessing.PolynomialFeatures([degree, ...])
preprocessing.PowerTransformer([method, ...])
preprocessing.QuantileTransformer([...])
preprocessing.RobustScaler([with_centering, ...])
preprocessing.StandardScaler([copy, ...])
```

Encodage

Normalisation

Preprocessing

```
preprocessing.Binarizer([threshold, copy])
preprocessing.FunctionTransformer([func, ...])
preprocessing.KBinsDiscretizer([n_bins, ...])
preprocessing.KernelCenterer()
preprocessing.LabelBinarizer([neg_label, ...])
preprocessing.LabelEncoder
preprocessing.MultiLabelBinarizer([classes, ...])
preprocessing.MaxAbsScaler([copy])
preprocessing.MinMaxScaler([feature_range, copy])
preprocessing.Normalizer([norm, copy])
preprocessing.OneHotEncoder([categories, ...])
preprocessing.OrdinalEncoder([categories, dtype])
preprocessing.PolynomialFeatures([degree, ...])
preprocessing.PowerTransformer([method, ...])
preprocessing.QuantileTransformer([...])
preprocessing.RobustScaler([with_centering, ...])
preprocessing.StandardScaler([copy, ...])
```

Encodage

Normalisation

Création de polynômes

Preprocessing

```
preprocessing.Binarizer([threshold, copy])
preprocessing.FunctionTransformer([func, ...])
preprocessing.KBinsDiscretizer([n_bins, ...])
preprocessing.KernelCenterer()
preprocessing.LabelBinarizer([neg_label, ...])
preprocessing.LabelEncoder
preprocessing.MultiLabelBinarizer([classes, ...])
preprocessing.MaxAbsScaler([copy])
preprocessing.MinMaxScaler([feature_range, copy])
preprocessing.Normalizer([norm, copy])
preprocessing.OneHotEncoder([categories, ...])
preprocessing.OrdinalEncoder([categories, dtype])
preprocessing.PolynomialFeatures([degree, ...])
preprocessing.PowerTransformer([method, ...])
preprocessing.QuantileTransformer([...])
preprocessing.RobustScaler([with_centering, ...])
preprocessing.StandardScaler([copy, ...])
```

Encodage

Normalisation

Création de polynômes

Transformation non linéaire

Preprocessing

```
preprocessing.Binarizer([threshold, copy])
preprocessing.FunctionTransformer([func, ...])
preprocessing.KBinsDiscretizer([n_bins, ...])
preprocessing.KernelCenterer()
preprocessing.LabelBinarizer([neg_label, ...])
preprocessing.LabelEncoder
preprocessing.MultiLabelBinarizer([classes, ...])
preprocessing.MaxAbsScaler([copy])
preprocessing.MinMaxScaler([feature_range, copy])
preprocessing.Normalizer([norm, copy])
preprocessing.OneHotEncoder([categories, ...])
preprocessing.OrdinalEncoder([categories, dtype])
preprocessing.PolynomialFeatures([degree, ...])
preprocessing.PowerTransformer([method, ...])
preprocessing.QuantileTransformer([...])
preprocessing.RobustScaler([with_centering, ...])
preprocessing.StandardScaler([copy, ...])
```

Encodage

Normalisation

Création de polynômes

Transformation non linéaire

Discrétisation

Preprocessing

```
preprocessing.Binarizer([threshold, copy])
preprocessing.FunctionTransformer([func, ...])
preprocessing.KBinsDiscretizer([n_bins, ...])
preprocessing.KernelCenterer()
preprocessing.LabelBinarizer([neg_label, ...])
preprocessing.LabelEncoder
preprocessing.MultiLabelBinarizer([classes, ...])
preprocessing.MaxAbsScaler([copy])
preprocessing.MinMaxScaler([feature_range, copy])
preprocessing.Normalizer([norm, copy])
preprocessing.OneHotEncoder([categories, ...])
preprocessing.OrdinalEncoder([categories, dtype])
preprocessing.PolynomialFeatures([degree, ...])
preprocessing.PowerTransformer([method, ...])
preprocessing.QuantileTransformer([...])
preprocessing.RobustScaler([with_centering, ...])
preprocessing.StandardScaler([copy, ...])
