# Régression linéaire

Noura BENHAJJI
noura.benhajji@gmail.com

# Introduction

- La régression sert à trouver la **relation** d'une variable par rapport à une ou plusieurs autres.
- Le but de la régression est d'estimer une **valeur** (numérique) de sortie à partir des valeurs d'un ensemble de caractéristiques en entrée.
- Le problème revient à estimer une fonction de calcul en se basant sur les données d'entraînement.

# Algorithmes de régression

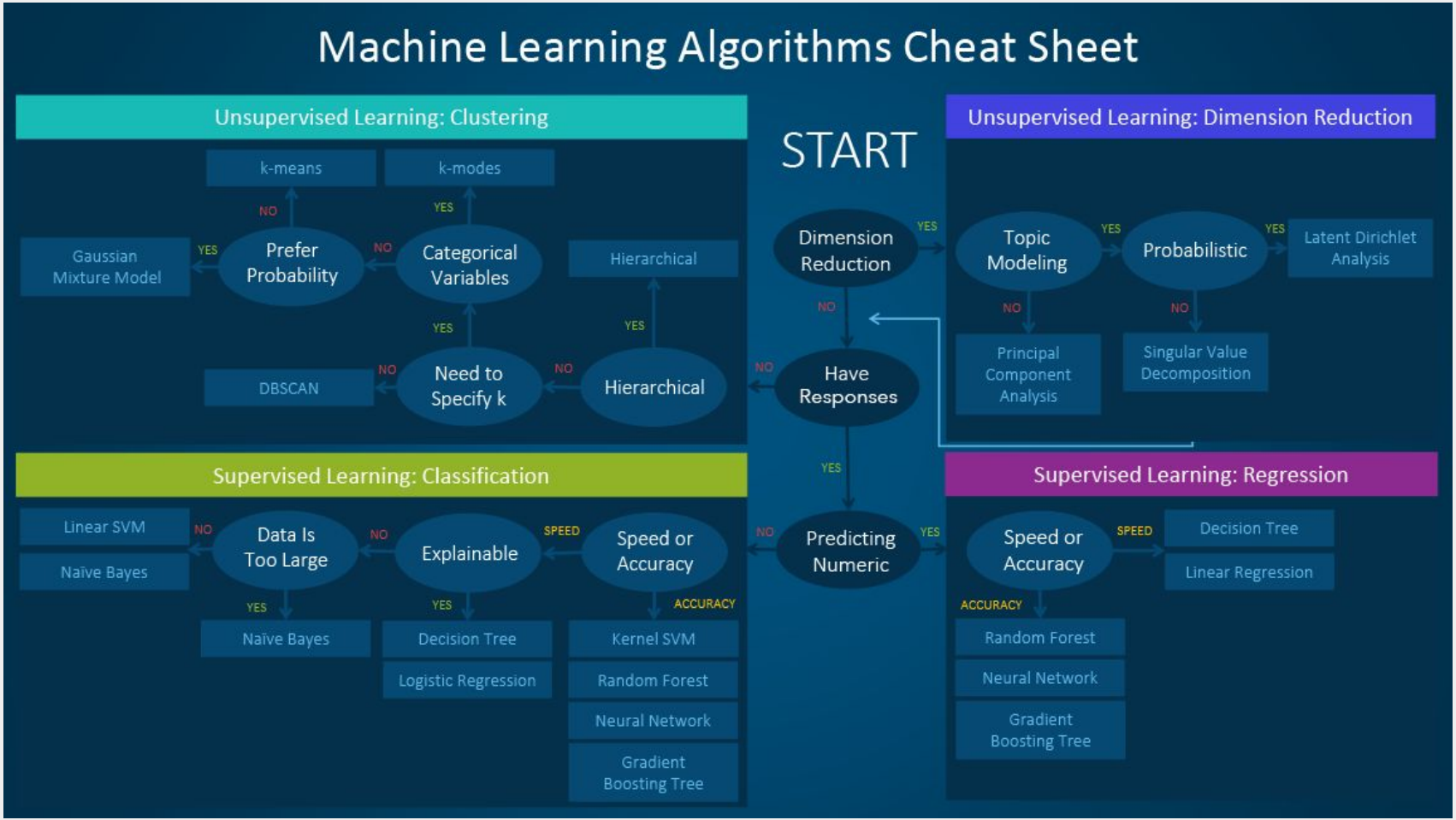
- **Régression linéaire**
  - Trouver une droite au milieu d'un nuage de points.
- **Régression polynomiale**
  - Trouver une relation de forme non-linéaire entre la réponse (y) et la ou les variables explicatives (x)
  - pour prendre en charge cette forme non linéaire, les algorithmes de régressions intègrent des polynômes dans leurs équations.

