MINISTERE DE L’ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE

(MESR)

REPUBLIQUE TOGOLAISE

*Travail-Liberté-Patrie*

**ECOLE SUPERIEURE DE GESTION D’INFORMATIQUE ET DES SCIENCES**

**Document de rapport**

**Domaine :** Sciences de l’ingénieur

**Mention :** Informatique

**Spécialité :** Intelligence Artificielle et Big Data

**EVALUATION FINALE : INTELLIGENCE ARTIFICIELLE**

(SWLTI)

Rédigé par :

**Abalaliwa LOMDO**

**MAHAMAT Cherif Issa**

**MASSINOU Kokou Luc**

Superviseur du cours :

**NADJAK KAN’SAM**

Enseignant à ESGIS

**Année académique : 2023-2024**

# SOMMAIRE

[INTRODUCTION GENERALE 1](#_Toc158248649)

[REPOSNES AUX QUESTIONS 2](#_Toc158248650)

[1. L’objet de notre dataset : pourquoi le choix de ce dataset? L’intérêt / but de cette étude et vos attente / utilités : 2](#_Toc158248651)

[2. Outils et Packages 2](#_Toc158248652)

[3. Affichage et interprétations des graphes de chaque variable d’entrée avec la variable de sortie. 5](#_Toc158248653)

[4. Notre conclusion sur le Principe de corrélation entre chaque variable d’entrée et la sortie 9](#_Toc158248654)

[5. Les méthodes de ML/AI peut être utilisé dans notre cas est : 9](#_Toc158248655)

[6. Voici une explication détaillée de chaque étape, ainsi que des commentaires sur les résultats à chaque phase : 10](#_Toc158248656)

[7. Commentons nos résultat à chaque phase ( traning et testing) 11](#_Toc158248657)

[8. Voici une conclusion en utilisant les termes demandés (bais, variance, underfitting, overfitting) : 12](#_Toc158248658)

[9. Les difficultés et faisabilités 13](#_Toc158248659)

[CONCLUSION GENERALE 14](#_Toc158248660)

[LE CODE I](#_Toc158248661)

# INTRODUCTION GENERALE

Dans le domaine en constante expansion de la science des données, il est crucial de comprendre les tendances salariales et les facteurs qui influent sur les rémunérations des professionnels. Ce projet vise à explorer et à analyser un ensemble de données concernant les salaires des différents métiers liés à la science des données, tels que les Data Scientists, Data Engineers et Data Analysts. L'objectif principal de cette étude est d'identifier les variables qui ont le plus d'impact sur les salaires dans ce domaine en croissance rapide. Nous examinerons diverses caractéristiques telles que l'expérience, le titre du poste, le lieu de travail, le niveau de formation, et d'autres encore, pour déterminer leur corrélation avec les salaires des professionnels des données.

En comprenant mieux les tendances salariales et les facteurs qui les influencent, les entreprises pourront prendre des décisions éclairées en matière de rémunération, les professionnels pourront mieux négocier leurs salaires et les chercheurs pourront identifier les domaines où il est nécessaire de concentrer leurs efforts de formation et de développement professionnel.

# REPOSNES AUX QUESTIONS

## L’objet de notre dataset : pourquoi le choix de ce dataset? L’intérêt / but de cette étude et vos attente / utilités :

Le dataset sur les salaires des professionnels de la data tels que les Data Scientists, Data Engineers ou Data Analysts a été choisi pour plusieurs raisons. Tout d'abord, ces professions sont devenues cruciales dans de nombreux secteurs avec l'essor de l'analyse de données et de l'intelligence artificielle. Comprendre les tendances salariales dans ces domaines est donc essentiel pour les professionnels aspirant à ces postes ainsi que pour les entreprises qui cherchent à attirer et à retenir les talents.

L'objectif principal de cette étude est d'analyser les facteurs qui influencent les salaires des professionnels de la science, de l'ingénierie et de l'analyse des données. Cela comprend l'examination des variables telles que l'expérience professionnelle, le niveau d'éducation, le lieu de travail, le secteur d'activité, etc., afin de déterminer leur impact sur les rémunérations perçues par ces professionnels.

Les attentes associées à cette étude sont multiples. Premièrement, elle devrait nous permettre de mieux comprendre les tendances salariales dans ces domaines ; ce qui pourrait être utile pour les professionnels qui cherchent à négocier leur salaire ou à planifier leur carrière. De plus, elle pourrait fournir des informations précieuses aux entreprises pour élaborer des politiques de rémunération compétitives et pour recruter et retenir les meilleurs talents. Enfin, cette étude pourrait également être utile aux chercheurs et décideurs qui s'intéressent à l'évolution du marché du travail dans le domaine de l'analyse des données.

## Outils et Packages

Les outils et packages utilisés pour traiter les données sont les suivants :

**Pandas** : Une bibliothèque Python qui offre des structures de données et des outils de manipulation de données puissants et flexibles, notamment des DataFrames pour organiser les données tabulaires.

**NumPy** : Une bibliothèque Python qui prend en charge les tableaux multidimensionnels et les matrices, offrant une grande variété de fonctions mathématiques pour effectuer des opérations sur les structures de données.

**Sklearn.model\_selection.train\_test\_split** : est une fonction de Scikit-Learn qui permet de diviser un ensemble de données en ensembles d'entraînement et de test pour évaluer les performances d'un modèle.

**Matplotlib** : Une bibliothèque de visualisation Python permettant de créer des graphiques et des visualisations de données. Pyplot est un module de Matplotlib qui offre une interface similaire à MATLAB pour la création de graphiques.

**Plotly** : est une bibliothèque de visualisation interactive qui permet de créer des graphiques interactifs et des tableaux de bord. Plotly Express est une interface de haut niveau de Plotly qui simplifie la création de graphiques.

**Seaborn** : Une bibliothèque de visualisation de données basée sur Matplotlib, offrant une syntaxe simplifiée pour créer des graphiques statistiques attrayants.

**TableOne** : Un package permettant de créer un tableau croiser statistiques récapitulatif baser sur un fichier Excel, tout en faisant une analyse sur une p-value pouvant varier de 0-1.

Une P-Value inférieure à 0,05 est généralement considérée comme statistiquement significative, auquel cas l'hypothèse nulle doit être rejetée. Une valeur p supérieure à 0,05 signifie que l’écart par rapport à l’hypothèse nulle n’est pas statistiquement significatif et que l’hypothèse nulle n’est pas rejetée.

**Plotly.offline.iplot** et **Plotly.offline.plot** : Des fonctions permettant de créer des graphiques interactifs avec Plotly et de les afficher dans un notebook Jupyter ou de les sauvegarder localement.

**Warnings** : Ce module permet de gérer les avertissements et les messages d'avertissement émis par Python ou les bibliothèques utilisées.

**Sklearn.preprocessing.LabelEncoder** : LabelEncoder est une classe de scikit-learn utilisée pour encoder des étiquettes catégorielles en nombres entiers pour les utiliser dans les modèles de Machine Learning.

**LinearRegression** : Cette classe est utilisée pour créer un modèle de régression linéaire. Elle est souvent utilisée lorsque vous avez une variable cible continue que vous essayez de prédire en fonction d'une ou plusieurs variables explicatives. Le modèle tente de modéliser la relation linéaire entre les variables explicatives et la variable cible en ajustant une ligne ou un hyperplan à travers les données.

**RandomForestRegressor** : Cette classe est utilisée pour créer un modèle de régression basé sur l'algorithme de forêt aléatoire. Un modèle de forêt aléatoire est construit à partir d'un grand nombre d'arbres de décision, où chaque arbre de décision est entraîné sur un sous-ensemble aléatoire des données. Ensuite, les prédictions de chaque arbre sont agrégées pour produire la prédiction finale.

