Entrée [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Chargement des données

Entrée [2]:
```
data=pd.read_excel("Insurance-data.xlsx")
data.head(10)
```

Out[2]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520
5	31	female	25.740	0	no	southeast	3756.62160
6	46	female	33.440	1	no	southeast	8240.58960
7	37	female	27.740	3	no	northwest	7281.50560
8	37	male	29.830	2	no	northeast	6406.41070
9	60	female	25.840	0	no	northwest	28923.13692

a/Description des caractéristiques

Entrée [3]:
```
description=pd.read_excel("Insurance-data.xlsx",sheet_name="Description")
description.head()
```

Out[3]:

	Description des variables	Unnamed: 1
0	NaN	NaN
1	age	âge du principal bénéficiaire
2	sex	sexe de l'assureur, femme, homme
3	bmi	Indice de masse corporelle, permettant de com...
4	children	Nombre d'enfants couverts par l'assurance mal...

Entrée [4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       1000 non-null   int64  
 1   sex       1000 non-null   object 
 2   bmi       1000 non-null   float64
 3   children  1000 non-null   int64  
 4   smoker    1000 non-null   object 
 5   region    1000 non-null   object 
 6   charges   1000 non-null   float64
dtypes: float64(2), int64(2), object(3)
memory usage: 54.8+ KB
```

Entrée [5]:
```
#Pour avoir une desciption des données numériques
data.describe()
```

Out[5]:

	age	bmi	children	charges
count	1000.000000	1000.000000	1000.000000	1000.000000
mean	39.640000	30.865565	1.080000	13099.629425
std	14.169586	6.046396	1.198765	11994.129978
min	18.000000	15.960000	0.000000	1121.873900
25%	27.000000	26.600000	0.000000	4719.683425
50%	40.000000	30.590000	1.000000	9286.850750
75%	52.000000	35.112500	2.000000	16073.095438
max	64.000000	50.380000	5.000000	63770.428010

Entrée [6]:
```
#Pour avoir une desciption des données catégorielles
data.describe(include="object")
```

Out[6]:

	sex	smoker	region
count	1000	1000	1000
unique	2	2	4
top	male	no	southeast
freq	505	803	278

b/Analyse descriptive des données

Analyse Univariée des variables quantitaves

D'après la matrice info visualisé précédemennt,on note que les colonnes numériques sont:"age","bmi","children","charges"

Entrée [7]:
```
#Gestion des valeurs manquantes
data.isna().sum()
```

Out[7]:
```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

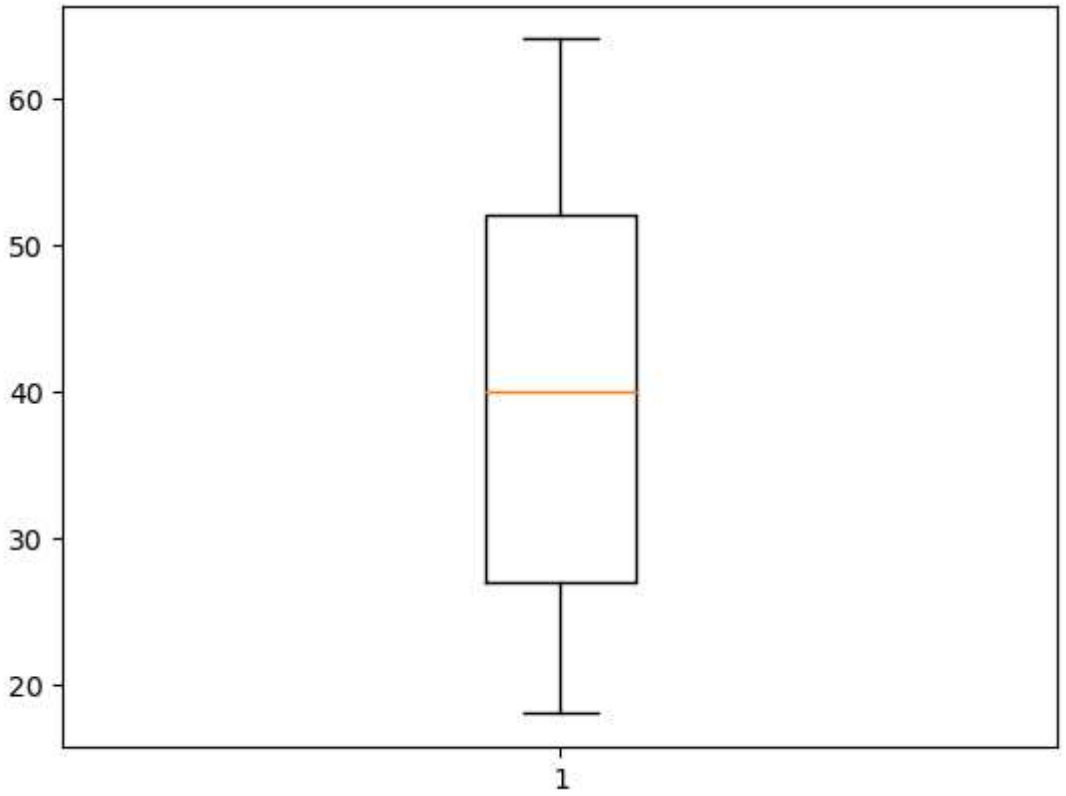
Entrée [8]:
```
data.describe()
```

Out[8]:

	age	bmi	children	charges
count	1000.000000	1000.000000	1000.000000	1000.000000
mean	39.640000	30.865565	1.080000	13099.629425
std	14.169586	6.046396	1.198765	11994.129978
min	18.000000	15.960000	0.000000	1121.873900
25%	27.000000	26.600000	0.000000	4719.683425
50%	40.000000	30.590000	1.000000	9286.850750
75%	52.000000	35.112500	2.000000	16073.095438
max	64.000000	50.380000	5.000000	63770.428010

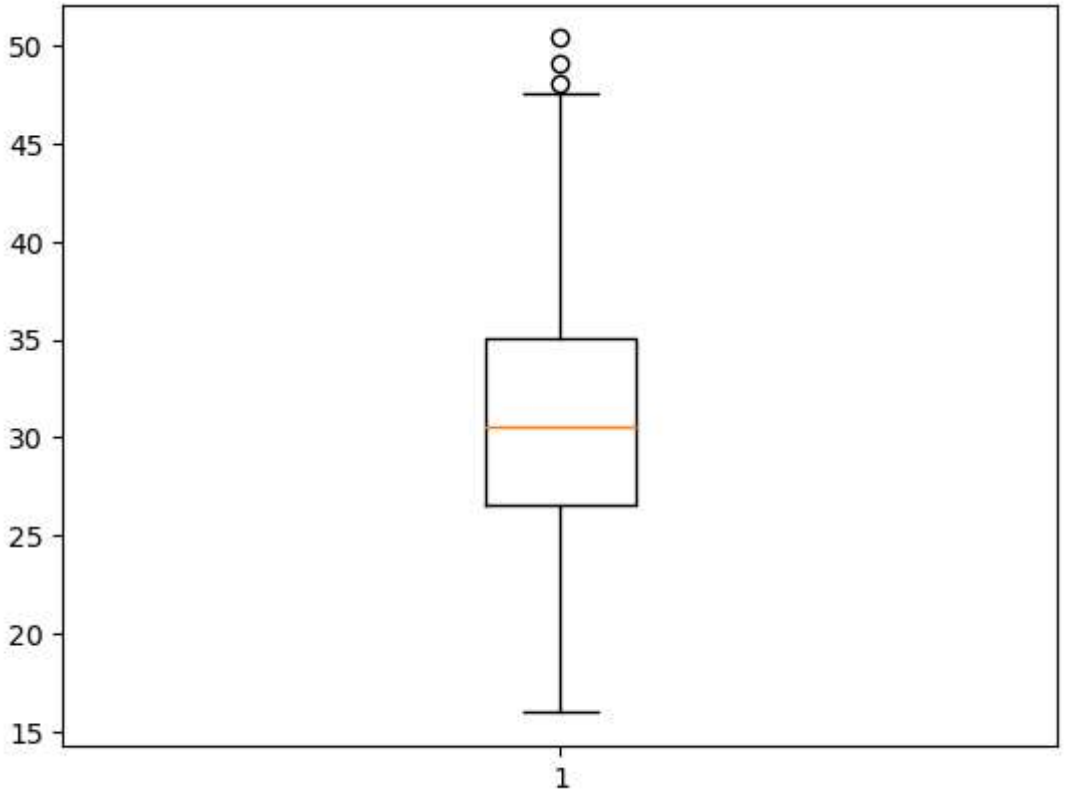
Par défaut la méthode describe ne touche que les valeurs numériques et ici toutes nos variables quantitatives.On peut remarquer que le nombre d'éléments dans chacune de ces colonnes est de 1000 à chaque niveau ,ce qui atteste aussi qu'il n'y a pas de valeur manquantes.De plus on peut aussi remarquer la présence de la moyenne qui est approximativement égale à la médiane(50%),de même qu'avec le "bmi" "children" et significativement différent de la médiane lorsqu'on est sur la colonne "charges".On peut visualiser chacune de ces données pour comprendre ce qui arrive.

Entrée [9]:
```
#Gestion des valeurs aberrantes
plt.boxplot(data["age"])
plt.show()
```



On ne remarque pas de valeurs aberrantes dans cette colonne,caractérisé par un boxplot n'affichant pas de valeurs en dehors de la plage du min-max,la différence entre la moyenne et la mediane n'était donc pas importante.(39.64-40)

Entrée [10]:
```
plt.boxplot(data["bmi"])
plt.show()
```



On peut déjà remarquer que certaines valeurs sont en dehors de la plage comme vu sur le plot.On peut étudier de près ces valeurs et voir si elles sont à éliminer ou à conserver.