# Algorithmes de classification supervisée

- **Le random Forest**
  - calculer la moyenne des prévisions obtenues
- **Le perceptron multicouches**
  - Un ensemble de neurones connectés.
  - Il est composé d'une couche d'entrée, de n couches cachées, et d'une couche de sortie.

# Algorithmes de classification supervisée

- **SVM**
  - le principe des SVM consiste à ramener un problème de classification à un hyperplan dans lequel les données sont séparées en plusieurs classes dont la frontière est la plus éloignée possible des points de données.
  - un SVM permet de résoudre les problèmes de régression en utilisant la technique du noyau (à venir)




Source : SAS Algorithm Flowchart

Veille individuelle **60min**

# Régression linéaire

- ❑ Qu'est ce qu'une régression linéaire ?
- ❑ Quelles sont les étapes de construction d'un modèle régression linéaire ?
  - ❑ fonction coût
  - ❑ minimiser la fonction coût
  - ❑ méthode de résolution du problème (descente de gradient, méthode des moindres carrés)
  - ❑ préparation des données

*Ressources* :

- La régression linéaire pour comprendre les grands principes du Machine Learning
- Comprendre la descente de gradient en 3 étapes et 12 dessins
- L'algorithme de Gradient Descent
- Notions fondamentales de la régression linéaire
- Préparation des données

# Régression linéaire

- La **régression linéaire** est un **modèle statistique** qui permet de **prédire** une **variable continue** en fonction d'autres variables **explicatives**.
- Le modèle de régression linéaire est basé sur l'hypothèse selon laquelle il existe une **relation linéaire** entre les variables explicatives et la variable expliquée à prédire.
- la régression linéaire consiste à trouver la droite qui s'ajuste le mieux aux données, en minimisant la **somme des carrés des écarts** entre les valeurs prédites et les valeurs observées

# Régression linéaire

- La régression linéaire est un algorithme d'apprentissage supervisé très basique qui vise à trouver la meilleure droite possible à l'aide d'une ou plusieurs variables explicatives.
- La fonction recherchée est sous la forme : **y = f(X)** avec f une fonction linéaire.
- Un modèle de **régression linéaire multiple** est sous la forme :

$$Y = ax_1 + bx_2 + cx_3 + \cdots + K + \varepsilon \text{ où } f(X) = aX + b$$

  - Y variable cible
  - a, … , k les coefficients
  - X=(x1, … , xq) variables explicatives
  - ε variable aléatoire qui représente l'erreur

# Régression linéaire

La régression linéaire est un outil puissant pour la compréhension des relations entre les variables et pour la prédiction de la valeur d'une variable à partir de valeurs connues d'autres variables.

Elle est largement utilisée dans de nombreux domaines, tels que l'économie, la finance, la médecine, l'ingénierie, etc.

# Construire un modèle de régression linéaire

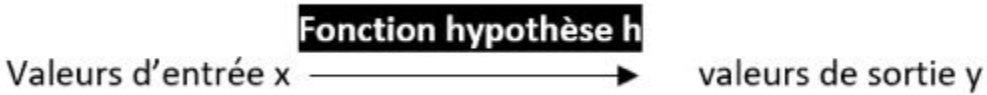
- Définir une **fonction coût** qui permet de calculer l'erreur
- **Minimiser** la fonction coût en trouvant les bons paramètres du modèle qui minimisent l'erreur.
- Choisir une **méthode de résolution**

# Construire un modèle de régression linéaire

1. **Préparation des données** : sélection des variables, la normalisation des données, le traitement des valeurs manquantes, etc.
2. **Sélection des variables explicatives** : Il est important de choisir les variables explicatives appropriées pour construire un modèle de régression linéaire (l'analyse de corrélation, l'analyse de variance (ANOVA), etc.)
3. **Division des données en ensembles d'entraînement et de test**
4. **Entraînement du modèle** : trouver les coefficients β qui minimisent la somme des carrés des écarts entre les valeurs prédites et les valeurs observées.
5. **Évaluation du modèle**
6. **Interprétation des coefficients** : comprendre l'influence de chaque variable explicative sur la variable à prédire.
7. **Utilisation du modèle pour des prédictions** : effectuer des prédictions sur de nouvelles données.