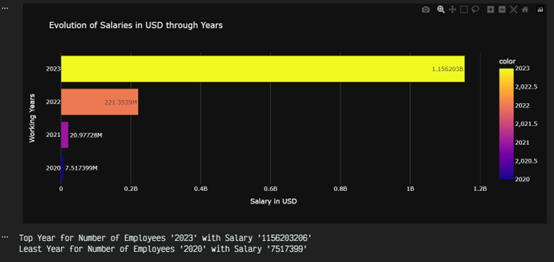
**GradientBoostingRegressor** : Cette classe est utilisée pour créer un modèle de régression basé sur l'algorithme de boosting de gradient. Le boosting de gradient est une technique d'ensemble où les modèles de prédiction faibles (souvent des arbres de décision peu profonds) sont ajoutés séquentiellement au modèle, en mettant l'accent sur les données mal prédites par les modèles précédents.

**Mean\_squared\_error (MSE)** : Cette fonction est utilisée pour évaluer les performances d'un modèle de régression en calculant l'erreur quadratique moyenne entre les valeurs prédites par le modèle et les valeurs réelles de la variable cible. Une erreur quadratique plus faible indique un meilleur ajustement du modèle aux données.

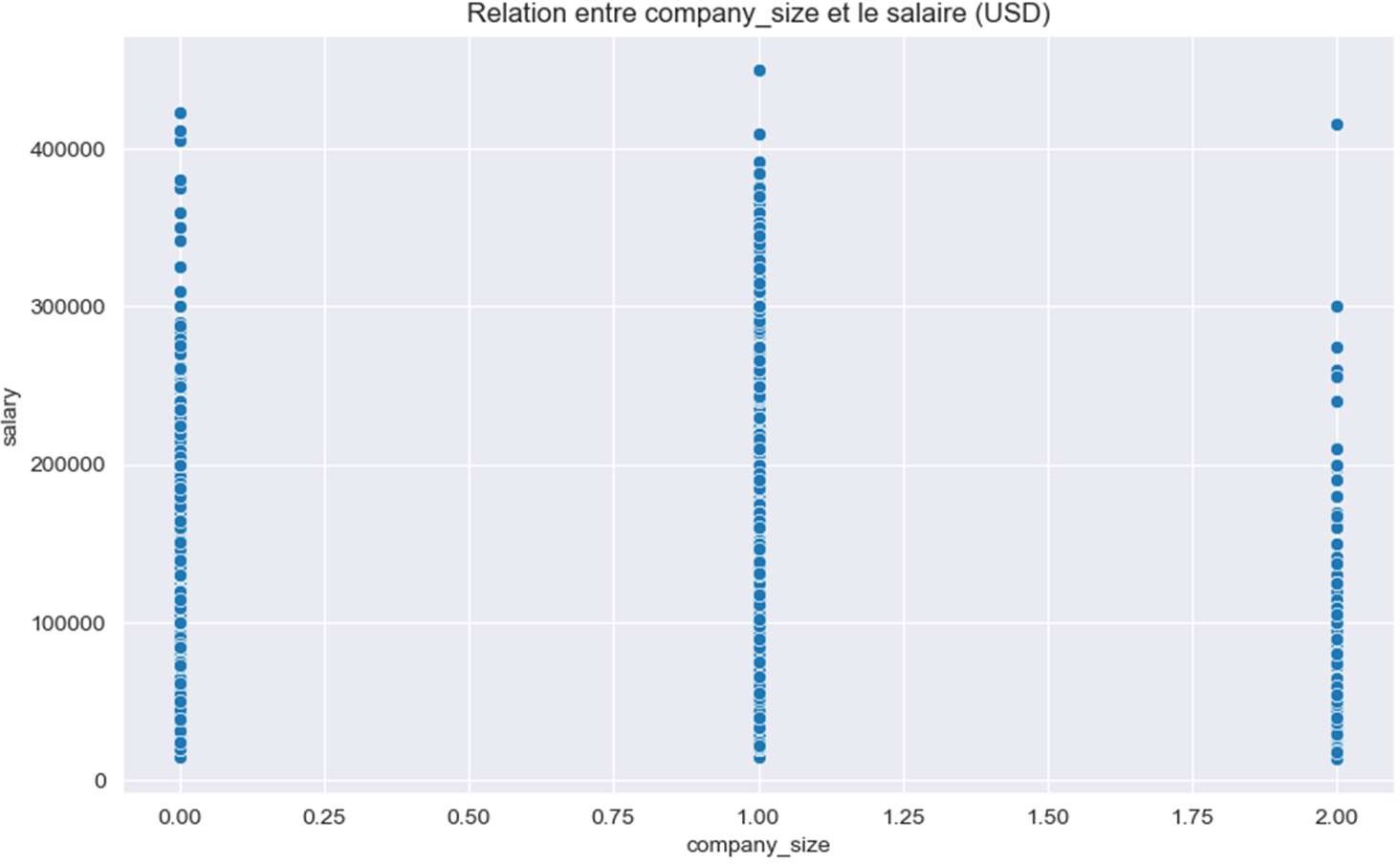
**Mean\_absolute\_error (MAE)** : Cette fonction est utilisée pour évaluer les performances d'un modèle de régression en calculant l'erreur absolue moyenne entre les valeurs prédites par le modèle et les valeurs réelles de la variable cible. Une erreur absolue moyenne plus faible indique un meilleur ajustement du modèle aux données.