Entrée [11]:
```
Q1=data['bmi'].quantile(0.25)
Q3=data['bmi'].quantile(0.75)
IQR=Q3-Q1
outliers_bmi=data['bmi'][(data['bmi']<Q1-1.5*IQR)|(data['bmi']>Q3+1.5*IQR)]
outliers_bmi.describe()
```

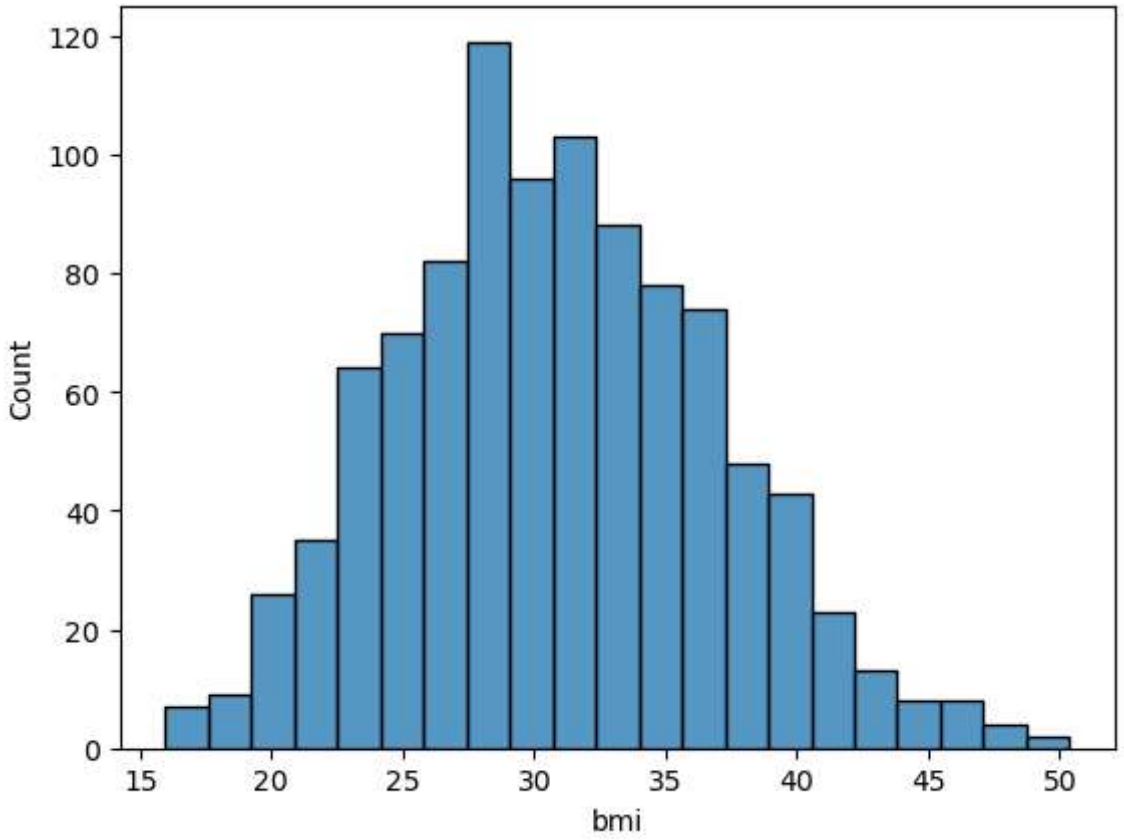
Out[11]:
```
count     3.000000
mean     49.170000
std       1.158922
min      48.070000
25%      48.565000
50%      49.060000
75%      49.720000
max      50.380000
Name: bmi, dtype: float64
```

Donc il s'agit de 3 valeurs dites "outliers",on peut regarder maintenant s'il faut ou non les supprimer

Entrée [12]:
```
data['bmi'].describe()
```

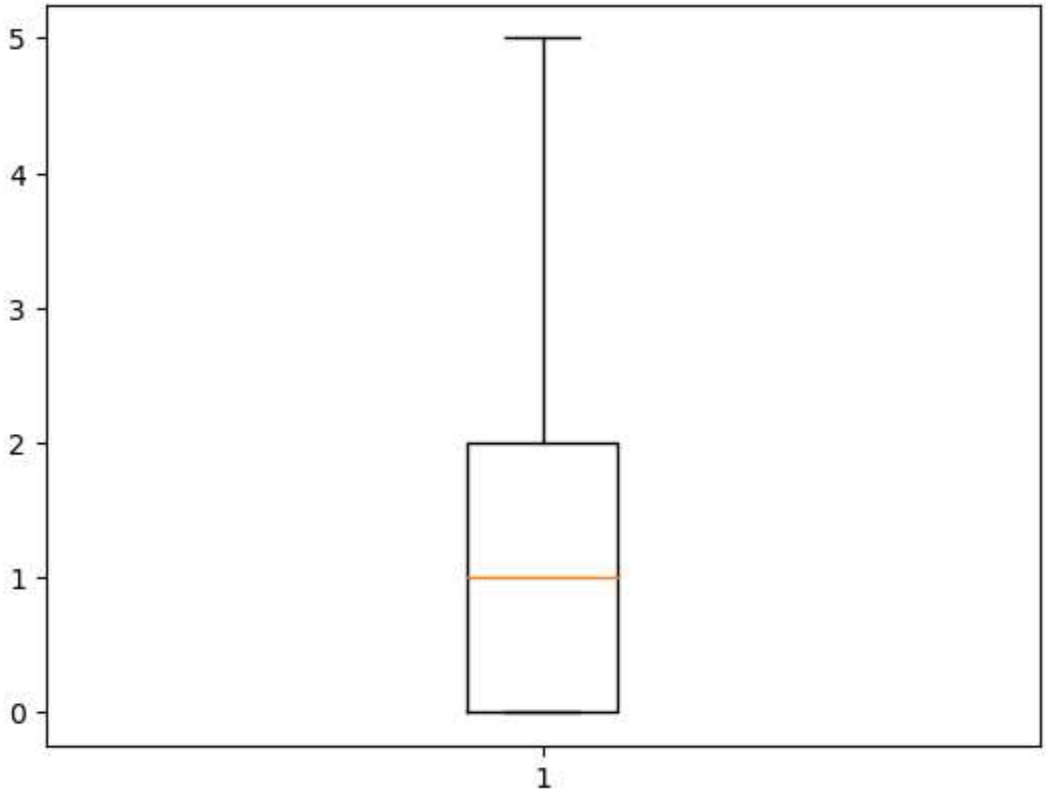
Out[12]:
```
count    1000.000000
mean       30.865565
std         6.046396
min        15.960000
25%        26.600000
50%        30.590000
75%        35.112500
max        50.380000
Name: bmi, dtype: float64
```

Entrée [13]:
```
sns.histplot(data['bmi'])
plt.show()
```



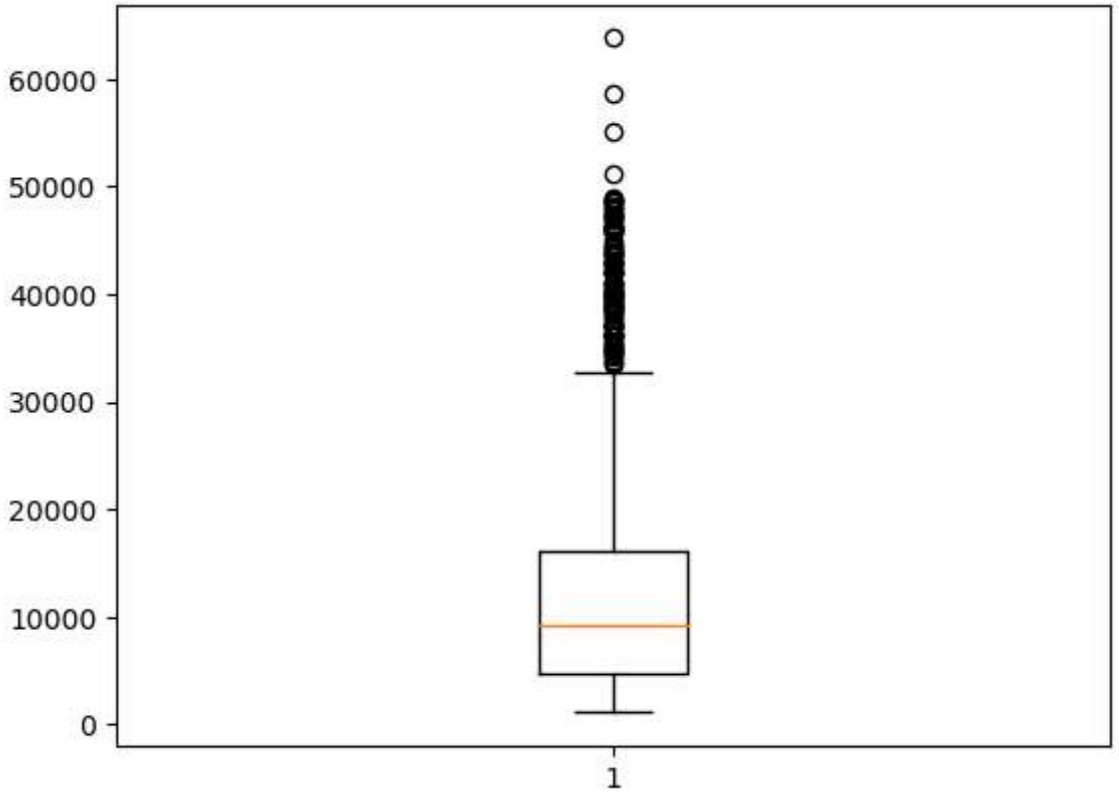
On peut constater que ces outliers ne modifient pas la distributions de la variable qui est normale donc il n'est pas pertinent de supprimer ces valeurs.