# Fonction hypothèse

- Mathématiquement, on cherche à déterminer la meilleure fonction hypothèse qui permet de trouver une **relation linéaire approximative** entre les variables d'entrée et la variable de sortie.




- Pour chaque point, la fonction hypothèse associe une valeur définie par h(xi) qui est plus au moins proche de yi.
- On définit l'**erreur unitaire** pour xi par

$$\sum_{k=1}^{P} (h(x_i) - y_i)^2$$

# Fonction coût

- La **fonction coût** s'écrit en pondérant la somme des erreurs quadratiques par le nombre de points p dans la base d'apprentissage.

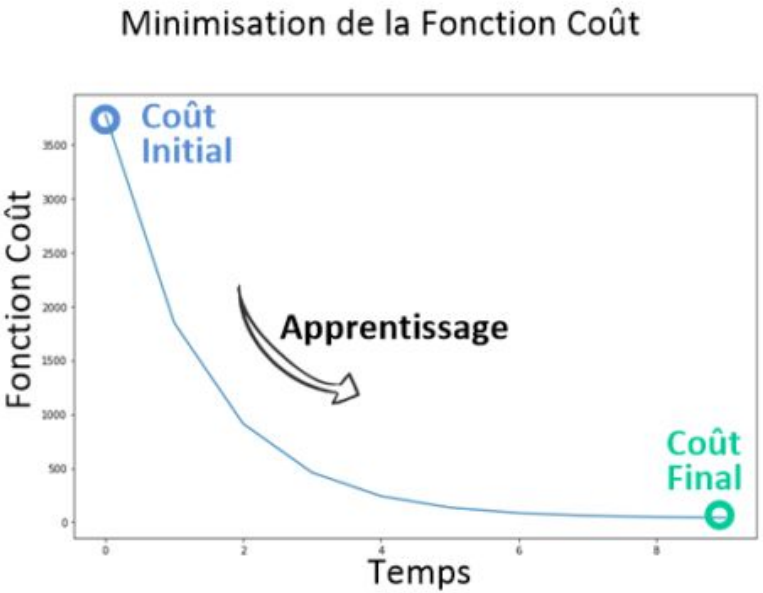
$$C(h) = \frac{1}{2p} \sum_{k=1}^{P} (h(x_i) - y_i)^2$$

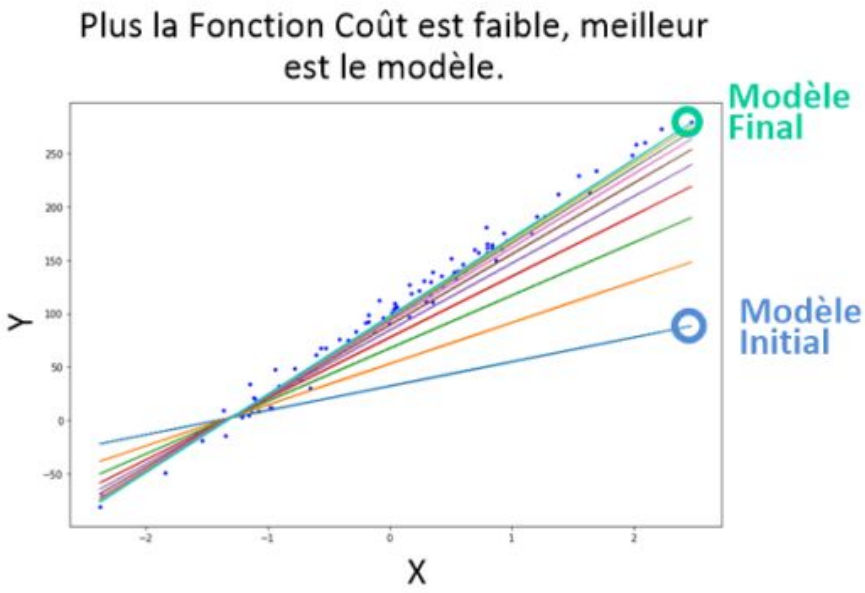
- Déterminer les meilleurs paramètres pour la fonction hypothèse *h* revient à trouver la meilleure droite, celle qui minimise la somme de toutes les erreurs unitaires.
- Il s'agit de trouver **minimum** de la fonction coût.

# La descente de Gradient

- La descente de Gradient est un algorithme d'**optimisation** pour trouver le **minimum** de n'importe quelle fonction convexe en convergeant progressivement vers celui-ci.
- En machine learning, la descente de gradient est utilisée dans les problèmes d'apprentissage supervisé pour **minimiser la fonction coût** (l'erreur quadratique moyenne par exemple)
- Grâce à cet algorithme que la machine apprend. i.e, trouve le meilleur modèle.