```

Encodage

Normalisation

Création de polynômes

Transformation non linéaire

Discrétisation

Personnalisation

Preprocessing

```
preprocessing.Binarizer([threshold, copy])
preprocessing.FunctionTransformer([func, ...])
preprocessing.KBinsDiscretizer([n_bins, ...])
preprocessing.KernelCenterer()
preprocessing.LabelBinarizer([neg_label, ...])
preprocessing.LabelEncoder
preprocessing.MultiLabelBinarizer([classes, ...])
preprocessing.MaxAbsScaler([copy])
preprocessing.MinMaxScaler([feature_range, copy])
preprocessing.Normalizer([norm, copy])
preprocessing.OneHotEncoder([categories, ...])
preprocessing.OrdinalEncoder([categories, dtype])
preprocessing.PolynomialFeatures([degree, ...])
preprocessing.PowerTransformer([method, ...])
preprocessing.QuantileTransformer([...])
preprocessing.RobustScaler([with_centering, ...])
preprocessing.StandardScaler([copy, ...])
```

Encodage

Normalisation

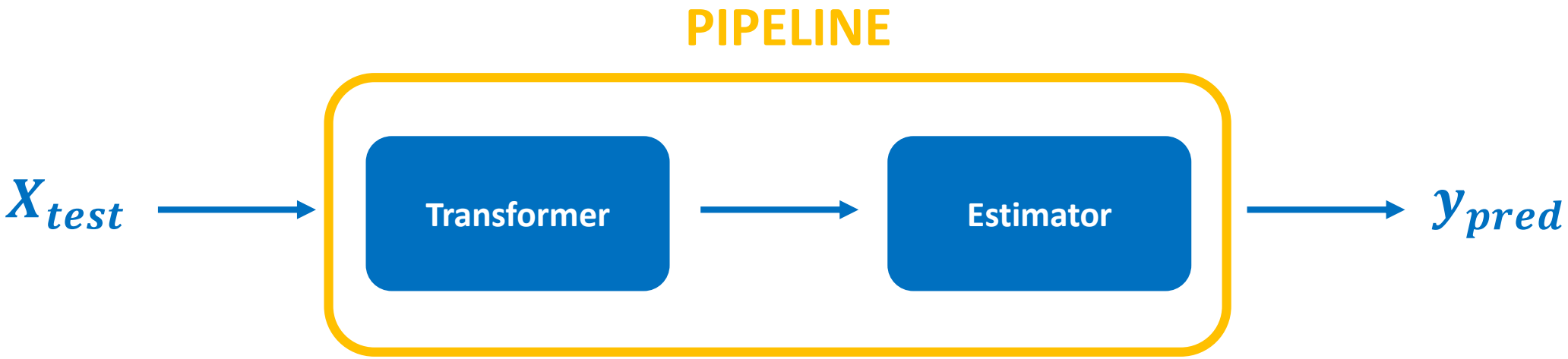
Création de polynômes

Si vous débutez

Preprocessing Transformers

preprocessing.Binarizer([threshold, copy])
preprocessing.FunctionTransformer([func, ...])
preprocessing.KBinsDiscretizer([n_bins, ...])
preprocessing.KernelCenterer()
preprocessing.LabelBinarizer([neg_label, ...])
preprocessing.LabelEncoder
preprocessing.MultiLabelBinarizer([classes, ...])
preprocessing.MaxAbsScaler([copy])
preprocessing.MinMaxScaler([feature_range, copy])
preprocessing.Normalizer([norm, copy])
preprocessing.OneHotEncoder([categories, ...])
preprocessing.OrdinalEncoder([categories, dtype])
preprocessing.PolynomialFeatures([degree, ...])
preprocessing.PowerTransformer([method, ...])
preprocessing.QuantileTransformer([...])
preprocessing.RobustScaler([with_centering, ...])
preprocessing.StandardScaler([copy, ...])