Entrée [14]:
```
plt.boxplot(data["children"])
plt.show()
```

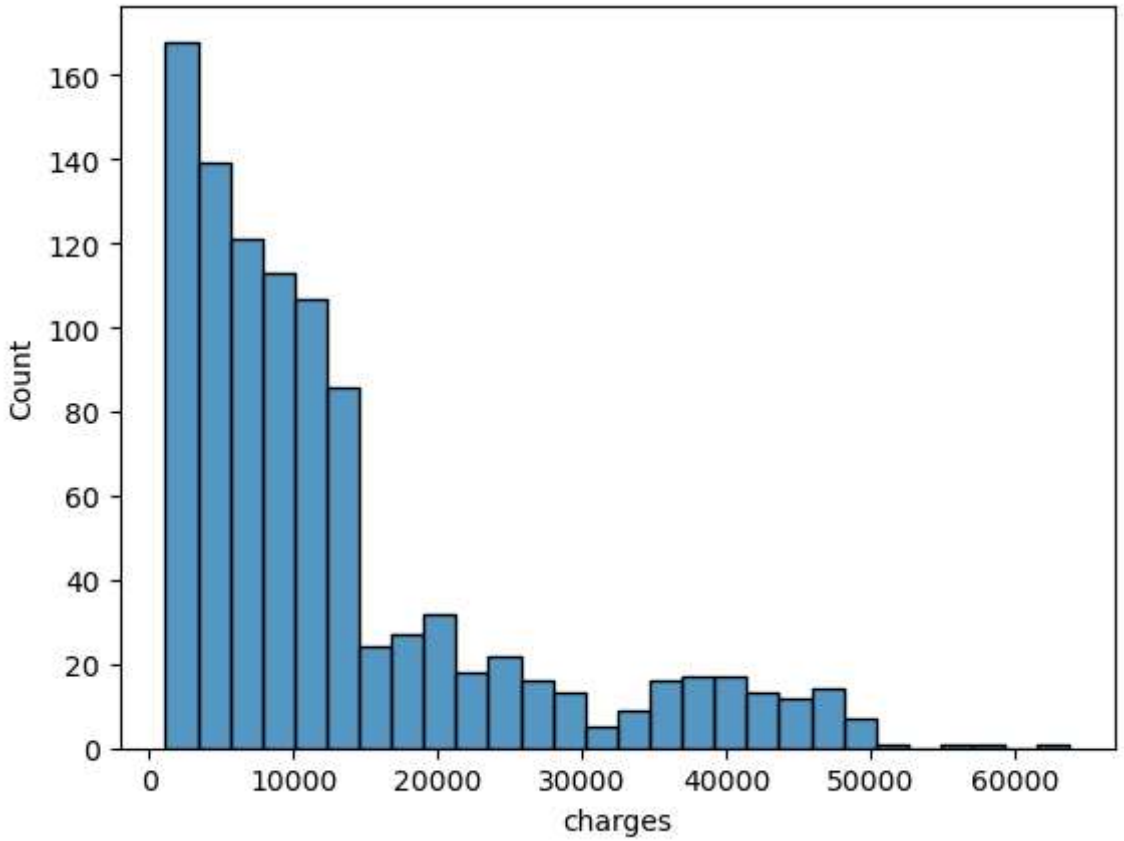


On peut déjà remarquer qu'il n'y pas d'outliers sur ce boxplot.

Entrée [15]:
```
plt.boxplot(data["charges"])
plt.show()
```



Entrée [16]:
```
sns.histplot(data['charges'])
plt.show()
```



On peut remarquer que la distribution n'est pas normale et est asymétrique,on peut voir l'impact de ces outliers sur la distribution de cette variable.

Entrée [17]:
```
Q1=data['charges'].quantile(0.25)
Q3=data['charges'].quantile(0.75)
IQR=Q3-Q1
outliers_charges=data['charges'][(data['charges']<Q1-1.5*IQR)|(data['charges']>Q3+1.5*IQR)]
describe_charges_outliers=outliers_charges.describe()
describe_charges_outliers
```

Out[17]:
```
count      106.000000
mean     41708.003564
std       5428.167921
min      33471.971890
25%      37378.082600
50%      40953.297200
75%      45708.161475
max      63770.428010
Name: charges, dtype: float64
```

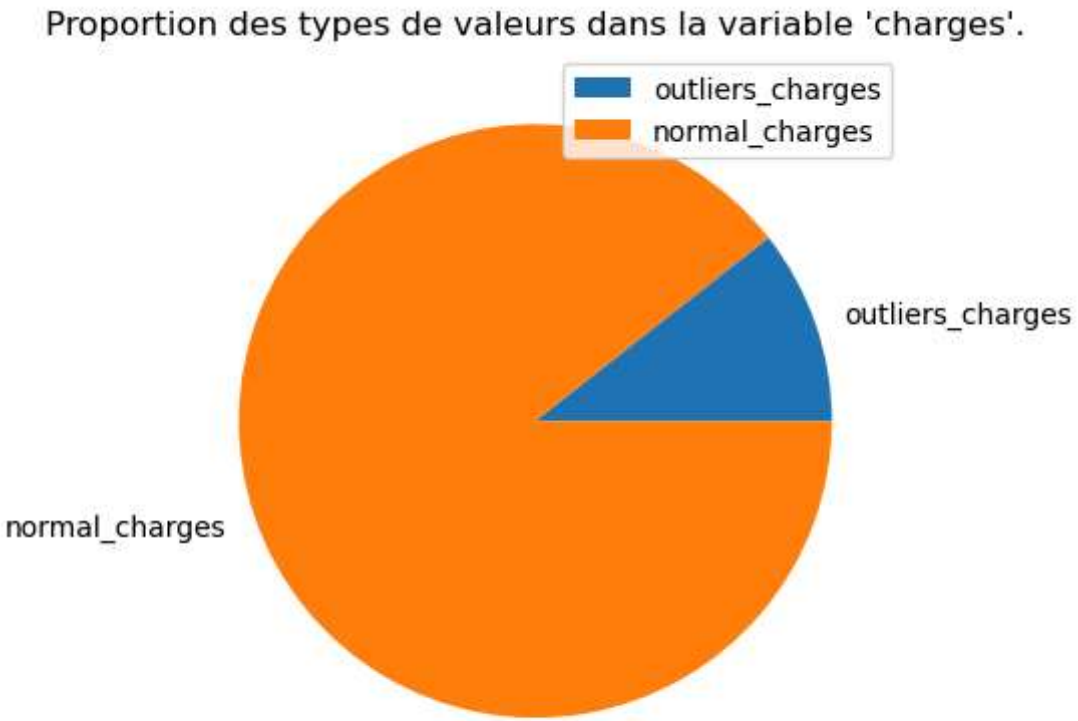
Entrée [18]:
```
describe_charges_not_outliers=data['charges'][(data['charges']>Q1-1.5*IQR)&(data['charges']<Q3+1.5*IQR)].describe()
describe_charges_not_outliers
```

Out[18]:
```
count      894.000000
mean      9707.585064
std       6985.875598
min       1121.873900
25%       4324.741937
50%       8236.843550
75%      12810.886117
max      32787.458590
Name: charges, dtype: float64
```

Entrée [19]:
```
count_charges=[]
count_charges.append(describe_charges_outliers["count"])
count_charges.append(describe_charges_not_outliers["count"])
print(count_charges)
```

```
[106.0, 894.0]
```

Entrée [20]:
```
plt.pie(count_charges,labels=["outliers_charges","normal_charges"])
plt.legend()
plt.title("Proportion des types de valeurs dans la variable 'charges'.")
plt.show()
```



On peut voir qu'il y a une quantité non-négligeable d'outliers sur cette variable soit 0.106% .D'après la description,"charges" correspond aux frais médicaux individuels (primes) facturés par l'assurance maladie.Il est normal que certaines rares personnes aient des assurances maladies plus élévés.On ne peut donc pas les supprimer directement mais attenuer l'effet de ces outliers en procédant à une transformation logarithmique. Le problème serait d'avoir un modèle incapable de généraliser corecement avec de grosses valeur lors que les données sont remises à l'échelle pour la prédiction,on va donc conserver la colonne telle qu'elle est.

Entrée [21]:
```
data['charges'].describe()
```

Out[21]:
```
count     1000.000000
mean     13099.629425
std      11994.129978
min       1121.873900
25%       4719.683425
50%       9286.850750
75%      16073.095438
max      63770.428010
Name: charges, dtype: float64
```

Analyse descriptives des variables qualitatives.

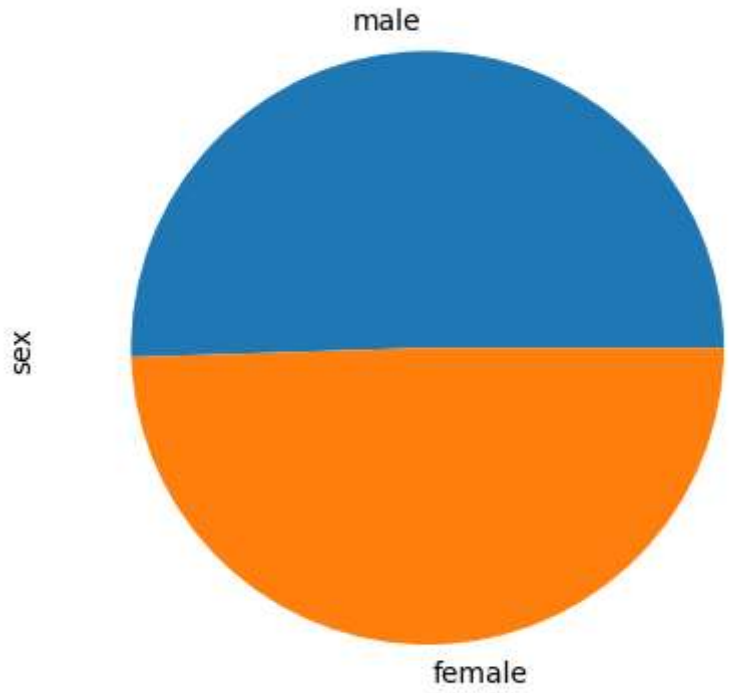
Entrée [22]:
```
data.describe(include="object")
```

Out[22]:

	sex	smoker	region
count	1000	1000	1000
unique	2	2	4
top	male	no	southeast
freq	505	803	278

Donc les variables qualitatives sont "sex" ,"smoker" et "region"

Entrée [23]:
```
data["sex"].value_counts().plot(kind="pie")
plt.show()
```

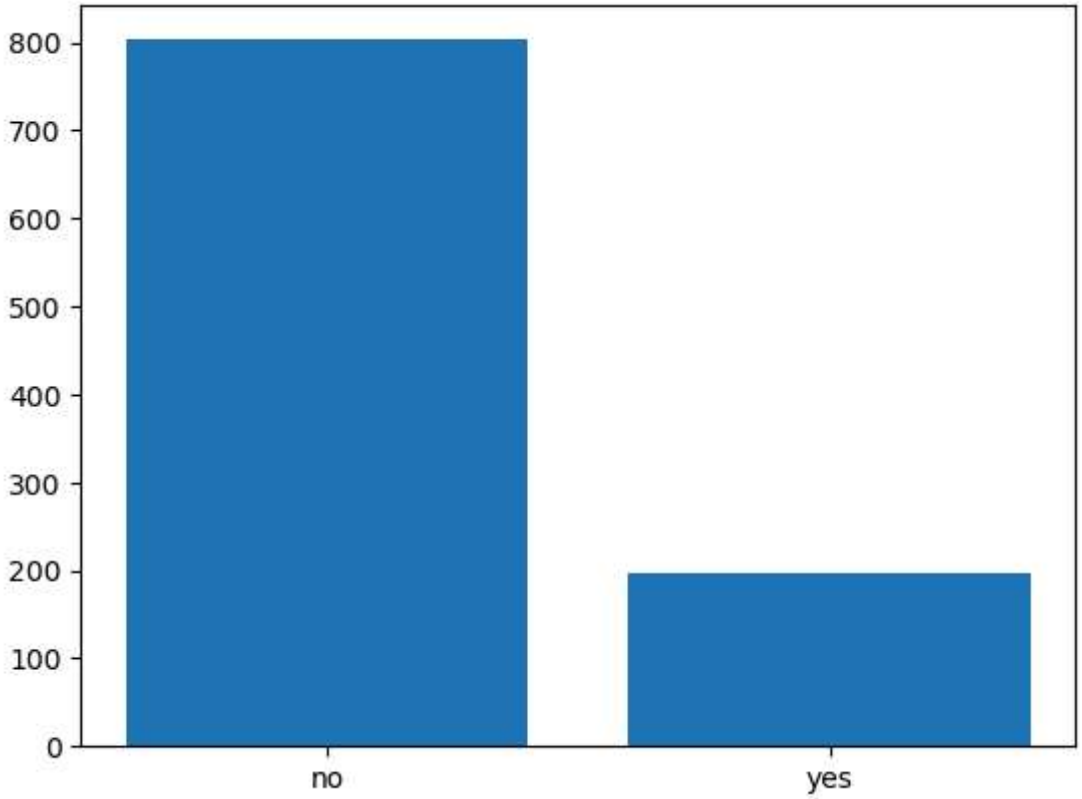


On peut dire qu'il y a 2 classes "male et "female" avec des proportions très proches .