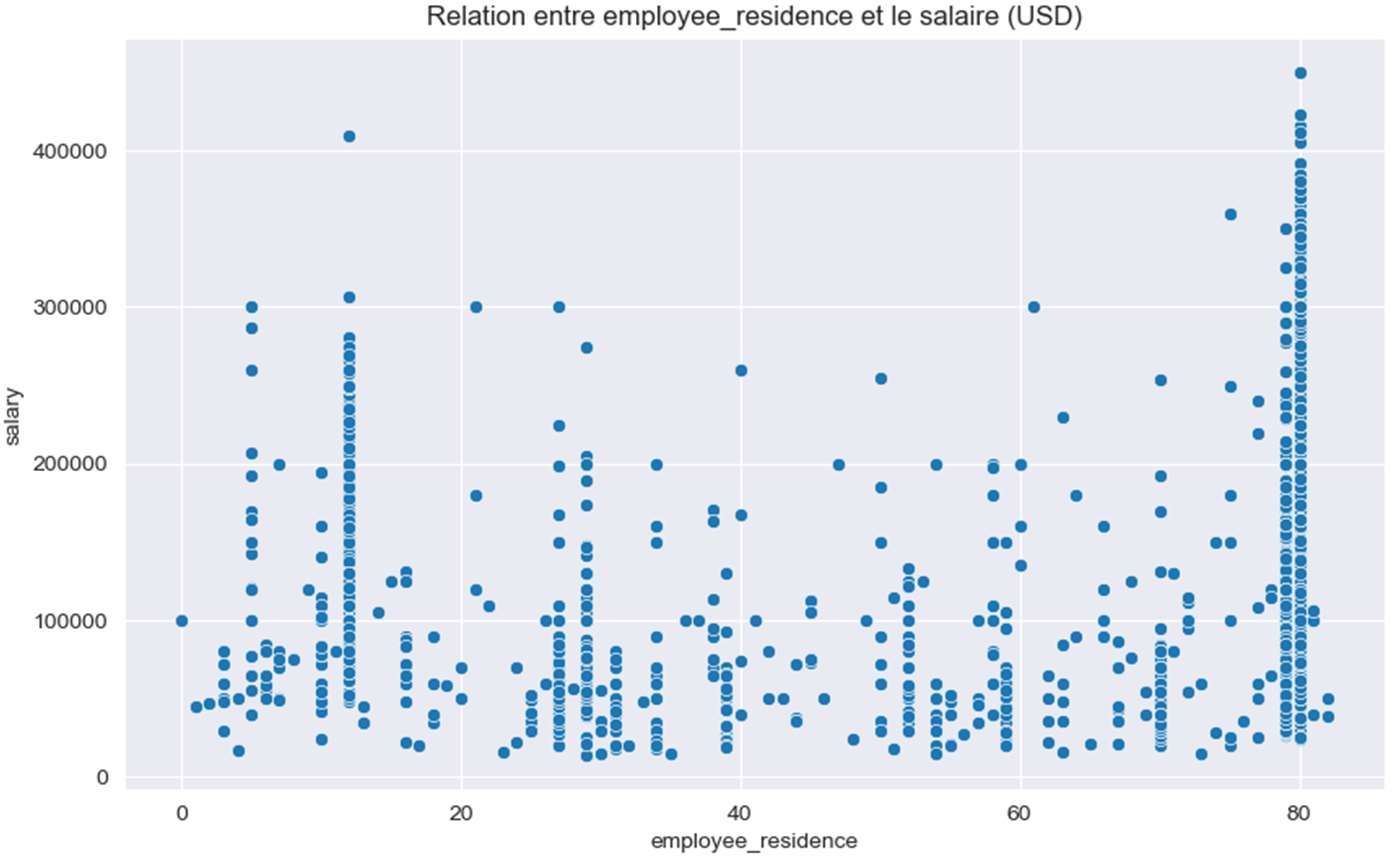
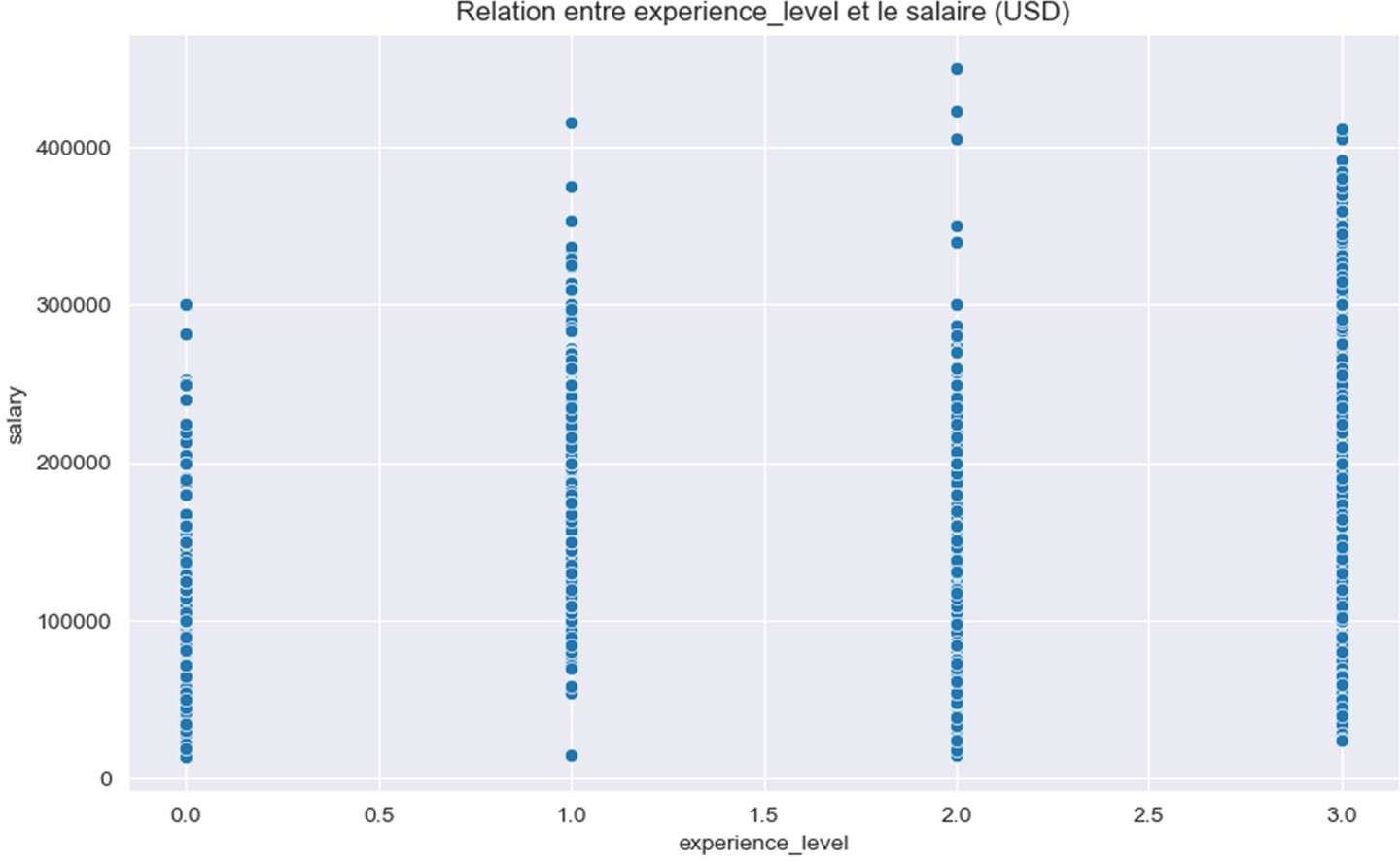
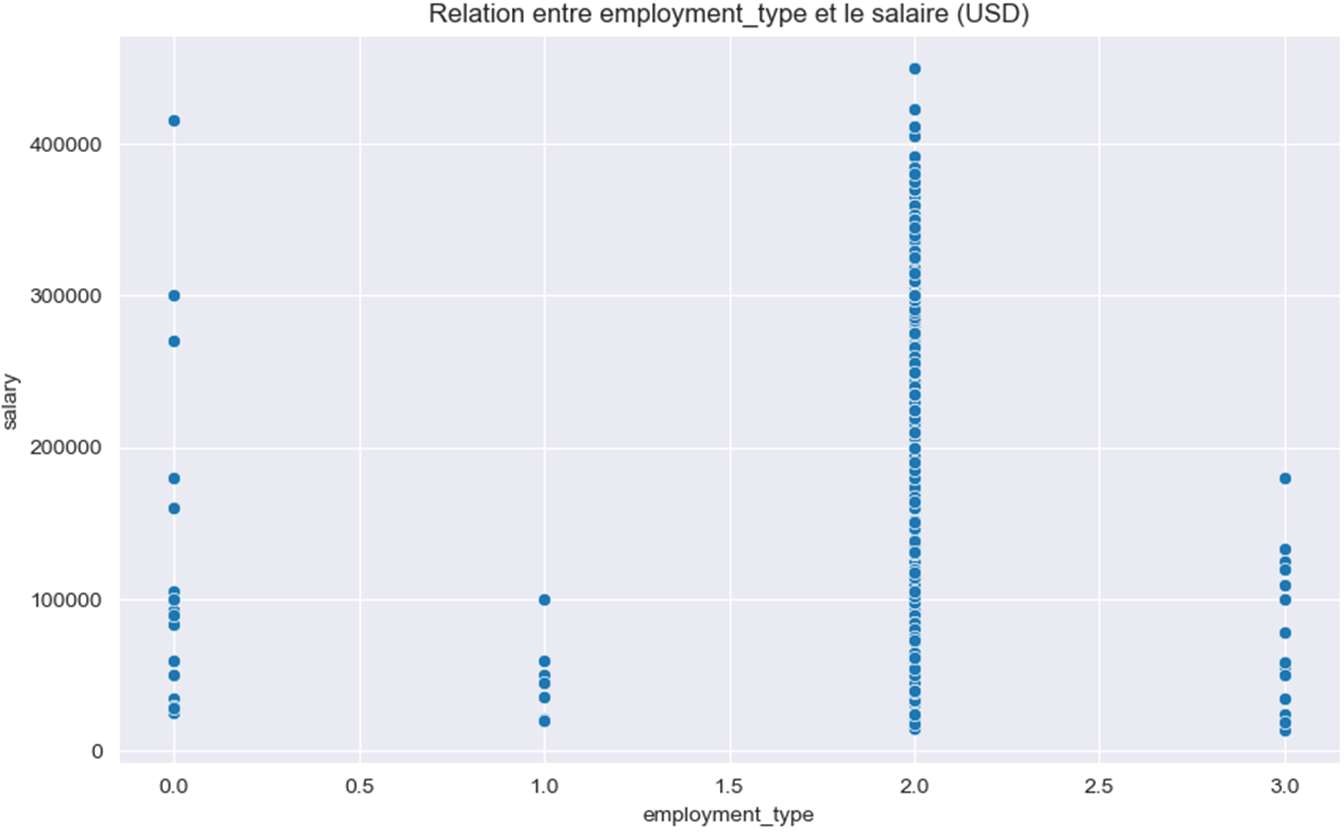
**r2\_score (R²)** : Cette fonction est utilisée pour évaluer les performances d'un modèle de régression en calculant le coefficient de détermination R². Ce coefficient mesure la proportion de la variance de la variable cible qui est expliquée par le modèle il est compris entre 0 et 1. Un R² plus élevé indique un meilleur ajustement du modèle aux données.