# La descente de Gradient




Source : [Descente de gradient - Machine Learnia]

# La descente de Gradient

- La formule générale est : $x_{t+1} = x_t - \eta \Delta x_t$
  - $\eta$ taux d'apprentissage
  - $\Delta x_t$ direction de la descente

Note : L'algorithme de descente du gradient décide de suivre comme direction de descente l'opposé du gradient (dérivée). Le gradient indique la croissance maximale d'une fonction à partir d'un point.

# La descente de Gradient

L'algorithme de descente de gradient est comme suit :

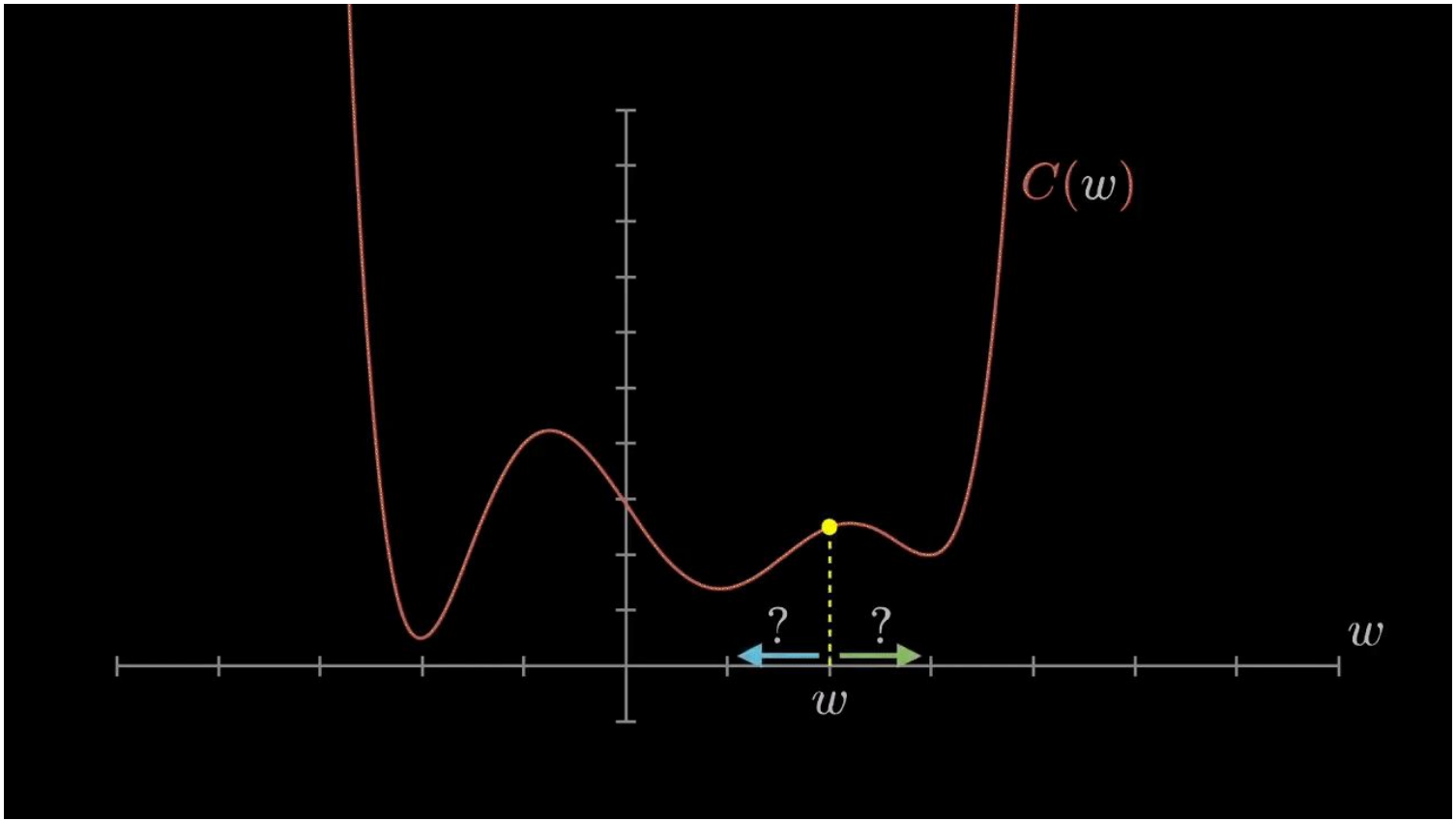
1. Soit un point d'initialisation x_0
2. calculer f(xt)
3. mettre à jour les coordonnées : $x_{t+1} = x_t - \eta \Delta f(x_t)$
4. répéter 2 et 3 jusqu'au critère d'arrêt $|f x_{t+1} - f x_t| \leq \varepsilon$