Entrée [24]:
```
data["smoker"].value_counts()
```

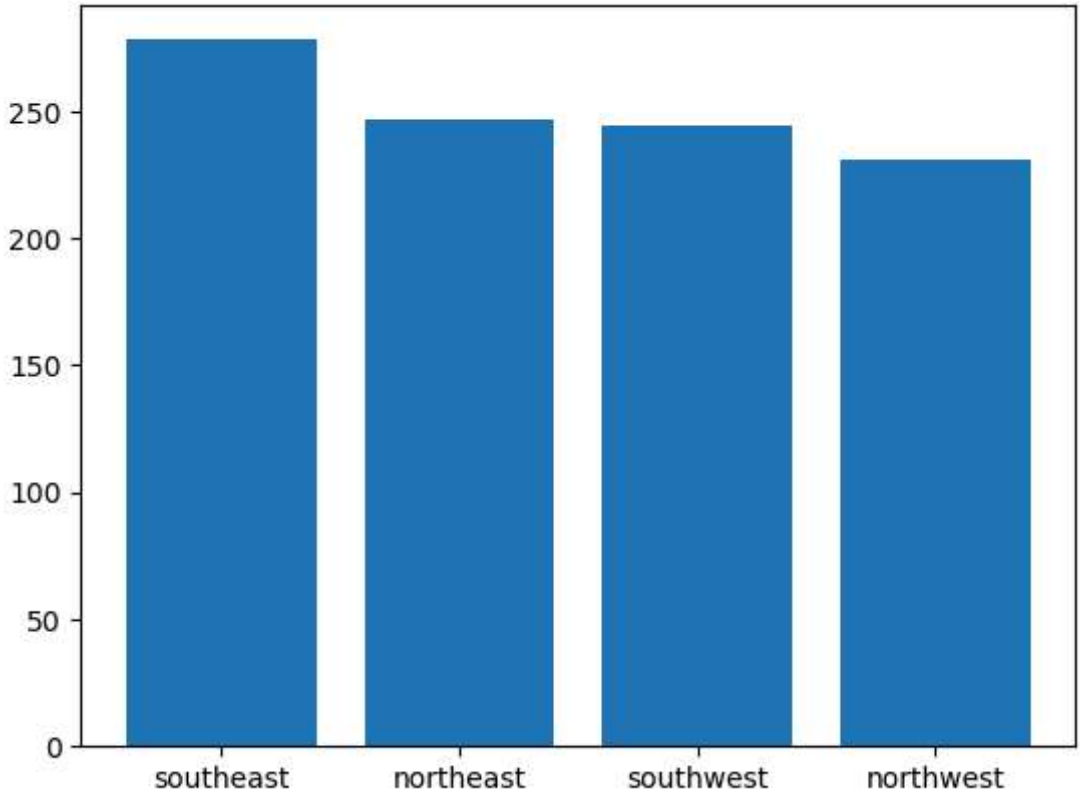
Out[24]:
```
no     803
yes    197
Name: smoker, dtype: int64
```

Entrée [25]:
```
plt.bar(data['smoker'].value_counts().index,data['smoker'].value_counts().values)
plt.show()
```



Il y a donc largment plus de non fumeur que de fumeur concernant la variable "smoker"

Entrée [26]:
```
plt.bar(data["region"].value_counts().index,data["region"].value_counts().values)
plt.show()
```

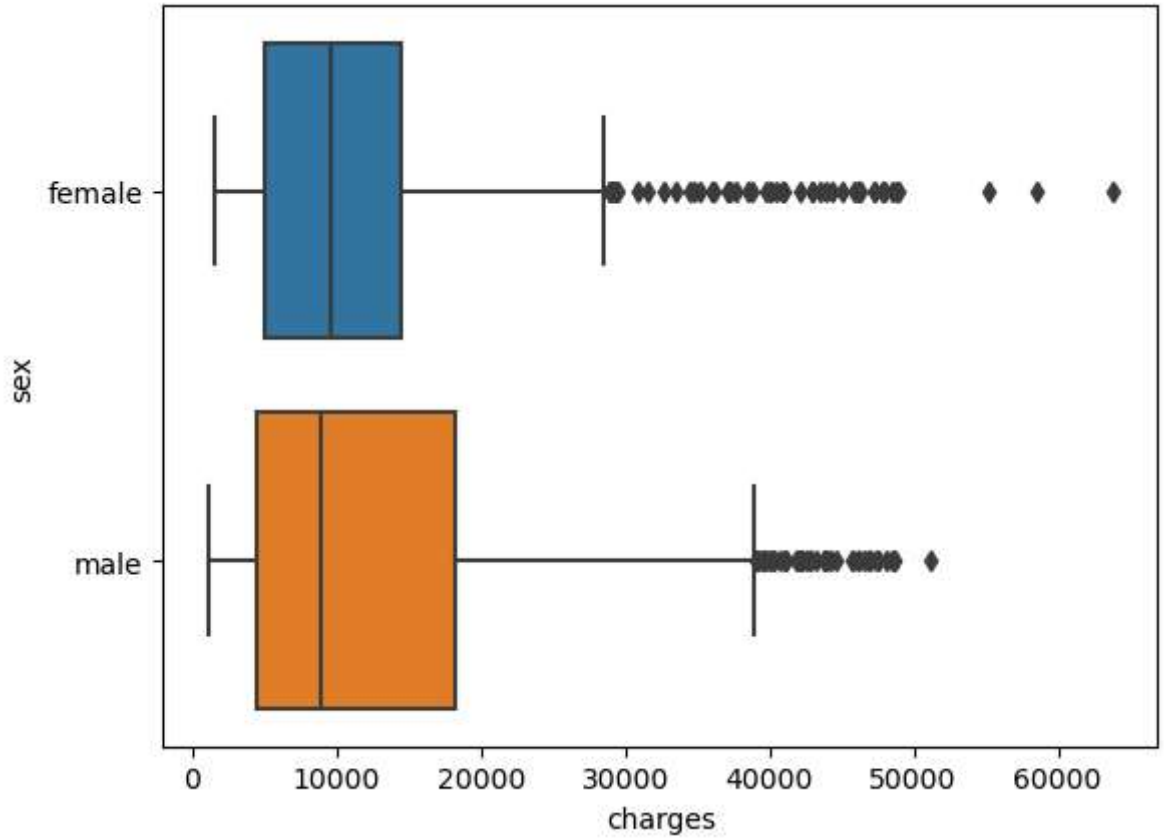


On peut remarquer que les bénéficiaire sont issus de toute part des Etats-Unis mais ceux du Sud-Est et par la même occasion de l'Est,sont ceux qui profite mieux de cette assurance.

Analyse descriptive bivariée entre la variable cible et les variables qualitatives.

Entrée [27]:
```
sns.boxplot(x=data['charges'],y=data['sex'])
```

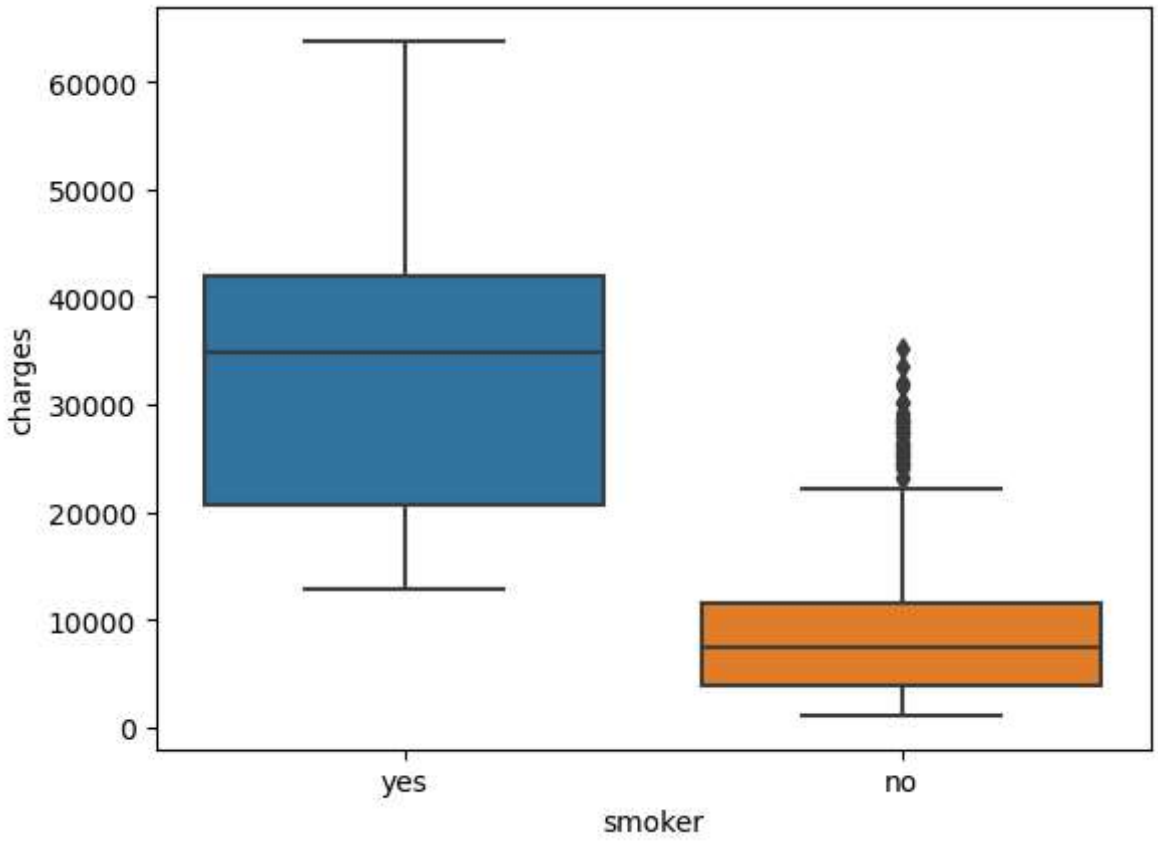
Out[27]:
```
<AxesSubplot:xlabel='charges', ylabel='sex'>
```



On a donc tendance à accorder la prise en charge plus au personnes dont les garants sont des hommes.

