Solution de l'Exercice 1 : Ajustement d'un modèle linéaire

Données

x	1	2	3	4	5
y	2	3	5	4	6

Relation linéaire

$$y = \beta_0 + \beta_1 x$$

Méthode des moindres carrés

Les formules des coefficients sont données par :

$$\beta_1 = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2}, \quad \beta_0 = \bar{y} - \beta_1 \bar{x}.$$

Calcul des moyennes

$$\bar{x} = \frac{1+2+3+4+5}{5} = 3, \quad \bar{y} = \frac{2+3+5+4+6}{5} = 4.$$

Calcul de β_1

$$\beta_1 = \frac{(1-3)(2-4) + (2-3)(3-4) + (3-3)(5-4) + (4-3)(4-4) + (5-3)(6-4)}{(1-3)^2 + (2-3)^2 + (3-3)^2 + (4-3)^2 + (5-3)^2}.$$

$$\beta_1 = \frac{4+1+0+0+4}{4+1+0+1+4} = \frac{9}{10} = 0.9.$$

Calcul de β_0

$$\beta_0 = \bar{y} - \beta_1 \bar{x} = 4 - 0.9 \cdot 3 = 1.3.$$

Équation de la droite de régression

$$y = 1.3 + 0.9x.$$

Prédiction pour $x = 6$

$$y = 1.3 + 0.9 \cdot 6 = 6.7.$$

Erreur quadratique moyenne (MSE)

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2.$$

Les prédictions sont :

$$\hat{y} = \{2.2, 3.1, 4.0, 4.9, 5.8\}.$$

Les erreurs au carré sont :

$$(2-2.2)^2 = 0.04, \quad (3-3.1)^2 = 0.01, \quad (5-4.0)^2 = 1.0, \quad (4-4.9)^2 = 0.81, \quad (6-5.8)^2 = 0.04.$$

$$\text{MSE} = \frac{0.04 + 0.01 + 1.0 + 0.81 + 0.04}{5} = 0.38.$$

Solution de l'Exercice 2 : Implémentation Python

Voici le code Python correspondant :