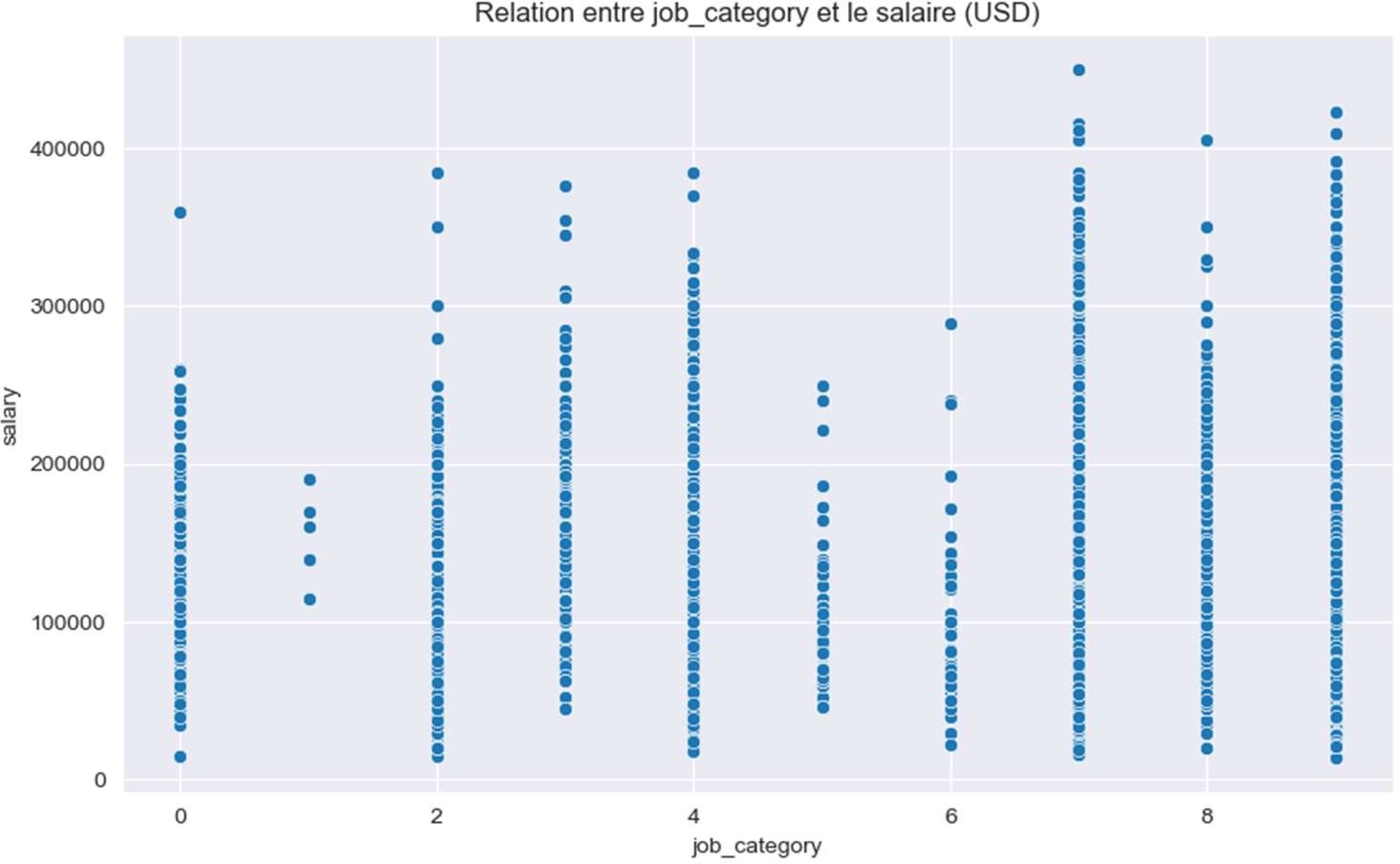
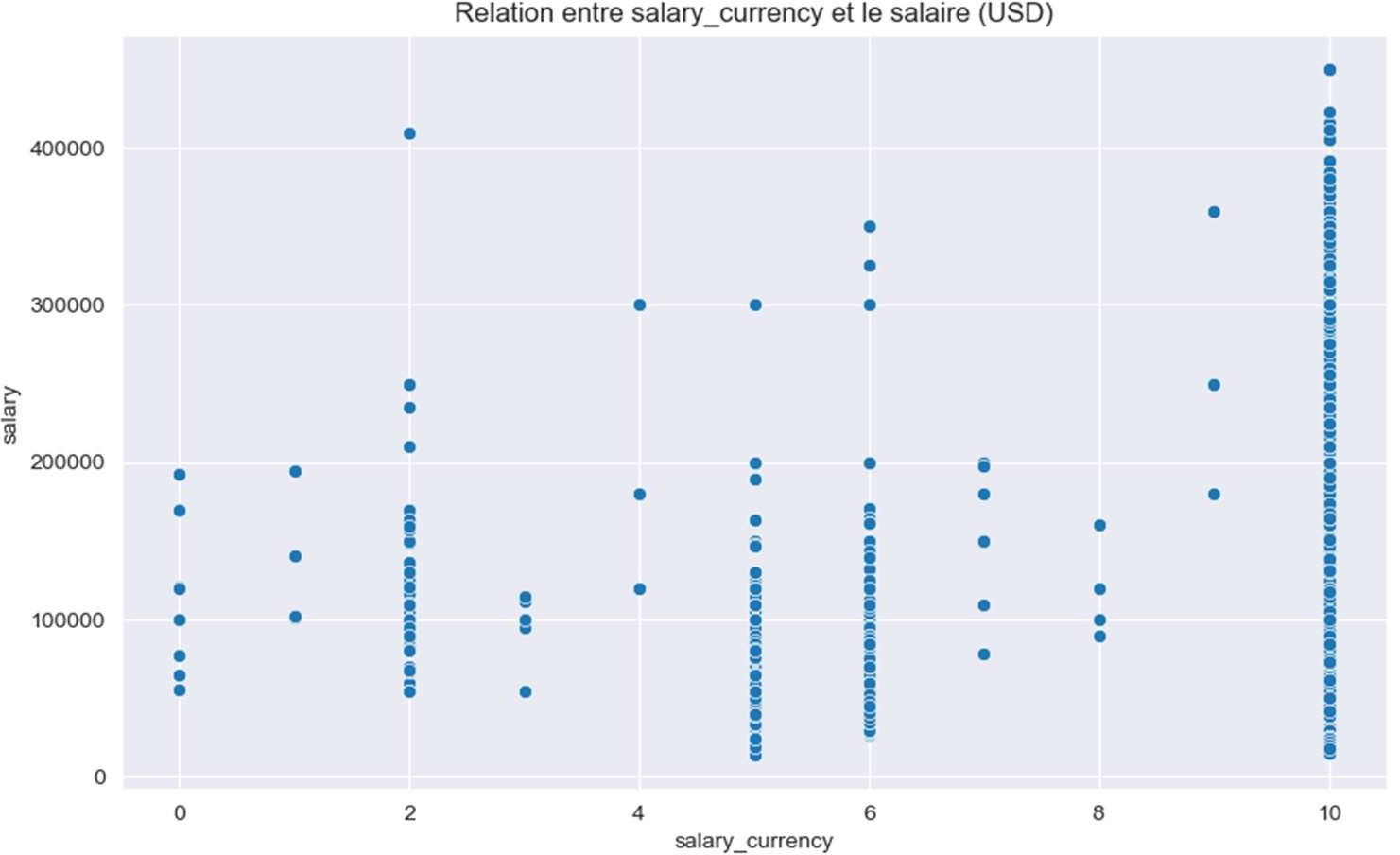
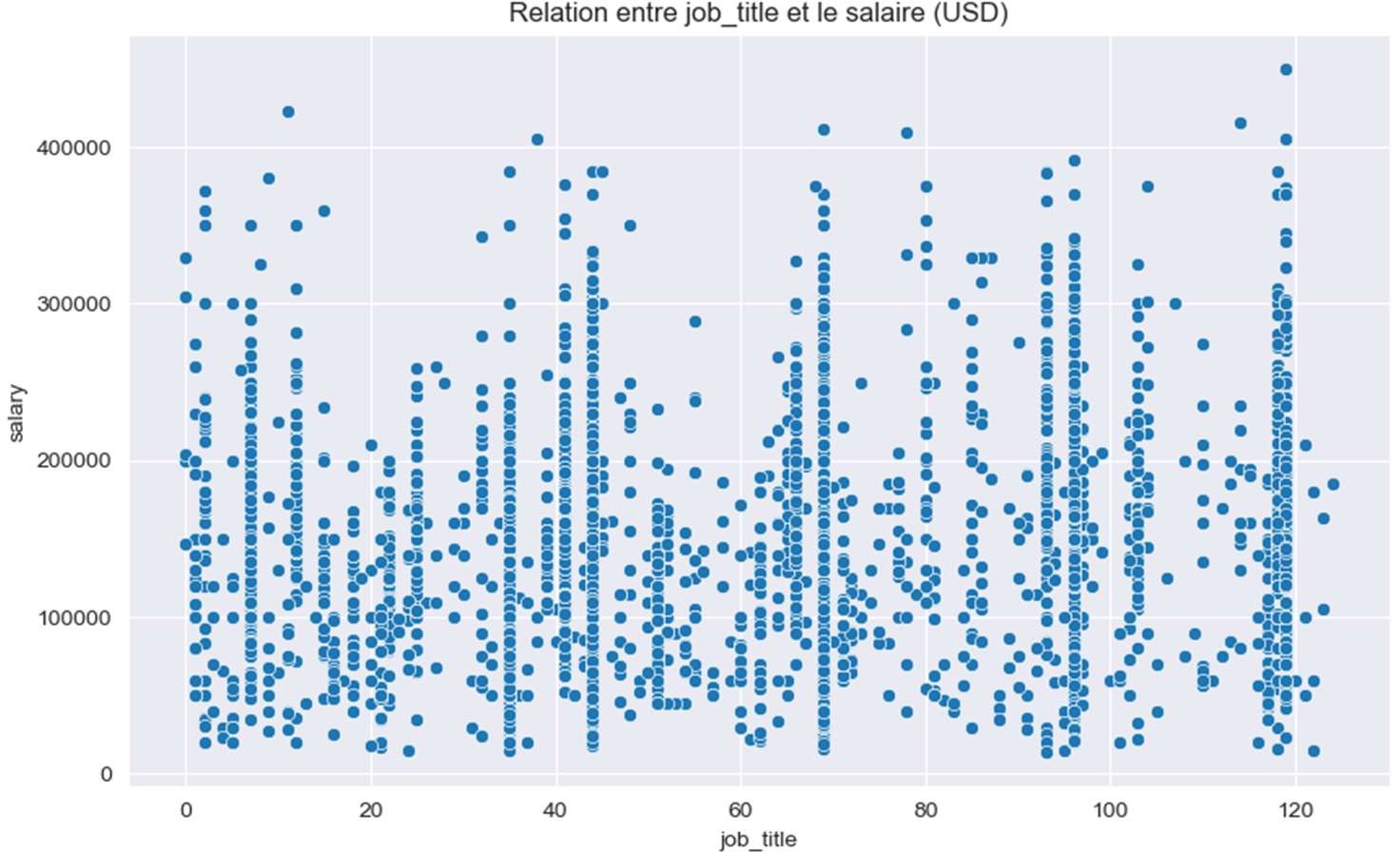
## Affichage et interprétations des graphes de chaque variable d’entrée avec la variable de sortie.