Entrée [28]:
```
sns.boxplot(x=data['smoker'],y=data['charges'])
```

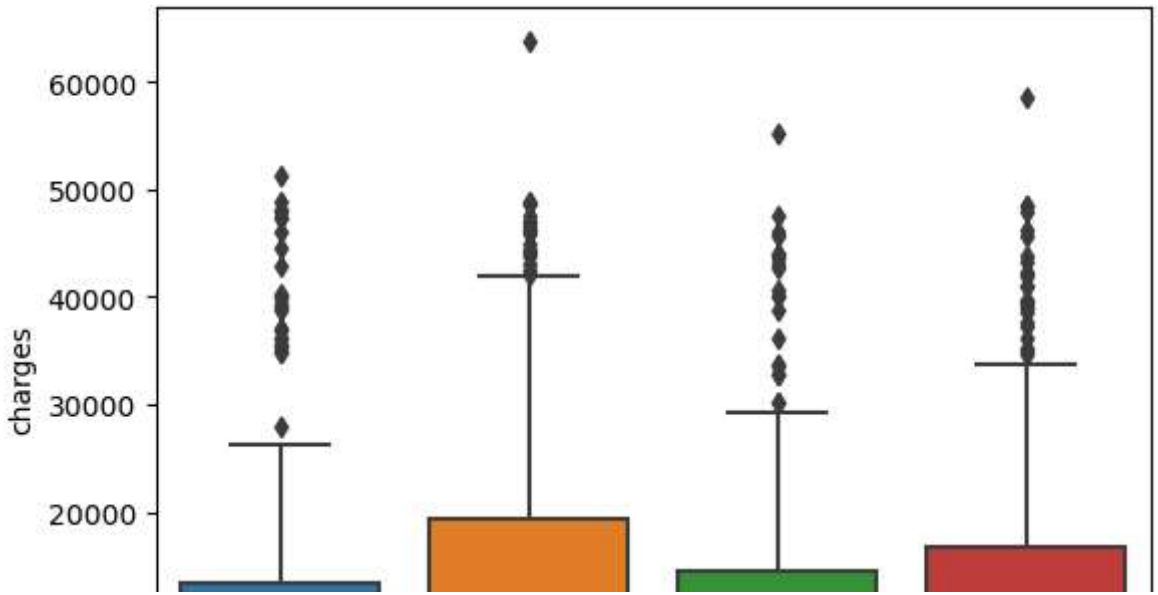
Out[28]:
```
<AxesSubplot:xlabel='smoker', ylabel='charges'>
```



On peut donc retenir que les fumeurs reçoivent plus la prise en charges médicale que les non-fumeurs et pas seulement,lorsqu'ils en reçoivent,c'est vraiment lus élévé cmparativement à ce qu'on purrait donner à un non-fumeur.Généralement un fumeur a une prise en charge plus élevé qu'un non-fumeur.On pourraitdéduire que l'état de santé de l'individu ifluence le montant de sa prise en charge.

Entrée [29]:
```
sns.boxplot(x=data['region'],y=data['charges'])
```

Out[29]: <AxesSubplot:xlabel='region', ylabel='charges'>

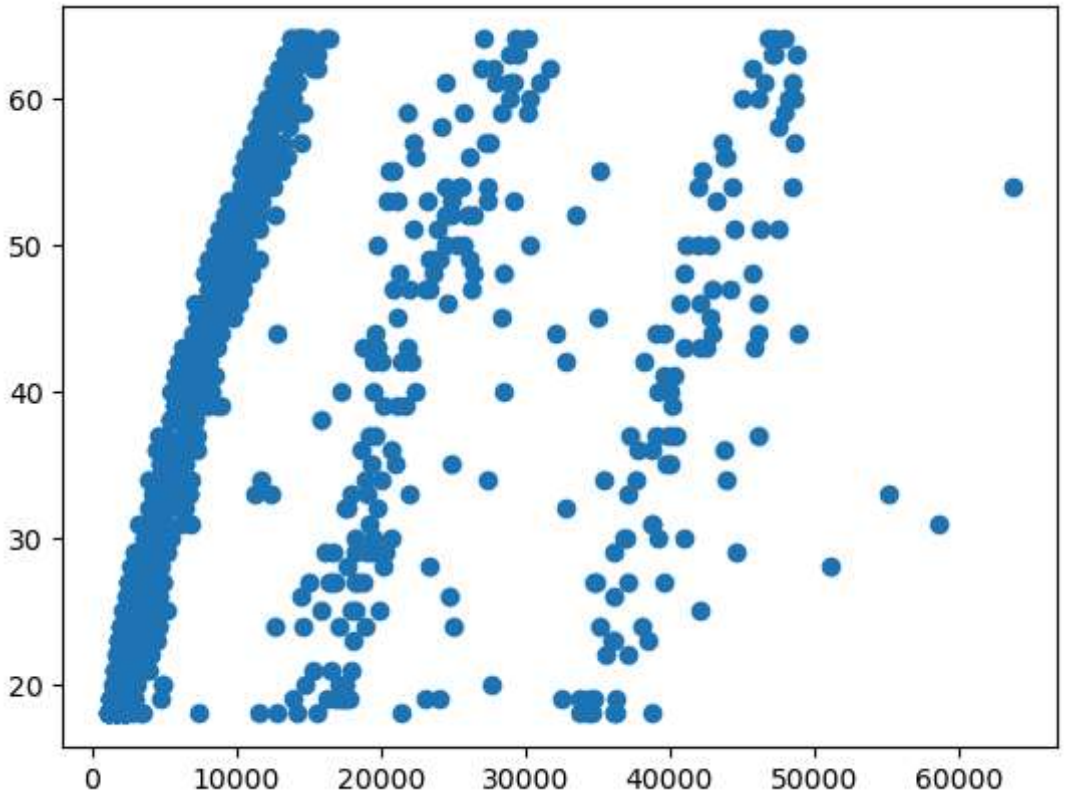


On peut constater que ceux résident dans l'Est américain reçoivent plus de prises en charges comparativement à ceux de l'Ouest.

Analyse descriptive bivariée entre la variable cible et les variables quantitatives

Entrée [30]:
```
#"age","bmi","children","charges"
plt.scatter(x=data["charges"],y=data["age"])
```

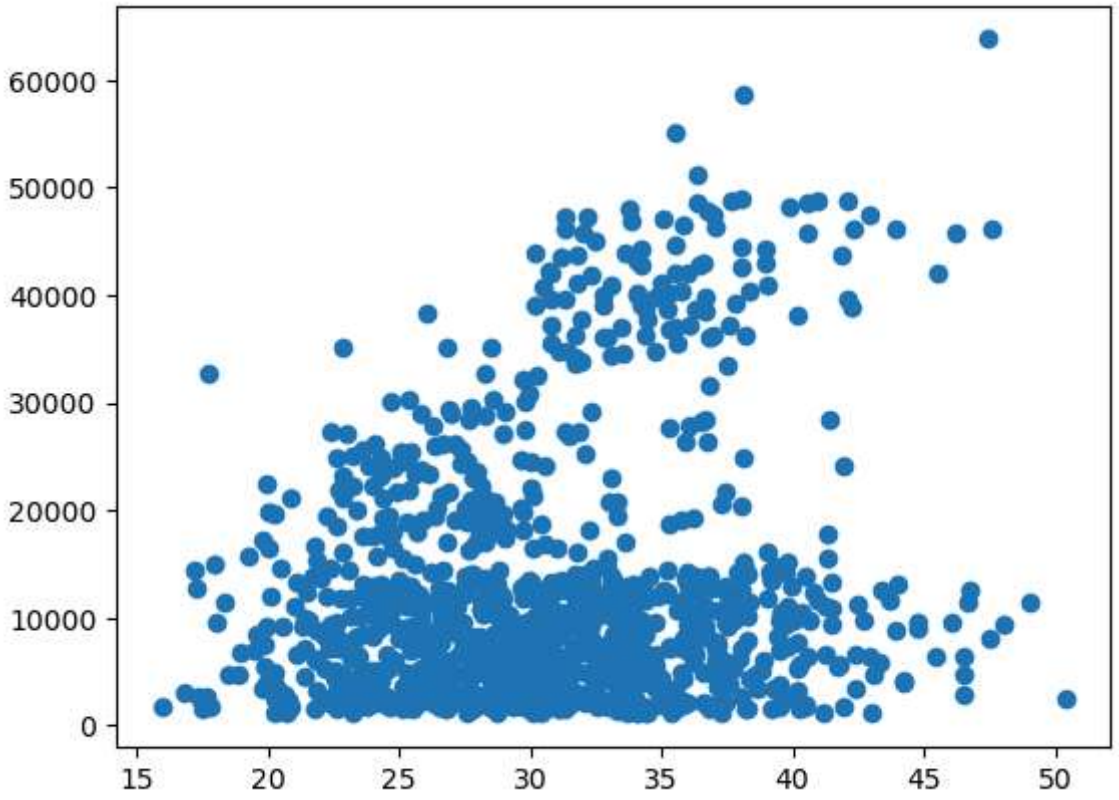
Out[30]: <matplotlib.collections.PathCollection at 0x17827dd02b0>



On peut constater que la prise en charge augmente avec l'âge de l'individu.