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
Données
X = np.array([1, 2, 3, 4, 5, 6]).reshape(-1, 1)
y = np.array([2, 4, 5, 4, 5, 7])
Modèle de régression linéaire
model = LinearRegression()
model.fit(X, y)
Coefficients
beta0 = model.intercept_  # Ordonnée à l'origine
beta1 = model.coef_[0]  # Pente
print(f"Équation : y = {beta0:.2f} + {beta1:.2f}x")
Droite de régression
y_pred = model.predict(X)
Graphique
plt.scatter(X, y, color='blue', label='Données')
plt.plot(X, y_pred, color='red', label='Régression')
plt.xlabel("x")
plt.ylabel("y")
plt.legend()
plt.show()
Prédiction pour x = 7
x_new = np.array([[7]])
y_new = model.predict(x_new)
print(f"Prédiction pour x=7 : y = {y_new[0]:.2f}")
```

Solution de l'exercice 3

Les données suivantes représentent la taille (x) et le poids (y) de différentes personnes :

x (taille, en cm)	150	160	170	180	190
y (poids, en kg)	50	60	65	70	80

1. Ajustement d'une droite de régression linéaire

La droite de régression a la forme :

$$y = \beta_0 + \beta_1 x$$

Les coefficients β_0 et β_1 sont calculés par :

$$\beta_1 = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2}, \quad \beta_0 = \bar{y} - \beta_1 \bar{x}$$

Calcul des moyennes :

$$\bar{x} = \frac{150 + 160 + 170 + 180 + 190}{5} = 170, \quad \bar{y} = \frac{50 + 60 + 65 + 70 + 80}{5} = 65$$

Calcul des coefficients :

$$\beta_1 = \frac{700}{1000} = 0.7, \quad \beta_0 = 65 - 0.7 \cdot 170 = -54$$

La droite de régression est donc :

$$y = -54 + 0.7x$$

2. Calcul du coefficient de détermination R^2

Prédictions et résidus

x_i	y_i	$\hat{y}_i = -54 + 0.7x_i$	$y_i - \hat{y}_i$	$(y_i - \hat{y}_i)^2$	$y_i - \bar{y}$	$(y_i - \bar{y})^2$
150	50	51	-1	1	-15	225
160	60	58	2	4	-5	25
170	65	65	0	0	0	0
180	70	72	-2	4	5	25
190	80	79	1	1	15	225
Somme				10		500

Le coefficient de détermination R^2 est donné par :

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$$

Substituons les valeurs calculées :

$$R^2 = 1 - \frac{10}{500} = 0.98$$

3. Explication de l'ajustement

Avec $R^2 = 0.98$, la droite de régression explique 98% de la variance des données. Le modèle est donc un excellent ajustement.

4. Ajout d'un nouveau point ($x = 200, y = 90$)

Nouvelles données :

x_i	y_i
150	50
160	60
170	65
180	70
190	80
200	90

Calcul des moyennes :

$$\bar{x} = \frac{150 + 160 + 170 + 180 + 190 + 200}{6} = 175, \quad \bar{y} = \frac{50 + 60 + 65 + 70 + 80 + 90}{6} = 69.17$$

Calcul des coefficients :

$x_i - \bar{x}$	$y_i - \bar{y}$	$(x_i - \bar{x})(y_i - \bar{y})$	$(x_i - \bar{x})^2$
-25	-19.17	479.25	625
-15	-9.17	137.55	225
-5	-4.17	20.85	25
5	0.83	4.15	25
15	10.83	162.45	225
25	20.83	520.75	625
Somme		1325	1750

Les coefficients sont calculés comme suit :

$$\beta_1 = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2} = \frac{1325}{1750} = 0.757$$

$$\beta_0 = \bar{y} - \beta_1 \bar{x} = 69.17 - 0.757 \cdot 175 = -63.71$$

La nouvelle droite de régression est donc :

$$y = -63.71 + 0.757x$$

Nouveau R^2 :

Après l'ajout du nouveau point, le calcul de R^2 montre qu'il reste proche de 0.98, mais peut diminuer légèrement car le nouveau point ne suit pas parfaitement la tendance des données précédentes.

Conclusion :

Le modèle initial est un très bon ajustement ($R^2 = 0.98$). Avec l'ajout du nouveau point, la pente (β_1) et l'ordonnée à l'origine (β_0) changent légèrement. Cependant, le modèle reste robuste avec un R^2 élevé.

Solution de l'exercice4

1. **Générez des données artificielles avec du bruit en suivant la relation :**

   $$y = 2x + 1 + \varepsilon,$$

   où ε est une erreur générée aléatoirement selon une loi normale $\mathcal{N}(0, 1)$.

   Nous allons générer les données x dans l'intervalle $[0, 10]$ et la variable aléatoire ε selon une distribution normale avec une moyenne de 0 et un écart type de 1.

   La relation entre x et y est modélisée par la fonction linéaire $y = 2x+1$, avec l'ajout du bruit ε qui perturbe les valeurs de y.

2. **Générez 50 points x aléatoires dans l'intervalle $[0, 10]$.**

   On génère 50 points x aléatoires dans cet intervalle et on applique la relation définie précédemment pour obtenir les valeurs correspondantes de y. Voici le code Python correspondant à cette étape :

```
import numpy as np
import matplotlib.pyplot as plt

# Générer 50 points x aléatoires dans l'intervalle [0, 10]
x = np.random.uniform(0, 10, 50)

# Générer l'erreur epsilon selon une loi normale N(0, 1)
epsilon = np.random.normal(0, 1, 50)

# Calculer les valeurs de y
y = 2 * x + 1 + epsilon
```

3. **Ajustez une régression linéaire à ces données et tracez le graphique.**

   Nous utilisons la régression linéaire pour ajuster une droite aux données générées. Le modèle linéaire aura la forme :

   $$y = \theta_0 + \theta_1 x.$$

   Le processus d'ajustement consiste à trouver les coefficients θ_0 et θ_1 qui minimisent la somme des carrés des erreurs entre les valeurs prédites et les valeurs observées.

   Voici le code Python pour ajuster la régression linéaire et tracer le graphique :