Transformer et Estimator



Transformer Estimator

X_{train}

↓

Transformer .fit_transform(X_{train})

Estimator .fit($X_{train_{transformed}}$, y_{train})

X_{test}

↓

Transformer .transform(X_{test})

Estimator .predit($X_{test_{transformed}}$)

Transformer Estimator

X_{train}

↓

Transformer .fit_transform(X_{train})

Estimator .fit($X_{train_{transformed}}$, y_{train})

X_{test}

↓

Transformer .transform(X_{test})

Estimator .predit($X_{test_{transformed}}$)

Transformer Estimator

X_{train}

↓

Transformer .fit_transform(X_{train})

Estimator .fit($X_{train_{transformed}}$, y_{train})

X_{test}

↓

Transformer .transform(X_{test})

Estimator .predit($X_{test_{transformed}}$)

Transformer Estimator

X_{train}

Transformer .fit_transform(X_{train})

Estimator .fit($X_{train_{transformed}}$, y_{train})

X_{test}

Transformer .transform(X_{test})

Estimator .predit($X_{test_{transformed}}$)

Transformer Estimator

X_{train}

↓

Transformer .fit_transform(X_{train})

Estimator .fit($X_{train_{transformed}}$, y_{train})

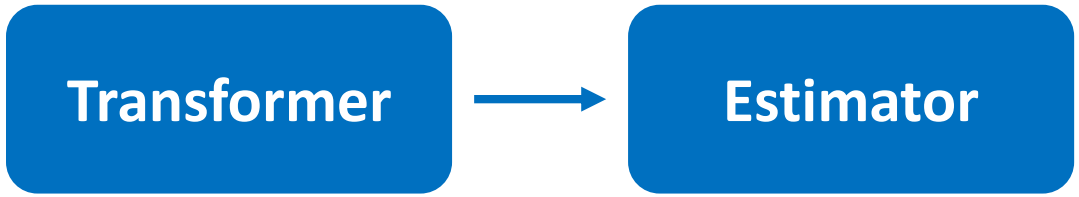
X_{test}

↓

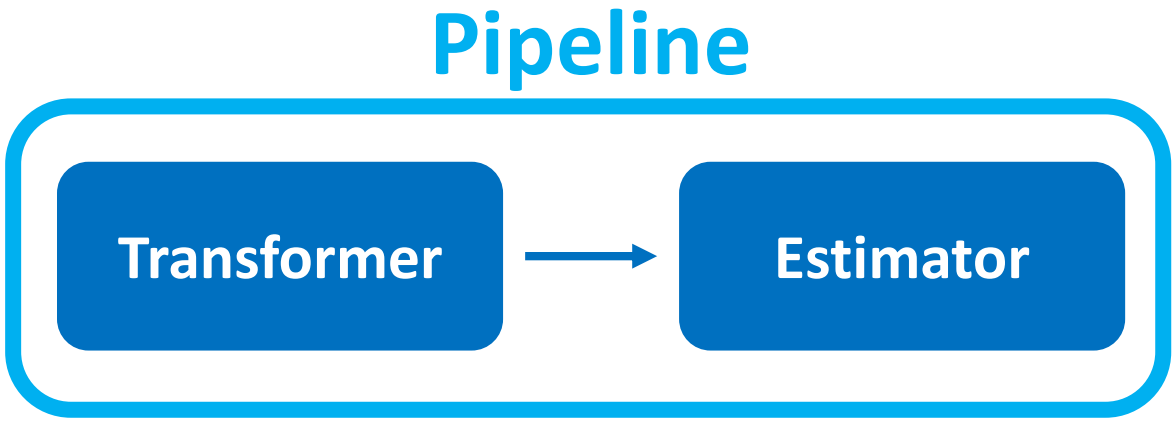
Transformer .transform(X_{test})

Estimator .predit($X_{test_{transformed}}$)

Pipeline et Estimateur Composite



Pipeline et Estimateur Composite



Pipeline et Estimateur Composite

Pipeline

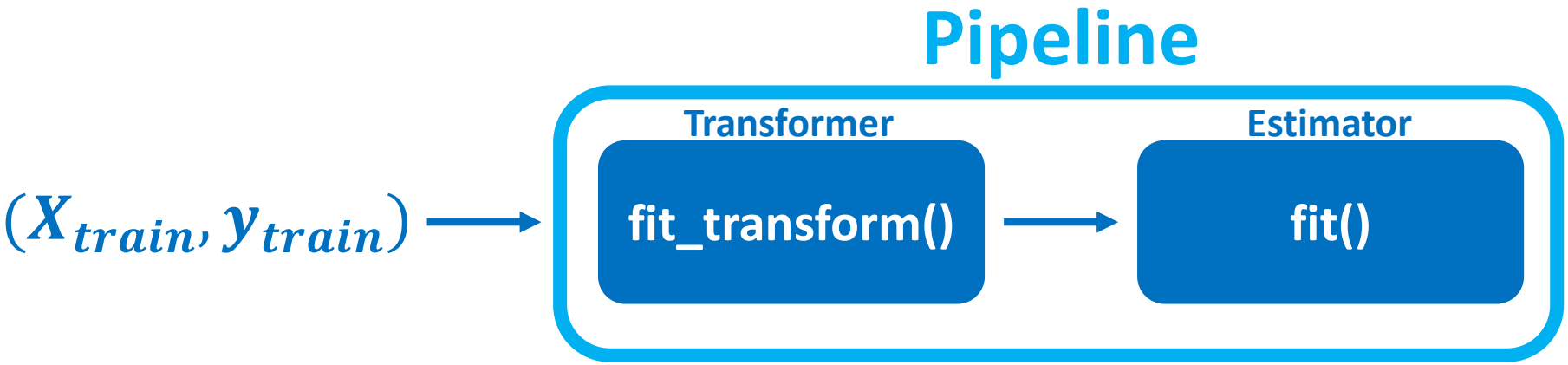
Composite Estimator

Pipeline et Estimateur Composite

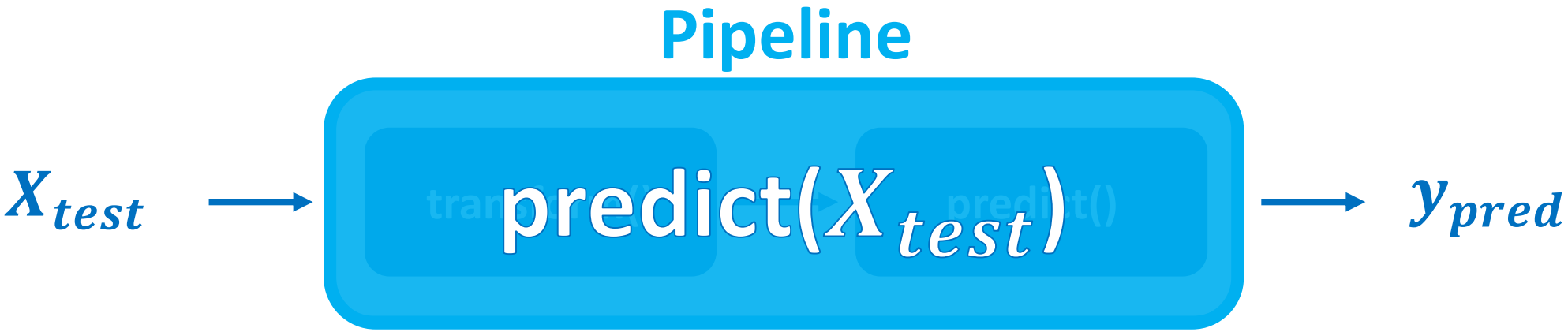
Pipeline

$(X_{train}, y_{train}) \longrightarrow$ fit(X_{train}, y_{train})

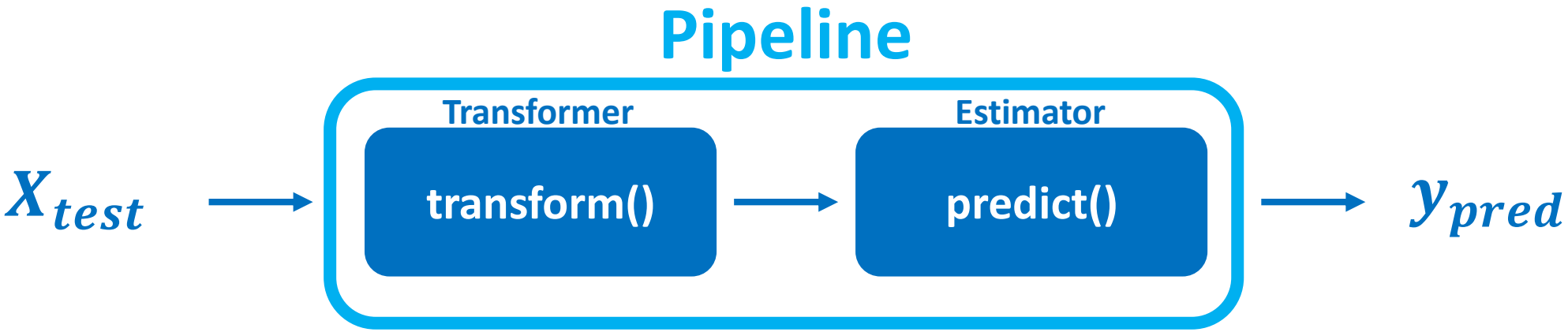
Pipeline et Estimateur Composite



Pipeline et Estimateur Composite



Pipeline et Estimateur Composite



Pipeline

Pour créer une **Pipeline**, 2 options :