Entrée [31]:
```
plt.scatter(x=data["bmi"],y=data["charges"])
```

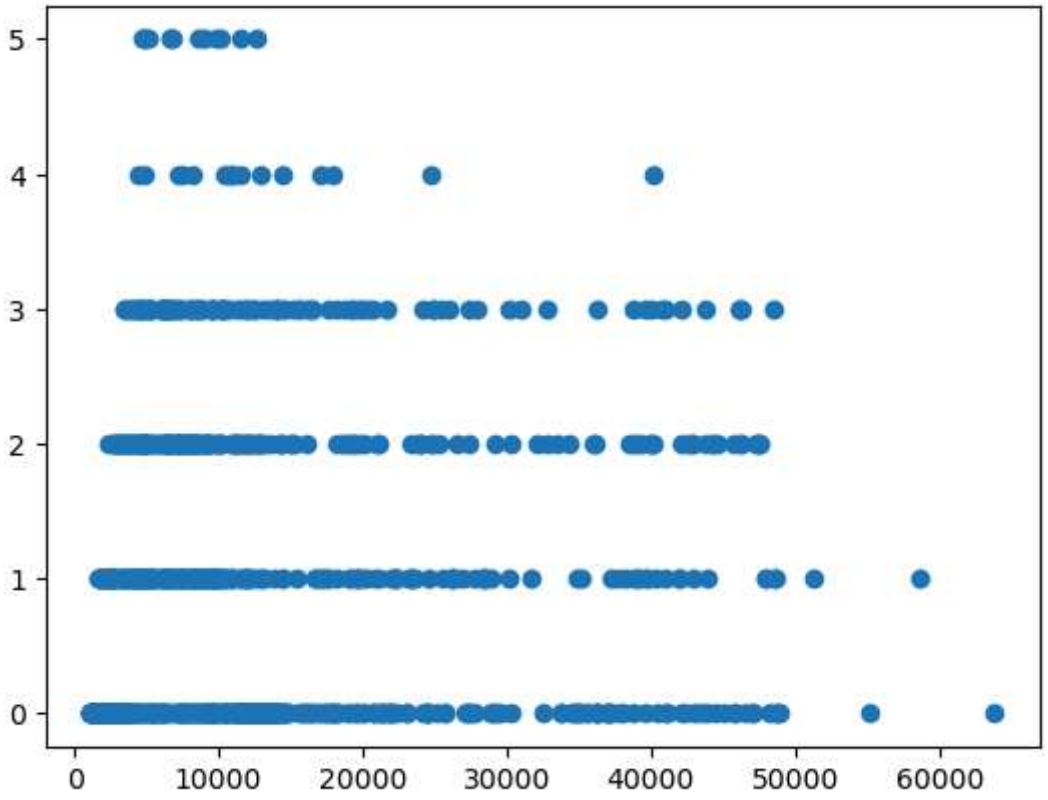
Out[31]:
```
<matplotlib.collections.PathCollection at 0x17826d00fd0>
```



On peut dire que généralement la prise en charge augmente avec la masse corporelle mais c'est beaucoup d'avoir moins de 30000 quand la masse corporelle varie entre 15 et 45.

Entrée [32]:
```
plt.scatter(x=data["charges"],y=data["children"])
```

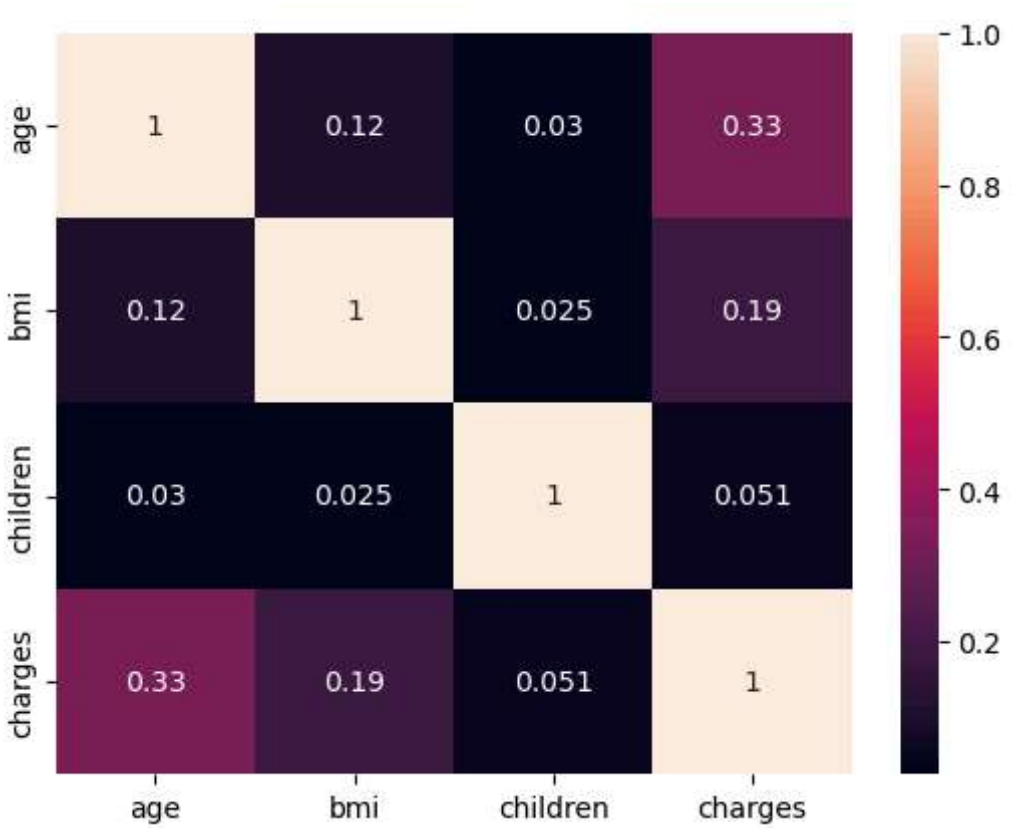
Out[32]:
```
<matplotlib.collections.PathCollection at 0x17826c23760>
```



On constate que les familles avec 0 enfants sont ceux qui reçoivent le plus la prise en charge et un peu moins en passant par les foyers à 1 ,2 et 3 enfants.Par ailleurs,les foyers avec 4 ou 5 enfants sont rarement concernés par la prise en charge.

Entrée [33]:
```
sns.heatmap(data[["age","bmi","children","charges"]].corr(),annot=True)
```

Out[33]: <AxesSubplot:>



L'âge est directement corrélé à la prise en charge de l'assurance de même que le bmi alors que le nombre d'enfant est quand très faiblement corrélé à la prise en charge.On pourrait penser que le nombre d'enfant n'influence pas la prise.

Interprétation

De l'ensemble des résultats de l'analyse descriptive on peut retenir que pour bénéficier d'une prise en charge médicale de façon générale il n'y a pas de distinction,mais plus vous êtes âgés,plus vous en recevrez cela multiplie par la même occasion les chaces d'avoir une masse corporelle plus élevé également responsable de la hauuse de la prise en charge.De façon parallèle votre état de santé,comme le fait d'être fumeur favorise l'obtention de cette prise en charge mais également le sexe du garant.Les personnes ayant des garants masculin ont plus de facilité à avoir cette

prise en charge.Par ailleurs,les plus chanceux sont ceux qui vivent dans l'est des Etats-Unis.Cette préférence régional est dû à des facteurs externes comme nos recherches l'ont révélées avec ses sources entre autres:

- Kaiser Family Foundation : https://www.kff.org/private-insurance/ (https://www.kff.org/private-insurance/)
- Commonwealth Fund : https://www.commonwealthfund.org/ (https://www.commonwealthfund.org/)
- Centre for Medicare and Medicaid Services : [https://www.cms.gov/] (https://www.cms.gov/%5D)(https://www.cms.gov/ (https://www.cms.gov/) En effet,l'Est se distingue par une couverture d'assurance plus élevée, une réglementation plus stricte de l'assurance, des coûts de santé généralement plus bas et un accès plus facile aux soins, tandis que l'Ouest offre un marché de l'assurance privée plus vaste, une utilisation accrue des soins préventifs, mais avec des disparités plus importantes dans la couverture, les coûts et l'accès aux soins, en particulier dans les zones rurales.

C'est surtout à cause de ces raisons que l'Est est beaucoup plus touché par la prise en charge que l'Ouest comme on n'a pu le remarquer plutôt.

c Pré-traitement des données

Entrée [34]:
```
data.head()
```

Out[34]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

Entrée [35]:

```
data['sex'].replace(['male','female'],[0,1],inplace=True)
data['smoker'].replace(['no','yes'],[0,1],inplace=True)
data.head()
```

Out[35]:

	age	sex	bmi	children	smoker	region	charges
0	19	1	27.900	0	1	southwest	16884.92400
1	18	0	33.770	1	0	southeast	1725.55230
2	28	0	33.000	3	0	southeast	4449.46200
3	33	0	22.705	0	0	northwest	21984.47061
4	32	0	28.880	0	0	northwest	3866.85520

Entrée [36]:

```
from sklearn.preprocessing import OrdinalEncoder
encoder=OrdinalEncoder()
data["region"]=encoder.fit_transform(data[["region"]])
data.head()
```

Out[36]:

	age	sex	bmi	children	smoker	region	charges
0	19	1	27.900	0	1	3.0	16884.92400
1	18	0	33.770	1	0	2.0	1725.55230
2	28	0	33.000	3	0	2.0	4449.46200
3	33	0	22.705	0	0	1.0	21984.47061
4	32	0	28.880	0	0	1.0	3866.85520

Le choix s'est porté sur l'Ordinal encodeur au lieu du one hot parce que d'après l'analyse les régions de provenance conditionne la prise en charge au vu du nombre de concerné dans chacune d'elle,ainsi il y a un ordre de priorité qui s'applique.

Entrée [37]:
```
data.drop_duplicates()
data.shape
```

Out[37]:
```
(1000, 7)
```

Normalisation:On va utiliser le Min max parce ue certaines variables sont à l'échelles de 1 et de plus toutes les variables ne suivent pas une distribution normale.