```
from sklearn.linear_model import LinearRegression

# Reshaper les données x pour les adapter à la régression linéaire
x_reshaped = x.reshape(-1, 1)

# Créer et ajuster le modèle de régression linéaire
model = LinearRegression()
model.fit(x_reshaped, y)
```

```
# Prédire les valeurs de y
y_pred = model.predict(x_reshaped)

# Tracer les données et la droite de régression
plt.scatter(x, y, color='blue', label='Données')
plt.plot(x, y_pred, color='red', label='Régression linéaire')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.title('Régression Linéaire avec Bruit')
plt.show()
```

4. **Analysez les résidus (différence entre les y observés et les y prévus).**

   Les résidus représentent l'écart entre les valeurs observées y et les valeurs prédites $y_{\text{prédit}}$. On peut calculer les résidus comme suit :

   $$\text{résidu}_i = y_i - \hat{y}_i.$$

   Les résidus doivent être analysés pour vérifier si le modèle de régression linéaire est bien ajusté. Si les résidus sont aléatoires et ne suivent pas de structure particulière, cela signifie que le modèle est bien adapté aux données.

   Voici le code pour calculer et afficher les résidus :

```
# Calculer les résidus
residuals = y - y_pred

# Tracer les résidus
plt.scatter(x, residuals, color='green')
plt.axhline(y=0, color='red', linestyle='--')
plt.xlabel('x')
plt.ylabel('Résidus')
plt.title('Analyse des Résidus')
plt.show()
```

5. **Expliquez l'impact du bruit sur la précision de la régression.**

   Le bruit introduit dans les données a un impact direct sur la précision du modèle de régression. Lorsque des erreurs aléatoires sont ajoutées à la relation linéaire, la droite de régression s'adapte à ces fluctuations et peut devenir moins précise pour prédire les valeurs de y. Si le bruit est important, la pente et l'ordonnée à l'origine du modèle de régression peuvent être biaisées, ce qui conduit à des prédictions moins fiables.

   De plus, les résidus vont montrer une certaine dispersion autour de la ligne des résidus égale à 0. Si les résidus ne sont pas distribués de manière aléatoire (par exemple, s'ils montrent une tendance particulière), cela pourrait indiquer que le modèle de régression linéaire n'est pas le meilleur ajustement pour les données.

- Si le bruit est faible, la régression linéaire donnera une bonne estimation des paramètres.
- Si le bruit est fort, le modèle risque de surajuster les données et de ne pas bien généraliser.

Analyse des résidus dans un modèle de régression

Lorsque l'on ajuste un modèle de régression, l'analyse des **résidus** est une étape cruciale pour évaluer la qualité de l'ajustement du modèle. Les résidus mesurent l'erreur de prédiction pour chaque observation et sont définis comme suit :

$$\text{Résidu} = y_i - \hat{y}_i,$$

où y_i est la valeur observée et $\hat{y}_i$ est la valeur prédite par le modèle.

Rôle de l'analyse des résidus

L'objectif est de vérifier si les résidus sont répartis de manière aléatoire ou s'ils présentent une tendance particulière. Cela permet d'évaluer si le modèle capture correctement la relation entre les variables explicatives (x) et la variable dépendante (y).

Cas où les résidus semblent aléatoires : Si les résidus sont répartis de manière **aléatoire** autour de $y = 0$, cela indique que :

- le modèle capture correctement la relation entre x et y ;
- il n'y a pas de structure dans les erreurs de prédiction.

Cas où les résidus présentent une tendance : Si les résidus présentent une **tendance** (par exemple, une courbure ou une augmentation systématique), cela signifie que le modèle est mal ajusté. Cela peut indiquer :

- une relation non linéaire entre x et y, mal capturée par un modèle linéaire ;
- une mauvaise spécification du modèle (par exemple, des variables explicatives importantes sont manquantes) ;
- une hétéroscédasticité, c'est-à-dire que la variance des résidus dépend de x.

Exemple pratique

Supposons que l'on ajuste un modèle linéaire aux données. Après avoir obtenu les prédictions, on trace un graphique des résidus :