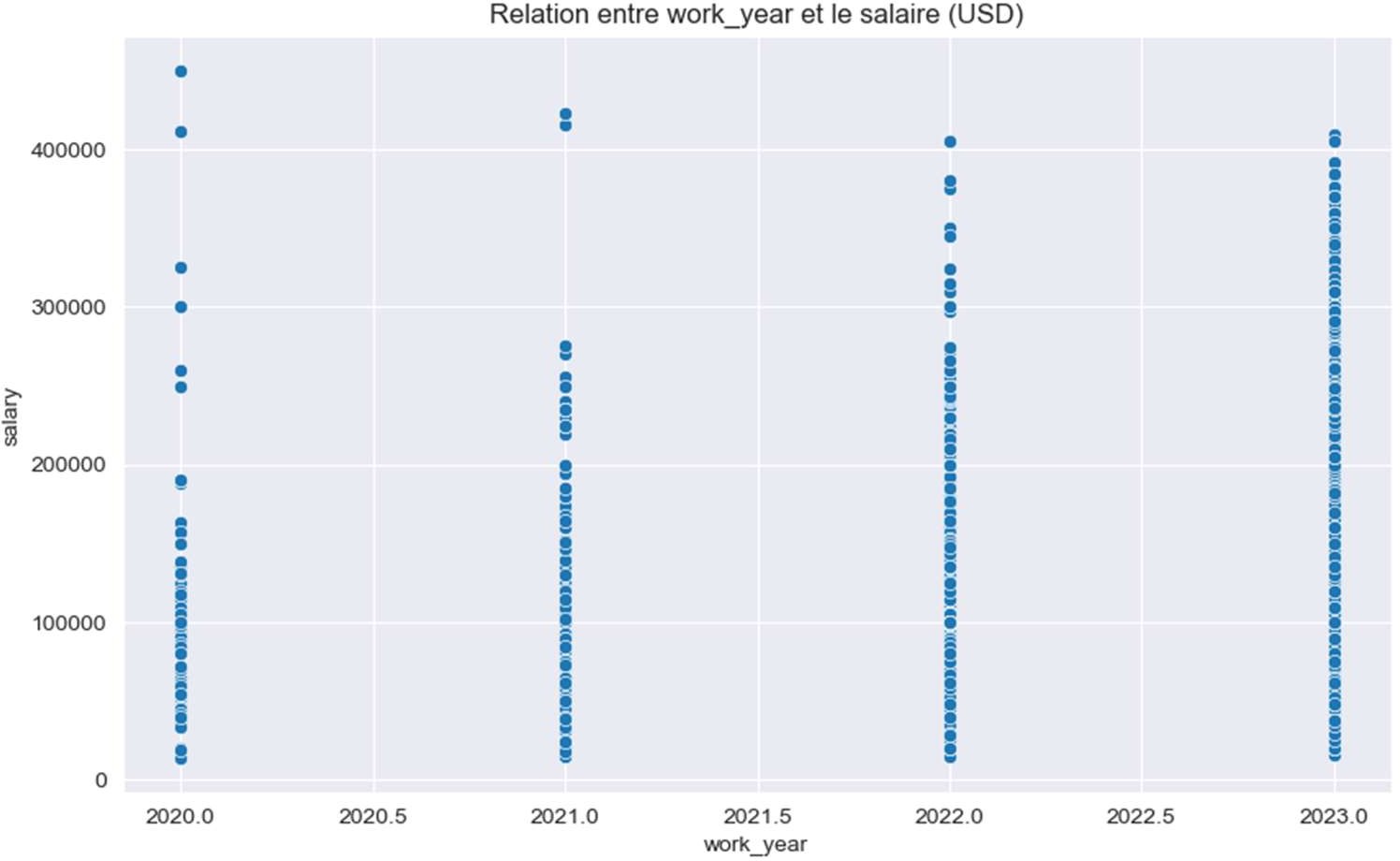
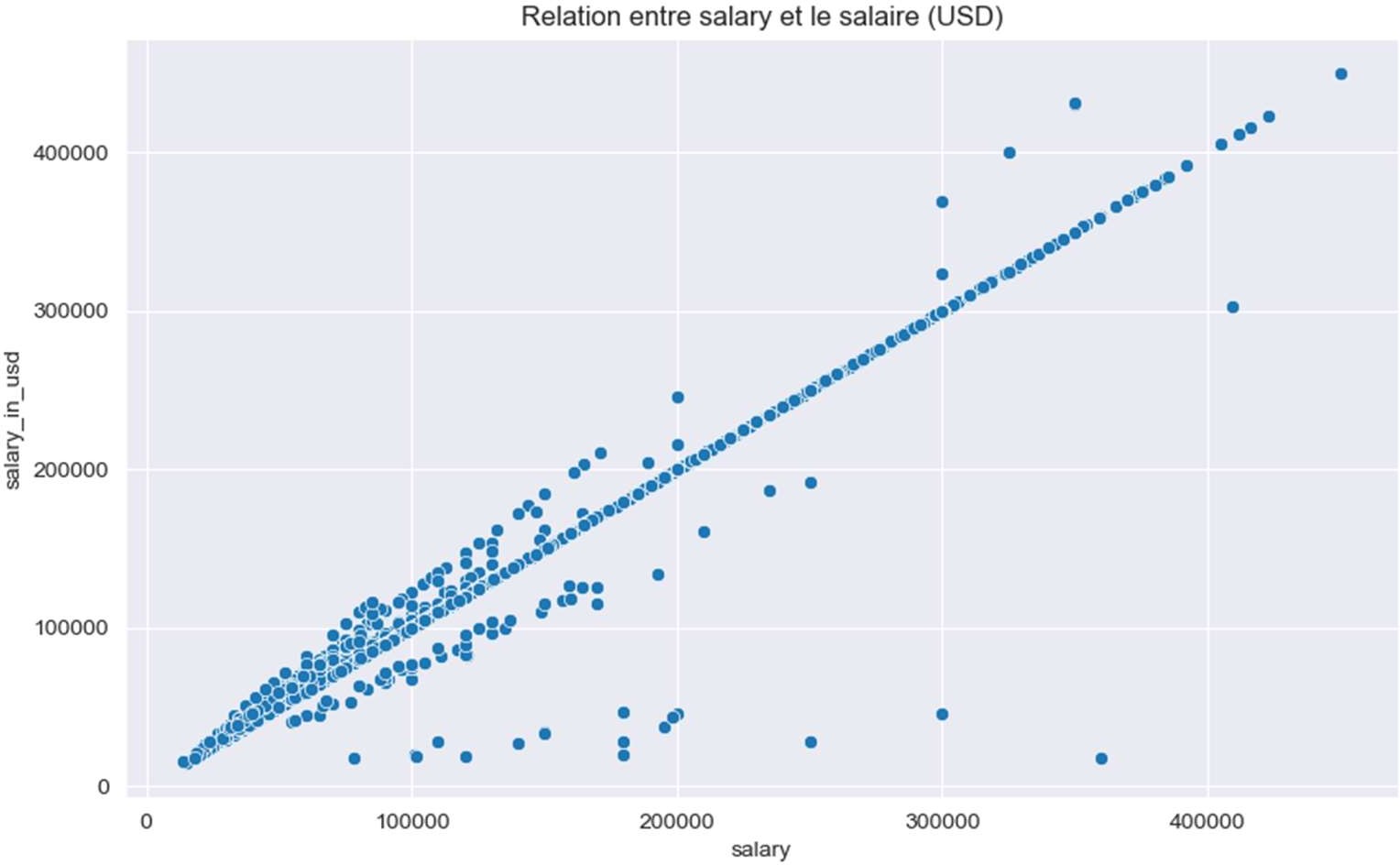
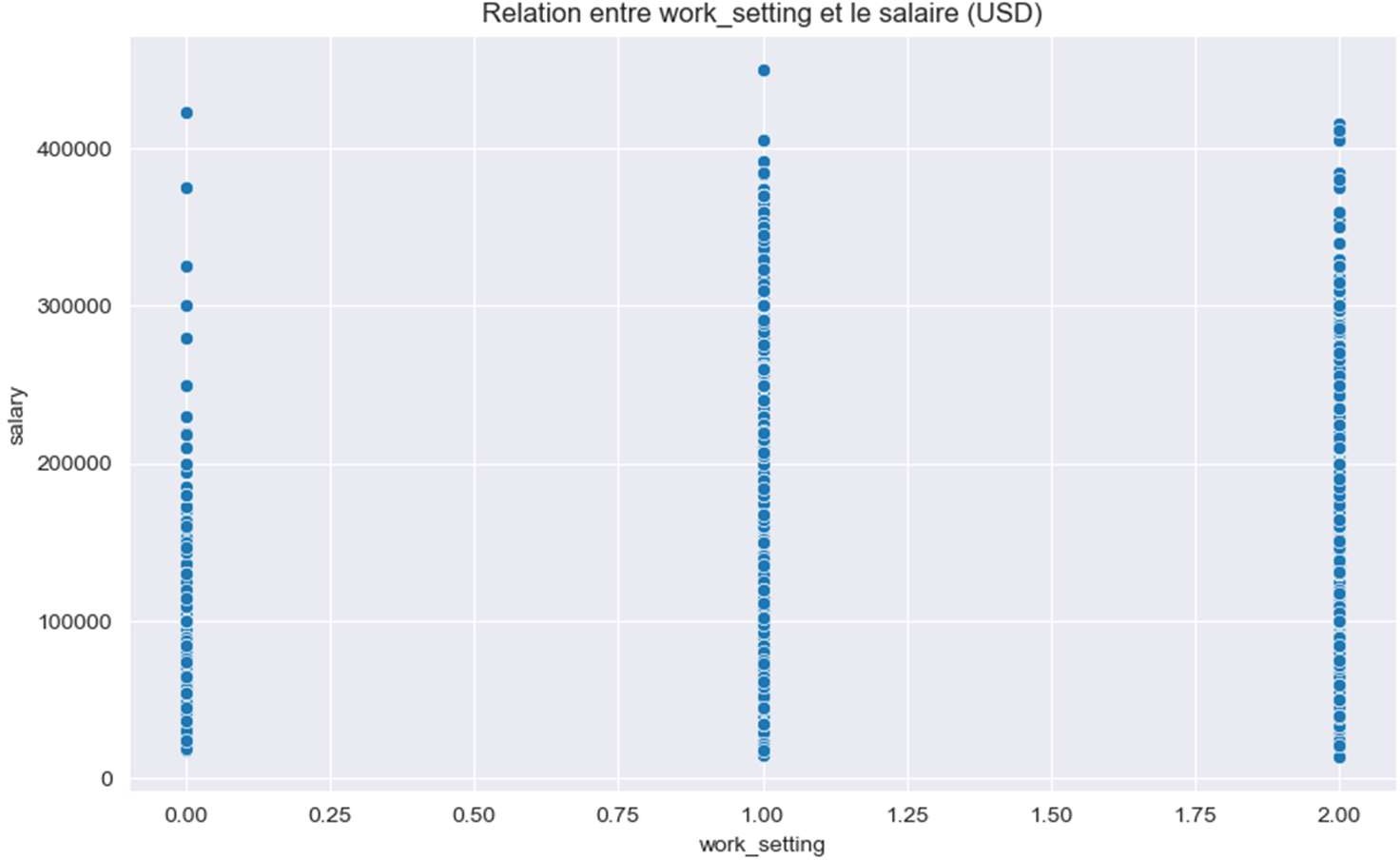