Entrée [38]:
```
from sklearn.preprocessing import MinMaxScaler

scalers_used={}
for i in data.columns[:-1]:
    scaler=MinMaxScaler()
    data[f"{i}"]=scaler.fit_transform(data[[f"{i}"]])
    scalers_used[f"{i}"]=scaler
df=data.copy()
data.head()
```

Out[38]:

	age	sex	bmi	children	smoker	region	charges
0	0.021739	1.0	0.346891	0.0	1.0	1.000000	16884.92400
1	0.000000	0.0	0.517432	0.2	0.0	0.666667	1725.55230
2	0.217391	0.0	0.495061	0.6	0.0	0.666667	4449.46200
3	0.326087	0.0	0.195962	0.0	0.0	0.333333	21984.47061
4	0.304348	0.0	0.375363	0.0	0.0	0.333333	3866.85520

Diviser les données,Séparer la variable cible

Entrée [39]:
```
from sklearn.model_selection import train_test_split
X=data.drop("charges",axis=1)
y=data["charges"]
```

Entrée [40]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.20,random_state=2)
```

d/Construction du modèle

-Modèle M0

Entrée [41]:
```
from sklearn.neighbors import KNeighborsRegressor
M0=KNeighborsRegressor()
M0.fit(X_train,y_train)
```

Out[41]:

KNeighborsRegressor (https://scikit-learn.org/1.5/modules/generated/sklearn.neighbors.KNeighborsRegressor.html)

KNeighborsRegressor()

Modèles M1,M2,M3

Entrée [42]:
```
case=[3,7,10]
models={}
for i in range(len(case)):
    models[f"M{case.index(case[i])+1}"]=KNeighborsRegressor(n_neighbors=case[i])
    models[f"M{case.index(case[i])+1}"].fit(X_train,y_train)
```

Entrée [43]:
```
print(models)
```

```
{'M1': KNeighborsRegressor(n_neighbors=3), 'M2': KNeighborsRegressor(n_neighbors=7), 'M3': KNeighborsRegressor(n_neig
hbors=10)}
```

Evaluation des performances

Entrée [44]:
```
from sklearn.metrics import mean_squared_error
models_mse={}
for i in models.keys():
    y_pred_=models[i].predict(X_test)
    models_mse[f"{i}"]=mean_squared_error(y_test,y_pred_)
print(f"Mean Squared Error{models_mse}")
```

```
Mean Squared Error{'M1': 36082709.31319956, 'M2': 27273567.712235246, 'M3': 29421748.57657776}
```

Entrée [45]:
```
y_pred=M0.predict(X_test)
M0_mse=mean_squared_error(y_test,y_pred)
print(f"M0_mse : {M0_mse}")
```

```
M0_mse : 28278300.82576779
```

Interprétations
On peut déjà remarquer que le modèle par défaut fait moins d'erreur que le modèle M1 et M3 mais plus élevé que le M2.

Par ailleurs,on remarque qu'en changeant le nombre de voisins,les performances du modèle sont assi influencées donc la performance du modèle repose sur le choix d'un k idéal.

e/Optimiser le modèle pour K=10

Optmiser les hyperparamètres weights,p, et algorithm

Entrée [46]:
```
from sklearn.model_selection import GridSearchCV
params_grid={
    'weights':['uniform','distance'],
    'p':[1,2],
    'algorithm':['ball_tree','kd_tree','brute','auto']
}

grid_search=GridSearchCV(models['M3'],params_grid,cv=5,scoring='neg_mean_squared_error',n_jobs=-1)


grid_search.fit(X_train,y_train)

best_prams=grid_search.best_params_
print(f"Meilleurs hyper paramètres :{best_prams}")

best_model=grid_search.best_estimator_
```

```
Meilleurs hyper paramètres :{'algorithm': 'ball_tree', 'p': 1, 'weights': 'distance'}
```

Evaluer les performances du modèle

Entrée [47]:
```
y_pred_gs=best_model.predict(X_test)
mse=mean_squared_error(y_test,y_pred_gs)
print("MSE du meilleur modèle :",mse)
```

```
MSE du meilleur modèle : 27471033.81482392
```

Entrée [48]:
```
from sklearn.metrics import r2_score
y_pred3=models['M3'].predict(X_test)
y_pred2=models['M2'].predict(X_test)
R2_score=r2_score(y_test,y_pred)
R2_score3=r2_score(y_test,y_pred3)
R2_score2=r2_score(y_test,y_pred2)
R2_score_gs=r2_score(y_test,y_pred_gs)
print(f"On a pour le modèle pr défaut un R2 de {R2_score} et pour le modèle après gread_search un R2 de {R2_score_gs},
```

```
On a pour le modèle pr défaut un R2 de 0.8376748079313231 et pour le modèle après gread_search un R2 de 0.84230874168
17825, M2 qui donne 0.8434422511959239 et M3 donnant 0.8311111046552906
```

Entrée [49]:
```
print(R2_score3)
```

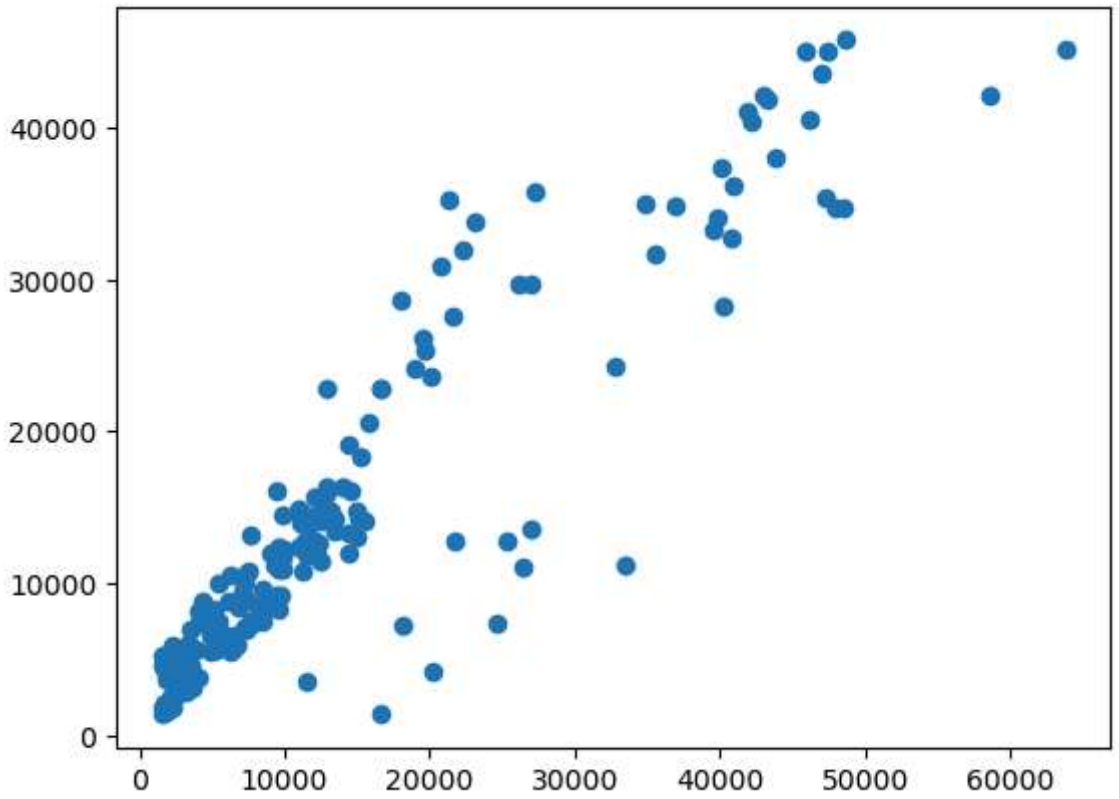
```
0.8311111046552906
```

Comparer les résultats au modèle par défaut
On peut retenir que le modèle par défaut est certes performant mais moins que celui obtenu par grid search qui est meilleur que M3.Le grid search a donc amélioré les performances de bases du modèle.Cependantle M2 reste le meilleur modèle boteu sur la série.Le choix du k idéal est donc crucial avant de penser au grid search.

Choix du modèle optimal
Le modèle optimal est M2 car il est celui qui fait le moins d'erreur et a le meilleur r squared.

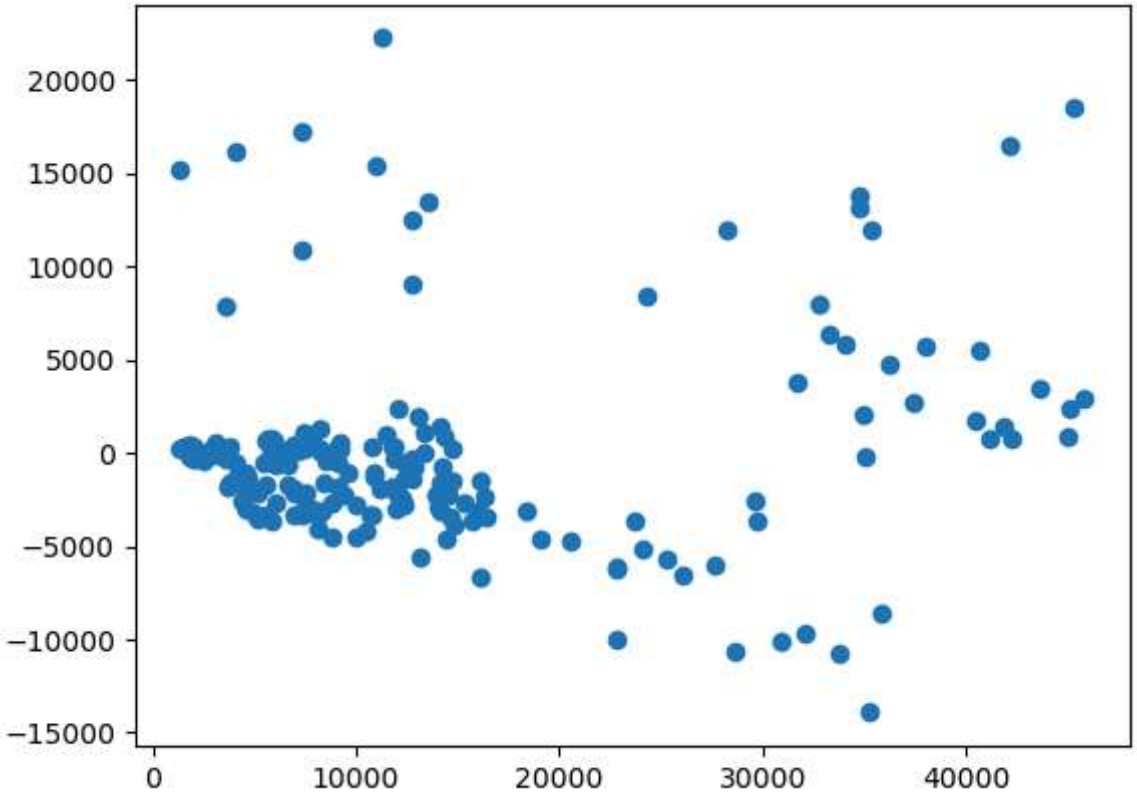
Analyser le meilleur modèle

Entrée [50]:
```
plt.scatter(x=y_test,y=y_pred2)
plt.show()
```



On peut comprendre qu'au fur et à mesure que valeur réelle augmente,la valeur prédite augmente aussi ce qui est positif pour l'analyse montrant que le modèle évolue dans le bon sens

Entrée [51]:
```
plt.scatter(x=y_pred2,y=y_test-y_pred2)
plt.show()
```



Le nuage de point des valeurs réisudelles ne montre aucune forme ou tendance,les points sont concentrés autour de 0,en montant on voit que les points se dispersent.On peut donc dire que le modèle est généralement bien ajusté mais qu'il y a des risques que l'erreur augmente lorsque le modèle a affaire à de grandes données.

Interprétation

Grâce à ces nuages de point on peut dire que le fait qu'il y ait une corrélation positive entre les valeurs réelles et celles prédites,de même qu'une dispersion des points autour de 0 pour les résidus de modèle montre qu'il peut gloablement généraliser.

g/Synthèse générale

Commentaires
Au terme de cette étude nous pouvons retenir que l'obtention d'une prise en charge médicale par l'assurance dépend de nombreux paramètres à l'instar de la zone de provenance,l'état de santé de l'individu,le nombre d'enfant.Par ailleurs,le modèle obtenu est généralement performant mais comme le montre le nuage de point des résidus,on peut comprendre que l'erreur est plus grand avec des valeurs plus grandes,ce qui est provoqué par la rareté des grandes données dans le dataset de départ suggérant qu'il s'agissait de valeurs aberrantes que nous avons quand même garder pour permettre au modèle de quand même mieux généraliser même si l'erreur est toujours présente.Aussi notons qu'en dehors des facteurs étudiés d'autres facteurs externes entre en jeux et pouvant permettre d'améliorer ce modèle ,cela prouvé car la l'assurance est aussi conditionné par la zone de provenance des individus.

Conclusion
Avoir une prise en charge médicale ,c'est avoir de l'âge,un état de santé négatif,mais aussi le moins d'enfants possible à charge.

Solution
Le plus gros problème surtout lié à la rareté des grandes sommes pour la prise en charge et la disparité entre les zones même s'il réflète la réalité des choix sont quand même des sources de biais qu'il faudra corriger avec plus de données et l'intégration de nouvelles variables explicatives en plus des anciennes afin d'avoir une meilleure description de la réalité.

Entrée [52]:
```
#sauvegarde du modèle
import joblib
joblib.dump(models['M2'],'best_insurance.pkl')
#Sauvegarde des scaler qui a servi à normaliser
joblib.dump(scalers_used,'scalers.pkl')
#Sauvegarder l'encodeur de région
joblib.dump(encoder,'region_encoder.pkl')
```

Out[52]:
```
['region_encoder.pkl']
```

Entrée [53]:

```
def Predict(data):
    #data=pd.DataFrame({"age":age,"bmi":bmi,"children":children,"smoker":smoker,"region":region})
    best_model=joblib.load('best_insurance.pkl')
    scalers=joblib.load('scalers.pkl')
    region_encoder=joblib.load("region_encoder.pkl")
    new_data=data.copy()
    new_data['sex'].replace(['male','female'],[0,1],inplace=True)
    new_data['smoker'].replace(['no','yes'],[0,1],inplace=True)
    new_data["region"]=region_encoder.transform(new_data[["region"]])
    new_data.head()
    print(scalers)

    for i in new_data.columns:
        if i in scalers:
            print(i)
            scaler=scalers[f"{i}"]
            print(scaler)
            new_data[[f"{i}"]]=scaler.transform(new_data[[f"{i}"]])
            #new_data.head()


    charges=best_model.predict(new_data)
    return charges
```

Entrée [54]:
```
predict=pd.read_excel('insurance-data.xlsx',sheet_name="PREDICT")
predict.drop('a ',axis=1,inplace=True)
#predict.head()
predictions=Predict(predict)
#Predict(predict)==data.columns[:-1]
```

```
{'age': MinMaxScaler(), 'sex': MinMaxScaler(), 'bmi': MinMaxScaler(), 'children': MinMaxScaler(), 'smoker': MinMaxSca
ler(), 'region': MinMaxScaler()}
age
MinMaxScaler()
sex
MinMaxScaler()
bmi
MinMaxScaler()
children
MinMaxScaler()
smoker
MinMaxScaler()
region
MinMaxScaler()
```

Entrée [55]:
```
pd.DataFrame(predictions,columns=["Prédictions"]).to_excel("Predict.xlsx")
```

Partie 2: Autres modèles

Régression Linéaire

Entrée [56]:
```
from sklearn.linear_model import LinearRegression
model=LinearRegression()
model.fit(X_train,y_train)
```

Out[56]:
```
LinearRegression
(https://scikit-learn.org/1.5/modules/generated/sklearn.linear_model.LinearRegression.html)
LinearRegression()
```

Entrée [57]:
```
y_pred=model.predict(X_test)

mse=mean_squared_error(y_test,y_pred)
r2=r2_score(y_test,y_pred)
print(f"mse Linear Regression: {mse}")
print(f"Linear Regression R2: {r2}")
```

```
mse Linear Regression: 34556753.07202567
Linear Regression R2: 0.8016347723914523
```

Decision Tree Regressor

Entrée [58]:
```
from sklearn.tree import DecisionTreeRegressor
model_tree=DecisionTreeRegressor(random_state=42)
model_tree.fit(X_train,y_train)
```

Out[58]:
```
▾ DecisionTreeRegressor (https://scikit-learn.org/1.5/modules/generated/sklearn.tree.DecisionTreeRegressor.html)

DecisionTreeRegressor(random_state=42)
```

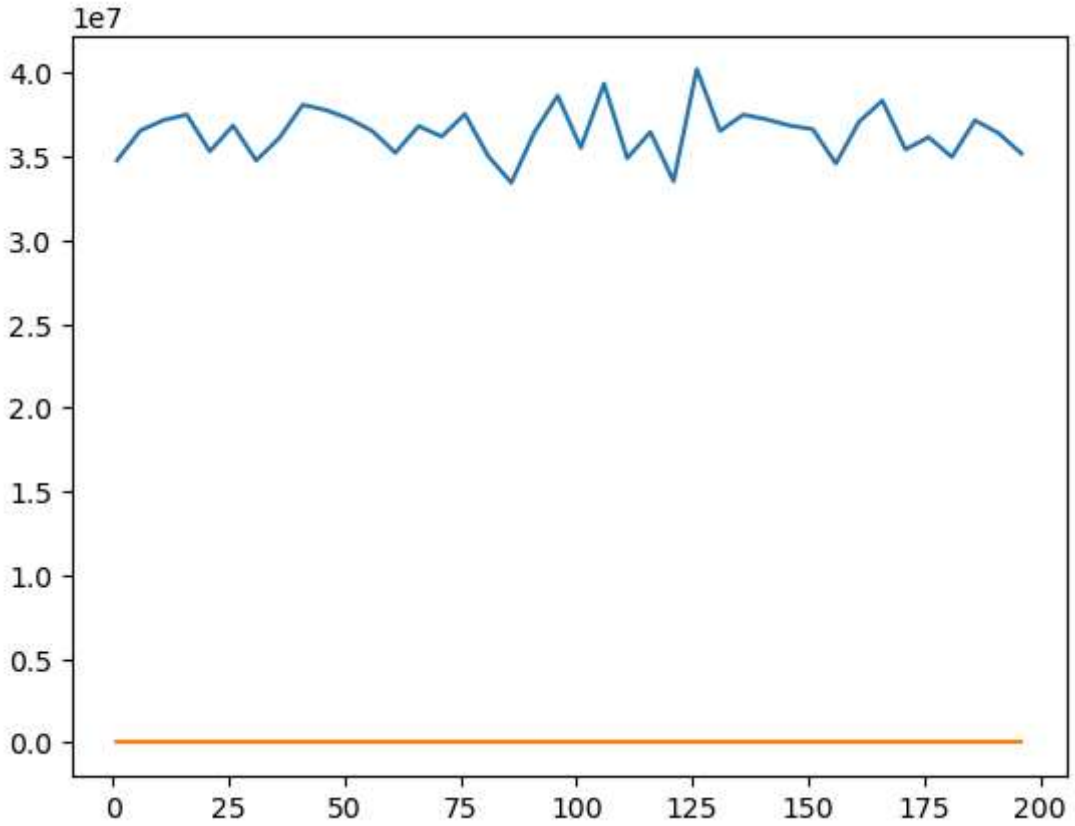
Entrée [59]:
```
y_pred_tree=model_tree.predict(X_test)

mse_tree=mean_squared_error(y_test,y_pred_tree)
r2_tree=r2_score(y_test,y_pred_tree)
print(f"mse Linear Regression: {mse_tree}")
print(f"Linear Regression R2: {r2_tree}")
```

```
mse Linear Regression: 35372408.895633586
Linear Regression R2: 0.7969526845586345
```

Entrée [60]:

```
delta_r2=[]
delta_mse=[]
lot=range(1,200,5)
for i in lot:
    model_tree=DecisionTreeRegressor(random_state=i)
    model_tree.fit(X_train,y_train)
    y_pred_tree=model_tree.predict(X_test)
    mse_tree=mean_squared_error(y_test,y_pred_tree)
    r2_tree=r2_score(y_test,y_pred_tree)
    delta_mse.append(mse_tree)
    delta_r2.append(r2_tree)
plt.plot(lot,delta_mse)
plt.plot(lot,delta_r2)
plt.show()
```



Entrée [61]:
```
delta_mse.index(min(delta_mse))
```

Out[61]:
```
17
```

Entrée [62]:
```
delta_r2[17]
```

Out[62]:
```
0.80812176492001
```

Entrée [63]:
```
max(delta_r2)
```

Out[63]:
```
0.80812176492001
```

Donc on peut se dire que la meilleure graine sur une rangée de 200 par bon de 5 est 17 car c'est celui qui permet d'avoir les meilleures perfomances sur cet intervalle.

Support Vector Machines

Entrée [66]:
```
from sklearn.svm import SVR
model_svr=SVR(kernel='rbf')
model_svr.fit(X_train,y_train)
```

Out[66]:
```
SVR (https://scikit-learn.org/1.5/modules/generated/sklearn.svm.SVR.html)
SVR()
```

Entrée [69]:
```
y_pred_svr=model_svr.predict(X_test)

mse_svr=mean_squared_error(y_test,y_pred_svr)
r2_svr=r2_score(y_test,y_pred_svr)
print(f"MSE SVR : {mse_svr}")
print(f"r2 SVR: {r2_svr}")
```

```
MSE SVR : 191989484.30677563
r2 SVR: -0.10207222517647252
```

On peut donc constater que dans ce cas un modèle non-linéaire n'est pas adapté pour réussir le travail.