1. **Résidus aléatoires :** Les résidus sont répartis de manière homogène autour de $y = 0$. Cela indique que le modèle est approprié pour les données.
2. **Résidus avec tendance :** Si les résidus suivent un motif en "U", cela pourrait signifier que la relation entre x et y est quadratique. Dans ce cas, un modèle linéaire serait insuffisant. Une transformation ou un modèle polynomial pourrait être nécessaire.

Transformations ou modèles plus complexes

Si les résidus montrent une tendance particulière :

- On peut appliquer des **transformations** (par exemple, logarithmiques ou quadratiques) pour mieux capturer la relation.
- Si les données sont complexes, il est possible d'utiliser des **modèles plus avancés**, tels que :
  - des régressions non linéaires ;
  - des modèles à base d'arbres (arbres de décision, forêts aléatoires) ;
  - des réseaux neuronaux pour des relations très complexes.

Conclusion

En résumé, un modèle bien ajusté produit des résidus aléatoires sans tendance particulière. Si ce n'est pas le cas, il peut être nécessaire d'explorer des modèles ou transformations alternatifs pour mieux décrire les données.

Exercice 5 : Régression non linéaire

Les données suivantes sont fournies :

x	1	2	3	4	5	6
y	1	4	9	16	25	36

Étape 1 : Vérification de la linéarité

On ajuste une droite de régression linéaire $y = \beta_0 + \beta_1 x$ aux données. Le coefficient de détermination R^2 est calculé pour évaluer l'ajustement.

Étape 2 : Transformation linéarisante

Les données y suivent une tendance quadratique ($y = x^2$). Pour linéariser cette relation, nous prenons $y' = \sqrt{y}$. Ainsi, nous avons $y' \approx \beta_0 + \beta_1 x$, ce qui permet d'utiliser une régression linéaire.

Étape 3 : Implémentation en Python

Voici le code Python pour effectuer les calculs et tracer les graphiques.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

# Données initiales
x = np.array([1, 2, 3, 4, 5, 6]).reshape(-1, 1)  # Variable explicative
```

```
y = np.array([1, 4, 9, 16, 25, 36])  # Variable cible

# Étape 1 : Ajustement d'un modèle linéaire sur les données non transformées
model_linear = LinearRegression()
model_linear.fit(x, y)
y_pred_linear = model_linear.predict(x)
r2_linear = r2_score(y, y_pred_linear)

# Visualisation des données et de la régression linéaire
plt.figure(figsize=(10, 6))
plt.scatter(x, y, color="blue", label="Données originales")
plt.plot(x, y_pred_linear, color="red", label="Régression linéaire")
plt.title("Régression linéaire sur les données non transformées")
plt.xlabel("x")
plt.ylabel("y")
plt.legend()
plt.show()

print("Modèle linéaire sur données originales:")
print(f"Intercept (): {model_linear.intercept_:.2f}")
print(f"Coefficient pour x (): {model_linear.coef_[0]:.2f}")
print(f"R²: {r2_linear:.4f}")

# Étape 2 : Transformation linéarisante y' = sqrt(y)
y_transformed = np.sqrt(y)

# Ajustement d'un modèle linéaire sur les données transformées
model_transformed = LinearRegression()
model_transformed.fit(x, y_transformed)
y_pred_transformed = model_transformed.predict(x)
r2_transformed = r2_score(y_transformed, y_pred_transformed)

# Visualisation des données transformées et de la régression linéaire
plt.figure(figsize=(10, 6))
plt.scatter(x, y_transformed, color="green", label="Données transformées (y' = sqrt(y)
plt.plot(x, y_pred_transformed, color="orange", label="Régression linéaire sur y'")
plt.title("Régression linéaire sur les données transformées")
plt.xlabel("x")
plt.ylabel("y'")
plt.legend()
plt.show()

print("\nModèle linéaire sur données transformées:")
print(f"Intercept (): {model_transformed.intercept_:.2f}")
print(f"Coefficient pour x (): {model_transformed.coef_[0]:.2f}")
print(f"R²: {r2_transformed:.4f}")

# Étape 3 : Comparaison des modèles
```

```
print("\nComparaison des modèles:")
print(f"R² sur données originales: {r2_linear:.4f}")
print(f"R² sur données transformées: {r2_transformed:.4f}")
```