Graphe comparatif entre année de travail et salaire en usb









## Notre conclusion sur le Principe de corrélation entre chaque variable d’entrée et la sortie

En conclusion, cette analyse de corrélation met en lumière plusieurs variables qui sont étroitement liées au salaire perçu par les employés. Cela peut être utile pour comprendre les facteurs qui influencent les niveaux de rémunération et orienter les décisions de gestion liées à la politique salariale et au recrutement.

## Les méthodes de ML/AI peut être utilisé dans notre cas est :

Dans notre cas, plusieurs méthodes de Machine Learning (ML) et d'Intelligence Artificielle (IA) peuvent être utilisées en fonction de différents critères tels que la précision, la complexité du modèle, la capacité à gérer les données de grande taille, et la capacité à traiter les relations complexes entre les caractéristiques et la cible. Sur la base des performances observées dans nos résultats, ainsi que des considérations de complexité et de capacité de généralisation, les méthodes suivantes sont recommandées pour notre cas :

* Random Forest
* Gradient Boosting,
* Régression Linéaire
* Réseau de Neurones
* Nearest Neighbors
* Decision Trees

## Voici une explication détaillée de chaque étape, ainsi que des commentaires sur les résultats à chaque phase :

*Étape 1 : Partition du jeu de données (training and test data)*

* Explication :
* Cette étape consiste à diviser votre jeu de données en deux ensembles distincts : un ensemble d'entraînement (training set) et un ensemble de test (test set).
* L'ensemble d'entraînement est utilisé pour entraîner les modèles d'apprentissage automatique.
* L'ensemble de test est utilisé pour évaluer les performances des modèles sur des données non vues lors de l'entraînement.
* Commentaires :
* Il est crucial de choisir une bonne proportion entre l'ensemble d'entraînement et l'ensemble de test pour garantir des performances fiables du modèle.
* Une répartition courante est de 70-80% pour l'ensemble d'entraînement et 20-30% pour l'ensemble de test.

*Étape 2 : Phase d'Entraînement (Training Phase)*

* Explication :
* Pendant la phase d'entraînement, les modèles d'apprentissage automatique sont entraînés sur l'ensemble d'entraînement.
* Les modèles utilisent les caractéristiques (features) de l'ensemble d'entraînement pour apprendre les relations entre les caractéristiques et la cible.
* Commentaires :
* L'objectif de cette phase est d'ajuster les paramètres du modèle pour minimiser une fonction de perte ou d'erreur, ce qui permet de trouver la meilleure représentation des données.
* Il est important de surveiller les performances du modèle pendant l'entraînement pour éviter le surajustement ou le sous-ajustement.

*Étape 3 : Phase de Test (Testing Phase)*

* Explication :
* Pendant la phase de test, les modèles entraînés sont évalués sur l'ensemble de test.
* Les modèles font des prédictions sur l'ensemble de test en utilisant les caractéristiques de cet ensemble.
* Commentaires :
* Cette phase est cruciale pour évaluer la capacité du modèle à généraliser à de nouvelles données non vues lors de l'entraînement.
* Les métriques de performance telles que l'erreur quadratique moyenne (MSE), l'erreur absolue moyenne (MAE) et le coefficient de détermination (R²) sont souvent utilisées pour évaluer les performances du modèle.

## Commentons nos résultat à chaque phase ( traning et testing)

* Sur la phase d'entraînement :
* Les performances sur l'ensemble d'entraînement peuvent sembler très bonnes, mais il est important de vérifier si le modèle n'a pas trop ajusté les données d'entraînement (surajustement).
* Un modèle qui sur ajuste les données d'entraînement peut ne pas généraliser correctement aux données de test.
* Sur la phase de test :
* Les performances sur l'ensemble de test sont cruciales pour évaluer la capacité du modèle à généraliser.
* Il est important de rechercher un écart significatif entre les performances sur l'ensemble d'entraînement et sur l'ensemble de test, ce qui pourrait indiquer un sura justement ou un sous-ajustement du modèle.
* Comparaison entre les performances d'entraînement et de test :
* Idéalement, les performances du modèle sur l'ensemble de test devraient être similaires à celles sur l'ensemble d'entraînement, indiquant que le modèle généralise bien aux données non vues.
* Si les performances sur l'ensemble de test sont significativement inférieures à celles sur l'ensemble d'entraînement, cela peut indiquer un surajustement du modèle.

## Voici une conclusion en utilisant les termes demandés (bais, variance, underfitting, overfitting) :

Interprétation de l’output de nos modèles (Algorithmes) :

**Model *Biais (Bias) Variance Underfitting/Overfitting***

|  |  |  |  |
| --- | --- | --- | --- |
| *Régression Linéaire* | Faible | Faible | Aucun |
| *Random Forest*  *Gradient Boosting* | Faible | Faible | Aucun |
| Faible | Faible | Aucun |

Les trois modèles (Régression Linéaire, Random Forest, et Gradient Boosting) semblent avoir des performances équilibrées avec une faible erreur de biais et de variance, indiquant qu'ils sont bien adaptés à la tâche de prédiction des salaires sur votre ensemble de données. Aucun signe clair de sous-ajustement ou de surajustement n'est observé, ce qui suggère qu'ils sont capables de bien généraliser aux données de test.

## Les difficultés et faisabilités

|  |  |  |
| --- | --- | --- |
| Aspect | Difficulté | Faisabilité |
| Régression | Interprétation des résultats peut-être complexe | Algorithmes disponibles pour une analyse approfondie |
| Linéaire | Assumptions strictes sur la linéarité des données | Données linéaires ou approximation acceptable |
| Sensible aux outliers et aux  valeurs aberrantes | Prétraitement des données pour gérer ces problèmes |
| Random Forest | Configuration des hyperparamètres peut être complexe | Méthodes d'optimisation automatique disponibles |
| Temps de calcul potentiellement élevé avec de grandes | Capacités de parallélisation pour accélérer le processus |
| Ensembles d'arbres | Librairies disponibles pour la  mise en œuvre |
| Gradient | Sensible au surajustement  avec des hyperparamètres mal | Techniques de validation croisée pour le réglage |
| Boosting | ajustés | Libraries fournissant des  implémentations efficaces |
| Plus de susceptibilité au bruit et aux valeurs  aberrantes | Ensembles de données de taille moyenne à grande taille |

# CONCLUSION GENERALE

À la lumière de notre analyse des salaires dans le domaine de la science des données, plusieurs conclusions peuvent être tirées. Tout d'abord, nous avons constaté que certaines variables, telles que le niveau d'expérience et le titre du poste, ont une corrélation significative avec les salaires des professionnels des données. Les Data Scientists et les Data Engineers semblent bénéficier de rémunérations plus élevées que les Data Analysts, ce qui est probablement dû à leurs compétences spécialisées et à leurs responsabilités plus avancées. De plus, des facteurs tels que la localisation géographique et le type d'entreprise peuvent également jouer un rôle important dans la détermination des salaires. Les professionnels des données travaillant dans des zones à forte demande ou pour des entreprises de grande taille ont tendance à gagner des salaires plus élevés.

Cette analyse met en évidence l'importance pour les entreprises de prendre en compte une variété de facteurs lorsqu'elles déterminent les salaires des professionnels des données. De même, pour les individus cherchant à progresser dans ce domaine, il est essentiel de reconnaître les tendances salariales et les variables qui peuvent influencer leur rémunération. En utilisant ces informations, les entreprises et les professionnels peuvent prendre des décisions plus éclairées et mieux adapter leurs stratégies de rémunération et de développement de carrière.