# La descente de Gradient

**Attention**

- deux éléments cruciaux pour le bon fonctionnement de l'algorithme de descente de gradient :
  - le point d'initialisation x° et la valeur du taux d'apprentissage.
- Un mauvais point d'initialisation ou un taux d'apprentissage peu adapté peut empêcher l'algorithme de converger vers le minimum.
- Plus le taux d'apprentissage est elevé, plus on suivra loin la direction indiquée par le gradient. -> minimum local

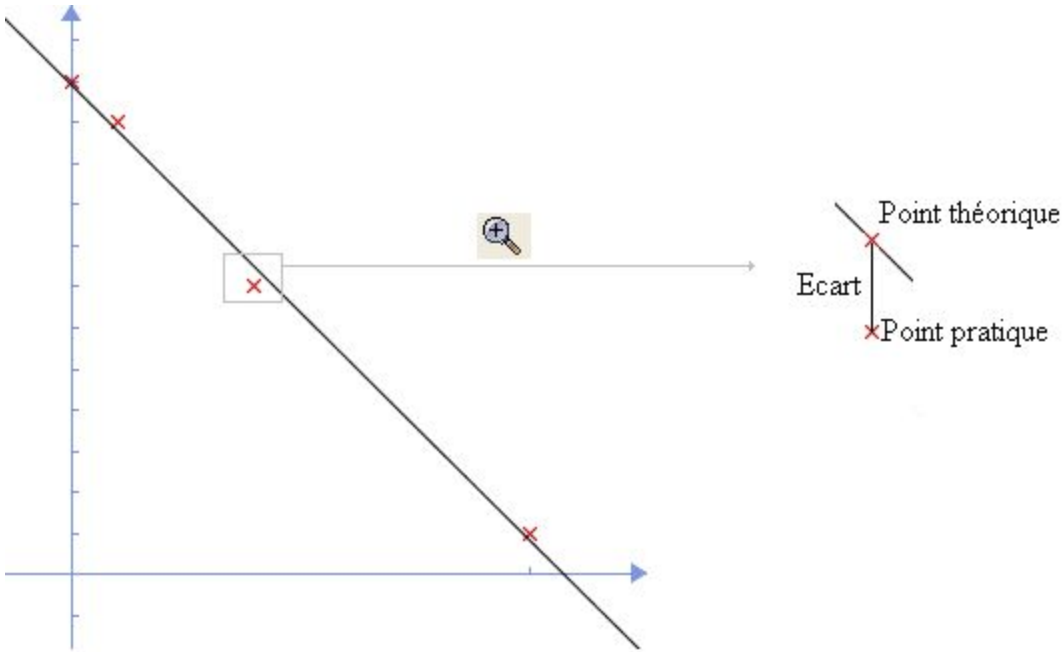
# La descente de Gradient



# La descente de Gradient

- La **méthode des moindres carrés**, permet de comparer des données expérimentales, généralement entachées d'erreurs de mesure, à un modèle mathématique censé décrire ces données.
- On parle de régression pour exprimer la **diminution** de la somme des écarts.
- Le principe des moindres carrés consiste à choisir les valeurs de a et b qui minimisent les erreurs de prédiction ou les résidus sur un jeu de données d'apprentissage.

$$\varepsilon = \sum_{i=0}^{P} \left(Y_i - (aX_i + b)\right)^2$$




En rouge, on a dessiné les points expérimentaux, et en noir, on a tracé une droite de régression. Le point théorique qui correspond au point pratique est celui situé sur la droite à la même abscisse. La méthode des moindres carrés consiste à prendre la somme des écarts au carré, et à la minimiser.

# Préparation des données

La préparation des données est une étape cruciale pour construire un modèle de régression linéaire efficace.