Résultats

- **Régression linéaire sur données originales :**

$$\beta_0 = 0.00$$
$$\beta_1 = 7.00$$
$$R^2 = 0.943$$

  Ce modèle ne capture pas bien la relation quadratique entre x et y.

- **Régression linéaire sur données transformées ($y' = \sqrt{y}$) :**

$$\beta_0 = 0.00$$
$$\beta_1 = 1.00$$
$$R^2 = 1.000$$

  Après transformation, le modèle ajuste parfaitement les données.

Interprétation

La transformation $y' = \sqrt{y}$ linéarise les données, permettant au modèle linéaire d'expliquer parfaitement la relation entre x et y'. Comparativement, le modèle initial est inadapté en raison de la nature quadratique des données.

Exercice 6 : Régression multiple

Données :

Surface (x_1, en m^2)	50	60	70	80	90
Chambres (x_2)	2	2	3	3	4
Prix (y, en k€)	200	220	250	280	310

1. Formulation du modèle linéaire

Le modèle de régression multiple s'écrit sous la forme :

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2,$$

où :

- y est le prix en milliers d'euros (k),
- x_1 est la surface en m^2,
- x_2 est le nombre de chambres,
- β_0, β_1, et β_2 sont les coefficients à estimer.

2. Estimation des coefficients par la méthode des moindres carrés

La méthode des moindres carrés repose sur la minimisation de l'erreur quadratique, donnée par la formule :

$$\boldsymbol{\beta} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y},$$

où :

- $\mathbf{X}$ est la matrice des variables explicatives :

$$\mathbf{X} = \begin{bmatrix} 1 & 50 & 2 \\ 1 & 60 & 2 \\ 1 & 70 & 3 \\ 1 & 80 & 3 \\ 1 & 90 & 4 \end{bmatrix},$$

- $\mathbf{y}$ est le vecteur des observations :

$$\mathbf{y} = \begin{bmatrix} 200 \\ 220 \\ 250 \\ 280 \\ 310 \end{bmatrix},$$

- $\boldsymbol{\beta} = [\beta_0, \beta_1, \beta_2]^\top$ est le vecteur des coefficients.

Étapes de calcul

1. **Calcul de $\mathbf{X}^\top \mathbf{X}$:**

$$\mathbf{X}^\top \mathbf{X} = \begin{bmatrix} 5 & 350 & 14 \\ 350 & 25500 & 1020 \\ 14 & 1020 & 42 \end{bmatrix}.$$

2. **Calcul de $\mathbf{X}^\top \mathbf{y}$:**

$$\mathbf{X}^\top \mathbf{y} = \begin{bmatrix} 1260 \\ 90000 \\ 3580 \end{bmatrix}.$$