# LE CODE

1. # Importation des modules nécessaires
2. import pandas as pd
3. import numpy as np
4. from sklearn.model\_selection import train\_test\_split
5. import matplotlib.pyplot as plt
6. import plotly.express as px
7. import seaborn as sns
8. from plotly.offline import iplot, plot
9. import plotly.offline as pyo
10. import warnings
11. from sklearn.preprocessing import LabelEncoder
12. from sklearn.linear\_model import LinearRegression
13. from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
14. from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error, r2\_score
15. # Ignorer tous les avertissements pour une sortie plus propre
16. warnings.filterwarnings("ignore")
17. # Lecture des données à partir du fichier Excel
18. df = pd.read\_excel('jobs\_in\_data.xlsx')
19. # Affichage de quelques échantillons de données pour vérifier leur lecture
20. df.sample(10)
21. # Vérification des valeurs manquantes dans chaque colonne
22. df.isna().sum()
23. # Analyse de la distribution des catégories d'emplois
24. df\_job\_category\_general = df['job\_category'].value\_counts()
25. df\_job\_category\_general
26. # Affichage du top et du bas des catégories d'emplois
27. print(f"Top Job Needed in Four Year'{df\_job\_category\_general.idxmax()}' with Value '{df\_job\_category\_general.max()}")
28. print(f"Least Job Needed in Four Year'{df\_job\_category\_general.idxmin()}' with Value '{df\_job\_category\_general.min()}")
29. # Visualisation de la distribution des catégories d'emplois
30. iplot(
31. px.bar(
32. df\_job\_category\_general,
33. labels={'job\_category': 'Job Category'},
34. title=f'Needed of Job Category in 4 years ago',
35. color\_discrete\_sequence=['#b3079c'],
36. template='plotly\_dark',
37. text\_auto=True
38. )
39. )
40. # Description statistique des données
41. df.describe()
42. # Groupement des catégories d'emplois par année
43. df\_job\_category = df.groupby('work\_year')['job\_category'].value\_counts()
44. # Création de graphiques de barres pour chaque année
45. colors = ['#ccaa14','#8807b3','#07b324','#1007b3']
46. j=0
47. for i in range(2020,2024):
48. fig = px.bar(
49. df\_job\_category.get(i),
50. labels={
51. 'job\_category':'Job Category',
52. 'value':'Numbre of Employees'
53. },title=f'Needed of Job Category in {i}',
54. color\_discrete\_sequence=[colors[j]],
55. template='plotly\_dark',text\_auto=True
56. )
57. pyo.iplot(fig)
58. j+=1
59. print(j)
60. df\_job\_category = df.groupby('work\_year')['job\_category'].value\_counts()
61. colors = ['#ccaa14','#8807b3','#07b324','#1007b3']
62. j=0
63. for i in range(2020,2024):
64. fig = px.bar(
65. df\_job\_category.get(i),
66. labels={
67. 'job\_category':'Job Category',
68. 'value':'Numbre of Employees'
69. },title=f'Needed of Job Category in {i}',
70. color\_discrete\_sequence=[colors[j]],
71. template='plotly\_dark',text\_auto=True
72. )
73. pyo.iplot(fig)
74. j+=1
75. print(j)
76. df.groupby('work\_year')['experience\_level'].value\_counts(normalize=True).sort\_values(ascending=True).unstack('experience\_level').plot(kind = 'bar', stacked=True)
78. df.sort\_index().plot(kind='bar', stacked=True)
80. top\_5\_countries = df[df['work\_year']==2023]['company\_location'].value\_counts(normalize = True).head(5)
81. top\_5\_countries
83. boxplot\_dataframe = df[(df['work\_year'] == 2023) & (df['company\_location'].isin(top\_5\_countries.index))]
84. boxplot\_dataframe
86. medians = boxplot\_dataframe.groupby('company\_location')['salary\_in\_usd'].median().sort\_values(ascending=False).index
87. medians
88. plt.figure(figsize=(15,5))
89. sns.boxplot(
90. data = boxplot\_dataframe,
91. x = 'company\_location',
92. y = 'salary\_in\_usd',
93. hue = 'experience\_level',
94. order = medians
95. )
96. plt.show()
97. level\_order = ['Entry-level', 'Mid-level', 'Senior', 'Executive']
98. top\_paid\_USA = boxplot\_dataframe[boxplot\_dataframe['company\_location']=="United States"]
99. mean\_pay = top\_paid\_USA.groupby(['job\_category','experience\_level'])['salary\_in\_usd'].mean().unstack('experience\_level')
100. mean\_pay = mean\_pay[level\_order].sort\_values(by = 'Senior', ascending = False)
101. mean\_pay\_k = mean\_pay/1000
102. plot\_order = mean\_pay\_k.index
104. # Encodage des variables catégorielles pour la préparation des modèles
105. encoder = LabelEncoder()
106. df['job\_category'] = encoder.fit\_transform(df['job\_category'])
107. df['experience\_level'] = encoder.fit\_transform(df['experience\_level'])
108. df['job\_title'] = encoder.fit\_transform(df['job\_title'])
109. df['salary\_currency'] = encoder.fit\_transform(df['salary\_currency'])
110. df['employee\_residence'] = encoder.fit\_transform(df['employee\_residence'])
111. df['employment\_type'] = encoder.fit\_transform(df['employment\_type'])
112. df['work\_setting'] = encoder.fit\_transform(df['work\_setting'])
113. df['company\_location'] = encoder.fit\_transform(df['company\_location'])
114. df['company\_size'] = encoder.fit\_transform(df['company\_size'])
115. df['company\_size'] = encoder.fit\_transform(df['company\_size'])
116. df['company\_size'] = encoder.fit\_transform(df['company\_size'])
117. df
118. df.describe()
119. df.columns.tolist()
120. df.shape
121. # Calcul de la matrice de corrélation entre les variables
122. correlation\_matrix = df.corr()
123. # Créer une figure
124. plt.figure(figsize=(10, 8))
125. # Dessiner la heatmap de la matrice de corrélation
126. sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm', fmt=".2f")
127. # Ajouter un titre
128. plt.title('Matrice de Corrélation')
129. # save the figure
130. plt.savefig('correlation\_matrix.png')
131. # Afficher la figure
132. plt.show()
133. # Diviser les données en ensembles d'entraînement et de test
134. X = df.drop(['salary\_in\_usd'], axis=1)  # Features
135. y = df[['salary\_in\_usd']]  # Target variable
136. X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.35, random\_state=42)
137. # Initialiser les modèles pour chaque colonne cible
138. models\_salary\_in\_usd = {
139. "Linear Regression": LinearRegression(),
140. "Random Forest": RandomForestRegressor(),
141. "Gradient Boosting": GradientBoostingRegressor(),
142. }
143. # Entraîner et évaluer de chaque modèle pour la colonne 'salary\_in\_usd'
144. for model\_name, model in models\_salary\_in\_usd.items():
145. print(f"Training {model\_name} for 'salary\_in\_usd' column...")
146. model.fit(X\_train, y\_train['salary\_in\_usd'])
147. y\_pred = model.predict(X\_test)
148. mse = mean\_squared\_error(y\_test['salary\_in\_usd'], y\_pred)
149. mae = mean\_absolute\_error(y\_test['salary\_in\_usd'], y\_pred)
150. r2 = r2\_score(y\_test['salary\_in\_usd'], y\_pred)
151. print(f"{model\_name} - MSE: {mse}, MAE: {mae}, R²: {r2}")
152. # Supposons que vous ayez déjà vos données dans un DataFrame appelé df
153. # Par exemple, vos variables d'entrée sont dans une liste nommée X et votre variable de sortie est y
154. # Créer un graphique de dispersion pour chaque variable d'entrée par rapport à la variable de sortie
155. for variable in X:
156. plt.figure(figsize=(10, 6))
157. sns.scatterplot(data=df, x=variable, y='salary\_in\_usd')
158. plt.title(f"Relation entre {variable} et le salaire (USD)")
159. # save figure
160. plt.savefig(f'{variable}\_vs\_salary\_in\_usd.png')
161. plt.xlabel(f"Variable d'entrée : {variable}")
162. plt.ylabel("Salaire (USD)")
163. plt.show()