1. **Sélection des variables** : Il est important de choisir les variables explicatives appropriées en utilisant des méthodes telles que l'analyse de corrélation, l'analyse de variance (ANOVA), etc.
2. **Nettoyage des données** : éliminer les erreurs, les duplicatas et les valeurs manquantes. Les valeurs manquantes peuvent être remplacées par des valeurs moyennes, des valeurs médianes ou par une imputation basée sur les valeurs des autres variables.
3. **Normalisation des données** : Il est souvent nécessaire de normaliser les données pour éviter que certaines variables n'aient une influence excessive sur les résultats du modèle, en utilisant des techniques telles que la normalisation Z-score ou la normalisation min-max.
4. **Transformation des variables** : Certaines variables peuvent avoir une distribution non normale qui peut affecter les résultats du modèle de régression linéaire. Il peut être nécessaire de les transformer en utilisant des techniques telles que la transformation logarithmique

Travail en groupe **45min**

# Régression linéaire

- ❏ Notebook : 1. Régression linéaire multiple
- ❏ Veille : matrice de corrélation

# Préparation des données

La préparation des données est une étape cruciale pour construire un modèle de régression linéaire efficace.

1. **Sélection des variables** : Il est important de choisir les variables explicatives appropriées en utilisant des méthodes telles que l'analyse de corrélation, l'analyse de variance (ANOVA), etc.
2. **Nettoyage des données** : éliminer les erreurs, les duplicatas et les valeurs manquantes. Les valeurs manquantes peuvent être remplacées par des valeurs moyennes, des valeurs médianes ou par une imputation basée sur les valeurs des autres variables.
3. **Normalisation des données** : Il est souvent nécessaire de normaliser les données pour éviter que certaines variables n'aient une influence excessive sur les résultats du modèle, en utilisant des techniques telles que la normalisation Z-score ou la normalisation min-max.
4. **Transformation des variables** : Certaines variables peuvent avoir une distribution non normale qui peut affecter les résultats du modèle de régression linéaire. Il peut être nécessaire de les transformer en utilisant des techniques telles que la transformation logarithmique

# Préparation des données

```python
import pandas as pd

# Chargement des données dans un DataFrame
data = pd.read_csv('path/to/data.csv')

# Séparation des variables en entrées (X) et en sorties (y)
X = data.drop('target_variable', axis=1)
y = data['target_variable']

# Prétraitement des données
# Encodage des variables catégorielles
X = pd.get_dummies(X, columns=['categorical_variable'])

# Remplacement des valeurs manquantes
X.fillna(X.mean(), inplace=True)

# Normalisation des données
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X = scaler.fit_transform(X)
```

# Préparation des données

```python
import pandas as pd
import numpy as np
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler, OneHotEncoder

# Chargement des données dans un DataFrame
data = pd.read_csv('path/to/data.csv')

# Séparation des variables en entrées (X) et en sorties (y)
X = data.drop('target_variable', axis=1)
y = data['target_variable']

# Prétraitement des données
# Encodage des variables catégorielles
numerical_features = X.select_dtypes(include=np.number).columns
categorical_features = X.select_dtypes(exclude=np.number).columns
```

```python
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numerical_features),
        ('cat', OneHotEncoder(), categorical_features)
    ])

# Remplacement des valeurs manquantes
imputer = SimpleImputer(strategy='mean')

# Construction du pipeline
pipe = Pipeline([
    ('preprocessor', preprocessor),
    ('imputer', imputer)
])

# Transformation des données
X = pipe.fit_transform(X)
```

Veille individuelle **45 min**

# Comment évaluer un modèle de régression ?

- ❑ Coefficient de détermination
- ❑ Mean Absolute Error
- ❑ Mean Squared Error
- ❑ Root Mean Square Error

# Evaluation

Les métriques couramment utilisées pour évaluer les modèles de régression sont :

- **Erreur quadratique moyenne (MSE)** : Cette métrique mesure la moyenne de l'erreur quadratique entre les valeurs prédites et les valeurs réelles. Cette métrique est souvent utilisée pour évaluer la qualité de la prédiction, car elle considère toutes les erreurs, qu'elles soient positives ou négatives, de la même manière.
- **Score R2** : Cette métrique mesure la qualité de la prédiction en comparant la variance expliquée par le modèle à la variance totale de la variable cible. Plus le score R2 est proche de 1, plus le modèle explique bien la variance de la variable cible.
- **Mean Absolute Error (MAE)** : Cette métrique mesure la moyenne de l'erreur absolue entre les valeurs prédites et les valeurs réelles. Elle donne une idée de la magnitude moyenne des erreurs sans tenir compte de leur signe.