3. **Calcul de $\boldsymbol{\beta}$:**

$$\boldsymbol{\beta} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}.$$

En résolvant cette équation à l'aide d'un solveur numérique, nous obtenons :

$$\boldsymbol{\beta} = \begin{bmatrix} 60.67 \\ 2.47 \\ 6.67 \end{bmatrix}.$$

Ainsi, le modèle estimé est :

$$y = 60.67 + 2.47x_1 + 6.67x_2.$$

3. Prédiction pour une maison de 85 m² avec 3 chambres

En substituant $x_1 = 85$ et $x_2 = 3$ dans le modèle, on obtient :

$$y = 60.67 + 2.47 \cdot 85 + 6.67 \cdot 3.$$

Effectuons les calculs :

$$y = 290.33\,k.$$

4. Comparaison avec une bibliothèque Python

En utilisant la bibliothèque `sklearn` en Python, le même calcul peut être effectué :

```
import numpy as np
from sklearn.linear_model import LinearRegression

# Données
X = np.array([[50, 2], [60, 2], [70, 3], [80, 3], [90, 4]])
y = np.array([200, 220, 250, 280, 310])

# Modèle
model = LinearRegression()
model.fit(X, y)

# Coefficients
beta_0 = model.intercept_
beta_1, beta_2 = model.coef_

# Prédiction
y_pred = model.predict([[85, 3]])
```

Les résultats obtenus avec Python sont identiques :

$$\beta_0 = 60.67, \quad \beta_1 = 2.47, \quad \beta_2 = 6.67,$$

et la prédiction pour une maison de 85 m² avec 3 chambres est :

$$y = 290.33\,k.$$

Conclusion

Le modèle estimé est cohérent avec les données, et les calculs manuels et numériques concordent. Les coefficients $\beta_0 = 60.67$, $\beta_1 = 2.47$, et $\beta_2 = 6.67$ décrivent bien la relation entre les variables explicatives et y.

1. Définition du biais

Un paramètre (comme la pente β_1 ou l'ordonnée à l'origine β_0) ou une prédiction est biaisé si la valeur moyenne estimée par le modèle, à travers de multiples ensembles de données ou simulations, est différente de la vraie valeur de ce paramètre.

Matériellement, si on note la vraie valeur d'un paramètre θ comme θ^* et son estimation par le modèle comme $\hat{\theta}$, le biais est défini par :

$$\text{Biais} = \mathbb{E}[\hat{\theta}] - \theta^*.$$

Si Biais $\neq 0$, alors l'estimation est biaisée.

2. Pourquoi les estimations peuvent-elles être biaisées ?

Les biais surviennent lorsque :

- **Modèle incorrect** : Le modèle choisi ne reflète pas correctement la relation entre les variables. Par exemple, utiliser un modèle linéaire pour une relation quadratique.
- **Bruit élevé** : Lorsque les données sont très bruitées, le modèle peut attribuer des variations aléatoires à la relation entre les variables explicatives et la cible.
- **Sous-échantillonnage** : Avec un faible volume de données ou un échantillon non représentatif, le modèle ne peut pas bien estimer les paramètres.
- **Biais dans les données** : Les données elles-mêmes peuvent contenir des erreurs ou ne pas représenter la réalité correctement.

3. Impact d'estimations biaisées

Si les estimations des coefficients (β_0, β_1) sont biaisées, cela signifie que le modèle produit systématiquement des prédictions incorrectes, même sur les données qu'il n'a pas encore vues.

- Par exemple, une pente biaisée (β_1) pourrait sous-estimer ou surestimer l'importance d'une variable explicative (x) sur la variable cible (y).

4. Exemple simple

Supposons que la vraie relation entre x et y soit donnée par :

$$y = 3x + 5.$$

Cependant, lors de l'estimation, à cause de données bruitées ou d'une erreur dans le modèle, le modèle estime :

$$\hat{y} = 2.5x + 4.$$

Dans ce cas :

- $\hat{\beta}_1 = 2.5$ est biaisée par rapport à la vraie valeur $\beta_1 = 3$.
- $\hat{\beta}_0 = 4$ est biaisée par rapport à la vraie valeur $\beta_0 = 5$.