- **Classe** Pipeline
- **Fonction** **make_pipeline**

model = **make_pipeline**(*StandardScaler(), SGDClassifier()*)

model.**fit**(X_{train}, y_{train})
model.**score**(X_{test}, y_{test})
model.**predict**(X_{test})

Pipeline GridSearchCV

On peut utiliser **GridSearchCV** pour trouver les meilleurs paramètres de la pipeline.

grid = GridSearchCV(pipeline, params, cv)

params = {

<Composant>_ _<paramètre> : [….]

}

grid.fit(X_{train}, y_{train})

grid.best_estimator

Column Transformer

ColumnTransformer permet d'appliquer vos transformers sur les **colonnes** que vous sélectionnez.

make_column_transformer ((transformer, [*'colonne 1', 'colonne 2', …*]))

Column Transformer

ColumnTransformer permet d'appliquer vos transformers sur les **colonnes** que vous sélectionnez.

make_column_transformer ((transformer, [*'colonne 1', 'colonne 2', …*]))

Application Classique :

Séparez les ***catégories*** et variables ***numériques*** *en* **2 listes**, puis utilisez ColumnTransformer pour traiter chaque liste de variables séparément.

Pipeline Avancee

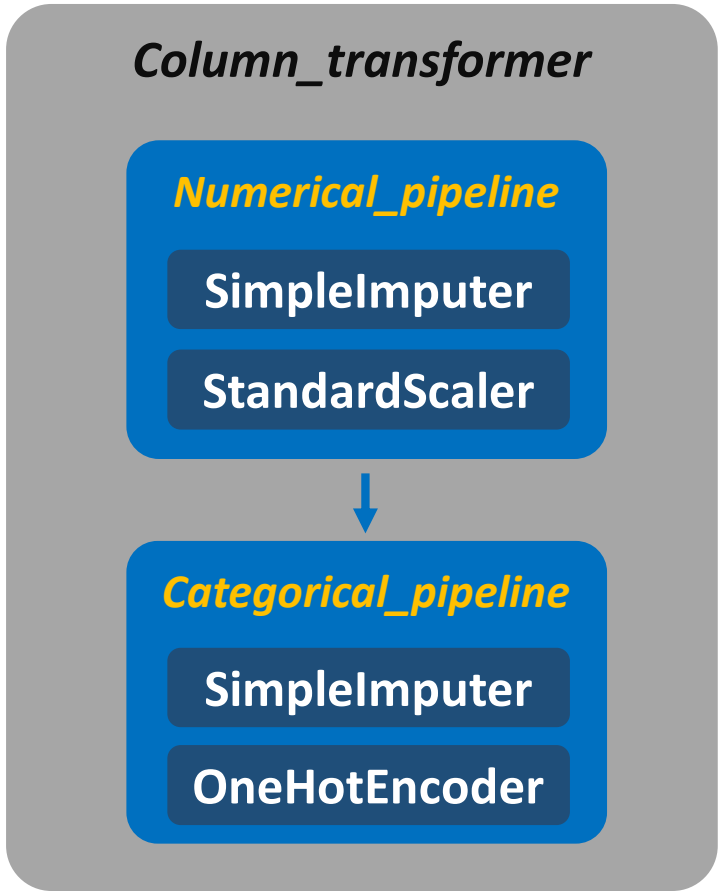
Numerical_pipeline

- SimpleImputer
- StandardScaler

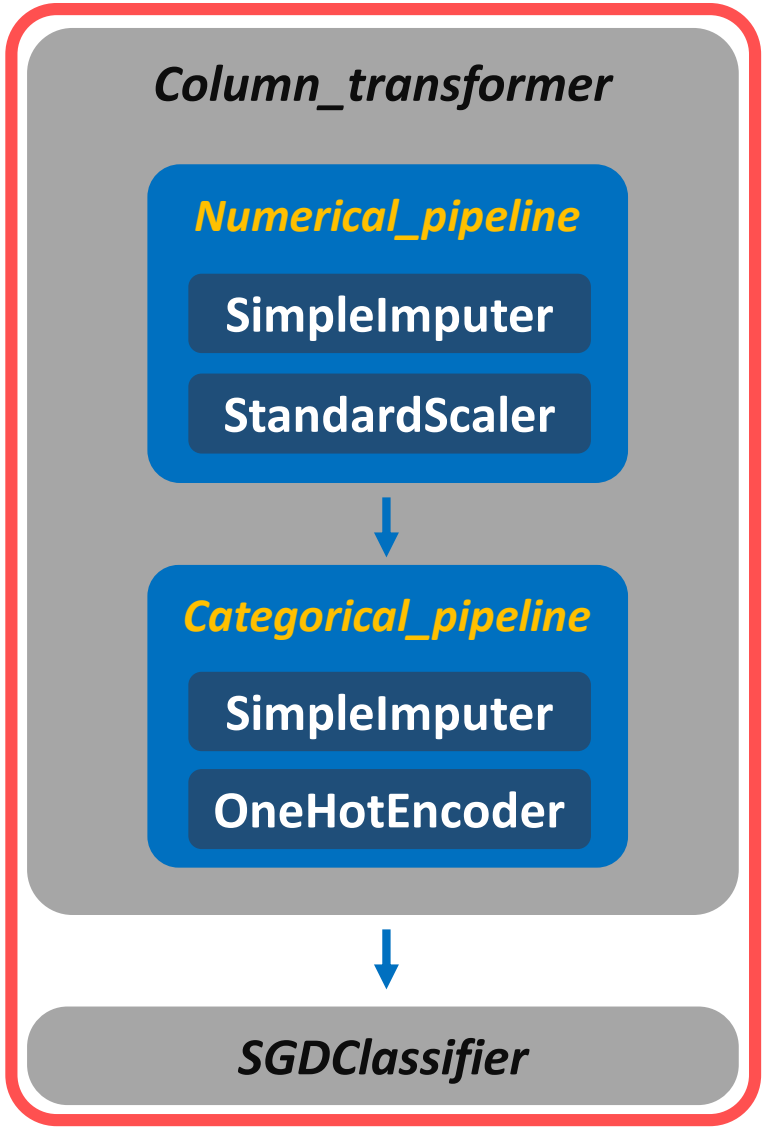
Categorical_pipeline

- SimpleImputer
- OneHotEncoder

Pipeline Avancee



Pipeline Avancee



Column Selector

Depuis la version 0.22 de sklearn, il existe **make_column_selector** qui permet de sélectionner certains ***types*** de colonnes.

3 arguments au choix :

dtype_include

dtype_exclude

pattern (avec des *Regular Expressions*)

Exemple de types :

np.number → variables numériques

object → variables catégorielles

Feature Union

make_union() permet de créer des pipelines **parallèles**. Les résultats sont **concaténés** à la sortie du transformer.