# Evaluation

| | | |
|---|---|---|
| **Erreur moyenne absolue (MAE)** | lorsque les erreurs ont des conséquences similaires, peu importe leur direction | prix |
| **Erreur quadratique moyenne (MSE)** | Elle est souvent utilisée lorsque les erreurs plus élevées ont une plus grande conséquence | temps de réponse |
| **Coefficient de détermination ($R^2$)** | Il est souvent utilisé pour évaluer la qualité de la régression et varie entre 0 et 1, avec des valeurs plus élevées indiquant un modèle plus fort. | |

# Evaluation

Il y a plusieurs points de vigilance à prendre en compte lors de l'évaluation d'un modèle de régression :

- **Sur-apprentissage** : prédictions inappropriées pour les nouvelles données. Pour éviter cela, vous pouvez utiliser une validation croisée ou séparer les données en jeux d'entraînement et de test.
- **Choix de la métrique** : la métrique d'évaluation appropriée pour votre problème en fonction de la distribution de vos données et de vos objectifs d'analyse. Par exemple, la moyenne des erreurs absolues (MAE) est souvent utilisée pour les données à distributions gaussiennes, tandis que la moyenne des erreurs quadratiques (MSE) est utilisée pour les données à distributions non gaussiennes.

# Evaluation

- **Les données manquantes** : les données manquantes sont prises en compte dans votre modèle de régression. Vous pouvez les remplacer par une valeur moyenne ou médiane ou les supprimer, mais cela peut affecter les résultats de votre modèle.
- **Les données aberrantes** : Détectez et traitez les données aberrantes qui peuvent affecter les résultats de votre modèle de régression. Vous pouvez les détecter en utilisant des techniques de détection d'outliers, telles que la méthode Z-Score ou la boîte à moustache .
- **La normalité des données** : les données suivent une distribution gaussienne ou approchent une distribution gaussienne. Si ce n'est pas le cas, les normaliser en utilisant des techniques telles que la transformation logarithmique ou Box-Cox.
- La **multicollinéarité** : Évitez la multicollinéarité dans les variables indépendantes, car cela peut entraîner des coefficients de régression non fiables et des erreurs d'estimation plus élevées.

Travail en groupe 1 h

# Evaluation d'un modèle de régression

- Notebook : [Régression linéaire] Métriques – Apprenants

Veille individuelle **60 min**

# Sélection de variables

- ❑ Quel intérêt ?
- ❑ Filter Method
- ❑ Wrapper Method
- ❑ Embedded Method

# Sélection de variables

Il existe plusieurs méthodes pour sélectionner les variables les plus importantes pour un modèle de machine learning, telles que :

- **Sélection de variables basée sur la statistique** : utilise des statistiques telles que la corrélation, l'ANOVA ou le test T pour évaluer l'importance de chaque variable. Les variables qui ont un impact significatif sur les résultats sont retenues.
- **Sélection de variables basée sur les algorithmes** : utilise des algorithmes tels que la **régression lasso**, le **backward elimination** ou le **forward selection** pour sélectionner les variables les plus importantes.
- **Sélection de variables basée sur les modèles** : utilise des modèles tels que les arbres de décision, les forêts aléatoires ou les réseaux de neurones pour évaluer l'importance de chaque variable.
- **Sélection de variables basée sur les scores** : Cette méthode utilise des scores tels que le **$R^2$**, l'**AIC** ou le **BIC** pour évaluer la performance du modèle pour chaque ensemble de variables. Les variables qui ont un impact positif sur les scores sont retenues.

# Sélection de variables

La sélection de variables basée sur la statistique

la classe ***SelectKBest*** du module ***sklearn.feature_selection*** est utilisée pour sélectionner les **k** variables les plus importantes à partir de différentes méthodes de sélection de variables statistiques

- **f_regression** : qui calcule le coefficient de corrélation de Pearson entre les variables explicatives et la variable cible
- **mutual_info_regression** : qui mesure l'information mutuelle entre les variables explicatives et la variable cible

```python
import numpy as np
import pandas as pd
from sklearn.feature_selection import SelectKBest, f_regression

# Charger les données
data = pd.read_csv('data.csv')

# Séparation des variables explicatives et de la variable cible
X = data.drop('target', axis=1)
y = data['target']

# Utilisation de la fonction SelectKBest pour sélectionner les variables les plus
kbest = SelectKBest(f_regression, k=5)
kbest.fit(X, y)

# Affichage des variables sélectionnées
selected_features = X.columns[kbest.get_support()]
print("Variables sélectionnées :", selected_features)
```