5. Comment réduire le biais ?

Pour minimiser le biais dans les estimations :

- **Utiliser un modèle adapté** : Choisir un modèle qui reflète la vraie relation entre les variables.
- **Augmenter les données** : Travailler avec un échantillon plus grand et représentatif.
- **Réduction du bruit** : Filtrer ou réduire les perturbations dans les données.

En résumé

Des estimations biaisées ne sont pas centrées autour de la vraie valeur, ce qui diminue la qualité et la fiabilité du modèle.

Exercice 7 : Régression et biais/variance

1. **Générez 100 points selon la relation $y = 3x + 2$ avec différents niveaux de bruit (par exemple, bruit faible, bruit moyen, bruit élevé).**

   Nous générons les points de données en utilisant la relation linéaire $y = 3x + 2$, en ajoutant différents niveaux de bruit, c'est-à-dire en ajoutant une valeur aléatoire à chaque point, dont l'écart-type varie en fonction du niveau de bruit. Trois niveaux de bruit sont utilisés : faible, moyen et élevé.

2. **Ajustez un modèle linéaire pour chaque jeu de données.**

   Nous ajustons un modèle linéaire aux trois jeux de données (avec bruit faible, moyen et élevé) en utilisant la méthode des moindres carrés.

3. **Tracez les données, les modèles ajustés, et discutez de l'impact du bruit en termes de biais et variance.**

   Impact du bruit :

   - **Bruit faible :** Lorsque le bruit est faible, les données sont proches de la relation $y = 3x + 2$, et le modèle ajusté suit bien cette relation. Le biais est faible et la variance est également faible.
   - **Bruit moyen :** Avec un bruit moyen, les données commencent à être légèrement dispersées autour de la relation linéaire $y = 3x + 2$. Le biais reste encore faible, mais la variance est plus élevée.
   - **Bruit élevé :** Lorsqu'un bruit élevé est ajouté, les données sont fortement dispersées, et le modèle ajusté commence à suivre trop de variations aléatoires dans les données. Cela augmente la variance et le biais, car le modèle n'est plus capable de capturer la vraie relation sous-jacente.

4. **Expliquez comment la taille des données affecte la performance de la régression.**

   La taille des données a un impact significatif sur la performance du modèle de régression. En général :

   - **Petite taille d'échantillon :** Si l'échantillon est petit, le modèle peut souffrir de sur-ajustement (overfitting) ou de sous-ajustement (underfitting), surtout si le bruit est élevé. Un petit nombre de points de données ne permet pas au modèle de bien généraliser.
   - **Grande taille d'échantillon :** Avec un grand nombre de points de données, le modèle a plus de chances d'apprendre les tendances sous-jacentes et de mieux généraliser. Cela peut réduire à la fois le biais et la variance, car les variations spécifiques aux petits échantillons sont lissées.

Conclusion

- Un faible niveau de bruit donne un modèle avec un faible biais et une faible variance.
- Un bruit plus élevé augmente la variance du modèle, ce qui peut conduire à un sur-ajustement des données (overfitting).

- La taille de l'échantillon joue un rôle crucial : un grand nombre de points de données permet de mieux généraliser et d'améliorer la stabilité des prédictions.

Exercice 8 : Interprétation des résidus

1. **Après avoir ajusté une régression linéaire sur des données, calculez et tracez les résidus (différence entre y_i observés et $\hat{y}_i$ prédits).**

   Après avoir ajusté un modèle de régression linéaire aux données, nous pouvons calculer les résidus e_i pour chaque observation i comme suit :

   $$e_i = y_i - \hat{y}_i$$

   où y_i est la valeur observée et $\hat{y}_i$ est la valeur prédite par le modèle. Pour calculer les résidus et les tracer, nous pouvons utiliser Python et Matplotlib.