# Sélection de variables

La sélection de variables basée sur les algorithmes

la classe SelectFromModel du module sklearn.feature_selection. Cette classe peut être utilisée pour sélectionner des variables en utilisant un modèle d'apprentissage automatique. Le modèle peut être un algorithme de régression ou de classification.

```python
import numpy as np
import pandas as pd
from sklearn.feature_selection import SelectFromModel
from sklearn.linear_model import LassoCV

# Charger les données
data = pd.read_csv('data.csv')

# Séparation des variables explicatives et de la variable cible
X = data.drop('target', axis=1)
y = data['target']

# Utilisation de LassoCV pour sélectionner les variables les plus
lasso = LassoCV(cv=5)
sfm = SelectFromModel(lasso, threshold=0.05)
sfm.fit(X, y)

# Affichage des variables sélectionnées
selected_features = X.columns[sfm.get_support()]
print("Variables sélectionnées :", selected_features)
```

# Sélection de variables

```
while True:
    best_feature = None
    best_pvalue = 1

    # Boucle pour tester chaque variable non encore incluse dans le modèle
    for feature in X.columns:
        if feature not in selected_features:
            X_temp = sm.add_constant(X[selected_features + [feature]])
            model = sm.OLS(y, X_temp).fit()
            pvalue = model.pvalues[feature]

            # Mise à jour de la meilleure variable si nécessaire
            if pvalue < best_pvalue:
                best_feature = feature
                best_pvalue = pvalue

    # Si aucune variable n'a été ajoutée, la boucle s'arrête
    if best_feature is None:
        break

    # Ajout de la meilleure variable à la liste des variables sélectionnées
    selected_features.append(best_feature)
```

La bibliothèque statsmodels permet d'entraîner un modèle de régression linéaire OLS à chaque étape de la boucle.
Le modèle est entraîné en incluant une constante pour prendre en compte l'interception, et en ajoutant une variable à la fois à partir de la liste des variables non encore sélectionnées.
Les valeurs p des coefficients de chaque variable sont alors comparées et la variable avec la plus petite valeur p est ajoutée à la liste des variables sélectionnées.
La boucle s'arrête lorsqu'il n'y a plus de variables à ajouter.

# Sélection de variables

La RFE est d'itérativement éliminer les variables les moins importantes en utilisant un algorithme de modèle déjà formé.

Il commence par considérer toutes les variables disponibles, et utilise l'algorithme de modèle formé pour évaluer la pertinence de chaque variable pour la prédiction de la variable cible.

La variable ayant la pertinence la plus faible est ensuite éliminée, et le processus est répété jusqu'à ce que le nombre souhaité de variables soit atteint.

```python
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.feature_selection import RFE

# Charger les données
data = pd.read_csv('data.csv')

# Séparation des variables explicatives et de la variable cible
X = data.drop('target', axis=1)
y = data['target']

# Définition du nombre de variables à conserver
num_features = 5

# Initialisation de la sélection de variables RFE
selector = RFE(LinearRegression(), num_features, step=1)
selector = selector.fit(X, y)

# Affichage des variables sélectionnées
selected_features = X.columns[selector.support_]
print("Variables sélectionnées :", selected_features)
```

# Sélection de variables

```
# Initialisation des modèles
lr = LinearRegression()
rf = RandomForestRegressor()

# Calcul des scores de validation croisée pour chaque modèle
lr_scores = np.abs(cross_val_score(lr, X, y, scoring='neg_mean_absolute_error', 
rf_scores = np.abs(cross_val_score(rf, X, y, scoring='neg_mean_absolute_error', 

# Sélection du modèle avec les meilleurs scores de validation croisée
if lr_scores.mean() < rf_scores.mean():
    model = lr
else:
    model = rf

# Entraînement du modèle final sur l'ensemble des données
model.fit(X, y)

# Affichage des coefficients pour les variables explicatives (uniquement pour la
if isinstance(model, LinearRegression):
    print("Coefficients :", model.coef_)
```

Travail en groupe **60 min**

# Sélection de variables

- Tuto : Feature selection python datacamp : A Case study in Python

Veille individuelle **60 min**

# Hypothèses de régression linéaire

- ❑ Exogénéité
- ❑ Homoscédasticité
- ❑ Erreurs indépendantes
- ❑ Normalité des erreurs
- ❑ Non colinéarité des variables indépendantes

Travail en groupe

**90 min**

# Modèle de prédiction de prix de diamants

- data
- Etapes :
  - exploration des données
  - préparation de données
  - modèle 1 : régression linéaire sans sélection de variables
  - evaluation modèle 1
  - interprétation modèle 1 (coef)
  - modèle 2 : régression linéaire avec sélection de variables
  - evaluation modèle 2
  - comparer modèle 1 et modèle 2
  - interprétation modèle 1 (coef)