   Code Python :

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

# Données (exemple simple)
x = np.array([1, 2, 3, 4, 5])
y = np.array([1.5, 2.5, 3.5, 4.5, 5.5])

# Reshape pour la régression
x_reshape = x.reshape(-1, 1)

# Ajuster la régression linéaire
model = LinearRegression()
model.fit(x_reshape, y)

# Prédictions et résidus
y_pred = model.predict(x_reshape)
residuals = y - y_pred

# Tracer les résidus
plt.scatter(x, residuals)
plt.axhline(0, color='red', linestyle='--')
plt.title('Résidus')
plt.xlabel('x')
plt.ylabel('Résidus')
plt.show()
```

   Ce code calcule les résidus et les trace sous forme de points, avec une ligne rouge représentant l'axe des résidus (à $y = 0$).

Résultats et Interprétation

Coefficients appris par le modèle :

- $\beta_0 = 0.5$ (ordonnée à l'origine).
- $\beta_1 = 1.0$ (pente).

La droite ajustée est donc :

$$\hat{y} = x + 0.5.$$

Prédictions :

Les prédictions ($\hat{y}$) sont parfaites car les données suivent exactement une relation linéaire :

$$\hat{y} = [1.5, 2.5, 3.5, 4.5, 5.5].$$

Résidus :

Les résidus sont tous égaux à zéro car il n'y a pas de bruit ou d'erreur dans les données :

$$\text{Résidus} = y - \hat{y} = 0 \quad \text{pour chaque point.}$$

Le graphique des résidus montre que tous les points sont alignés sur la ligne rouge ($y = 0$).

Validation du modèle :

Comme les résidus ne présentent aucun motif ou tendance (ils sont constants à zéro), cela confirme que le modèle est parfaitement ajusté.

2. **Vérifiez si les résidus sont distribués aléatoirement autour de 0.**

   Une fois les résidus calculés et tracés, il est important d'examiner s'ils sont distribués de manière aléatoire autour de 0. Si les résidus sont aléatoires, cela signifie que le modèle de régression linéaire est bien adapté aux données. En revanche, s'il existe une tendance ou une structure évidente dans les résidus, cela indique que le modèle ne capture pas bien la relation sous-jacente entre les variables.

   Interprétation des résidus : - Distribution aléatoire autour de 0 : Si les résidus sont distribués aléatoirement, cela suggère que le modèle de régression linéaire est approprié pour ces données. Cela signifie que l'erreur de prédiction est distribuée de manière homogène et qu'il n'y a pas de structure systématique non modélisée. - **Présence d'une tendance dans les résidus :** Si une tendance est visible dans les résidus (par exemple, une courbure ou une structure), cela suggère que le modèle de régression linéaire ne capture pas bien la relation entre les variables. Cela peut indiquer qu'un modèle non linéaire pourrait être plus adapté aux données.

3. **Discutez ce que cela signifie si une tendance est visible dans les résidus.**

   Si une tendance est visible dans les résidus, cela signifie que le modèle de régression linéaire ne capture pas toutes les relations pertinentes entre les variables. Voici quelques explications possibles : - **Non-linéarité :** La relation entre x et y pourrait être non linéaire, et une régression linéaire ne suffit pas à expliquer cette relation. Par exemple, une courbe ou une fonction quadratique pourrait mieux représenter les données. - **Variables manquantes :** Il est possible qu'il manque des variables explicatives dans le modèle. Si certaines variables importantes ne sont pas incluses dans le modèle, cela peut se traduire par des résidus structurés. - **Hétéroscédasticité :** Si la variance des résidus n'est pas constante (par exemple, si les résidus sont plus dispersés pour certaines valeurs de x que pour d'autres), cela peut suggérer un problème d'hétéroscédasticité, ce qui signifie que la variance de l'erreur n'est pas constante.

   Pour résoudre ce problème, il peut être nécessaire d'essayer d'autres types de modèles (par exemple, régression polynomiale ou modèles non linéaires) ou de transformer les données.